The Third International Conference on
Artificial Intelligence Applications on Wall Street
Tactical and Strategic Computing Technologies

Editor
Roy S. Freedman

June 6-9, 1995 • New York City

Sponsored by
Pace University
School of Computer Science and Information Systems

In Cooperation with:
American Association for Artificial Intelligence (AAAI)
Association for Computing Machinery/SIGART (ACM/SIGART)
Society for the Management of AI Resources and Technology - Financial Services (SMART-F$)
International Association of Knowledge Engineers (IAKE)
Society for Quantitative Analysis (SQA)

Supporting Publications:
AI Expert• AI in Finance • Annals of Mathematics and AI • Wall Street & Technology • Stocks and Commodities

Proceedings

The Third Annual International Conference on Artificial Intelligence Applications on Wall Street

Dr. Roy S. Freedman, *Editor*

Proceedings

The Third Annual International Conference on Artificial Intelligence Applications on Wall Street

Dr. Roy S. Freedman, *Editor*

June 6 - 9, 1995 • New York, New York

Sponsored by:
Pace University
School of Computer Science and Information Systems

In Cooperation with:
American Association of Artificial Intelligence (AAAI)
Association for Computing Machinery/SIGART (ACM/SIGART)
Society for the Management of AI Resources and Technology - Financial Services (SMART-F$)
International Assocation of Knowledge Engineers (IAKE)
Society for Quantitative Analysis (SQA)

Published by the Software Engineering Press

Copyright © 1995 by the Software Engineering Press. All rights reserved. Abstracting or reprinting is permitted with credit to the source. Copying without fee is permitted provided that the copies are not made or distributed for direct commerical advantage and credit to the source is given.

ISBN 0-938801-09-0

Order from:
Software Engineering Press
973C Russell Avenue
Gaithersburg, MD 20879

Cover Design: Dawn Graphics, Washington, D.C.

Logistics: International Association of Knowledge Engineers (IAKE)

Printed in the United States

Chairman's Introduction

> Ils disent queles éclipses présagent malheur, parce que les malheurs sont ordinaires, de sorte qu'il arrive si souvent du mal, qu'ils devinent souvent; au lieu que s'ils disaient qu'elles présagent bonheur, ils meutiraient souvent. Ils ne donnent le bonheur qu'à des rencontres du ciel rares; ainsi ils manquent peu souvent à deviner.
>
> They say that eclipses predict bad luck. But bad luck is common, so that when bad things happen, they frequently predict it. On the other hand, if they would say that eclipses predict good luck, they would often be lying. They only attribute good luck to rare heavenly conjunctions; therefore they fail less often in prediction.
>
> Blaise Pascal (1623-1662), Penseés, no. 173.

The *Third International Conference on Artificial Intelligence Applications on Wall Street* is organized to continue the momentum generated from the conferences held in 1991 and 1993. Our goal is to provide a serious international forum where the newest applications of knowledge-based technologies for trading, asset allocation, and regulation can be discussed and evaluated.

The last two years have emphasized the fact that we live in a world full of risk. In some sense, risk is a statistical description of danger — an attribute that rational beings seek to minimize. Predicting natural (earthquakes, hurricanes, famine), political (war, terrorism), and financial disasters are difficult, to say the least, unless one is as pessimistic as Pascal. His point is that since disasters happen so often, then on average, any indicator will predict one. On the other hand, a description of risk purely in terms of mathematical expectation is misleading, since this sense of risk replaces the risk of an individual with the risk of an average. This is what can get investors into trouble.

Over the past year, we have seen how the risks associated with currency devaluation, interest rate movements, and leverage in derivative markets led to near-disasters for some market participants investors, and governments. Some of these situations could have been avoided if the dangers associated with these instruments were made more comprehensible. I believe that one of the responsibilities of conferences like this is to help make financial risk understandable.

This conference could not have happened without the help of numerous people. I first want to acknowledge the support of this year's sponsor, the Pace University School of Computer Science and Information Systems, and in particular, Dean Susan Merritt and Professor David Sachs. Thanks to Susan Atwell of Pace for smoothly coordinating the paper submission and review process. I also want to acknowledge the help of the cooperating societies and supporting publications; our paper session and panel session chairs; our invited speakers; and of course, our international Program Committee, who did a thorough job in carefully reviewing the many submitted papers. Special thanks to Pat White of Systemsware Corporation, for her diligence in preparing for this international event and for publishing the quality Proceedings. At Inductive Solutions, I want to thank Marie De Luca and Stacy Pennebaker for their help in organizing the program.

Roy S. Freedman
Program Chairman — AI/WS-95
Inductive Solutions, Inc.
New York City. June, 1995

Program Committee

Program Chair: Roy S. Freedman, *Inductive Solutions*

Chidanand Apte, *IBM*

Michael Benaroch, *Syracuse University*

John DeSaix, *NASD*

Rinaldo DiGiorgio, *Sun Microsystems*

Don Dueweke, *New York Stock Exchange*

Susan Garavaglia, *Dun & Bradstreet*

Mike Gargano, *Pace University*

Arnold Gia-Shuh Jang, *Springfield (Hong Kong)*

Ypke Hiemstra, *Vrije Universitei (Amsterdam)*

Ken Kleinberg, *Gartner Group*

Yuval Lirov, *Lehman Brothers*

Joseph Mathai, *Fidelity Investments*

Ross Miller, *General Electric*

Dan Schutlzer, *Citibank*

Stephen Slade, *New York University Stern School of Business*

Kar Yan Tam, Hong Kong University of Science and Technology

Milton White, *DATANAMICS, Inc.*

Takahira Yamaguchi, *Shizuoka University (Japan)*

Bradford Leach, *New York Mercantile Exchange*

Table of Contents

Infrastructure Modeling

Advanced Forecasting Techniques

Expert Systems and Hybrid Approaches

Risk Management

Data Analysis, Modeling, and Representation

Optimization: Portfolios and Profit

Improving Neural Network Models

Fundamental and Value Strategies

Derivatives

Trading Floor Support

Paper Session: Infrastructure Modeling

Chair: Rinaldo DiGiorgio, Sun Microsystems

Modeling Business Applications with the OODB Ownership Relationship

Michael Halper
Dept. of Math & Computer Science
Kean College of New Jersey
Union, NJ 07083 USA

Yehoshua Perl, Oscar (Ou) Yang, James Geller
CIS Department and CMS
New Jersey Institute of Technology
Newark, NJ 07102 USA

Abstract

Ownership is a very important relationship in the business world. It is endowed with rich semantics and various complexities with respect to both the owner and the property that is owned. In this paper, we present a formal model of "ownership" relationships in the context of an Object-Oriented Database (OODB) system. As our motivation, we employ three scenarios involving various ownership relationships that exhibit a wide range of distinctions. Essential aspects of ownership relationships are their related transactions such as sale, lease, and donation. Since certain of these can be applied with respect to specific kinds of ownership, while others cannot, we need to explicitly model this behavior in order to properly represent ownership in an OODB system. The ownership relationship, at times, exhibits inheritance behavior, where the values of certain attributes are derived with respect to it. With these issues in mind, we have identified and formally defined various characteristics (which we call the dimensions) of ownership. Our ownership model incorporates all these to capture the functionality of ownership's transactions and inheritance.

1 Introduction

Ownership is a very important relationship in the business world. It is endowed with rich semantics with respect to the owner and the property that is owned. As used in the corporate world, ownership can exhibit a hierarchical structure. For example, one company can own other companies.

Because of its complexity, modeling ownership in the context of a database system can be an extremely difficult task. In this paper, we introduce an "ownership" relationship model that can be integrated into an Object-Oriented Database (OODB) system. The use of this relationship greatly facilitates the problem of modeling real-world ownership and of enforcing its associated constraints.

To motivate the need for an ownership relationship and to see what kinds of problems one might encounter when trying to model it, we shall employ three example scenarios, pictured in Fig. 1, 2, and 3. Corresponding database schemata are included in the Figures, which have been drawn using our OOdini graphical notation [9]. Ownership is denoted by a bold, dotted arrow. This symbol was chosen for the mnemonic association between the dots and the "o" in "ownership."

Let us now describe these scenarios. In the first one, Jim and his business partner David own a manufacturing business that produces an item for which Jim holds a patent. The business resides in a building which Jim owns and for which a bank, First Nat'l Trust, holds a lien. Jim and David have a joint business bank account. Jim rents his house from Tom. Jim also uses a car that is legally owned by his business. A car owned by Jim is used by his son John. Jim and David each have individual bank accounts and investment portfolios, consisting of corporate stocks and government bonds. In addition, each possesses a life insurance policy and the appliances in their homes.

In the second scenario, Chrysler owns Jeep, Plymouth, and Dodge, each of which in turn owns subsidiaries, manufacturing plants, industrial equipment, etc. Dodge and Mitsubishi jointly own the Eagle Corporation. Chrysler, being a public company, issues stock that is owned by shareholders who are persons or other corporations.

The third scenario deals with an individual investor. Here, Jack owns several portfolios of investments, including stocks, bonds, and mutual funds. Together, these portfolios represent all of Jack's investments.

Against these scenarios, one is liable to carry out any of the various transactions associated with ownership, such as sales, leases, or donations. Such transactions are often restricted due to complex constraints. For example, under some circumstances, the sale of an object might be disallowed. Ownership also exhibits "inheritance" behavior, where val-

Copyright © 1995 Software Engineering Press

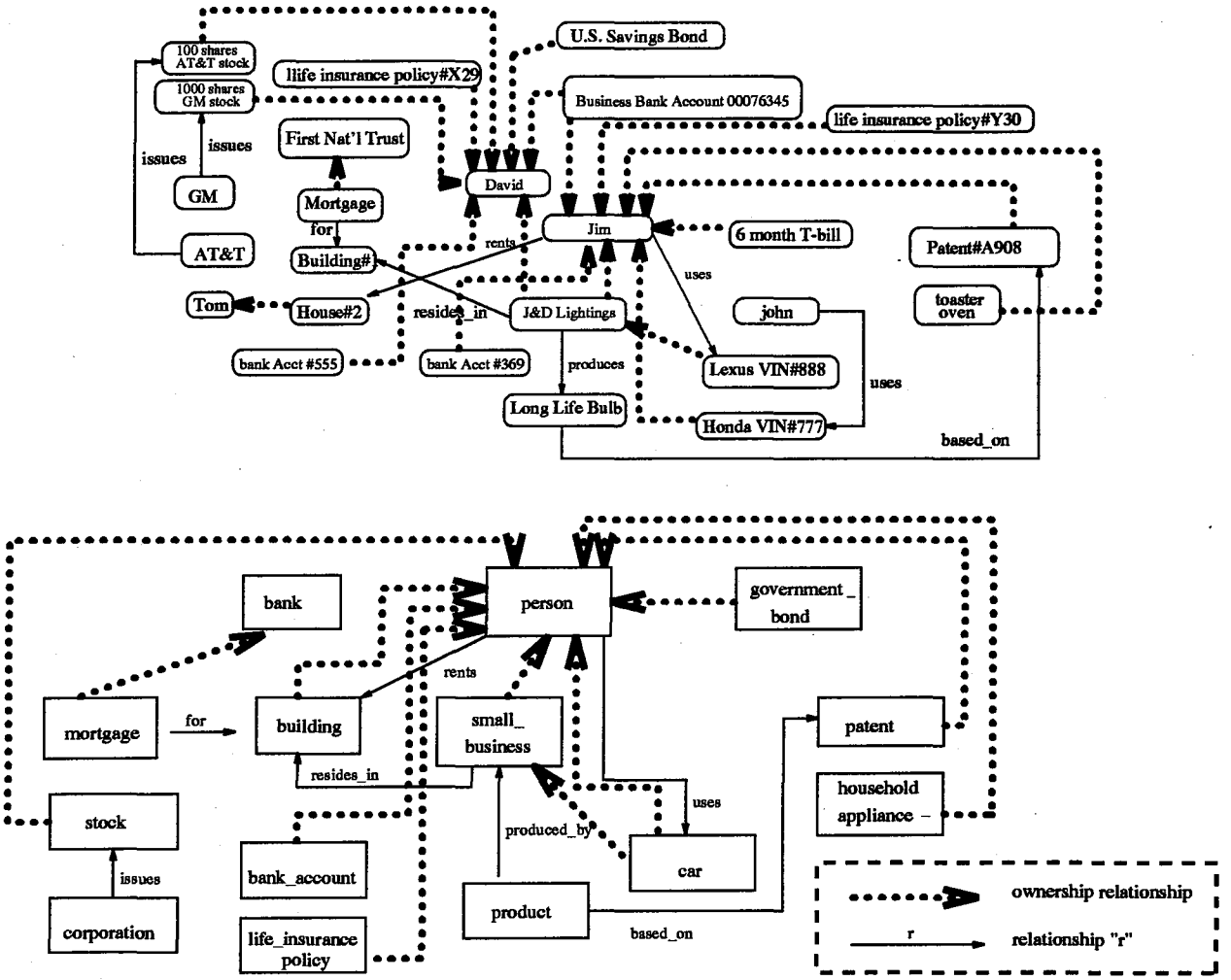


Figure 1: Instance and schema diagrams for first scenario.

ues of certain attributes of objects can be derived from other attributes via ownership relationships. We note that an investor's net worth can be determined directly from the values of his or her portfolios.

As can be gathered from the above, dealing with the issues of ownership can be very complex. To model such behavior in the context of a traditional relational database system [3, 4] would require that some programmer write programs (apart from the database system itself) to ensure that all business transactions are carried out in accordance with the semantics of ownership. However, if one were to bypass such programs, then no guarantee could be made about the database's integrity. Certain "illegal" ownership transactions may be allowed to commit, leaving various constraints violated and the database in an invalid state.

In an OODB system, the programmer could incorporate the additional constraint satisfaction code into the methods of any classes that participate in ownership relationships. However, this would still require manual programming labor. Furthermore, the correctness of the code would be very difficult to verify given all the subtleties of ownership. Ensuring that the programmer has properly accounted for all the necessary constraints in the program code would be nearly impossible.

In order to avoid such *ad hoc* programming approaches and remove the burden from the programmer, we present an ownership relationship model that expands and enhances an existing OODB data model. This relationship serves to encapsulate the rich semantics of ownership and its related trans-

Copyright © 1995 Software Engineering Press

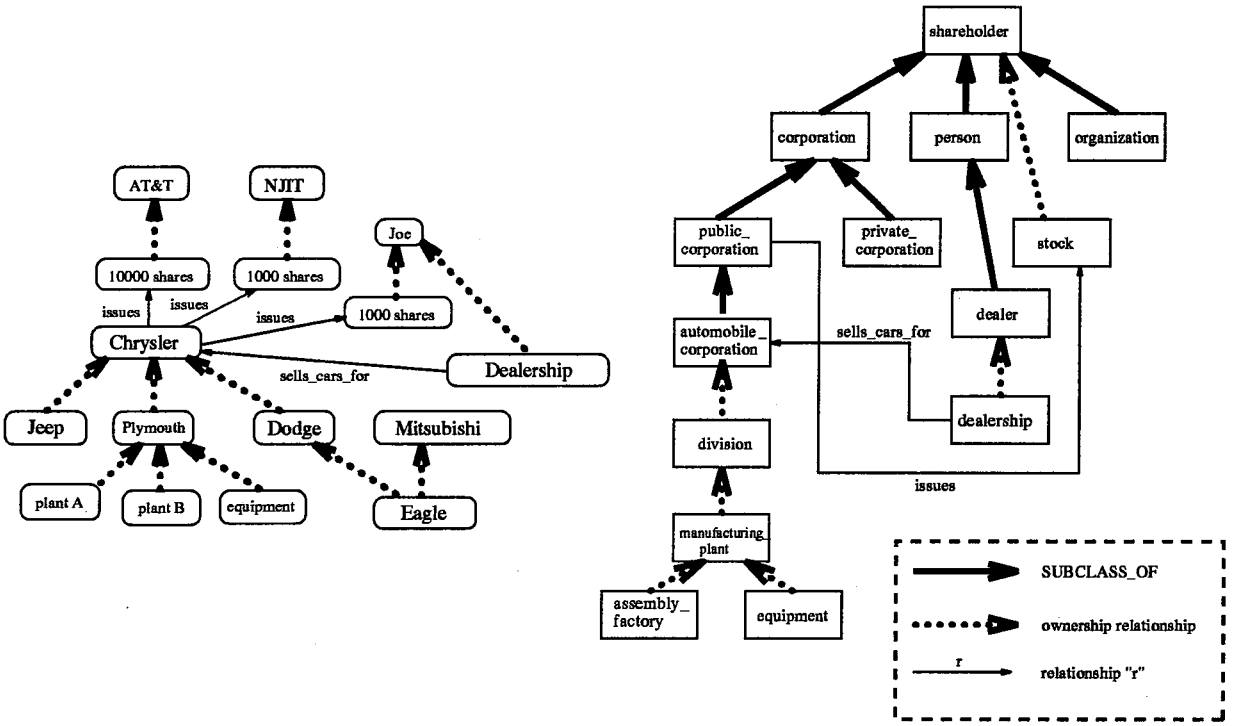


Figure 2: Instance and schema diagrams for second scenario.

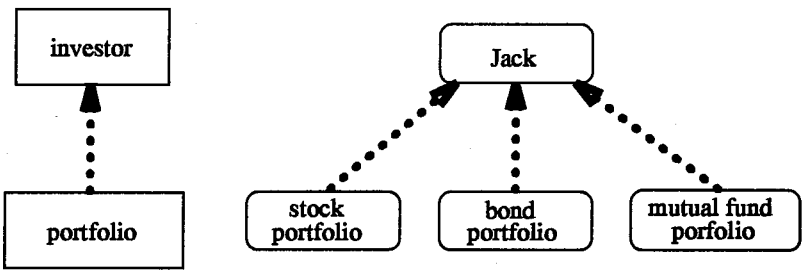


Figure 3: Schema and instance diagrams for third scenario.

actions. Using it, one can declaratively specify the desired behavior and then allow the OODB system to enforce it. In this sense, the ownership relationship is a "semantic relationship," meaning that its interpretation does not lie solely "in its name" [13] but rather in its constraint-satisfaction and inheritance mechanisms. In this light, our ownership relationship can be viewed as a modeling primitive of an OODB system with built-in semantics. In previous research, the IS-A (or SUBCLASS) relationship [2, 12] and the PART-OF relationship [6, 7, 8, 10, 11] have been modeled as semantic relationships.

The remainder of this paper is organized as follows. In Section 2, we discuss the legal definition of ownership. In Section 3, we formally define the ownership relationship and describe its "characteristic" dimensions, which capture the wide range of distinctions exhibited by ownership. Section 4 contains concluding remarks.

2 Definition of Ownership

When we describe a state of "ownership," we must, in general, include the following three features: (1) The owner, (2) the property that is owned, and (3) the characteristics of the relationship between the two. We are interested in identifying what types of objects can fill the roles of (1) and (2), and what the characteristics that distinguish the various kinds

Copyright © 1995 Software Engineering Press

of ownership are.

According to *Webster's Dictionary*, *ownership* is defined as follows:

1. The state or fact of being an owner.

2. Proprietorship; Legal right of property; Legal or just claim or title (to something); in law, the right to use for one's own advantage some property.

The owner referred to above can, by law, be a natural person, a corporation, or an organization. The latter two are, in general, referred to as *legal entities*. Under the law, legal entities are vested with certain powers, some of which are also held by natural persons. Others, like the power to exist in perpetuity, are unique to legal entities. In our databases, we see that Jim as a natural person owns his business. The Chrysler Corporation as a legal entity owns Dodge. In Fig. 1, *bank*, *small_business*, and *corporation* are legal entities. All "owner" classes in Fig. 2, except *person*, represent legal entities.

Ownership of an item is often distributed among persons and legal entities. E.g., Jim and David together own a business *J&D Lightings*, and a business bank account. Also, the Eagle Corporation is a joint venture of Chrysler and Mitsubishi. We describe such a situation as *joint ownership*. It is legitimate for a person and a company to jointly own a property. The ownership need not be divided into equal portions. Stock holdings partition the ownership of a public company into various percentages. Jim, e.g., owns thousand shares of Chrysler.

In law, *property* means the rights which one has in anything subject to ownership, whether it be mobile or immobile, tangible or intangible, visible or invisible. Ownership is used synonymously with rights in property. Thus, a person is said to be the owner of a property if he has certain rights in it. The term ownership is often used to indicate that one has the "highest rights" [1] in a property, but it may be used even when one does not have all the rights; thus, we say that a person is an owner of a house even though he has rented it to a tenant who has exclusive rights to the use of the house during the term of the lease [1].

A property can be classified as *real*, *intellectual*, or *personal*. A *real property* refers to the rights that one has in land or things closely related to it. An *intellectual property* is the rights held on an idea (e.g., the design of an invention) or a creative work (such as a musical composition or a novel). For such property, the rights apply to a potentiality—no claim is made on any tangible item. Copyrights and patents are the ordinary forms of intellectual property. *Personal property* encompasses everything that is not a real or intellectual property.

As examples, Jim's business resides in a building which is his real property. The patent (number A908) for the Long-Life Bulb is his intellectual property. Bank account 369 and the car used by John are his personal property. In Fig. 1, the class *building* denotes a real property. *Patent* is an intellectual property. The remainder of the "property" classes represent personal properties. In Fig. 2, the only real property is *manufacturing_plant*. The rest are personal properties.

One characteristic of the ownership relationship itself centers around the existence of a legal document that verifies the owner's rights to a property. A copyright owner, e.g., is granted a legal certificate giving him exclusive rights to possess, make, publish, and sell copies of his intellectual production, or to authorize others to do so. In contrast, the owner of a household item does not have a legal document to support his ownership, but he has the right to use it as he pleases. We call ownership of the former kind *documented* and ownership of the latter kind *undocumented*. So, Jim's patent is documented, while his ownership of a toaster oven is undocumented.

In Fig. 1, the following ownerships (written as: owner class-property class) are among those that can be classified as documented: *bank-mortgage*, *person-building*, *person-small_business*, *person-bank_account*, *person-patent*, and *small_business-car*. The relationship between the classes *person* and *household_appliance* is undocumented. All ownerships in Fig. 2 are documented.

As a final distinction, some kinds of ownership are acquired by operation of law, while some others are not. We call ownership of the former kind *de jure* and ownership of the latter kind *de facto*.

3 Ownership as an OODB Semantic Relationship

3.1 Transactions and Inheritance

As noted above, the most crucial aspects of ownership are the constraints that it imposes on its related transactions such as sale and lease. Certain transactions can be applied to specific kinds of ownership, while others cannot. For example, in the case of exclusive ownership, the owner can sell his belonging without restriction (and thus the transaction "sale" can be applied freely), while for joint ownership an owner cannot sell the property without

Copyright © 1995 Software Engineering Press

the consent of the other owners (so the use of "sale" must be controlled). When a person has accepted an offer to sell his house, he cannot accept another offer, even though he is still the owner, until that time when the first offer becomes invalid. We call ownership of this kind *action-limited.* Similarly, when one has bought a stock option, the ownership of it may expire after a certain period of time if it is not exercised. In this case, we say that the ownership is *time-limited.* Likewise, when one has ownership of some property like a car or a house, it cannot be sold without its supporting documentation.

Let us consider some of these complexities of ownership transactions in the context of our example scenarios. If Jim wants to sell the business, he needs the consent of David, his partner. If David wants to buy half of the business's building from Jim, then he must have the consent of First Nat'l Trust which owns the mortgage. What would happen if David wanted to sell his half of the company to a new partner? Depending on the partnership agreement, he may need Jim's approval. With respect to their joint checking account, do both Jim and David need to sign every check together? Clearly that depends on the nature of the account. What about the sale of properties that are being used by others? For example, can Tom sell the house that he is renting to Jim? Yes, but the new owner would be unable to occupy the house until the lease expired. Is John allowed to sell his father's car? No, because even though he is using the car, he does not possess the proper ownership documentation required to sell it.

Aside from the transactions, the ownership relationship plays a vital role in more accurately modeling various application domains via its inheritance mechanism, which allows values of certain attributes to be propagated across it. For example, Jack's net worth (i.e., his "value") can be determined directly as the sum of the values of the portfolios that he possesses. Consider also that to calculate Chrysler's profits for 1994, the profits of Dodge, Plymouth, and Jeep must be added together. Furthermore, the profits of Dodge must take into account the profits of Eagle. In all these examples, a value propagation between properties and owners is required.

From the above we see that to properly support transactions and inheritance with respect to ownership, we need to explicitly model the different characteristics (which we call the *dimensions*) of the ownership relationship. Our investigation has revealed six important dimensions. In this section, we will first formally define the ownership relationship and its constituent dimensions. Thereafter, we will examine two of the dimensions, *exclusiveness* and *value propagation*, in some detail. We will then briefly describe the others.

3.2 Formal Definition of the Ownership Relationship

Let $E(C)$ denote the extension of a class C, i.e., the set of all its instances. The ownership relationship between a property class B and an owner class A (denoted $O_{B,A}$) is defined as the following septuple:

$$O_{B,A} =< \Omega_B^A, \lambda, \beta, \alpha, \chi, \delta, \nu >$$

where Ω_B^A is a relation from $E(B)$ to $E(A)$. The pair $(b, a) \in \Omega_B^A$ indicates that the instance b of class B is the property of (i.e., is owned by) the instance a of class A. We will ordinarily express this fact as $b\Omega_B^A a$. The remaining elements of the septuple are the six characteristic dimensions, whose names are *Legality*, *Documentation*, *Limitation*, *Exclusiveness*, *Dependency*, and *Value Propagation*, respectively. For each, we list its domain in the following:

$\lambda \in \{$*de jure*, *de facto*$\}$,

$\beta \in \{$*registration-docum'ted*, *transfer-docum'ted*, *undocum'ted*$\}$,

$\alpha \in \{$*action-limited*, *time-limited*, *action&time-limited*, *unlimited*$\}$,

$\chi \in \{$*exclusive*, *free-joint*, *percentage-joint*, *global-percentage-joint*$\}$,

$\delta \in \{$*owner-to-property*, *nil*$\}$,

$\nu \in \{$*up*, *down*, *upTrans*, *downTrans*, *up&down*, *nil*$\}$.

The values of both δ and ν may be *nil*, indicating that the particular characteristic (dependency or value propagation) is inapplicable. For lack of space, formal descriptions of only two dimensions will be given. The rest of the dimensions are described formally in [5, 14]. (In [14], we had only four dimensions and the notion of transactions is not connected to the dimensions of ownership.) For the following definitions, assume an ownership relationship $O_{B,A}$.

Definition 1: $\forall a \in E(A)$, let $P_{\Omega_B^A}(a) = \{b \mid b \in E(B) \wedge b\Omega_B^A a\}$. $P_{\Omega_B^A}(a)$ is called the *property set* of a with respect to $O_{B,A}$, i.e., the set of instances of B which are properties of a.

Definition 2: $\forall b \in E(B)$, let $N_{\Omega_B^A}(b) = \{a \mid a \in E(A) \wedge b\Omega_B^A a\}$. $N_{\Omega_B^A}(b)$ is called the *owner set* of b with respect to the ownership $O_{B,A}$, i.e., the set of instances of A of which b is a property.

Copyright © 1995 Software Engineering Press

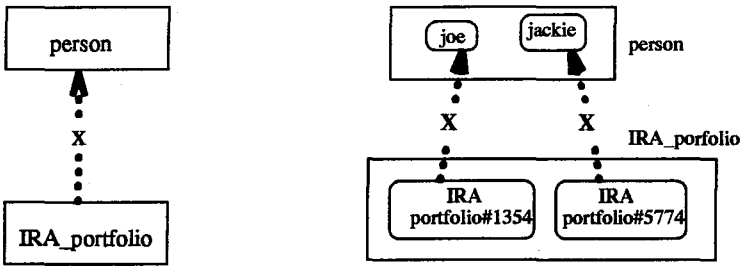


Figure 4: An example of exclusive ownership.

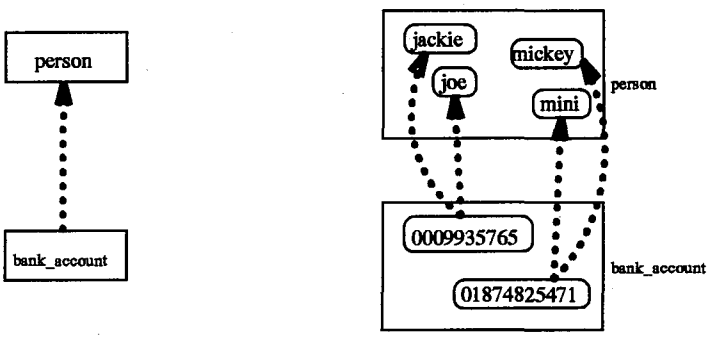


Figure 5: Jointly owned bank accounts.

3.3 Exclusiveness Dimension

Ownership can be classified as exclusive or joint. In other words, a property may be owned by one owner or jointly owned by several owners. The formal definition for the exclusive ownership relationship follows:

Definition 3: For the ownership relationship $O_{B,A}$, χ = *exclusive* implies that $\forall b \in E(B)$, $|N_{\Omega_B^A}(b)| \leq 1$. In other words, a property cannot have more than one owner.

To represent this graphically, we add an **X** to the dotted arrow to denote e**X**clusive (Fig. 4).

Those ownership relationships which are not exclusive are referred to as *joint*, in which case a property may be either jointly owned freely, i.e., there is no explicit partition of the rights of the joint owners in the property (e.g, a joint bank account is freely shared by a couple—we call this *free joint*), or jointly owned such that each owner takes a certain percentage of the ownership (e.g., husband and wife each own 50% of their house—we call this *percentage joint*). We call the case where all owners have the same percentage *equal joint*. Although the exclusiveness dimension has been included in some OODB models (e.g., SHOOD [11] and our part relationship model [7, 8]), percentage joint is unique to ownership. Percentage joint plays an important role in economic activities. A shareholder has the right to receive his percentage of dividends.

In our graphical notation, a plain dotted arrow indicates free joint (Fig. 5). Percentage joint and equal joint are denoted by labels of **P** and =, respectively (Fig. 6).

Definition 4: For the ownership relationship $O_{B,A}$, χ = *free-joint* implies that $\forall b \in E(B)$, $O_{B,A}$ does not impose any constraints on $|N_{\Omega_B^A}(b)|$. That is, each instance b may have any number of owners.

Definition 5a: For the ownership relationship $O_{B,A}$, χ = *percentage joint* implies that $\forall b \in E(B)$, each of its owners a has an associated number $p_{b,a}$ $(0 < p_{b,a} \leq 100)$ indicating a's percentage of ownership of b. The percentages $p_{b,a}$ associated with all the owners of b must total 100%.

Definition 5a defines the *percentage joint* ownership relationship when the property class has only one associated owner class. At times, the ownership of an object may be distributed among owners from different classes. This case is defined as follows.

Definition 5b: The ownership relationships O_{B,A_1}, $O_{B,A_2}, \ldots, O_{B,A_n}$ are *global percentage joint* if $\forall b \in E(B)$, each of its owners (regardless of their classes) own percentages of b totaling 100%.

Copyright © 1995 Software Engineering Press

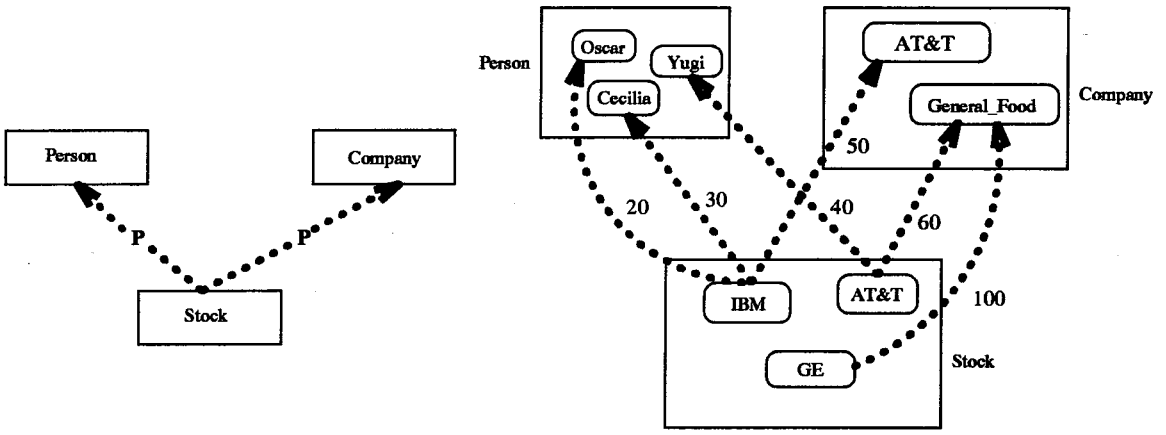


Figure 6: Stocks are owned (percentage) jointly by person and company.

To better understand Definition 5b, refer to Figure 6, where $O_{Stock,Person}$ and $O_{Stock,Company}$ are two *global percentage joint* relationships. For any instance of class *Stock*, the ownership is distributed among its owners such that each of them takes a certain percentage and the sum of the percentages is 100%. In Figure 6, the IBM stock owners are Oscar and Cecilia of class *Person*, and AT&T of class *Company*, with 20, 30, and 50 percent of the ownership, respectively.

3.4 Value Propagation Dimension

There are times when a certain feature of a property is naturally assimilated as a feature of its owner, or vice versa. E.g., the address of a person may be modeled as the address of his house rather than as an intrinsic attribute of the person. Likewise, the name that appears on the passport can be taken to be the name of its owner. In the former case, the value of *address*, rather than being duplicated, should be stored solely with the house and propagated upward on demand. *Address*, in this sense, is a derived attribute of person.

As another example, Jack's net worth can be determined directly from his portfolios. Specifically, Jack's net worth (denoted as his "value") is just the sum of the values of his various portfolios. As these fluctuate on a minute-to-minute basis, so too should Jack's worth. Therefore, it does not make sense to store this value statically. Rather, it should be derived dynamically from the appropriate sources on demand. The ownership relationship can automatically (i.e., without the need for manual programming) perform the necessary retrieval and computation.

Definition 6: Let $\pi_B : E(B) \rightarrow \tau$ be an attribute of B. The ownership relationship $O_{B,A}$ is said to be *invariant upward propagating* if it defines a property π_B on the class A such that the value of π_B for an instance $a \in E(A)$ is identically the value of π_B for that $b \in E(B)$ which is owned by a.

For example, if the property *address* is propagated from the class *house* to the class *person*, then the ownership relationship would define the property *address* on class *person* as follows:

$$address(a) = \begin{cases} address(b), & \text{if } \exists b \in P_{\Omega_B^A}(a), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Thus, the address of a person is identically that of the house that he or she owns. Invariant propagation in the other direction is defined analogously (see [5]).

Transformational upward value propagation is designed to take contributions for the value of the propagated (or inherited) attribute from any number of objects that are owned. The multiple values are transformed into a single value of the attribute's data type.

Definition 7: Let $\pi_B : E(B) \rightarrow \tau$ be an attribute of B. The ownership relationship $O_{B,A}$ is said to be *transformational upward propagating* if it defines a property π_B on the class A such that the value of π_B for an instance $a \in E(A)$ is derived by applying some transformation collectively to the values of π_B for all $b \in E(B)$ such that b is owned by a.

Here, instead of being identical to a value at a single "property" object, the value of the propagated attribute is derived through a transformation of values from many owned objects. For the example of the net worth of an individual, the propagated property *value* would have the following definition:

Copyright © 1995 Software Engineering Press

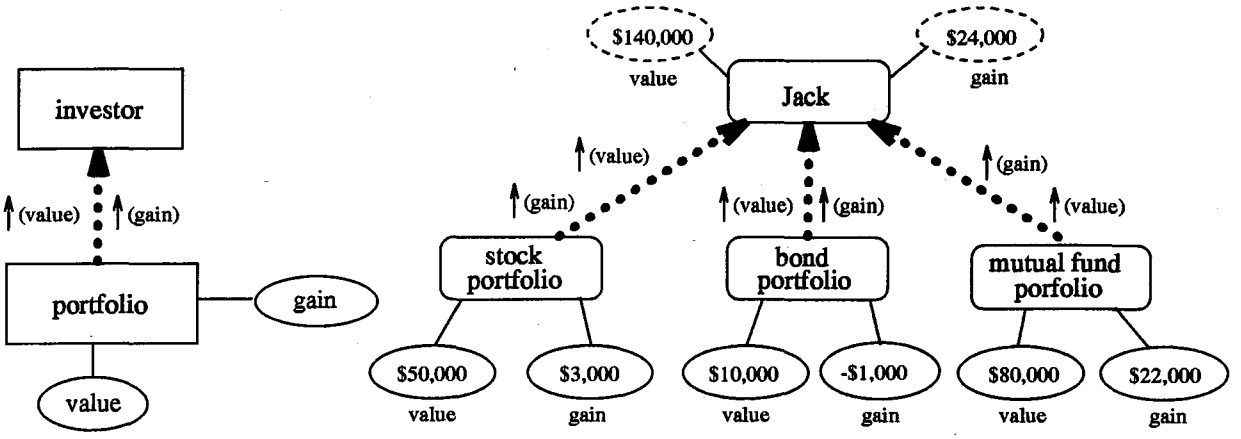


Figure 7: An example of value propagation.

$$value(a) = \begin{cases} \sum_{i=1}^{n} value(b_i), & \forall i,\ value(b_i) \text{ defined} \\ \text{undefined}, & \text{otherwise,} \end{cases}$$

where a is an investor and $b_1, b_2, \ldots, b_n$ are his portfolios. The above is shown graphically in Fig. 7, where we also show the specific example of Jack obtaining his net worth from his three portfolios. Another derived attribute, an investor's total gain (which is just the sum of the gains of the portfolios), is shown in the figure as well.

3.5 Additional Dimensions

Due to space limitations, we mention the issues of several other dimensions only briefly. For details, see [5, 14]. The dependency dimension regulates the semantics of deletion of owner class A or property class B. It defines when deletion of one should cause deletion of the other. Ownership can be either *documented*, or *undocumented*. Documented ownership always has a supporting legal document, while undocumented ownership does not.

Some kinds of ownership are acquired "by operation of law," i.e., through a formal legal procedure. We call such ownership *de jure*. Others are not, and are called *de facto*. These are the values for the legality dimension. Ownership is often used to indicate the "highest rights," but it may be used when one does not have all the rights. In other words, ownership may be limited in some aspects. For example, if the owner of a house has accepted an offer to sell that house to someone, then he cannot sell it to some other person, even though he is still the owner, unless the offer becomes invalid.

4 Conclusion

We have addressed the issue of representing ownership relationships in OODBs with a model that captures a variety of semantics. In particular, we have distinguished a number of aspects for the roles of the owner and property in such relationships. These aspects define notions like exclusive and joint owners. Formal definitions for various ownership relationships were presented. To complement these, we have presented graphical symbols for each of the ownership relationships which expand the OOdini graphical schema representation language for OODBs [9]. We have also investigated the interaction between the various ownership transactions and the ownership relationship's characteristic dimensions. We plan to integrate the ownership relationship that we have defined here into a commercial OODB system.

Acknowledgment

We thank Stewart Klein for reading an earlier version of this paper and for his many important remarks regarding the legal issues surrounding ownership.

References

[1] R. Anderson, W. Kumpf, and R. Kendrick. *Business Law — Principles and cases.* South-Western Publishing Co., Cincinnati, OH, 1971.

[2] R. J. Brachman. What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *Computer*, 16(10):30–36, Oct. 1983.

Copyright © 1995 Software Engineering Press

[3] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[4] C. J. Date. *An Introduction to Database Systems*, volume 1. Addison-Wesley Publishing Co., Inc., Reading, MA, fourth edition, 1986.

[5] J. Geller, M. Halper, O. Yang, and Y. Perl. Exploring the semantics of ownership relationships. In preparation, 1995.

[6] M. Halper. *A Comprehensive Part Model and Graphical Schema Representation for Object-Oriented Databases.* PhD thesis, CIS Department, New Jersey Institute of Technology, 1993.

[7] M. Halper, J. Geller, and Y. Perl. An OODB "part" relationship model. In *Proceedings of the First International Conference on Information and Knowledge Management*, pages 602–611, Baltimore, MD, 1992.

[8] M. Halper, J. Geller, and Y. Perl. Value propagation in object-oriented database part hierarchies. In *Proceedings of the 2nd Int'l Conference on Information and Knowledge Management*, pages 606–614. Washington, DC, 1993.

[9] M. Halper, J. Geller, Y. Perl, and E. J. Neuhold. A graphical schema representation for object-oriented databases. In R. Cooper, editor, *Interfaces to Database Systems*, pages 282–307. Springer-Verlag, London, 1993.

[10] W. Kim, E. Bertino, and J. Garza. Composite objects revisited. In *Proc. of the 1989 ACM SIGMOD International Conference on the Management of Data Portland, Oregon, appeared as SIGMOD RECORD*, pages 337–347, 1989.

[11] G. T. Nguyen and D. Rieu. Representing design objects. In *AI in Design'91.* Butterworth-Heinemann Ltd., 1991.

[12] A. Snyder. Encapsulation and inheritance in object-oriented programming languages. In *Proc. OOPSLA-86*, pages 38–45, 1986.

[13] W. A. Woods. What's in a link: Foundations for semantic networks. In D. G. Bobrow and A. M. Collins, editors, *Representation and Understanding*, pages 35–82. Academic Press, New York, NY, 1975.

[14] O. Yang, M. Halper, J. Geller, and Y. Perl. The OODB ownership relationship. In *Proceedings of the Int'l Conf. on Object Oriented Information Systems*, pages 389–403, London, England, 1994.

Copyright © 1995 Software Engineering Press

AN APPLICATION OF ARTIFICIAL INTELLIGENCE - SIMULATING THE BUSINESS ENVIRONMENT

By

Bryan Knower, Michael Gargano, and Frank Marchese.

I. INTRODUCTION

This study uses an artificial life paradigm . We developed a model of a localized business environment in an attempt to study long term business trends. The study investigated the relationship of non living dynamic systems to living ones. This was done using a model called CORPWORLD.

The model can be conveniently divided into three main component parts, namely: Artificial Life, Genetic Algorithms, and Corporate Behavior.

We will take up each of these topics in turn and discuss them in relation to each other. The main focus of this paper is the application of such a model to a simulated business environment.

II. BACKGROUND FOR THIS EXPERIMENT

The background for this experiment was a simulation of an artificial life system, called the *Kreecher Simulation*, inspired (and very loosely based) on the work of Thomas Ray.[11]

THE KREECHER SIMULATION

History:

Each individual (known as a Kreecher) competed with all the others based on simple interaction rules. The simulation accommodated three different genotypes, namely Passive, Coercive, and Destructive. All Kreechers were born with a specific vitality quotient. This indicator was depleted every cycle unless a particular individual was chosen via genetic selection for reproduction. Any genotype could reproduce. If selected, the vitality quotient of that individual was enhanced and the genotype was propagated via crossover with the genetic string of the Kreecher selected in the previous cycle. The simple rules for interaction were:

- Passive Kreechers actively avoided opponents, and had no direct effect on them.
- Coercive Kreechers transformed the genotypes of opponents to their own. The sphere of influence was limited, and meeting was random.
- Destructive Kreechers actively sought out opponents and destroyed them. The sphere of influence was limited but destructive types tended to migrate towards densely populated areas of the simulation grid.

RESULTS OF THE KREECHER SIMULATION

It appeared that populations of different types became dominant, ruled, and then decayed in a surprisingly cyclic manner. When a specific genotype became dominant, others tended to become suppressed, showing swings in population levels and vitality well below the stabilization level of the dominant genotype. As the dominant genotype decayed another would come up to take its place, and this process would go on indefinitely except in the case of a population explosion within the destructive genotype. Such an event led to mass destruction and eventually total extinction. The ordering of the dominance cycles showed no pattern, so that it was impossible to predict which genotype would become dominant next. Seeding the simulation with a passive genotype tended to produce the best results (i.e.. consistently long simulation runs with no chaotic disintegration), in the long term, while the other genotypes tended to produce random divergent behavior, probably based on their predatory behavior patterns.

An average run consisted of approximately five to ten thousand generations, within which the cyclic dominance behavior was readily evident. (A generation was one iteration of the simulation process). In the extreme long term, a pattern of chaotic behavior interspersed with long stretches of stable behavior became evident. In addition, long periods of cyclic behavior were interspersed with short periods of random chaotic behavior during which no dominant genotype could be identified.

Copyright © 1995 Software Engineering Press

III. ARTIFICIAL LIFE

Artificial Life is the study of man-made and non-living natural systems that exhibit behaviors characteristic of natural living systems. In order to simulate a living system, certain fundamental characteristics of living systems are assumed. The CORPWORLD model attempts to preserve the sense underlying these assumptions in the following propositions:

PROPERTIES OF LIFE

- Life is a pattern in space time rather than a specific material object.
- Self representation must be present.
- The organism should have a metabolism.
- Functional external interaction should be present.
- The organisms should eventually stabilize under most perturbations.
- Evolution must be present.

Evolution is taken to be present given the following characteristics;

HEREDITY- Offspring are similar to their parents.

VARIABILITY- Offspring are not identical to their parents or to each other. The two characteristics are complimentary. (i.e.. the copy process must produce uniformly similar offspring but not consistently identical ones over the life of the process.).

Both the *Kreecher* simulation and the *CORPWORLD* simulation satisfy these conditions.

IV. GENETIC ALGORITHMS

Genetic algorithms are selection procedures that work via evolutionary fitness and mutation. They are often used in the solution of optimization problems, and can be part of the machine learning process. In general, genetic algorithms incorporate a selection mechanism, coupled with a crossover mechanism and a mutation process.

The genetic algorithms used in the CORPWORLD simulation are simple. They use fitness functions to select the fittest individual and/or genotype at a specified point in time.

CROSSOVER

Genetic crossover is minimal in the model due to the fact that in a corporate environment, healthy corporations do not necessarily incorporate the behavior of successful corporations that have gone before them, especially in the area of corporate behavior. In *CORPWORLD*, therefore, crossover is restricted to the transference of genotype characteristics to offspring during reproduction, and transference of individual assets between peers during mergers or takeovers.

MUTATION

Mutation (in the model), transforms individuals from one genotype to another, and occurs during birth, reproduction, and even randomly on an infrequent basis. The four possibilities in the character matrix are: Conservative, Hyperactive, Expander and Virulent. Mutation ensures that variation exists in the execution of the various dynamic processes making up the life of the individual, and that the copying process in the genetic mechanism contains random imperfections, ensuring that it does not become a simple template function.

V. THE CORPWORLD MODEL

RATIONALE

The question of whether the principles of artificial life, (as they were demonstrated in the Kreecher Simulation), could be used to explore simulated interactions in a corporate environment, formed the basis of the experiment. It was decided to model four basic genotypes which would, (hopefully), incorporate variations in corporate behavior dynamics. Certain characteristics were immediately evident as dominant candidates for selection such as the raider mentality, acquisition of ailing companies by healthy ones, mergers in the face of increasing competition from large adversaries, traditionalism and conservative management, and aggressive entrepreneurship.

We decided to incorporate the raider mentality into a representation called the *Virulent* genotype; the expansion within a supportive market type corporation was represented by an *Expander* genotype; the traditional conservative management oriented type was represented by a *Conservative*

Copyright © 1995 Software Engineering Press

genotype, and the aggressive entrepreneurship type was represented by a Hyperactive genotype. Acquisitions and mergers were incorporated (in various degrees), into the common behavior patterns of all the aforementioned genotypes.

STRUCTURE

The simulation environment is a two dimensional plane on which individual corprobes exist and interact. Each corprobe is located at a specific point on the x,y plane. Its grid coordinates identify it uniquely. A corprobe's sphere of influence is the grid area within which the corprobe's presence has an effect on the market share of others, and within which its performance is affected by them. The further a competitor is from the center of a corprobe's sphere of influence, the lesser the effect it is likely to have on that corprobe's performance. A corprobe's health and vitality are measured by a rating known as the Financial Soundness Index (FSI). The higher the rating, the healthier the individual. Corprobes born via reproduction inherit the characteristics and vitality of their parents while those born via random birth inherit the characteristics and vitality of the initial seed population.

The following variable parameters can be used to control and limit the simulation.

- Percentage of each type as a fraction of one hundred percent.
- Size of initial seed population.
- Number of time steps to run or infinite horizon option.
- The option to re-seed the population if extinction approaches.
- Turn on (or off) a file trace with adjustable trace points.
- Turn on (or off) the graphical display with adjustable display points.

The system allows sampling of statistical data over a moving average of cycles which can be set to a desired level. Statistics are updated every cycle. Statistical data is also dumped to a file trace if this option has been turned on. In addition, the output trace prints out data points to a disk file or printer at fixed time intervals that can be varied. A summary of statistics for the entire run is written to a disk file in addition to the trace data. This summary is broken down by genotype and shows average values per cycle for FSI levels, production costs, marketing expenditure, and population levels.

GENOTYPES

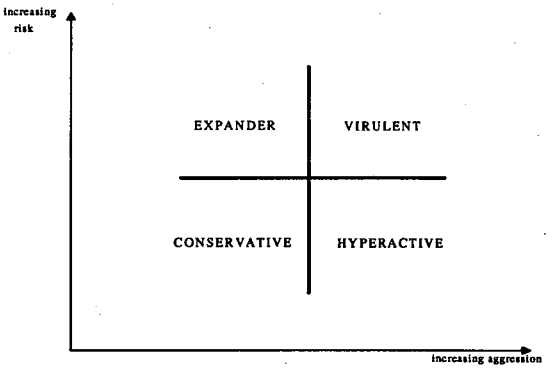


fig 1. Genotype Interrelationships

The four genotypes are differentiated by the strength (or weakness) of two characteristics called the *Risk Factor*, and the *Aggression Factor*. (Fig 1).

Expander

The Expander genotype uses aggressive marketing to expand market share. Expanders have a high risk factor coupled with a low aggression factor.

Conservative

The Conservative genotype keeps production well within demand. Both risk factor and aggression factor are low for this genotype.

Virulent

The Virulent genotype is destructive in nature and tends to keep production at high levels most of the time. Both risk and aggression factors are high in this genotype. Profits for the Virulent genotype are usually at the expense of other neighboring genotypes via decrease of market share.

Hyperactive

The Hyperactive genotype is a crossbreed between the Expander and the Conservative. It has a low risk factor coupled with a high aggression factor.

Copyright © 1995 Software Engineering Press

VARIABLE PARAMETERS

The Reproduction Threshold

The Reproduction Threshold controls the point any corprobe must reach before it can reproduce.

The Takeover Threshold

The Takeover Threshold is the point at which corprobes are deemed to be failing, and are tagged as candidates for take over. (Corprobes which are tagged and not taken over during the current cycle, are deemed bankrupt.).

The Merge Threshold

The Merge Threshold is a level that s aspiring corprobes need to reach in order to successfully take over a vulnerable corprobe. Increasing the Merge threshold makes it harder for a selected candidate to qualify for the take over process, and can lead to an increased failure rate and vice versa.

The Resource Multiplier

The Resource Multiplier is a factor in calculating consumer demand for the product of a corprobe. It is analogous to the initial number of customers per square area for a specified market, and is fixed for each run of the system. Current consumer demand for a particular corprobe's product is calculated using the Resource Multiplier, the number and location of other corprobes within the active corprobe's sphere of influence, and other factors. Current consumer demand is inversely related to the population density within the active corprobe's sphere of influence.

The Random Mutation Frequency

Random Mutation Frequency determines the probability that a corprobe will mutate during the current cycle. The actual probability of mutation is a function of the healthiness of the current genotype. Increasing the Random Mutation Frequency can lead to chaotic behavior.

The Random Birth Frequency

Random Birth Frequency determines the probability that a new corprobe will enter the system arbitrarily. Random Birth Frequency is intended to model new entrants to a market and also serves the purpose of rejuvenating the population.

The Selection Ratio

The Selection Ratio determines what percentage of the population (excluding the *best* corprobe), executes a standard cycle. Those corprobes not selected have their assets depleted via depreciation and become weaker relative to those that successfully execute a standard cycle.

THE LIFE CYCLE

Each cycle consists of a selection via fitness function of the healthiest individual in the population. The indicator used is the Financial Soundness Index (FSI) which tracks the vitality of an individual corprobe. The selected individual executes an enhanced cycle. A random sampling of the remaining population execute a standard cycle. All corprobes have their assets depreciated by a fixed percentage each cycle. This depletion models overhead costs, which are incurred regardless of production.

SELECTION VIA FITNESS

The method used in the selection process for the *CORPWORLD* model is a biased roulette wheel. In the case of selection of the fittest individual, the fitness function assigns weights based on the health of each individual. (Health is given by the Financial Soundness Index indicator (FSI) for each individual). In the case of selection of the fittest genotype, the fitness function assigns weights to each genotype based on the cumulative health index of the individuals belonging to that genotype.

THE ENHANCED AND STANDARD CYCLES

Cost of production is calculated for both the healthiest selected individual and those selected via random sampling of the population remainder. This cost is based on Current asset level, Current demand level, Characteristic type, and Market Factors. Consumer demand for active corprobe's product during the current cycle is given by

$$\textit{Consumer Demand} \leftarrow \oint(\textit{Gridsize}, \textit{ResourceMultiplier})$$

Once Consumer Demand has been determined, the projected production figures, (Estimated Sales), for each corprobe are calculated. Estimated Sales for the active corprobe is given by

Copyright © 1995 Software Engineering Press

Estimated Sales← ∮*(Cost, MarkUp)*
Cost ← ∮*(Marketing, Research&Development)*
Marketing ← ∮*(Current_Liquid_Assets,Risk, Aggression, Financial_Soundness_Index)*
Research&Development ← ∮*(Liquid_Assets, Risk, Aggression, Financial_Soundness_Index)*

Penalties are incurred for overproduction, ensuring that indiscriminate expansion does not occur. Also, random market fluctuations are simulated by additions to or subtraction from asset level recalculations for the active individual. New asset levels for the corprobe are calculated by

Liquid_Assets ← ∮*(Cost, Sales, Diversity, Quality, PreviousSales, Random_Market_Fluctuations)*
Quality ← ∮*(Liquid_Assets, Sales, Research&Development)*
Diversity ← ∮*(Marketing, Quality, Sales)*

Fixed assets are also subject to fluctuation due to market conditions, and are depreciated at a rate given by the variable parameter *Depreciation.* If fixed assets fall below a specific level they are renewed via transfers from liquid assets. The Financial Soundness Index is recalculated for the new asset levels and a revised sphere of influence is also established. These are given by

Financial_Soundness_Index ← ∮*(TotalAssets)*
GridSize ← ∮*(Financial_Soundness_Index)*

This process constitutes a standard cycle and is executed by both the healthiest selected individual and the random sampling of the population remainder based on the variable parameter *Select Ratio*. The healthiest selected individual also executes a special reward procedure which enhances its asset levels and hence its FSI and related sphere of influence. In addition, the selected individual is also made eligible for reproduction. Those corprobes that do not execute any form of the business cycle do not earn profit for the current cycle but incur depreciation and overhead costs.

THE TAKEOVER MECHANISM

Every cycle, those corprobes whose liquid asset levels fall below the *Takeover Threshold* are marked as vulnerable and open for acquisition. For each of these individuals, an acquirer is selected. The selection is based on the strength of the Financial Soundness Index, ensuring that only the healthiest individuals are selected as acquirers. A prospective acquirer must have adequate resources to complete the acquisition. Adequate resources are indicated by comparison with a threshold known as the *Merge Threshold.* The acquirer must minimally have liquid assets at the level of this threshold to acquire the target corprobe. A corprobe can acquire only a single target during a single cycle. This prevents a single individual from acquiring the assets of all failing corprobes for a single cycle. If a merger cannot go through because the selected acquirer has insufficient resources to complete the procedure the target corprobe goes bankrupt and is removed from the system during the next business cycle.

THE REPRODUCTIVE MECHANISM

During each life cycle, the corprobe that is selected as the healthiest individual (i.e.. the corprobe that executes an enhanced cycle), is also made eligible to reproduce. In order for reproduction to take place, the selected corprobe must have a liquid asset level of *Reproduction Threshold* or greater. If the selected corprobe, (the parent), meets this criteria, a fresh node is added to the active corprobe list and given the same behavior characteristics as the parent. Asset levels for the child corprobe are based on a percentage of the parent corprobe's asset levels and are therefore linked to the parent health index. Newly created corprobes can mutate directly after birth, ensuring that the copy process is not a template function but a variable process. Entry to the mutation procedure does not imply that mutation takes place. Parent corprobes undergo a reduction in asset levels to reflect the energy expended in the creation of a new individual. Corprobes can also be added via the Random Birth mechanism.

VI. RESULTS AND CONCLUSIONS

The simulation was run at varying Resource Multiplier levels ranging from one thousand through one million. Also, thresholds were varied to find out if there was significant behavior change at differing threshold levels. In all cases, a single parameter was varied while others were held constant, in order to measure the amount of change

Copyright © 1995 Software Engineering Press

in a controlled situation. Data was sampled via a moving average to smooth out short term anomalies as well as to observe long term trends. One hundred runs were done with each parameter set, and the results were averaged over the life of that specific run to smooth out random fluctuations.

We discovered that in the Resource Multiplier range of four thousand to twelve thousand, there was a steady, consistent increase in population growth, along with corresponding increases in FSI levels for all four genotypes. (See Appendix A: fig 1-2). Growth varied among the genotypes, but the overall picture showed a linear increase that was relatively consistent over all four types. Beyond this point, the population remained stable while FSI levels, which had increased dramatically towards the latter portion of the range, dropped to about a half of their peak value, and stabilized in a cyclic pattern. (See Appendix A: fig 3-4). The system stabilized at the transition point and levels remained consistent thereafter. Varying thresholds (such as the *Reproduction Threshold* and the *Takeover Threshold*) produced an amplitude shift, with the focus of the shift being concentrated in the population graph. (See Appendix A: fig 5-6). In all cases studied, varying the parameters did not essentially change the shape of the initial growth curve or the consequent transition point. Effects were mainly visible in the stabilization levels produced thereafter.

We think that the above results indicate that the model optimizes the population levels of the available genotypes for a specific resource level (i.e.. consumer demand) and a particular set of controls (the various thresholds and limits). The model was able to optimize over an extremely large range of values for the *Resource Multiplier*. The tested range ran from four thousand to one million. The statistical data indicated that large increases in the *Resource Multiplier* showed up as correspondingly large up-shifts in the graph as a whole.

The growth curve leading to the transition ledge is analogous to a youthful dynamic market which gradually becomes saturated over time. We think that the transition ledge indicates the point of market saturation. The dramatic plunge in FSI levels, from the transition point to stabilization levels is indicative of companies that invested too heavily in growth and expansion at the tail end of the market expansion curve, and were forced to pay a penalty when predicted demand did not meet expectations.

From the heights of the peaks in the FSI graph levels, it can be seen that the different genotypes show different growth patterns. The Conservative genotype shows the lowest peaks on average, and this indicates its nature. The Expander genotype generally has the highest average peaks, (indicative of its nature) and therefore shows more fluctuations in health over time. Overall, it would seem that the Conservative genotype is less susceptible to market fluctuations than the other three types, and that the Expander and Hyperactive genotypes have consistently larger market share. The model seems to show that the amount of risk undertaken is directly proportional to overall market share and vulnerability to market fluctuations.

Further experiments were conducted, in which the simulation was run with the absence of a particular genotype until a specific time frame was reached. As before, the system stabilized itself for the existing three genotypes. After stabilization, the absent genotype was introduced, and mutation and birth procedures were allowed to make use of this genotype. The system soon restabilized itself for the new configuration, albeit at a lower level from the one established earlier. (See Appendix A: fig 7-8). Introduction of a fresh genotype to a stable system resulted in the incorporation of the new genotype into the environment matrix at the cost of production decreases and health index depletion in the existing gene pool. In the context of the corporate environment, the model seems to imply that new entrepreneurs with characteristics widely differing from those already established, can find a niche in the marketplace, with the result that there is less market share for everybody, and a decreased likelihood of extended growth.

Changes in the threshold parameters tended to make the system less stable under extreme conditions. For example, increasing the Reproduction Threshold, the Merge Threshold, and the Takeover Threshold simultaneously produced a situation where it was harder for a corprobe to reproduce itself, and also harder to take over the

Copyright © 1995 Software Engineering Press

resources of a failing corprobe. Additionally, corprobes failed at a higher rate. In such a case, the rate of entropy for the system as a whole, tended to be higher than the rate of growth, leading eventually to population depletion, or market crash, (i.e.. failure of simulation due to inadequate number of corprobes, or inadequate number of healthy ones).

Raising the Reproduction Threshold made it more difficult for corprobes to reproduce, but those that did were correspondingly stronger than in a situation where the threshold was lower. Since reproducing corprobes supported their new offspring by direct asset transfer, the new corprobes, mostly of the same genotype as the parents, were correspondingly stronger financially and healthwise. The result was higher FSI levels for the genotype as a whole, with the FSI graph plots showing higher peaks corresponding to increased corporate stability.

VII. SUMMARY

In conclusion, it was observed that CORPWORLD does effectively model certain aspects of the corporate environment and the behavior of corporations within the limitations imposed by simulation's design. It can be viewed as a prototypical model that can explore behavior patterns in the corporate environment using such tools as Artificial Life and Genetic Algorithms. Further enhancements to the model, in order to make realistic projections regarding specific markets would involve:

- Maintaining a gene pool of successful phenotypes, which could be used to incorporate true genetic breeding. In this case, genetic evolution would be directed towards the best possible survivor in an environment.
- Changing the genotype over time to enable it to adapt more flexibly to its environment than at present. This would be a pre-requirement for genetic evolution. In particular, the *Risk Factor* and *Aggression Factor* could be made continuously variable, and active over a wider range.
- Changing the behavior patterns for each genotype to more aggressively reflect its nominal characteristics. In particular, the Virulent genotype could be designed to actively seek out targets within its sphere of influence, while the Expander genotype could be altered to factor in the healthiness of adversaries as well as its own, within its sphere of influence .
- Giving individual corprobes rudimentary intelligence, at least to the level of being able to make minor adjustments (such as adjusting sales and production to population levels), to its immediate environment based on its perception of that environment.
- Making resources (consumers) migratory, rather than fixed, as at present.
- Linking corprobes over relative distances on the grid, based on their familial links. This would include the ability to move assets back and forth between nodes of a larger distributed corporation.
- Making production a function of multiple variable markets rather than a single fixed one.
- Being able to consider many different types of environments based on economic variables.

CORPWORLD has the potential to become a valuable tool for experimentation and exploration of various corporate environments.

Copyright © 1995 Software Engineering Press

SELECTED BIBLIOGRAPHY

1. Ackley, D. H., and M. S. Littman. *Learning From Natural Selection in an Artificial Environment.* In Proceedings of the International Joint Conference on Neural Networks, Nol. I, Theory Track, Neural and Cognitive Science Track, (Washington, DC, Winter 1990) Hillsdale, NJ: Lawrence Erlbaum Associates, 1990.

2. Beer, Randall D., Hillel J. Chiel, and Leon S. Sterling. *An Artificial Insect.* Scientfic American, Vol. 79. September-October, 1991.

3. Cariani, P. *Emergence and Artificial Life.* The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems held 1990, in Los Alamos, New Mexico. Volume X. California: Addison-Wesley Publishing Company, Inc. 1992.

4. Dawkins, R. *The Evolution of Evoluability.* In *Artificial Life,* ed. C. Langton. The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems held 1987, in Los Alamos, New Mexico. Volume VI. California: Addison-Wesley Publishing Company, Inc. 1989.

5. Doolittle, W. F., and C. Sapienza. *Selfish Genes, the Phenotype Paradigm and Genome Evolution.* Nature 284 (1980): 601 - 603.

6. Farmer, J. D., and A. Belin. *Artificial Life: The Coming Evolution.* The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems held 1990, in Los Alamos, New Mexico. Volume X. California: Addison-Wesley Publishing Company, Inc. 1992.

7. Goldberg, David E. *Genetic Algorithms in Search, Optimization and Machine Learning.* New York: Addison-Wesley Publishing Company, Inc. 1989.

8. Gould, S. J., and N. Eldredge. *Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered.* Paleobiology 3 (1977): 115 - 151.

9. Knower, B., and F. Marchese. *The Kreecher Simulation: An Experiment in Simulated Artificial Life.* Unpublished manuscript (1992).

10. Langton, Christopher G., ed. *Artificial Life.* The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems held 1987, in Los Alamos, New Mexico. Volume VI. California: Addison-Wesley Publishing Company, Inc. 1989.

11. Langton, Christopher G., Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, eds. *Artificial Life II.* The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems held 1990, in Los Alamos, New Mexico. Volume X. California: Addison-Wesley Publishing Company, Inc. 1992.

12. Smith., J. Maynard. *Mathematical Ideas In Biology.* Cambridge: Cambridge University Press, 1968.

13. Volterra, V. *Variations and Fluctuations of the Numbers of Individuals in Animal Species Living Together.* In *Animal Ecology,* edited by R. N. Chapman, 409 - 448. New York: McGraw-Hill, 1976.

14. Wilson, E. D., and W. H. Bossert. *A Primer of Population Biology.* Stamford, CN: Sinauers, 1971.

Copyright © 1995 Software Engineering Press

AN APPLICATION OF ARTIFICIAL INTELLIGENCE - SIMULATING THE BUSINESS ENVIRONMENT

Appendix

By
Bryan Knower, Michael Gargano, and Frank Marchese.

Pace University
(212) 346-1336

Copyright © 1995 Software Engineering Press

APPENDIX A

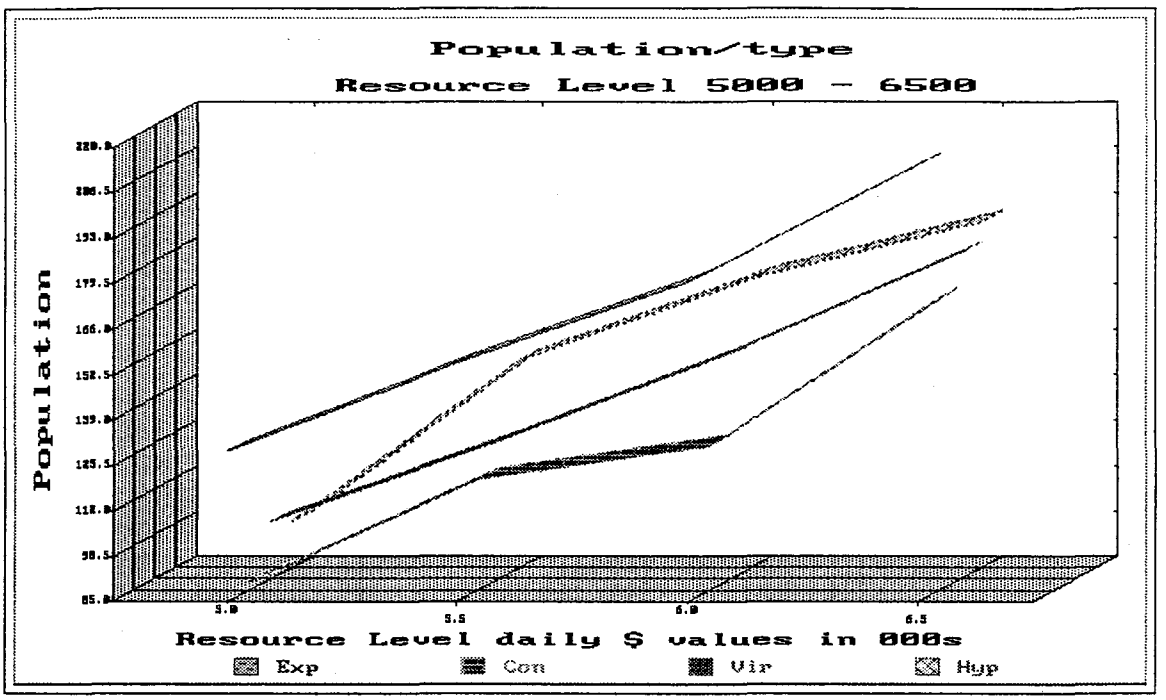


fig 1. Population levels with resource levels varying from 5000 to 6500.

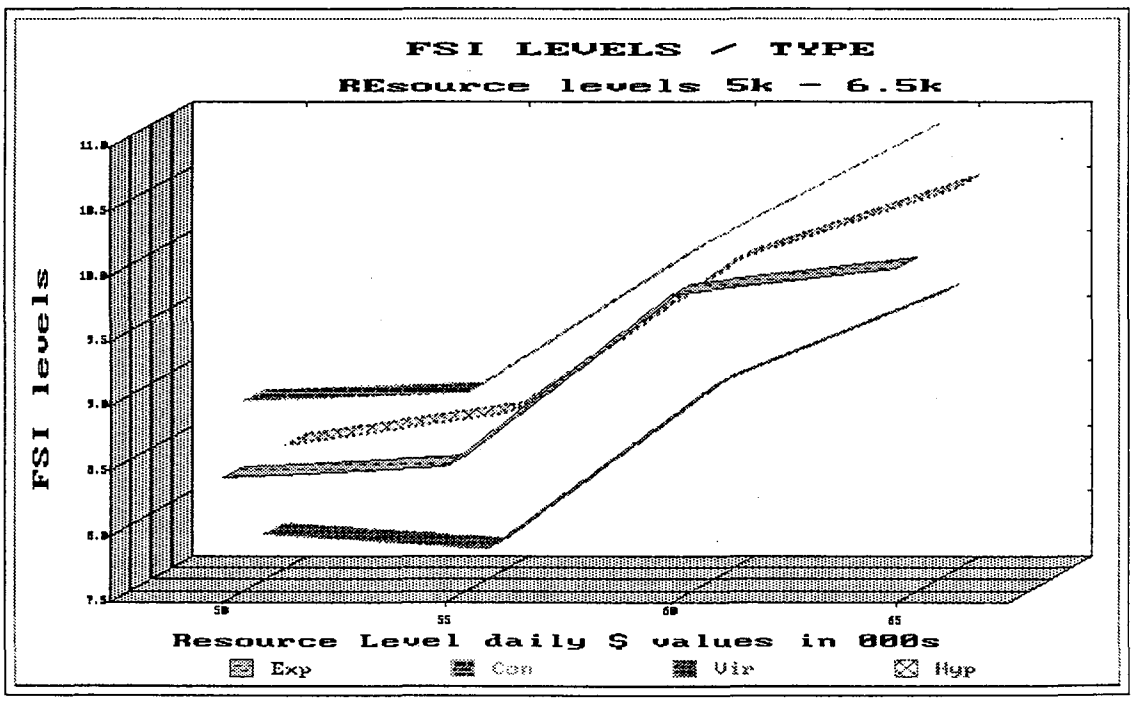


fig 2. FSI levels with resource levels varying from 5000 to 6500.

Copyright © 1995 Software Engineering Press

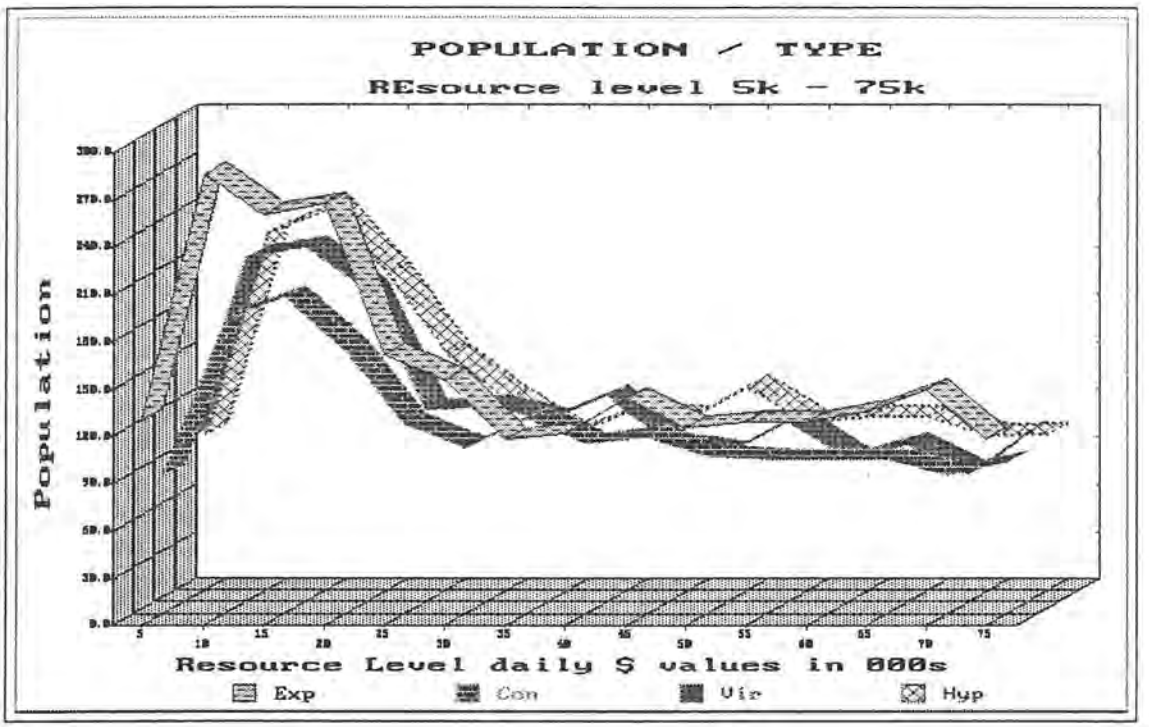


fig 3. Population levels with resource levels varying from 5000 to 75,000.

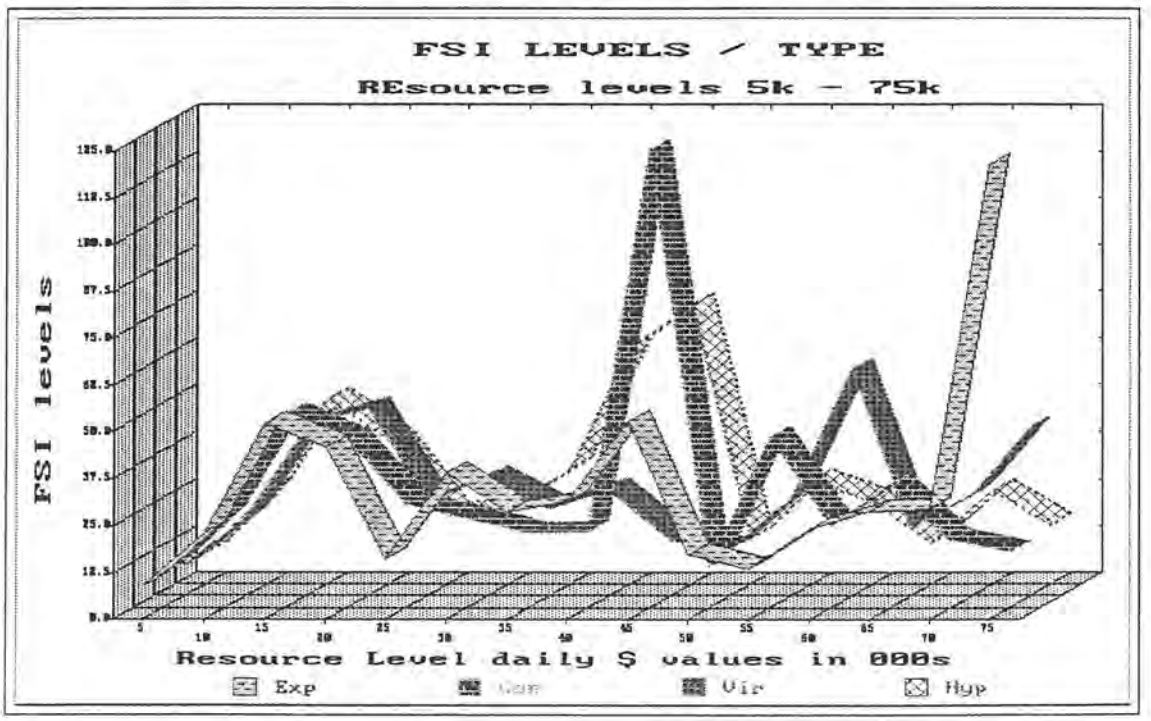


fig 4. FSI levels with resource levels varying from 5000 to 75,000.

Copyright © 1995 Software Engineering Press

APPENDIX A

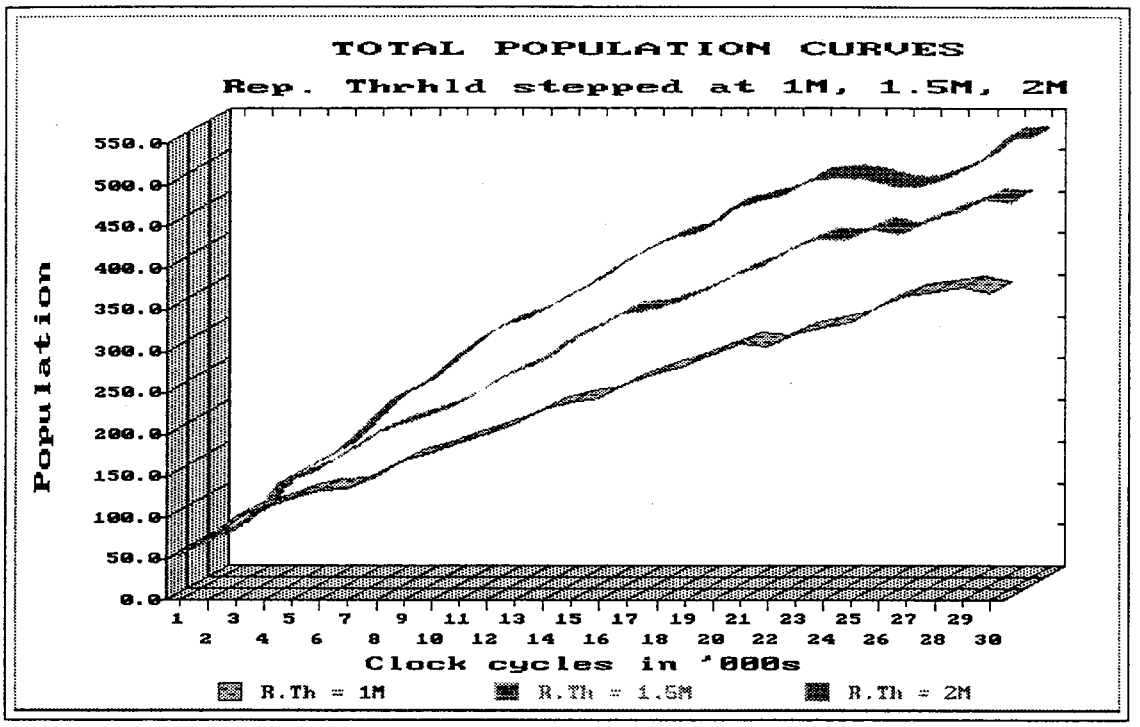


fig 5. Total population levels at three levels of the Reproduction Threshold. (1M, 1.5M, 2M).

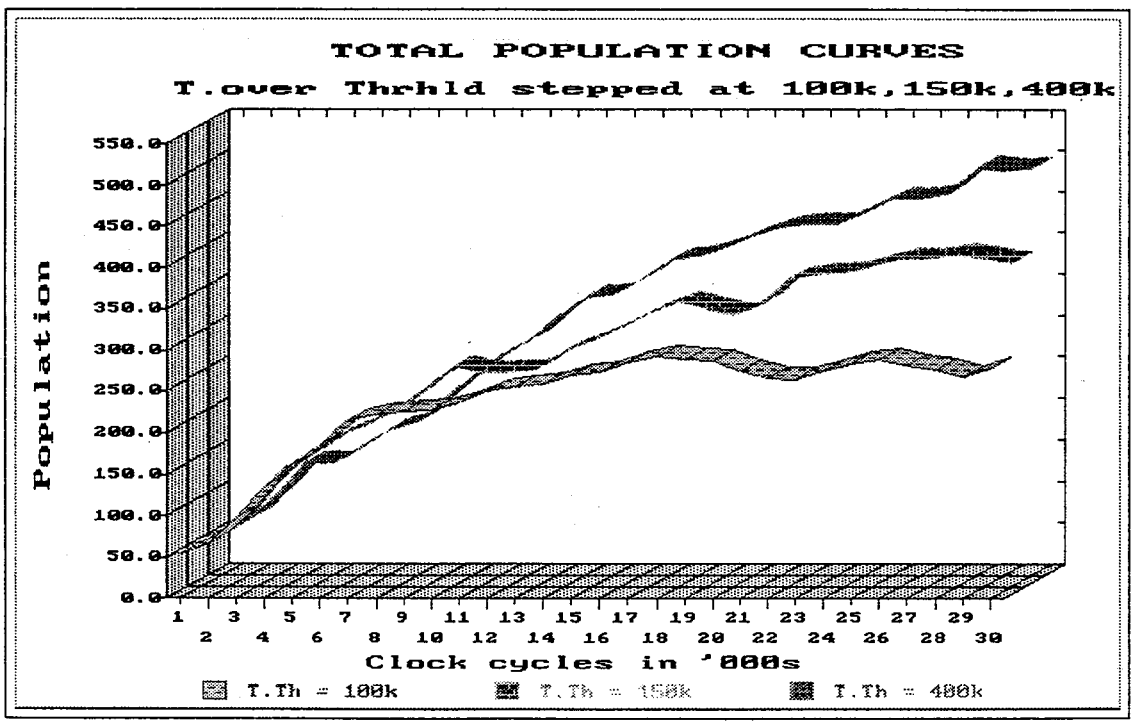


fig 6. Total population levels with Takeover Threshold stepped at 100k, 150k and 200k.

Copyright © 1995 Software Engineering Press

APPENDIX A

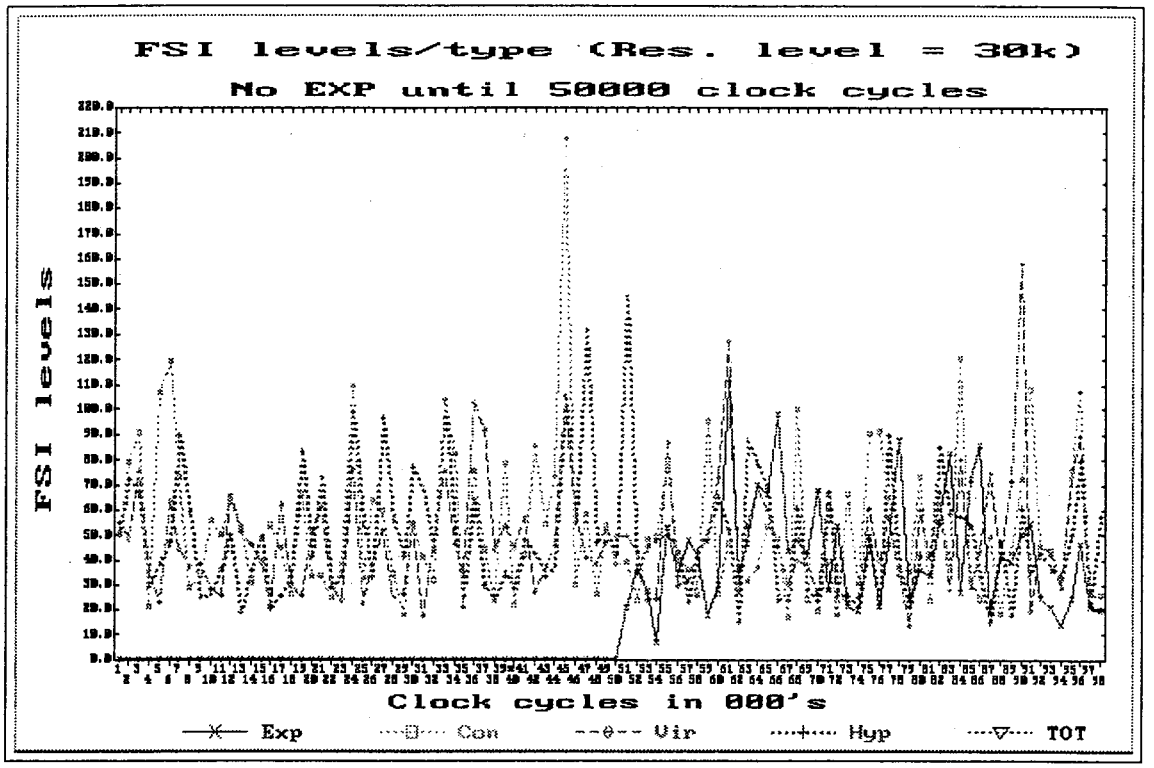


fig 7. FSI levels with one genotype with-held until 50,000 clock cycles.

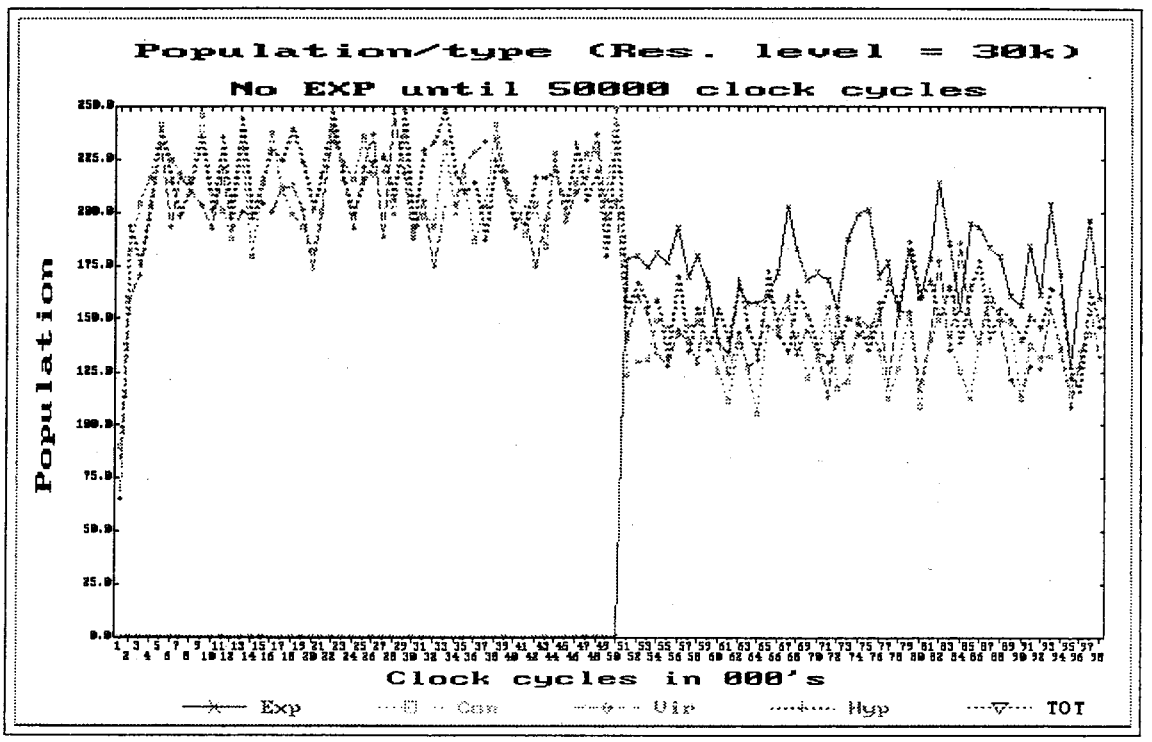


fig 8. Population levels with one genotype with-held until 50,000 clock cycles.

Copyright © 1995 Software Engineering Press

ALCOD: An IDSS for Stock Market Surveillance

Peter Goldschmidt
The University of Western Australia,
Department of Accounting and Finance
Nedlands, Western Australia 6907. Email:
pgold@ecel.uwa.edu.au[1]

Abstract

ALCOD is a cooperative multi-agent intelligent decision support system to assist stock market surveillance teams in the classification of alerted non-compliant events transacted on the exchange. ALCOD facilitates the review of the classifications. The system combines heuristic, approximate and causal reasoning and is centred around a relational database which is used as a control blackboard.

1. INTRODUCTION

1.1 Stock Market Surveillance and Decision Processes

Major international capital market providers currently use various surveillance techniques to ensure that the market participants are well informed and that illegal activities are detected, Keyes (1991). The stated mission of the Australian Stock Exchange (ASX) is "to provide, for the benefit of all participants, the most internationally competitive and fair market for financial securities and derivatives so as to enhance Australia's position as a regional financial centre."[2]

When the ASX was reorganised in 1989, the need for a formal market surveillance function was recognised. Prior to this date, equities market surveillance was divided among the ASX Companies and Membership Divisions, the federal National Companies and Securities Commission (NCSC) and the Corporate Affairs Commissions and the six separate states, the Australian Capital Territory, and the Companies Office of the Northern Territory. The surveillance Division was an outgrowth of the reorganisation.

The Surveillance Division's role is to monitor the market to ensure trading is fully informed. As a result, it may detect unusual patterns of market behaviour that might instance market manipulation, insider trading and similar practices. Once an unusual pattern is detected, if no adequate explanation is found and there appears to have been a breach of the ASX rules, it is reported to the Exchange's Companies Division (if a listed company is involved), ASX Membership Division (if a broker is involved), or the Derivatives Division (if derivatives are involved). Where there appears to have been a breach of the law, the matter is reported to the federal government body that administers the corporations law, namely the Australian Securities Commission (ASC).

When it was formed, the Surveillance Division commenced a program that combined computer-based decision support systems to analyse market events, with communications software, text retrieval and graphics. This program resulted in the Surveillance of Market Activity (SOMA) and related subsystems such as real-time monitoring of market events, news display, market replay,
and alerts' history. The SOMA system originated from the NYSE's STOCK WATCH system and has been modified for the Australian context.

The SOMA system is evolutionary, to keep pace with improvements in surveillance methods, changes in technology, and evolving market behaviour. The complete SOMA system is written in Cobol and C and runs on PCs supported by a Local Area Network.

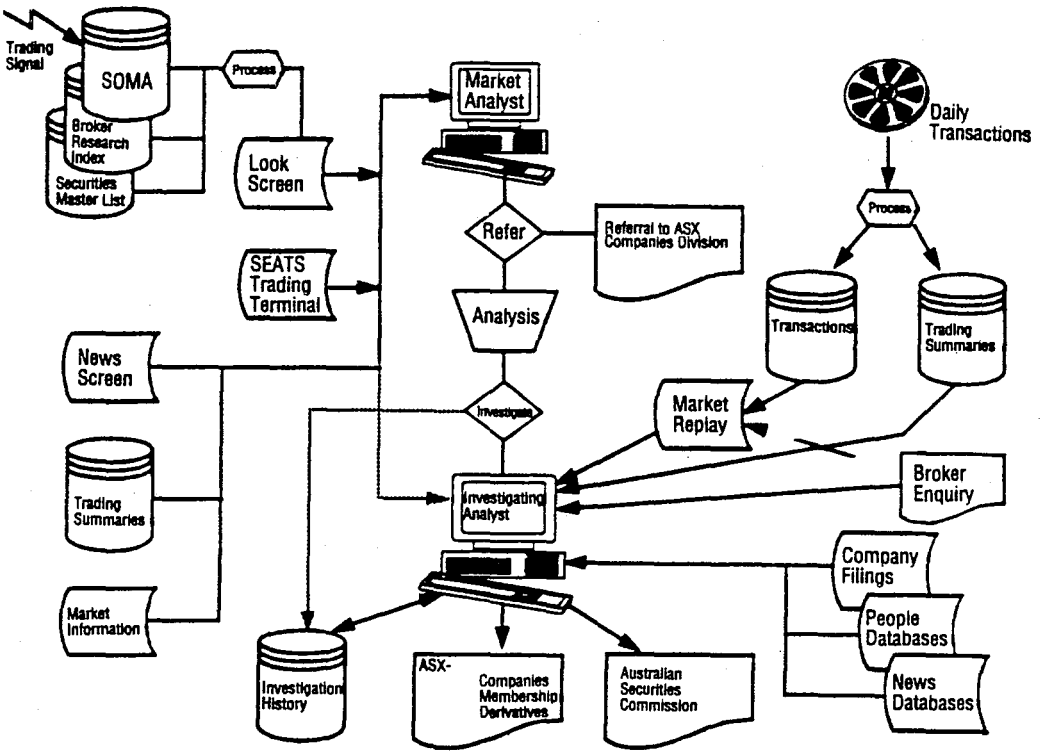
---

[1]We gratefully acknowledge the cooperation of the Surveillance Division of the Australian Stock Exchange.
The author is indebted to Professor Philip Brown , The Department of Accounting and Finance and Dr Paul Hadingham, The Department of Computer Science, The University of Western Australia for their advice and encouragement.

[2] ASX Annual Report (1994, p. 2).

Copyright © 1995 Software Engineering Press

Figure 1. How ASX Surveillance Works
Source: ASX Surveillance Division



Copyright © 1995 Software Engineering Press

The Surveillance Monitoring System at the ASX comprises two components: i) the automated system for the initial monitoring, and ii) the market analyst's review, comprising the subsequent decision processes. The automated system generates an "**alert**", the structured component, which a market analyst then scrutinises, the unstructured component. An alert is generated when there is an occurrence in the trading of a security that is outside pre-set values of any of a number of parameters. If the market analyst isolates unexplained unusual patterns of market behaviour and concludes an investigation is justified, the details are forwarded to the assistant manager of Surveillance who then determines if an investigation is required. If one is required, the report is sent to an investigating analyst who calls for brokers' records, conducts an analysis of all the available data, and decides whether a report of the activity to the relevant regulator is warranted (Berry and Yanco, 1990).

1.3. The Surveillance Operations Tasks

The SOMA model monitors between 50,000 and 80,000 trading entries per day and includes priorities that are determined by the type of alert generated. For example, when there is a volume type alert (e.g., when there is an extraordinarily large volume of trades), the number of days since the stock was previously traded will be a factor that contributes to the choice of the alert's priority. Surveillance operations con be broken down into a sequence of steps as follows.

- Once the automated system detects unusual market activity, it produces an alert. The type of alert depends on the nature of the unusual activity.
- SOMA separates the alerts into those that relate to one of the top 96 **liquid stocks**[3] and the rest (about another 1,000 stocks), which are classified as **illiquid stocks.** An analyst is responsible for each category. Liquid stocks are, by their nature, well researched by market participants. They make up a large part of the market index and can be seen as representing "the market". Alerts generated by events are sent to the one market analyst for scrutiny. Alerts for the illiquid stocks, which make up the bulk of the alerts, are sent to an assistant market analyst who is supervised by a market analyst. To reduce the number of rejected alerts, all alerts are now prioritised into two categories: viz., **primary and secondary alerts.** Primary alerts are those that are maintained for scrutiny and subsequently recorded in an **alerts history file** for referral. Secondary alerts are recorded in an alerts history file for reference, if necessary, at a later date.
- At the start of analysis, the assistant market analyst is presented with graphic user interface (GUI), displaying the current alerts generated by the primary system, keyed by ASX code.
- When the assistant market analyst selects a stock code for which there is an alert, she is presented with the report that details why the alert was generated.
- The assistant market analyst then adds comments to the GUI, using as a guide a set of questions that are documented in her manual. The answers to these questions determine whether the alert is to be rejected or accepted for further scrutiny. They relate, for example, to issues of price or volume movements compared to previous movements of that stock and to the movement in the relevant share index, or the presence of company announcements, brokers' newsletters, etc. Reference is made to charts of the past trading patterns of the stock and the index, the stock alert history, news services, and other information that may be of interest. Comments are added to the alert (via the GUI) on anything that the assistant market analyst believes may help the market analyst in reviewing the alert.
- The assistant market analyst compares the stock's price and volume movements with its history and with movements in the relevant index, in addition to the comparisons made by the automated system.
- The assistant market analyst (when possible) inputs **alert codes.** These codes flag the alert status as judged by the assistant. They may indicate, for example,

[3] Market capitalisation and value of trades are the metrics used to measure liquidity.

Copyright © 1995 Software Engineering Press

that the alert is "not for analysis", "watch" or "in line with sector".

- If, at this stage of the review, an alert identifies a significant change in the market for a stock that is unexplained by news and other market information, then the circumstances are referred to ASX Companies Division personnel. They contact the relevant company or broker(s) should they deem it necessary.
- The next step of surveillance is conducted by the market analyst, who also has access to a database containing points of interest relating to news items, brokers' recommendations, public newsletters and journal recommendations, online charts, the response from the ASX Companies Division personnel or the ASX Membership Division (if either is applicable), and alert history files. On occasion she may alter the alert codes entered by the assistant analyst.
- If there is an unexplained pattern of trading it is brought to the market analyst's attention for further inquiry. After conducting a detailed analysis including (for example, an analysis of who bought and who sold, and an evaluation of the value of the transactions of the trading concerned proportionate to the stock's capitalisation) a report is prepared for the surveillance assistant manager (the senior analyst) who then determines if an investigation is justified. If so, the report is forwarded to a surveillance investigator who conducts an inquiry into the matter.

2.1. THE ALCOD SYSTEM

The ALCOD System [ALCOD], is an Intelligent Decision Support System [IDSS], Hotzman (1989), Gottinger and Weiman (1992), and Marzano (1992), developed to assist the surveillance team in classifying the alerts generated by the SOMA system. ALCOD functions in a complex environment where the information used by the analysts is often highly context sensitive in high volume.

ALCOD uses Multi-Agent Technology, Bond and Gasser (1989), Bird (1993), Jin and Levitt (1993), Morrison (1993), O'Valle (1994), to reflect the multi-agent surveillance team, as well as Fuzzy Modelling, Evidence Combining, Zadeh (1978,1983,1986), Yager (1983,1987) and Torasso and Console (1987, 1989a, 1989b), and the concept of Blackboard Control Architecture, Hayes-Roth (1985), Mookerjee and Chaturvedi (1993) to model aspects of the surveillance team's decision making tasks.

ALCOD's primary function is to suggest an appropriate alert code, and to present the evidence supporting this suggested code. One of the results of ALCOD is the generation of an audit trail of the decisions made by the team. This trail is being used for fine-tuning the ALCOD system, and potentially can be used for reviewing the threshold levels in the SOMA system, for case-based reasoning to evaluate historical alerts, codes and evidence, O'Leary (1992), Garner and Chen, (1992) and Mott (1993), as the decision outcomes are retained, and as a training tool for novice analysts.

2.2. Initial Monitoring

As mentioned, the initial monitoring facilitates structured decision making and is done by the real-time procedural, SOMA system. The threshold levels of SOMA can be adjusted each trading session by altering the start-of-day threshold parameters. SOMA produces 40 different types of alert reports including the alert details. This output is converted to produce input for ALCOD.

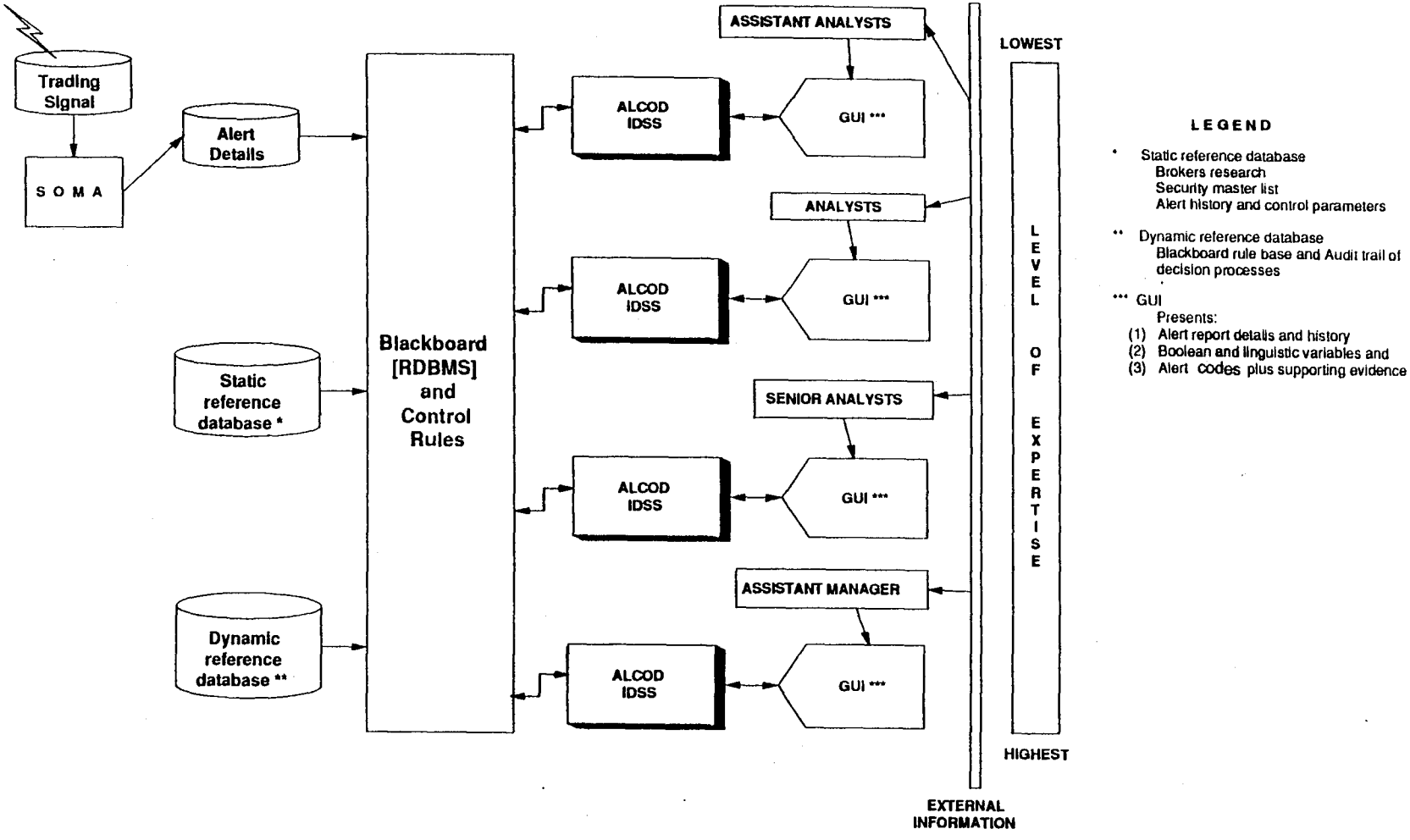
Typical alert reports include, for example: Sale Price versus Close of any of the last ***n*** Days, Sale Price versus Previous Close, Volume of ***n*** Days versus Past ***n*** Day Volume, and Today's Volume over ***n%*** of Issued Capital.

2.3. Secondary and Subsequent Monitoring and Decision Processes

The alerts generated by SOMA identify suspected breaches in compliance. However as the threshold levels have a high granularity, the alerts need to be evaluated to determine if they

Copyright © 1995 Software Engineering Press

FIGURE 2. The ALCOD System Schematic



Copyright © 1995 Software Engineering Press

are true positive alerts or false positives. Due to the temporal and context sensitive nature of the information required to evaluate each alert report in this complex environment, it is necessary for the analysts to use a large amount of information that may correspond to or conflict with the alert under review. The decision process involves the accumulation of evidence based on this information. This evidence is used to classify the alert. There are 24 different classifications of an alert, each requiring supporting evidence.

Typical classifications include, for example: Analysis Commenced, On-Market Buy Back Scheme, Investigation Commenced, Media Article, Company Announcement, Portfolio Adjustment, Watch, Watch and Ring Companies Department, In-line with Underlying Security, Substantial Shareholding Notice and Not for Analysis, for various reasons, such as insufficient volume, insufficient price, in-line with industry classification or in-line with market index.

Surveillance team members use internal and external information, as well as historical cases which include an alert code as well as its supporting evidence. Consequently, the analysts are typically faced with large amounts of temporal and context sensitive information, both directly and indirectly related to the event under scrutiny. This may lead to inconsistencies in the analysts' decision making and evidence production. Additionally analysts may present biases which manifest as inconsistencies over time and as inconsistencies between the analysts in the team.

Under this complex decision making environment, the decision makers can be assisted in their decision processing by computerised decision support systems [DSS], Gory and Scott-Morton (1971), Sprague (1980), Sprague and Watson (1986), Sviokla (1986), Silver (1990) and Todd and Benbassat (1990). The goal of this DSS is to assist the decision maker in matching events generated by an external agent, to known, or suspected, patterns of anomalous, agent behaviour. This goal can be seen as the terminal hypothesis supported by subgoals or node hypotheses. The objective of this DSS is to minimise the inconsistencies discussed above and to impose a normative framework for the combination of complementary and conflicting evidential information. This normative framework also assists the analyst in the management of the high volume of related external information.

2.3.1. ALCOD Model Dynamics

After initial alert generation (produced using quantitative methods with coarse-granule thresholds), the first task of the team is to determine if an alert is feasible. This and subsequent decisions are based on team members' expert determinations (supported by evidence) to classify the alert. The alert's classification is then reviewed by more senior members of the team, who use the alert code as a basis to review the evidence consistent with the assigned code. They may add, modify or disprove this evidence. The result of the review process is the retention or modification of the code plus evidence.

To assist in evaluating the information relevant to an alert, and to generate the evidence, ALCOD combines prototypical knowledge representation with heuristic, approximate and causal reasoning and evidence combination methodologies. This allows for combining heuristic and deep reasoning. The techniques used are discussed in detail in Torasso and Console (1989a).

Our basic model uses non-monotonic reasoning to apply defeasible logic on frame-based, Minsky (1975), knowledge structures. The fuzzy modelling and related evidence combining techniques are used in preference to the Bayesian framework measure of the strength of evidence, the likelihood ratio, because of the limitations with respect to the use of likelihood ratios as inputs which constrain the evidence aggregation. For details see Krishnamoorthy (1993). Additional justifications for the techniques used can be found in Torasso and Console (1989a), pp 3-26.

Copyright © 1995 Software Engineering Press

Figure 3. The frame Substantial Shareholder Notice, Short Term Price Movement (prototypical and control knowledge parts)

FRAME: Substantial Shareholder Notice [SSN]

TRIGGERS
Today's Price versus Previous Close
RM=1.0
NECESSARY FINDINGS
The level of importance of an SSN being lodged recently to the price move
<VI,0.75> <I,0.5> <SI,0.25> <NI,0.0>
RM=1.0
SECONDARY FINDINGS
The level of importance a particular broker being responsible for all or most of today's volume to the price move
<EI,1.0> <VI,0.75> <I,0.5> <SI,0.25> <NI,0.0>
RM=0.9

The level of importance of a particular broker having layers of bid and asks, and being noted in the history
<EI,1.0> <VI,0.75> <I,0.5> <SI,0.25> <NI,0.0>
RM=0.05

The level of importance of the company having been queried in the last few months about the top 20 shareholder because of an increase in the volume of trading AND this volume attributed to changes in the top 20 shareholders, to the price move
<EI,1.0> <VI,0.75> <I,0.5> <SI,0.25> <NI,0.0>
RM=0.5

The level of importance to the price move of enquiries by ASX to the company about an announcement (including periodic reports)
<EI,1.0> <VI,0.75> <I,0.5> <SI,0.25> <NI,0.0>
RM=0.05

VALIDATION RULES
confirm if SSN (timing, level of importance)
in context -
RM =1.0
ALTERNATE HYPOTHESES
confirm if alert previously classified as SSN (timing, level of importance)
in context -
RM =1.0
DEFAULT SPECIALISATION
The level of importance of an SSN being lodged recently to the price move
<EI,1.0>
RM=1.0

Copyright © 1995 Software Engineering Press

Knowledge representation [KR]

On the heuristic level, which is hierarchically organised, the KR is in the form of frames which contain structural knowledge slots, prototypal knowledge slots and control knowledge slots. The higher heuristic levels are coarse classification hypotheses whereas the lower levels are more specific. These high and lower levels are connected by a specialisation relationship connecting the frames that represent general classification hypotheses to frames representing more specific hypotheses.

Structural knowledge is formed by these generalisation and specialisation slots which describe a frame's position in the system. The specialisation slots contain the names of the classification hypotheses that are more specific than the one under consideration. The generalisation slot indicates the hypotheses generalising the one under consideration.

Prototypical knowledge is divided into subdivisions containing primary or necessary conditions, and secondary conditions. The primary conditions need to be satisfied in order for an hypothesis to be confirmed. The secondary conditions are a variation on Cravetto et. al. (1985) sufficient conditions, which establish the hypotheses under consideration when a sufficient condition is met. As these sufficient conditions are often difficult to define, they have been replaced by secondary conditions, which allow for the completion of primary conditions by describing more specific conditions that are not strictly necessary.

Control knowledge may have up to five slots: Triggers, Validation Rules, Associated Hypotheses, Alternate Hypotheses and Default Specialisation.

- Triggers are rules that need to be satisfied by actual data before a frame is instantiated.
- The slot Validation Rules contain production rules that involve specific data to confirm or reject the instantiation of a frame.
- The slot Associated Hypotheses contains a list of those hypotheses that are associated with the current one and the slot Alternate Hypotheses contains a list of those hypotheses that are alternative to the current one.
- The Default Specialisation slot contains a list of specialisations that are more common when a generic classification can be established. When a frame corresponding to a generic classification can be established by actual data and no specialisation frame can be triggered, this default specialisation can be used.

Relevance measures and evidence formulation

In order to weigh the importance of data or conditions on data, we use the concept of relevance measures [RM]. The RM metrics associated with each atomic condition in a complex condition lies on the [0,1] interval. An RM of 1 has the maximum relevance and conversely the minimum RM is 0. RMs are elicited from the experts as part of the initial knowledge acquisition, and are built into the system.

When data is input by the user in the form of a boolean variable, i.e. True or False, linguistic variables [LV] are associated with each positive response. There are five LVs ranging from extremely important, [EI] to not important, [NI]. These LVs are then combined with the RM to produce an evidence measure for each element concerned. Elements associated with the same classification goal are then combined to form an 'evidence chunk'. The control knowledge, heuristic knowledge, causal and approximate reasoning are used to evaluate the global degree of evidence for the hypotheses under consideration. The end result is the required alert code, plus its supporting evidence.

The Multi-Agent Component

ALCOD is centred around a relational database which contains the output from SOMA, and the reference databases. The RDB is also used as a blackboard on which results and various controlling parameters are recorded, and by

Copyright © 1995 Software Engineering Press

Figure 4. Influence of Relevance Measures on Evidence Evaluation

Adapted from: Torasso and Console (1989a, 1989b).

The combination of the relevance measures and the level of importance, takes the form of a connective,

$$f_{CONNECTIVE}: D_{EV} \times D_{RM} \rightarrow D_{EV}$$

where D_{EV} represents the domain of evidence, i.e. the level of importance, and D_{RM} is the domain of relevance measures. The 'corrected' evidence obtained by applying $f_{CONNECTIVE}$ to the pair <observed_evidence, RM) of a fact is the same as that used in the fuzzy evaluation of the complex condition. The general requirements for the connective function are shown in (1) and (2) for the two cases of AND and OR connectives respectively.

(1)
$f_{AND}(e,0)=1$
$f_{AND}(e,1)=e$
$f_{AND}(0,m)=1-m$
$f_{AND}(1,m)=1$
$f_{AND}(e,m) \geq e$ if $0<e<1$ and $0<m<1$

(2)
$f_{OR}(e,0)=0$
$f_{OR}(e,1)=e$
$f_{OR}(0,m)=0$
$f_{OR}(1,m)=m$
$f_{OR}(e,m) \leq e$ if $0<e<1$ and $0<m<1$
$f_{AND}(e,m)=m*e+(1-m)$

The first operand of both f_{AND} and f_{OR} represents the observed evidence of an atomic condition and the second one the relevance measure of the finding occurring in the atomic condition.

To constrain the form of the formulae which define f_{AND} and f_{OR} we use the functions (3) which satisfy (1) and (2). We are also experimenting with function (4).

(3)
$f_{AND}(e,m)= m*e + (1-m)$
$f_{OR}(e,m)=m*e$

(4)
$f_{AND}(e,m)=e^2(m^2-m)+e(2m-m^2)+(1-m)$
$f_{OR}(e,m)=e^2(m-m^2)+m^2 e$

Once the revised evidence degree has been evaluated for the elementary conditions, we use (5), (6) and (7) to combine the elementary evidence to form a chunk of evidence.

(5) $e(\text{AND } (T_1\ T_2\ \ldots\ T_n)) = \alpha+\beta * (\beta-\alpha)$

where $\alpha = \prod_{j=1}^{n} e(T_j)$ and $\beta = \min_{j=1}^{n} e(T_j)$

(6) $e(\text{NOT T}) = 1 - e(\text{T})$

(7) $e(\text{OR } (T_1\ T_2\ \ldots\ T_n)) = e(\text{NOT (AND ((NOT } T_1) \text{ (NOT } T_2) \ldots \text{ (NOT } T_n))))$

Copyright © 1995 Software Engineering Press

Figure 5. A Heuristic Approach to Evidence Combination

Adapted from: Torasso and Console (1989a, 1989b).

To combine the evidence degrees of the related knowledge chunks to form the global evidence degree of the terminal hypothesis, the starting point is the Bernoulli formula (8).

(8) $e_1 +_f e_2 = e_1+(1-e_1)*e_2$

However, as this considers the degrees of evidence as the same with no single value having a privileged position we proceed from (9) to formulate (10) to distinguish between the primary and secondary findings.

(9) $e_1 +_\lambda e_2 = e_1+(1-e_1)*e_2*g(e_1,\lambda)$

where the parameter λ represents the degree of privilege.

$g(e_1,1)=1$ (perfect privilege)

$g(e_1,0)=e_1$ (unfair privilege)

$g(e_1,\lambda)=X$ with $e_1 < X < 1$ when $0 < \lambda < 1$

(10) $g(e_1,\lambda) = e_1 + (1 - e_1) * \lambda$

By varying λ we can obtain an evidence combination scheme which assigns more or less predominance to the evidence obtained.

Finally to evaluate the overall global degree of evidence of the hypothesis under consideration we use (11) to combine the degrees of evidence of the separate knowledge chunks obtained from the primary evidence e(P), the secondary evidence e(S), the exclusion rule e(ER) and the comfirmation rule e(CR)

(11) $e(H) = ((e(P) +_u e(S)) O_f [1 - e(ER)] +_u e(CR))$.

Copyright © 1995 Software Engineering Press

which team members' communicate their results. The homogeneous and heterogeneous components constitute the multi-agent architecture; that is, i) the Analysts, ii) the KBSs, iii) the Blackboard, and iv) the Databases.

Once an alert code plus supporting evidence has been assigned to an alerted stock, the information is passed to the next team member for review. Modifications to this result can be performed either by manually editing this output or by using the hedging strategies. The results of each team member are added to the decision audit trail.

EXTENSIONS AND FUTURE RESEARCH

Further research is currently being conducted or planned including; increasing the range of LVs to include three points within each LV, temporal case-based reasoning to evaluate historical classifications and evidence, natural language processing of company announcements, the incorporation of a cost function to evaluate the risk of further analysis, and issues related to extending the analysis-team-wide modelling to include modelling the investigation task .

CONCLUSION

ALCOD is a proof of concept prototype currently under review by the Surveillance Division of ASX. The system is built using Smart Elements/Nexpert and MSAccess. It is PC based.

Preliminary real-time testing was conducted over a five day period on alert report, 'sale price compared to the previous close'. Classifications generated by a senior analyst were compared with those generated by ALCOD. The results showed that in 38 cases out of 50 reviewed, 76%, the ALCOD result agreed with the analyst's determination. In two cases, 4%, the analyst modified her classification to that recommended by ALCOD. In the remaining 10 cases, 20%, ALCOD ranked the analyst's recommendation as second with a difference in the degree of evidence between first and second being less than 11%. Further system development and fine-tuning is currently being conducted.

The ALCOD system is discussed in detail in Goldschmidt (forthcoming), and is used as an illustrative example of the concept of compliance monitoring for anomaly detection in a complex environment.

Bibliography

Berry, J. and G. Yanco, (1990). "Attacking Abuses in the Australian Stock Market ". Journal of the Securities Institute of Australia.

Bird, S., (1993). "Toward a taxonomy of multi-agent systems". International Journal of Man-Machine Studies, (39), 689-704.

Bond, A.H., and L. Gassers, (eds.), (1988). Readings in Distributed Artificial Intelligence, Morgan Kaufman, San Mateo, CA.

Cravetto, C., Lesmo, L., Molino, G., and P. Torasso, (1985). "LITO2: A Frame Based Expert System for Medical Diagnosis in Hepatology". In (I. De Lotto, M. Stefanelli eds.): Artificial Intelligence in Medicine, North-Holland, 107-119.

Garner, B., and F. Chen, (1992). "Case-Based Interaction for Fraud Detection in EDP". In Proceedings of International Conference on Information Processing & Systems, Beijing, China.

Goldschmidt, P.S., (forthcoming). "Compliance Monitoring for Anomaly Detection in a Complex Environment using Multi-Agent Technology". PhD Dissertation, The University of Western Australia.

Gory, G.A. and M.S. Scott-Morton, (1971). "A Framework for Management Information Systems". Sloan Management Review, 13(1), 55-70.

Gottinger, H.W., and P. Weiman, (1992). "Intelligent Decision Support Systems". Decision Support Systems, North-Holland, (8), 317-332.

Copyright © 1995 Software Engineering Press

Hayes-Roth, B., (1985) "A Blackboard Architecture for Control". Artificial Intelligence 26(3), 251-321.

Hotzman, S., (1989). Intelligent Decision Systems, Readings MA: Addison-Wesley.

Jin, Y. and R.E. Levitt, (1993). "i-AGENTS: Modelling Organisational Problem Solving in Multi-Agent Teams". Intelligent Systems in Accounting, Finance and Management. (2), 247-270.

Keyes, J., (1991). "Intelligent Financial Intelligence", Financial & Accounting Systems, (7), 12-15.

Krishnamoorthy, G., (1993). "Discussion Of Aggregation of Evidence in Auditing: A Likelihood Perspective". Auditing: Practice and Theory, (12), Supplement, 161-166.

Marzano, G., (1992). "IDSSs Opportunities and Problems: Steps to Development of an IDSS". AI & Society, (6), 115-139.

Mookerjee, V., and A.R.Chaturvedi, (1993). "A Blackboard Control Architecture for Model Selection and Sequencing", European Jrnl. of Information Systems, 2(1), 3-14.

Morrison, J., (1993). "Team Memory: Information Support for Business Teams". HICSS-26 (Proceedings of the 26th Annual Hawaii Int. Conf. on Systems Sciences), Wailea, Hawaii, 4, 122-131.

Mott, S., (1993). "Case Based Reasoning: market applications and fit with other technologies". Expert Systems with Applications, 6(1), 97-104.

O'Leary, D., (1992). "Case-Based Reasoning and Multi-agent Systems for Accounting Regulation Systems with Extensions". Journal of Intelligent Systems in Accounting, Finance and Management, 1(1), 41-52.

O'Valle, A., (1994). "Using Heterogeneous Multi-Agent Technology for Expert Systems Design and Development". Proceedings of the 1994 World Conference on Expert Systems.

Silver, M., (1990). "Decision Support Systems: Directed and Nondirected Change". Journal of Information Research, 1(1), 47-70.

Sprague, R., (1980). "A Framework for the Development of Decision Support Systems", MIS Quarterly, 4(4), 1-26.

Sprague, R. H., and H.J. Watson (Eds.), (1986). Decision Support Systems: Putting Theory into Practice. Prentice-Hall, Engelwood Cliffs, NJ.

Todd, P. and I. Benbassat, (1991). "An Experimental Investigation of the impact of Computer Based Decision Aids on Decision Making". Journal of Information Research, 2(2), 87-115.

Torasso, P. and L. Console, (1987). "Causal Reasoning in Diagnostic Expert Systems", in Proc. V. Int. Conf. on Applications of Artificial Intelligence, Orlando, 598-605.

Torasso, P. and L. Console, (1989a). Diagnostic Problem Solving, North Oxford Academic, London.

Torasso, P. and L. Console, (1989b). "Approximate Reasoning and Prototypical Knowledge", Int. Jrnl. of Approximate Reasoning, 3(2), 157-177.

Yager, R., (1983). "Quantified Propositions in a Linguistic Logic". International Journal of Man-Machine Studies, (19), 195-227.

Yager, R., (1987). "Using Approximate Reasoning to present Default Knowledge". Artificial Intelligence, (31), 99-112.

Zadeh, L., (1978). "Fuzzy Sets as a Basis for a Theory of Possibility". Fuzzy Sets and Systems, (1), 3-28.

Zadeh, L., (1983). "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems". Fuzzy Sets and Systems, (11), 199-227.

TRADEMARKS

DOS, Windows, and MSAccess are trademarks of Microsoft Corp,
Smart Elements / Nexpert is a trademark of Neuron Data.

Copyright © 1995 Software Engineering Press

NEW DEVELOPMENTS IN SOFTWARE PATENT PROTECTION[1]

Mikhail Lotvin
Pennie & Edmonds
1155 Avenue of the Americas
New York, New York 10036-2711

Richard Nemes
Pace University
One Pace Plaza
New York, New York 10038-1502

In the first proceedings of this conference, in which we presented a paper dealing with patent protection of financial software, we discussed those patents that afford exclusive rights to developers of software systems and their underlying financial applications. Since then, many patents for financial software applications have issued, and the importance of software patents in general has been growing with increased recognition of their value.

Recently, software patents have been given extensive coverage in the press. To cite an example, Stac Electronics won a $120 million jury verdict against Microsoft for infringement of data compression patents. (Subsequently, however, the case was settled with Microsoft paying millions to Stac.) This case once again underscores the great potential value of software patents. Similarly, controversy surrounding a patent issued to Compton's New Media, who claimed exclusive rights to searches in multimedia text/graphics environments, attracted a great deal of attention. In response to pressure from industry, the Patent Office, on its own motion, which is unusual, initiated proceedings to reexamine this patent. In addition, largely in response to the computer industry's reaction to this patent, the Patent Office held public hearings concerning software patents.

In our previously mentioned paper, we stressed that although software is entitled to patent protection (including software for financial applications), the courts continue to struggle with the problem of distinguishing patentable subject matter from so-called "mathematical algorithms" and from abstract ideas, both of which are not patent protectable.

In the early 1980's, the United States Court of Appeals for the Federal Circuit was created as a unified appellate jurisdiction for patent-related cases. Although the Supreme Court can, at its discretion, review the opinions of that court, it rarely does so, and therefore the decisions of the Federal Circuit constitute the most important body of patent law. Between 1982 and 1989 the Federal Circuit issued only two decisions relating to patentability of software, namely In Re Iwasha and In Re Grahams , both discussed in our previous paper. In 1992 it came out with another decision, Arithmia Research Technologies, Inc. v. Corazonix Corporation, and in 1994 it issued a whopping four

[1]This paper represents the opinions of the authors and not necessarily those of the organizations with which they are associated.

Copyright © 1995 Software Engineering Press

decisions related to the patentability of software.

Particularly relevant subject matter was discussed in the 1994 case In Re Schrader, where the Court addressed the patentability of a method for competitively bidding on auctioned items, and in In Re Trovato, which analyzed patent claims concerning a purely graph-theoretic algorithm. In In Re Warmerdam the patentability of software for a "bubble hierarchy" was discussed, and in In Re Alappat the Court considered the patentability of an anti-aliasing oscilloscope system. Although these cases do not display much consistency and clearly articulated guidance is nowhere to be found, a discussion of these decisions ought to provide some guidance to software developers.

Before analyzing these new decisions and discussing their relevance to the protection of AI systems for financial and business applications a few words regarding patents in general are in order. A patent typically consists of two main parts: a detailed *disclosure* of the invention, which is a detailed description of its embodiment intended to be understood by persons "skilled in the art," and the *claims*, which define the scope of intellectual property that is subject to patent protection. While the disclosure is detailed and specific, the claims, on the other hand, are written as broadly as possible, subject of course to originality and clarity requirements. Many factors are considered in determining whether a given invention is entitled to a patent, the most important being novelty and non-obviousness. While these factors are common to all areas of technology, software has an additional dimension, namely, the fundamental question regarding the patentability of software in the first place. Even though software is a main-stream branch of engineering, in the 1970's and early 80's it was confused with mathematics and abstract ideas, both of which are not entitled to patent protection. (Mathematics, in particular, is considered a phenomenon of nature that is always in the public domain.) This cast a shadow on the entire area of software technology.

Turning now to the cases cited above, in In Re Warmerdam the Federal Circuit considered the claims to an algorithm and data structure for implementing collision detection using "a bubble hierarchy." The invention is an improved technique for detecting the perimeter of an object using a hierarchically structured logical "bubble," which contains lower-level bubbles of smaller diameter, each of which encloses a portion of the object. The algorithm successively intercepts the object with bubbles moving down the hierarchy, and if at the lowest level at least one bubble is "burst", the program determines that a collision would occur. Those claims addressed to a method of generating a data structure were rejected. The court held that the method constituted a mathematical procedure and simply manipulated "abstract ideas." A claim directed to a "data structure" generated in accordance with the method of the invented algorithm was deemed to mean "nothing more than another way of describing the manipulation of ideas." (One of the reasons that the court was confused regarding the term "data structure" was that the patent application did not define the actual bubble data structure.) Interestingly, the court found that the claim directed to a machine having a memory that contains data representing a bubble hierarchy using this method was in fact statutory. The court in footnote 6 stated "our predecessor court has recognized that the storage of data in a memory physically alters the memory and thus in some sense gives rise to a new memory."

In analyzing this decision, it appears that the failure to articulate an area of application was an important factor. Had the claims referred to an application, particularly to physical actions for which the algorithm was obviously intended, the chances of success at the Patent Office would have been greater. Clearly, this court was confused by the term "data structure" and did not make the

Copyright © 1995 Software Engineering Press

connection to physical memory (i.e RAM) arranged in a certain way, which is how computer scientists view data structures.

In In Re Trovato the Federal Circuit addressed patentability of a heuristic for determining the shortest path in a graph. Its broadest claims relate to (a) storing a configuration space data structure, and (b) propagating cost waves in the configuration space data structure to fill the configuration space data structure with cost values according to a space variant metric. The claims were found nonstatutory by the Federal Circuit. The court concluded that "Trovato does not claim to have invented a new kind of computer which the recited mathematical algorithm controls. Nor do they claim that the recited mathematical algorithm has been combined with a new memory controlling a computer known to the art. Putting Trovato's claims in their most favorable light, the most they provide is a systemic way in which to compute a number representing the shortest part. A new way to calculate a number cannot be recognized as statutory subject matter."

Regarding "data structures," in In Re Lowry (1994), the Federal Circuit determined that a claim directed to data structure *is* proper. It characterized the data structure as a physical entity:

> Data models define permissible data structure -- organizational structures imposed upon the data used by the application program -- compatible with particular data processing systems. Data structures are the physical implementation of a data model's organization of the data. Data structures are often shared by more than one application program.

Thus, several decisions issued at approximately the same time came out with opposing conclusions regarding the meaning of "data structure." This means that, in general, the notion of data organization can be patented; care should be taken, however, to explain in a patent application that the reference is to the physical organization of a computer's memory. In inventions related to AI, where the innovation is entirely in software, care should be taken to explain in great detail that what takes place is not abstract, but rather that a machine under software control is directed to perform a new function. Although the disclosure is supposed to be addressed to "a person skilled in the art," for pragmatic reasons it is advisable to provide specifications that are at a much lower level than is customarily done when addressing programmers. This means that though all educated programmers know what a data structure is, the patent office and courts might not, so a prudent patent application writer will provide careful explanation of such notions using physical terms.

In In re Schrader the Federal Circuit rejected as unpatentable a method of conducting auctions. The claims essentially covered a way of determining an optimal combination of bids. The Federal Circuit held that the claims of this patent were directed to "a mathematical algorithm," which is not under the current laws subject to patent protection. The claim was drafted in purely logical terms and did not discuss any physical implementations whatsoever. Had this claim included a description, even at the most elementary level, showing how computers are used in auction environments, the outcome might have been different.

In Arithmi Research Technology v. Corazonix Corporation the Federal Circuit found that a method for analyzing electrocardiographic signals in order to determine certain characteristics of heart function was patentable. Though the process of analyzing electrocardiographic signals was implemented entirely in software, nevertheless, the court found that the claimed steps of " 'converting', 'implying', 'determining' and

Copyright © 1995 Software Engineering Press

'comparing' are physical process steps that transform one physical, electrical signal into another." One of the reasons that the court was persuaded that this software claim was patentable is that it was directed to a well articulated practical application. In addition, this court took a more liberal approach to software claims in general, stating that "computers came to be generally recognized as devices capable of performing or implementing process steps, or serving as components of an apparatus, without negating the patentability of the process or the apparatus."

In In Re Alappat the Federal Circuit addressed patentability of an anti-aliasing algorithm used for high quality oscilloscope displays. In essence, the invention amounts to a mathematical transformation of input voltage data into display output data that represents screen pixel intensities. Although the invented procedure could easily be implemented as a program, the patent application was artfully drafted in that its disclosure showed circuitry (AND, OR, NAND gates; ROM, ALU, etc.) instead of program code. The claims, however, were written strictly along software lines, making no explicit mention of circuit components. The court was persuaded that the disclosed circuit elements corresponded to the elements of the software claim and found that the claim related to statutory subject matter. The court was also persuaded that this claim clearly recited its intended application, i.e., a rasterizer for creating a smooth wave form. In the conclusion of the majority opinion, the court stated that "a computer operating pursuant to software may represent patentable subject matter. . . In any case, a computer, like a rasterizer, is apparatus not mathematics."

Several minority opinions were entered in this case that disputed the patentability of the invention. One minority opinion drew the analogy between a general-purpose computer and a player piano, with software being the counterpart to the piano roll, which merely provides the guiding pattern underlying the piano's operation. The argument said that a melody encoded on a piano roll cannot be patented, and similarly software, intrinsically, cannot be patented. Clearly, the majority opinion was not influenced by this argument, and to the contrary, it confirmed that a computer programmed in accordance with software is considered a new machine, provided that it is new in view of other requirements of patent law, such as novelty and non-obviousness.

This decision illustrates again that describing the underlying physical structure of an invention in detail helps in achieving patentability. Whether the hardware is described as a low-level circuit as in Allapart or as high-level computer architecture, such descriptions increase the chance that the patent examiner or the court will not object to software claims. This is especially true in view of the other cases that were rejected because the courts did not see sufficient physical basis.

What about AI systems? Since these tend to be high-level applications, frequently comprising, essentially, only number crunching, it is imperative that one describe an underlying hardware platform, even to a minimal degree (e.g. simple block diagram), when applying for a patent. It is also important, as indicated earlier, to provide an area of application and to connect the abstract (data structures, mathematical formulas, etc.) to physical structures. By carefully structuring patent applications it is possible to obtain practical patent protection for AI inventions.

Copyright © 1995 Software Engineering Press

Paper Session: Advanced Forecasting Techniques

Chair: Kar Yan Tam, Hong Kong University of Science and Technology

MULTICRITERIA ASSOCIATIVE MEMORY APPROACH FOR NONLINEAR SYSTEM PARAMETER ESTIMATION

Hany S. Gobreial, Ph.D.

The Aerospace Corporation
P.O. Box 92957
Los Angeles, CA 90009–2957
(310) 336–6420

SHORT ABSTRACT

A multicriteria associative memory (MAM) approach is proposed to encode associations between a set of stimulus vectors constituting a matrix $\boldsymbol{S}$ *and a corresponding set of response vectors constituting a matrix* $\boldsymbol{R}$*. Kohonen (1988) originally suggested that a "memory" matrix* $\boldsymbol{M}$ *be constructed such that* $\boldsymbol{MS}$ *is as close as possible to* $\boldsymbol{R}$ *in the least–squares sense. The solution to this problem is* $\hat{\boldsymbol{M}} = \boldsymbol{RS}^{+}$*, where* $\boldsymbol{S}^{+}$ *is the Moore–Penrose generalized inverse of* $\boldsymbol{S}$*. Unfortunately, though, when calculated this way,* $\hat{\boldsymbol{M}}$ *has low robustness to noise. The MAM approach suggested here increases robustness by separately assessing costs of the deviation of* $\boldsymbol{MS}$ *from* $\boldsymbol{R}$ *and for the sizes of elements in* $\hat{\boldsymbol{M}}$*.*

The major application of this MAM approach is to gain a priori information for nonlinear dynamical systems, such as the Solow–Swan Economic Growth Model, in order to initialize an extended Kalman Filter (EKF) for parameter and state estimation. Kalman Filtering is a method for system identification, which is identifying the properties of an unknown system from observations, to predict the system's future behavior. For the Solow–Swan nonlinear economic model, the MAM approach converts observations of system output into usable initial estimates for EKF to estimate the state vector. Discrepancy values generated from the MAM approach provide initial state estimates as well as the initial state covariance for the EKF. This, in turn, leads to a highly reliable estimate of system parameters.

Keywords: Artificial neural networks, linear associative memory, multicriteria optimization, nonlinear estimation, extended Kalman Filtering, Solow–Swan growth model

To be presented at the Third International Conference on Artificial Intelligence Applications on Wall Street June 7–9, 1995.

INTRODUCTION

Parameter estimation problems for nonlinear systems are typically formulated as nonlinear optimization problems in the presence of noise requiring an iterative approach to its solution. Several on–line and off–line methods have been developed in the past, most of them based on different variants of the Gauss–Newton method;Gray [4]. Probabilistic formulations, such as maximum likelihood, bayesian multi–model technique and the Extended Kalman Filter; Mehra [7] have also been developed. Yet the basic difficulty when facing identification of a large number of parameters from input–output data, is caused by the fact that the cost function's surface may have multiple minima and therefore, convergence to the "correct" parameters is iteratively possible only when one starts from a close enough initial guess of the parameters to be identified. For quadratically convergent iterative methods, such as Gauss–Newton methods, the components of the initial estimate for the parameter vector often have to be within ten to twenty percent of their true values; Cuyt [2]. Possible difficulties that might result from inaccurate estimates are slow rates of convergence, weakly observable set of observations as well as oscillatory behavior. Computational difficulties due to ill–conditioning may also arise.

An alternative method of performing parameter estimation is to use an associative memory approach. Here, rather than iteratively solving the inverse problem

Copyright © 1995 Software Engineering Press

for a given input–response pair as is commonly done in on–line identification, the forward problem is repeatedly solved for various input–response pairs and a memory matrix is adduced which optimally associates the inputs with the outputs. Thus when the estimation scheme is later presented with a given input (output), it can then estimate the output (input) that corresponds to it. Different learning schemes that develop the memory matrix that maps the input to the output have been devised.

THE ASSOCIATIVE MEMORY APPROACH

The goal of the approach of associative memories or associative mappings is to get initial estimates for nonlinear systems. For each parameter vector $\mathbf{r}_i$ in a selected training set $\{\mathbf{r}_1,\mathbf{r}_2,...,\mathbf{r}_q\}$, the system equations determine a vector $\mathbf{s}_i$ of system outputs. A "memory" matrix $\hat{\mathbf{M}}$ is then constructed to optimally (in the sense of least squares) associate each "stimulus" vector $\mathbf{s}_i$ with its corresponding "response" vector $\mathbf{r}_i$, in the sense of least squares. Specifically $\hat{\mathbf{M}}=\mathbf{RS}^{+}$, where $\mathbf{R}$ denotes the matrix $[\mathbf{r}_1,\mathbf{r}_2,...,\mathbf{r}_q]$ of training parameter vectors and $\mathbf{S}^{+}$ denotes the Moore–Penrose generalized inverse of the training output matrix $\mathbf{S} = [\mathbf{s}_1,\mathbf{s}_2,...\mathbf{s}_q]$. Given an observed system output vector $\mathbf{s}^*$, an estimate $\hat{\mathbf{r}}$ for the system parameter vector is obtained by setting $\hat{\mathbf{r}} = \mathbf{M}\mathbf{s}^*$.

Surprisingly accurate parameter estimates were obtained for an illustrative nonlinear image processing problem when the observation vectors $\mathbf{s}^*$ are noise free;Kalaba [6]. However, instability problems were encountered when memory matrices constructed from noise–free training vectors were subsequently used to recover parameter estimates from observation vectors corrupted with noise.

This paper describes development and application of a multicriteria associative memory (MAM) procedure to initialize a successive approximation scheme, such as an Extended Kalman Filter, which will lead to rapid convergence for nonlinear estimation problems. The MAM approach directly and systematically guard against the ill–conditioning of the memory matrix. The memory matrix $\mathbf{M} = \mathbf{RS}^{+}$ corresponds to the extreme point of the MAM frontier where the association cost C_a is minimized with no regard for the size cost C_z, i.e., with no separate consideration given to the size of the elements of $\mathbf{M}$. By moving away from this extreme point along the frontier, bias(training association error) is increased in return for a decrease in the variance of the resulting parameter estimates. The MAM frontier is thus analogous to the coefficient frontier obtained using ridge trace procedures in regression analysis;Vinod[11]. A key difference, however, is that the purpose of the MAM frontier is to provide one or more useful initial parameter estimates for some given nonlinear estimation problem. Consequently, the usual criticism of ridge regression procedures–that they do not necessarily lead to parameter estimators with optimal statistical properties– is not applicable to MAM;Judge[5].

State of the art spacecraft position determination systems utilize a Kalman Filter ;Chui[1] to blend position measurement data with angular outputs. Since the Kalman Filter presumes linear measurement equations, it should be appropriately initialized to avoid processing new measurements when errors are "large". Otherwise, the convergence of its error estimates may not be consistent with its error covariance predictions. In fact, its estimates may converge to bias errors which can be quite large with respect to the predicted bounds embedded in the covariance matrix. In this paper, modifications to the conventional Kalman Filter formulation which may improve its convergence properties are investigated. That is, reducing initial errors using the MAM approach in conjunction with a

Copyright © 1995 Software Engineering Press

nonlinear estimation technique, such as the Extended Kalman Filter can provide significant improvement in convergence time and precision.

MULTICRITERIA ASSOCIATIVE MEMORY SOLUTION METHODS

Let the solution to a basic differential equation be denoted by

$$(1) \quad k(t) = H(t;\beta,\gamma,\delta), \quad t \geq 0$$

A classical approach to the parameter estimation problem for the model represented by the differential equation above would be to pose it as a nonlinear least squares problem in which the sum of squared deviations

$$(2) \quad \sum_{j=1}^{m} [k^*(t_j) - H(t_j;\beta,\gamma,\delta)]^2$$

is to be minimized with respect to (β,γ,δ). For each different trajectory of observations $(k^*(t_1),k^*(t_2),...k^*(t_m))$, a different sum (2) would have to be minimized, typically by means of a successive approximation scheme. A major drawback of many successive approximation schemes, however, is the need to have a good initial estimate for the true parameter vector.

Once the possibility of imprecise calculations and observations is recognized, keeping the elements of the memory matrix small becomes an important criterion (in addition to the basic criterion of obtaining good training case associations). Two basic costs are associated with each possible memory matrix **M**: an association cost $C_a(\mathbf{M})$ measuring the extent to which **M** fails to associate each training output vector $\mathbf{s}_i$ with its corresponding training parameter vector $\mathbf{r}_i$ and a size cost $C_z(\mathbf{M})$ measuring the extent to which the elements of **M** differ from zero.

On the basis of both tractability and general intuitive appeal, the costs $C_a(\mathbf{M})$ and $C_z(\mathbf{M})$ are each expressed as sums of squared discrepancy terms. Specifically, the association cost entailed by **M** is taken to be

$$(3) \quad C_a(\mathbf{M}) = \|\mathbf{MS}-\mathbf{R}\|^2$$

and the size cost entailed by **M** is taken to be

$$(4) \quad C_z(\mathbf{M}) = \|\mathbf{M}\|^2$$

For any given training set $\{(\mathbf{r}_1,\mathbf{s}_1),(\mathbf{r}_2,\mathbf{s}_2),...,(\mathbf{r}_q,\mathbf{s}_q)\}$, a family of memory matrices is constructed, each having the following efficiency property: No other memory matrix achieves lower cost with respect to both the association and the size criteria. Such matrices are referred to as MAM matrices, and their associated cost vectors (C_a, C_z) are said to constitute the MAM frontier. By construction, the MAM frontier is a downward sloping strictly convex curve in the two-dimensional (C_a, C_z) cost plane. At one extreme of the MAM frontier is the cost vector incurred when the association cost $C_a(\mathbf{M})$ is minimized with no regard for the size cost $C_z(\mathbf{M})$. One memory matrix which minimizes the association cost is $\hat{\mathbf{M}}=\mathbf{RS}^+$, the memory matrix determined using the standard linear associative memory approach. Among all memory matrices which achieve the minimum association cost, the memory matrix $\hat{\mathbf{M}}$ is the one that has the smallest norm. At the other extreme of the MAM frontier is the cost vector incurred when the size cost $C_z(M)$ is minimized with no regard for the association cost $C_a(\mathbf{M})$. The memory matrix that uniquely solves this minimization problem is the zero matrix.

In view of the strict convexity of the MAM frontier, each point on this frontier solves a problem of the form "minimize C_z subject to C_a =constant." Consequently, each MAM matrix can be generated as the solution to a problem of the form

$$(5) \quad \min_{\mathbf{M}} [\alpha C_a(\mathbf{M}) + (1-\alpha) C_z(\mathbf{M})]$$

where α is a suitably chosen Lagrange multiplier lying between 0 and 1. The slope of the MAM frontier at the solution point

Copyright © 1995 Software Engineering Press

for (5) is given by $-(\alpha/[1-\alpha])$. Thus, α parameterizes the attainable trade–offs between association cost and size cost along the MAM frontier.

Given any nonlinear least squares problem such as (2), the MAM procedure can be used to generate a range of MAM estimates $\hat{r}(\alpha)$ for the underlying system parameter vector. The weight factor α is a tuning device which can be adjusted up or down to control for noise in the observation vectors as well as for noise due to round–off errors. The objective is to determine, through the training process,a range of values for α which result in one or more usable initial parameter estimates for the solution of the nonlinear least squares problem by a successive approximation scheme.

KALMAN FILTER FORMULATION BASED ON MAM ESTIMATES

Kalman Filtering is an optimal state estimation process applied to a dynamic system that involves random perturbations. More precisely, the Kalman Filter is a method for system identification, which is identifying the properties of an unknown system from observations, to predict the system's future behavior.The estimates generated by the Kalman Filter from noisy data taken at discrete points in real time are linear, unbiased, and have minimum error variance. It has been widely used in many areas of industrial & government applications such as video and laser tracking systems, satellite navigation, ballistic missile trajectory estimation, radar, and fire control. With the recent development of high–speed computers, the Kalman Filter has become more useful even for very complicated real time applications. For nonlinear models, a linearization procedure based on a Taylor series approximation is performed. The Kalman Filter so obtained will be called the Extended Kalman Filter

The Kalman formulation of the filtering problem assumes complete a priori knowledge of the process and measurement noise statistics. In most practical situations, these statistics are inexactly known. The use of wrong a priori statistics in the design of a Kalman filter can lead to large estimation errors or even to a divergence of errors. The purpose of initializing the Kalman Filter with MAM estimates is to reduce or bound these errors by adapting the Kalman Filter to the real data sampled by the training sets to get the MAM estimates. The estimation method relies on providing sufficient 'training'(i.e. exposure to different input–response pairs) for an adequate knowledge base to be acquired. Thus when the estimation scheme is later presented with a given input(output), it can then estimate the output(input) that corresponds to it. By starting out with just a few training pairs of inputs and responses, one can obtain reliably good initial estimates for the parameter vector to be estimated. Numerical results are shown for a nonlinear, growth of an economy at the macro level.

An Illustrative Economic Growth Problem

Consider an economy which produces a national product $Y(t)$ at each time $t \geq 0$ using capital and labor inputs $K(t)$ and $L(t)$. The production relation for the economy is given by

$$(6) \qquad Y(t)=F(K(t),L(t);\theta)$$

where θ is a parameter characterizing the production process. Denoting time t consumption by $C(t)$ and time t net investment $dK(t)/dt$ by $DK(t)$, and assuming that the amount of capital depreciation at each time t is a constant proportion $\delta \geq 0$ of the existing capital stock $K(t)$, supply equals demand in the time t product market if and only if

$$(7)\ Y(t) = C(t) + DK(t) + \delta K(t)$$

Time t gross savings $S(t) = Y(t) - C(t)$ are a constant proportion s of time t national product $Y(t)$, where the savings

Copyright © 1995 Software Engineering Press

rate s lies between 0 and 1. Thus,

$$(8) \qquad C(t) = [1-s]Y(t)$$

Substituting (8) into the product market clearing condition (7), and rearranging terms, the growth of the capital stock over times $t \geq 0$ is given by

$$(9) DK(t) = sF(K(t),L(t);\theta) - \delta K(t)$$

The labor force L(t) grows at a constant rate $g \geq 0$, with $L(0) > 0$. Let $k(t) = K(t)/L(t)$ and $y(t) = Y(t)/L(t)$ denote the time t capital–labor and income–labor ratios. Using the constant returns to scale assumption for $F(\cdot;\theta)$, the production relation (6) can be expressed in per capita terms as

$$(10) \; y(t) = F(k(t),1;\theta) \equiv f(k(t);\theta)$$

Also, the time rate of change of the capital–labor ratio k(t) satisfies

$$(11) \; Dk(t)/k(t) = DK(t)/K(t) - DL(t)/L(t) = DK(t)/K(t) - g$$

Finally, define $\lambda = [g + \delta]$, and divide each side of equation (9) by L(t). Making use of relations (10) and (11), it follows after some manipulation of terms that the time rate of change Dk(t) of the capital–labor ratio k(t) satisfies the differential equation

$$(12) \; Dk(t) = sf(k(t);\theta) - \lambda k(t), \; t \geq 0$$

where the initial capital–labor ratio k(0) is given by some historically determined value $k_0 > 0$. Equation (12) is the basic differential equation for the Solow–Swan growth model, a well–known macroeconomic model which is still very influential. See [8] and [9].

At each time $t_j, j=1,2,...,m$, an observation $k^*(t_j)$ is obtained on the capital–labor ratio $k(t_j)$ in accordance with the measurement relation

$$(13) \qquad s^* = \begin{bmatrix} k^*(t_1) \\ k^*(t_2) \\ \vdots \\ k^*(t_m) \end{bmatrix}$$

the problem is to estimate the parameters (k_0,θ,s,λ) which characterize the underlying data–generating process (11).

MAM Procedure for the Solow–Swan Growth Model

The first step in the MAM procedure is the construction of a finite set of training cases. Given any training parameter vector $r_i = (k_0,\theta,s,\lambda)^T$ for the Solow–Swan growth model, a corresponding training output vector $s_i = (k(t_1),k(t_2),...,k(t_m))^T$ can be generated by numerically integrating the basic Solow–Swan differential equation (12). A closed–form expression for the solution of this differential equation is not required.

For conceptual clarity, however, it is useful to focus on a special case in which a closed–form expression for the solution of (12) can be obtained. Specifically, suppose the per–capita production function $f(\cdot;\theta)$ for the Solow–Swan growth model takes the commonly used Cobb–Douglas form

$$(14) \qquad f(k;\theta) = k^{\theta}$$

for some $\theta \in (0,1)$. The production parameter θ then gives the capital share of the national product at each time t. That is, $\theta = p(t)k(t)/y(t)$, where the time t capital rental rate p(t) is taken to be the time t marginal product of capital $f'(k(t))$.

Given (14), the solution to the basic Solow–Swan differential equation (12) is

$$(15) \qquad k(t) = H(t;k_0,\theta,s,\lambda)$$
$$k(t) = ([[(k_0)^{1-\theta} - s/\lambda]e^{-(1-\theta)\lambda t} + s/\lambda)^{1/(1-\theta)}$$

Note that (15) is a highly nonlinear function

Copyright © 1995 Software Engineering Press

of the four model parameters k_0, θ, s, and λ. For each given parameter vector $(k_0, \theta, s, \lambda)$, the solution value (15) for the time t capital–labor k(t) ratio converges as t approaches infinity to the stationary solution value

$$(16) \qquad \bar{k} = (s/\lambda)^{1/(1-\theta)}$$

General Experimental Set–up

The MAM procedure can be implemented even if nothing is known a priori about the nonlinear least squares solution for the parameter vector. However, if a priori information is available concerning plausible values for this solution, the training parameter set should presumably be designed to encompass these values.

To demonstrate the MAM procedure for the Solow–Swan growth model over a plausible training grid, the training parameter sets $\{\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_q\}$ for all of the numerical reported below were constructed on the basis of the following guideline parameter values:

$$(17) \quad k_0 \approx 5.0;\ \theta \approx .29;\ s \approx .15;$$
$$\lambda = [g + \delta] \approx [.03 + .07] = .10$$

The guideline values for the capital share θ, the gross savings rate s, the effective labor growth rate g, and the depreciation rate δ in (17) were constructed using empirically determined ratios and magnitudes given in [3]. As noted in [10, p. 150], the meaning of the "capital stock" K(t), and hence the capital–labor ratio k(t) is a source of much controversy in growth theory. The guideline value for k_0 in (17) is for illustrative purposes only.

To simplify graphical depictions and comparisons, the training parameter set for each experiment consisted of 49 two–dimensional parameter vectors $\mathbf{r}_i$, with constant guideline values set for the remaining two parameters. One series of experiments ("Series I") was run for training parameter vectors of the form $\mathbf{r}_i = (k_0, \theta)^T$ and another series ("Series II") was run for training parameter vectors of the form $\mathbf{r}_i = (s, \theta)^T$.

In each experiment, seven different values were considered for each of the two training parameters, and these values were centered around the parameters' guideline values in (17). Consequently, in each experiment the "training parameter grid" consisted of a 7 x 7 square of two–dimensional parameter points approximately centered at a guideline point as determined from (17).

Each experiment in each series consisted of four basic steps.

- First, for each considered training parameter set $\{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{49}\}$, a corresponding training output set $\{\mathbf{s}_1, \mathbf{s}_2 \ldots, \mathbf{s}_{49}\}$ was generated using the Adams integration algorithm for equation (12). Each 15 x 1 training output vector $\mathbf{s}_i$ consisted of fifteen capital–labor ratios $k(t_j)$, j=1,2,...,15, calculated for the "observation times" $t_1 = 0.0, t_2 = 1.0, \ldots t_{15} = 14$. These training vectors were used to form training response and stimulus matrices.
- Second, MAM matrices $\hat{\mathbf{M}}(\alpha)$ are constructed, with α ranging from 0.10 to 1.0
- Third, the components of the training output vectors $\mathbf{s}_i$ were corrupted with additive noise, resulting in a set of noisy "observation vectors " of the form $\mathbf{s}_i^* = \mathbf{s}_i + \mathbf{n}_i$.
- Fourth, tests were conducted to determine the extent to which each MAM matrix $\hat{\mathbf{M}}(\alpha)$ is successfully able to recover the training parameter vector $\mathbf{r}_i$ when post–multiplied by the noisy observation vector $\mathbf{s}_i^*$, i=1,2,...,49. For each training case i, the measured used to judge the success of the recovery is the discrepancy, in percentage terms, between the jth component $\hat{r}_{ij}(\alpha)$ of the MAM estimate $\hat{\mathbf{r}}(\alpha) = \hat{\mathbf{M}}(\alpha)\mathbf{s}_i^*$ and the jth component r_{ij} of the actual training parameter vector $\mathbf{r}_i$, j=1,2:

Copyright © 1995 Software Engineering Press

(18) $dis_{ij}(\alpha)=([\hat{r}_{ij}(\alpha)-r_{ij}]/r_{ij})x100$

In each series of experiments, the effect of observation noise on the accuracy of the resulting parameter estimates is investigated

Suppose the observation vectors $\mathbf{s}_i^*$ are taken to be the training output vectors $\mathbf{s}_i$ corrupted by additive gaussian noise with mean 0 and variance σ^2. In this case, to achieve the minimum mse recovery of the training parameter vectors over the training grid, it follows from (13) that the penalty weight α in (16) should be set equal to

(19) $\alpha^0 = 1/[1 + q\sigma^2]$

In particular, for any given number of training cases q, the penalty weight α should be set close to 1.0 when observation noise is minimal ($\sigma^2 \approx 0$) and close to 0 when observation noise is extensive ($\sigma^2 \gg 0$).

A variety of numerical experiments were run for the Solow–Swan growth model with observation vectors $\mathbf{s}_i^*$ taken to be the training output vectors $\mathbf{s}_i$ with components corrupted by additive gaussian noise generated by means of a $N(0,\sigma^2)$ pseudorandom number generator.

The MAM estimates $\hat{\mathbf{r}}_i(\alpha)$ determined in these experiments were reasonably accurate α was set equal to a value in [.10,.99] which was roughly nearby the minimum mse value (19), e.g., a distance apart of .20 or less. Overall, high accuracy levels (discrepancies around one percent or less) were attained for the initial capital–labor ratio k_0, good accuracy levels (discrepancies around ten percent or less) were obtained for the gross savings rate s, and reasonably good accuracy levels (discrepancies around twenty percent or less) were obtained for the capital share θ. Moreover,recovery accuracy was generally good for all training parameter vectors lying near the center of the training parameter grids.
Some of these experiments will now be reported in more detail.

Consider, first, the case in which the observation vectors are noise free ($\sigma = 0$), so that the ith observation vector $\mathbf{s}_i^*$ coincides with the ith training output vector $\mathbf{s}_i$. The minimum mse value for α in this case is $\alpha^0 = 1.0$. To what extent are the MAM matrices $\hat{M}(\alpha)$ able to recover the training parameter vectors $\mathbf{r}_i$ when post–multiplied by the training output vectors $\mathbf{s}_i$?

Several interesting observations can be made. First, even when α takes on the value .10, and is thus very far from its minimum mse value 1.0, highly accurate estimates are obtained for the initial capital–labor ratio k_0, especially along the reverse diagonal. Moreover, the corresponding estimates for the capital share θ are highly accurate toward the center of the training grid, and reasonably accurate (less than twenty two percent) elsewhere with the exception of the column where θ takes on its smallest training value .20. Estimation accuracy improves with increases in α as long as α remains below 1.0.
Qualitatively similar results were obtained for Series II experiments with $\sigma = 0$. The discrepancies obtained for the gross savings rate s tended to be higher than those obtained for k_0, and the discrepancies obtained for θ tended to be significantly higher.
What happens in the noisy observation case $\sigma > 0$?

For the MAM estimates of k_0 in the Series I experiments, the answer is "not much." The percentage discrepancies for k_0 for the most part remained well below ten percent all along the MAM frontier for each σ in the tested range 0.05 to 0.40. Occasionally along the boundary of the training parameter grid the percentage discrepancies rose above ten percent, but only by a few percentage points. The corresponding MAM estimates for θ were reasonably accurate (discrepancies around twenty percent or less) over the interior of the training parameter grid for each tested σ value when the value of α was set roughly in the neighborhood of α^0, e.g., a distance apart of about .20 or less.

Copyright © 1995 Software Engineering Press

In the Series II experiments, reasonably accurate estimates(discrepancies around twenty percent or less) were obtained for both s and θ over the interior of the training parameter grid for each tested σ value when the value of α was set roughly near α^0,e.g., a distance apart of about .20 or less.

The Solow–Swan growth model with the guideline parameter values listed in (17) is the basis for trying to answer the following questions:

- First, How many training cases are necessary to gain a good and reliable MAM estimate for the system studied ? Does a need for a huge number of training cases guarantee convergence to a solution ?
- Second, Does spacing of training parameters over a grid of possible values have any effect on convergence to correct solution ?
- Third, How many measurements are needed for sufficient convergence with respect to discrepancy values ?
- Fourth, What should time spacing be between measurements ?

The results show that for 25 training cases, the convergence to the discrepancy values that are generated for 49 training cases is comparable (within 2 %). Closer the training points are taken within the training grid, the more reliable the MAM estimates are. A reduction in separation distance between data points with an increase in training cases produces the most accurate MAM estimates. Increasing the number of training cases with a fixed spacing results in a degradation of estimation accuracy. An increase in spacing with fixed number of training cases, also increases inaccuracies. For 1 second spacing between observations, the more observations utilized to generate the MAM estimates, the better the estimates. After 7 measurements, no significant improvement occurs. For the Solow–Swan model, taking two measurements, 5 seconds apart generates estimates that are as accurate as 5 measurements, 2 seconds apart and 10 measurements that are 1 second apart. This statement holds for the Series I and Series II experiments due to the relatively linear form that the Capital–labor ratios take as a function of time.

Implementation of the Extended Kalman Filter to the Solow–Swan Model

Consideration to the Solow–Swan model whose dynamics are known in structure is given in this section. The Extended Kalman Filter is utilized for identification of a nonlinear system. The identification method relies on providing sufficient 'training' (i.e., exposure to different input–response pairs) for an adequate knowledge base to be acquired . The MAM matrix is adduced which optimally associates the inputs with the outputs. Thus when the identification scheme is later presented with a given output, it can then estimate the inputs that correspond to it.

Given the Solow–Swan nonlinear differential equation with the Cobb–Douglas per–capita production function form:

$$(20)\quad Dk(t)=sk(t)^{\theta}-\lambda k(t) \quad t\geq 0$$

the Extended Kalman Filter equivalent formulation using the transformation $\zeta = k^{1-\theta}$ is derived as follows:

$$(21)\quad \dot{\zeta} = (1-\theta)s-(1-\theta)\lambda\zeta \quad t\geq 0$$

Let the state vector $\mathbf{x} = [\zeta,\theta,\lambda,s]^T$, equation 21 becomes:

$$(22)\qquad \dot{\mathbf{x}}= \begin{bmatrix} \{1-x(2)\}x(4)-\{1-x(2)\}x(3)x(1) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The nonlinear discretized system model which is generated by replacing $\mathbf{x}$ by $\mathbf{x}_i$ and $\dot{\mathbf{x}}$ by $(\mathbf{x}_{i+1}-\mathbf{x}_i)dt^{-1}$ where $dt>0$ denotes the sampling time, is as follows: $\mathbf{x}_{i+1} = \mathbf{f}_i\mathbf{x}_i$, where

Copyright © 1995 Software Engineering Press

$$(23)\mathbf{f}_i = \begin{bmatrix} 1 & dt\{x_i(3)x_i(1)-x_i(4)\} & -dtx_i(1) & dt \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The measurement model based on k(t) being the observations is derived as follows:

If $\zeta = k^{(1-\theta)}$: then $k = \zeta^{(1/1-\theta)} = \zeta^{(1-\theta)^{-1}} = x(1)^{(1-x(2))^{-1}}$, therefore,

$$(24)\mathbf{z}_i=\mathbf{h}_i\mathbf{x}_i+\mathbf{v}_i=x_i(1)^{(1-x_i(2))^{-1}}+\mathbf{v}_i$$

where $\mathbf{v}_i$ is the measurement noise, usually a Gaussian white noise process. This noise component is the equivalent to the additive noise with mean 0 and variance σ^2 that is added to the training output vector $\mathbf{s}_i$ for the MAM experiments. The Extended Kalman Filter is propagated through the following equations,

(25) State Estimate Propagation Equation $\hat{\mathbf{x}}_{i+1/i} = \mathbf{f}_i\hat{\mathbf{x}}_{i/i}$

(26) Error Covariance Propagation Equation $\mathbf{P}_{i+1/i} = \boldsymbol{\phi}_i\mathbf{P}_{i/i}\boldsymbol{\phi}_i^T$,

where $\boldsymbol{\phi}_i = [\partial \mathbf{f}_i/\partial \mathbf{x}_i(\hat{\mathbf{x}}_{i/i})]$

(27) Gain Matrix

$\mathbf{K}_i=\mathbf{P}_{i/i-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{i/i-1}\mathbf{H}^T+\mathbf{R})^{-1}$,

where $\mathbf{H}=[\partial \mathbf{h}_i/\partial \mathbf{x}_i(\hat{\mathbf{x}}_{i/i-1})]$

(28) State Estimate Update Equation

$\hat{\mathbf{x}}_{i/i} = \hat{\mathbf{x}}_{i/i-1} + \mathbf{K}_i(\mathbf{z}_i-\mathbf{h}\hat{\mathbf{x}}_{i/i-1})$

(29) Error Covariance Update Equation

$\mathbf{P}_{i/i} = (\mathbf{I}-\mathbf{K}\mathbf{H})\mathbf{P}_{i/i-1}$

where I is the identity matrix.

Extended Kalman Filter Results

Series II experiments for the (s,θ) training grid when $\alpha = 0.9$ in the noise free case results in discrepancy factors for θ when s=0.15 range from –10 to 10 % and –9 to 7 % for s when θ=0.29. The initial conditions for filter initialization are:

True State vector $\mathbf{x} = [5^{\cdot 71},.29,.10,.15]$

State Estimate vector

$\hat{\mathbf{x}}_{0/0} = [5^{\cdot 71},.29\pm 10\ \%,.10,.15-9\%]$

State Covariance Matrix

$\mathbf{P}_{0/0} = \mathrm{cov}[\mathbf{x}-\hat{\mathbf{x}}_{0/0},\mathbf{x}-\hat{\mathbf{x}}_{0/0}]$

Series I experiments produce discrepancy values of 0 for all k,

$\hat{\mathbf{x}}_{0/0}(1)=x(1)=5^{\cdot 71} = 3.135.$

A small gaussian white noise with 0 mean and a $(.025)^2$ variance was added to the observations in order to insure against underflow errors in the covariance matrix entries. Assuming no MAM estimates are considered for a priori information for the Eextended Kalman Filter, then the initial filter initialization is as follows:

$$\hat{\mathbf{x}}_{0/0} = [7.0^{\cdot 8},.20,.10,.09]$$
$$\mathbf{x}_{0/0} = [5.0^{\cdot 71},.29,.10,.15]$$

Table 1a
Discrepancy Percentages with and without MAM Estimates
Minimal Noise case, 100 data points

True State values	% Error MAM	% Error NO MAM
1.50	0.00	4.00
0.29	0.36	21.00
0.10	0.00	0.00
0.15	0.09	4.00

Copyright © 1995 Software Engineering Press

Table 1b Large Noise case, 100 data points

True State values	% Error MAM	% Error NO MAM
1.50	2.00	10.30
0.29	7.50	50.00
0.10	0.00	0.00
0.15	2.00	37.00

The results above show the improvement in convergence when a priori information is available. A priori information is generated through the enumeration of different training cases to generate a MAM estimate. As with MAM estimates, when observations are corrupted with noise, the resulting state estimates are not quite as accurate. This is also true due to the covariance matrix entries being corrupted by noise. Of primary significance is how 1σ value for the small noise case converges to zero. This indicates that the corresponding state estimates are highly accurate. The small error of the initial estimates of the filter from the true value allow the Extended Kalman Filter to be more robust to noise, even in the presence of nonlinearities. This robustness allows the 1σ error to converge towards zero, rather than reaching steady–state at a non–zero value. In the presence of a very small noise term, with accurate initial estimates (MAM A–Priori), the 1σ values converge to zero at a significantly faster rate.

CONCLUSIONS

Successful implementation of successive approximation procedures for nonlinear least squares problems typically require initial estimates for the parameter vector which are within ten to 30 percent of the actual solution vector. Experimental results reported in this paper suggest that one or more usable initial estimates for the parameter vector might be found by considering the MAM parameter vector estimates corresponding to a rough sample of a–points along the MAM frontier.

Specifically, in each experiment with observation noise, reasonably accurate MAM parameter estimates were obtained over the interior of the training parameter grid for a–values in [.10,.99] lying even roughly nearby the (generally unknown) "optimal" alpha values.That is, the discrepancies between these estimates and the true parameter values were around twenty percent or less. On the other hand, even in the absence of observation noise, highly inaccurate parameter estimates were obtained when a was set equal to 1.0, i.e., when the standard linear associative memory procedure was used.

Reduction of spacing between the training parameters together with increasing the number of training cases, encompassing a grid of values around the guideline parameter values, tended to generate low discrepancy values. Only a few data points with maximum separation between observation times are needed for the MAM approach to work. The above findings are also applicable to the noisy observation case. The MAM approach, thus, is quite robust to corruption of measurements by noise.

A priori information is essential for minimizing the initial error when implementing an Extended Kalman Filter as an estimator for a nonlinear dynamical system. Otherwise, the convergence of its error estimates may not be consistent with its error covariance predictions. In fact, its estimates may converge to bias errors which can be quite large with respect to the predicted bounds embedded in the covariance matrix. In the two examples implemented , it has been shown the significant improvement in the convergence of the error to zero, when the MAM estimate values are used to initialize the Extended Kalman Filter. Of particular interest, are discrepancy values that are two to three orders of magnitude larger for the noisy measurement case when no a–priori(no–MAM) information is utilized. For the low noise case, the convergence of the filter is solely based on how reliable the initial estimates and initial covariance are. The magnitude of the error, in the worst case scenario(Solow–Swan Model),is 58

Copyright © 1995 Software Engineering Press

times larger for the non–MAM case than the MAM–case. Rapid convergence of the filter for the MAM case, together with the reliability of convergence, brought about by having accurate initial estimates with bounded initial covariance errors, provides a very powerful tool in terms of precision and speed in the nonlinear dynamic model estimation area.

REFERENCES

[1] C. Chui and G. Chen, *Kalman Filtering with Real–Time Applications,* (Springer–Verlag, N.Y., 1991)

[2] A. Cuyt and L. Rall, "Computational implementation of the multivariate Halley method for solving nonlinear systems of equations," *ACM Transactions on Mathematical Software* 11 (1985) 20–36.

[3] E. Denison, *Trends in American Economic Growth, 1929–1982* (The Brookings Institution, Washington D.C., 1985).

[4] C. Gray, "Applications of Newton's Method to Attitude Determination," *AAS Guidance and Control Conference,* Keystone, Colorado 1992.

[5] G. Judge, W. Griffiths, R. Hill, H. Lutkepohl, and T.C. Lee, *The Theory and Practice of Econometrics,*(John Wiley, N.Y., 1985)

[6]R. Kalaba, Z. Lichtenstein, T. Simchony, and L. Tesfastion, "Linear and nonlinear associative memories for parameter estimation," *Inform. Sci.*, vol 61 (1992) 45–66.

[7] R. Mehra, "Approaches to Adaptive Filtering," *IEEE Transactions on Automatic Control,* vol. 10 (1972) 693–698.

[8] E. Prescott, "Robert M. Solow's neoclassic growth model: an influential contribution to economics," *Scand. J. of Econ.* 90 (1988) 7–12.

[9] R. Ramannathan, *Introduction to the Theory of Economic Growth* (Springer–Verlag, N.Y., 1982).

[10] J. Simon, "Great and almost great magnitudes in economics," *J. of Econ. Perspectives* 4 (1990) 149–156.

[11] H. Vinod and A. Ullah, *Recent Advances in Regression Methods* (MarcelDekker, N.Y., 1981).

Copyright © 1995 Software Engineering Press

Forecasting Currency Exchange Rates: Neural Networks and the Random Walk Model

Eric W. Tyree

Dept. of Business Computing
City University
London EC1V 0HB
United Kingdom
Tel: 017 - 477 - 8413
E-Mail: e.tyree@city.ac.uk

J. A. Long

Dept. of Business Computing
City University
London EC1V 0HB
United Kingdom
Tel: 017 - 477 - 3404
E-Mail: j.a.long@city.ac.uk

ABSTRACT

This work provides an evaluation of the use of neural networks as a technique for forecasting currency exchange rates. Recently, successful attempts at forecasting exchange rates such as the US$ - DM and US$ - SF have been reported in the literature (i.e. Refenes et al (1993, Weigend et al (1992))) but their methodologies have been less than stringent leaving them open to accusations of data mining. The work presented here will attempt to replicate some of this previous work and then subjugate the resulting neural network forecasts to a more stringent level of analysis. More specifically, standard backpropagated feedforward networks will be used to forecast the US$ - DM exchange rate 1, 5, and 20 trading days into the future with the resulting performances compared to the random walk forecasting model and to an autoregressve forecasting model. The experimental techniques used here are also proposed as a general framework which should be followed when making claims of the successful application of neural networks to financial time series generally seen as unforecastable.

INTRODUCTION

One of the more difficult problems in economics is the forecasting of financial markets. Traditional quantitative methods used to forecast the behaviour of financial markets often produce unsatisfactory if not dismal results given the complex interactions between a given market's behaviour and other economic phenomena. Part of the problem lies in the fact that the relationships existing between financial markets and the economy as a whole are often poorly understood. On top of this there are also a variety of political and psychological factors influencing the dynamics of the markets over time. Neural networks may provide some hope of producing a suitable methodology for overcoming some of these difficulties.

A number of successful claims of using neural network based market forecasting systems have been published. Unfortunately, much of this work suffers from inadequate documentation regarding methodology (Binks and Allinson (1991), Collard (1991), Lee and Park, (1992)) or claims of positive results not backed up by comparisons with other relevant forecasting techniques (Binks and Allinson (1991), Lee and Park, (1992), Collard (1991) Weigend *et al* (1992)). This makes it difficult to both replicate previous work and obtain an accurate assessment of just how well connectionist techniques really perform in comparison to other forecasting techniques. What previous work has been done using connectionist approaches to market forecasting can be roughly categorised based on how a forecast is being extracted from the input data with the neural network

Copyright © 1995 Software Engineering Press

model. Most have attempted to extrapolate the future behaviour of a market with a neural network based times series analysis by having the network output some value representing the future behaviour of the market (i.e. forecasting the price, expected return or degree of change etc...). This is usually done by giving the network information about the market's past behaviour (Refenes *et al* (1993)) or information about its past behaviour in conjunction with the dynamics of a variety of other economic variables used as additional input (Weigend *et al* (1992), Lee and Park (1992) and Hutchinson (1994)). Others have tried to train the network to recognise known market patterns (Binks and Allinson, (1991)) or attempt to train the network to learn an optimised trading strategy (Collard (1991) and Kimoto *et al* (1990)).

This report will attempt to apply a connectionist approach to the forecasting of a notoriously "unpredictable" financial market - currency exchange rates. Some relatively straight foreword methods of using standard backpropagated feedforward neural networks to forecast the US$ - DM exchange rate will be analysed and compared with other forecasting models. These experiments will include univariate forecasting of the exchange rate at 1, 5 and 20 days in advance and multivariate forecasts at 1 and 5 days in advance.

In addition, a methodological framework is also proposed for the use of neural networks in financial forecasting. The framework is quite simple and consists of two basic techniques. First, the performance of neural networks should be compared with other relevant forecasting models. Simply demonstrating that neural network based methods "work" is not enough as this does not shed any light on their relative performance to potentially simpler and more accurate forecasting methods. For this work, the random walk forecasting model will be use as the primary comparison model as currency exchange rates are widely viewed to be best explained as random walks (Diebold and Nason, (1990)). The random walk model simply states that due to market efficiency, current price changes are independent of past price changes. In other words, univariate forecasting should be impossible as past price changes do not offer any clues to what form the future behaviour of prices might take. Since price changes in efficient markets such as exchange rates are assumed to be a random distribution with 0 mean (see Pindyck and Rubinfeld (1991)) the best forecast one can make for any amount of time in the future is to assume the future price will be the same as today's price. As previous claims of success with univariate forecasting of the US$ - DM exchange rate contradict the random walk model, the random walk model is the most appropriate to base a comparison with neural networks with[1] in this instance.

Second, when claiming positive results steps should be taken to guard against accusations of data mining. It will be shown here that spurious results are not difficult to obtain in some instances. To help circumvent this problem, the approach taken here is to run our simulations on multiple portions of the data set to guard against the possibility of chance results.

METHODS

This study consists of five main experiments intended to examine the relative performance of neural networks and the random walk model in forecasting the US$ - DM exchange rate. The first experiment attempts to use a feedforward backpropagated network to forecast the US$ - DM exchange rate one trading day in advance using input consisting solely of daily US$ - DM data in much the same way as Refenes *et al* (1993). The second experiment attempts to fit the random walk model to the exchange rate data to more accurately ascertain the appropriateness of the random walk model as an explanation for the behaviour of the price changes in the exchange rate. The third experiment will use an identical technique as

[1]Note that the random walk model only refers to univariate or "technical" forecasting. It does not state that price changes in particular markets that follow random walks are also operating independent of other variables.

Copyright © 1995 Software Engineering Press

the first experiment to forecast 5 trading days (one week) and 20 trading days (one month) in advance. The fourth experiment will conclude the univariate forecasting by taking a multistep approach to forecasting. In multistep forecasting the output from the network after presentation of the final training pattern is taken as input to the network for the next forecast step. This process is then repeated for the entire length of the forecast lead period. Finally, the last experiment will attempt one and five trading day forecasts using multivariate input incorporating interest rates and other currencies.

In all the experiments, the networks consisted of standard feedforward networks with full connectivity between layers. Connections between units were restricted to being from one layer to the next. Learning was conducted with the standard backpropagation algorithm with momentum term and utilised the standard sum squared error cost function:

$$E = 0.5\sum_{p=1}^{N}(A_p - D_p)^2$$

where A is the network output for output pattern p, D is the desired output for pattern p and N is the number of patterns. The results of the neural network model were then analysed with respect to random walk forecasting model. The random walk forecasting model was defined as today's price being the best forecast for any point in the future. More formally, $y(t + N) = y(t - 1) + \varepsilon(t)$ where $y(t + N)$ is any point in the future and $\varepsilon(t)$ is an error term[2]. The relative performances of the models used in this work was analysed by comparing the mean squared errors of the models over the test sets. All currency and interest rate data used was daily data from the period beginning Jan. 1, 1990 and ending May 31, 1994.

[2]Other definitions of the random walk model exist which also accommodate systematic "drift" in the data. As no such long term drift was found, the simpler version was used here.

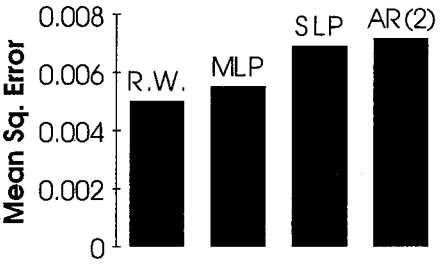


Fig. 1: Results of univariate single day forecasts.

RESULTS

Experiment 1: Single day univariate forecasting

In this first experiment the notion that a previous sequence of the US$ - DM exchange rate $W^I = y(t), y(t - 1), \ldots, y(t - n)$ can be used to forecast the value of the exchange rate one day in advance $W^O = y(t + 1)$ with a neural network was examined. The training data used consisted of the first 850 trading days of the exchange rate data and the test data consisted of the following 50 days. Various sizes of W^I were tried, ranging from 2 to 20 trading days, of which none provided satisfactory results. Increasing the number of hidden units did not produce any improvement either.

The results are shown in fig. 1 which compares the results from the random walk model, an autoregressive model, a multilayer perceptron with 10 inputs and 5 hidden units and a single layer perceptron with 10 inputs. The network results displayed were typical of that found in this experiment regardless of the hidden or input layer size. Clearly, the random walk model is producing more accurate forecasts. An autoregressive process used to fit the training data produced a second order linear model of the form:

$$y(t) = 1.0426y(t - 1) - 0.053y(t - 2) + \varepsilon(t)$$

Copyright © 1995 Software Engineering Press

where y(t) is the value of the US$ - DM at time t indicating that the value at time t is almost entirely dependant on the value at time t - 1. Although the networks did slightly better than the AR(2), none outperformed the random walk model. Even further, the best performing networks seemed to be implementing something approaching a random walk as can be seen in fig. 2. The top of fig. 2 displays the random walk forecast along with the test set while the bottom displays the output of the 10 - 5 - 1 network on the test data. Clearly, the network is simply using the previous value of the exchange rate as a forecast for the next day - a random walk.

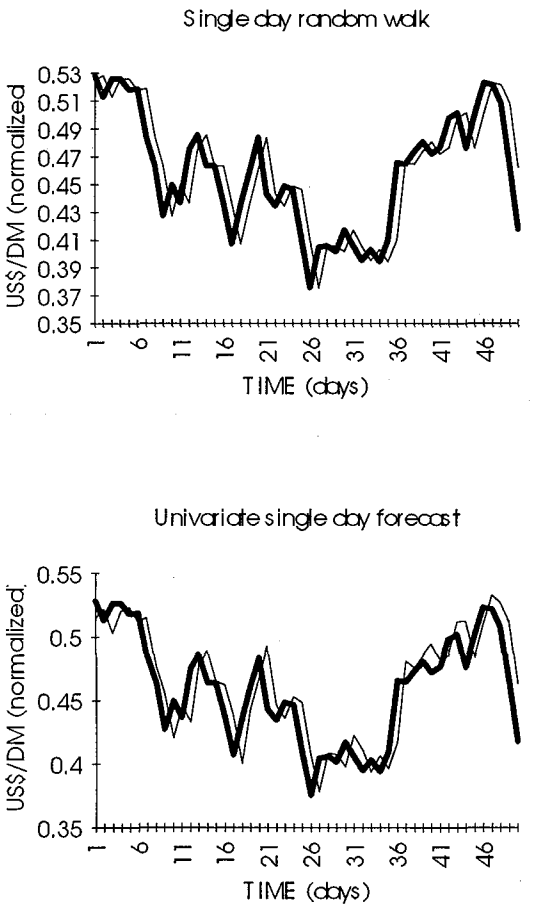


Fig. 2: Random walk (top) and neural network (bottom) single day forecasts. The thick lines are the actual exchange rate and the thin lines represent the forecasts.

Experiment 2: Fitting the random walk model

The superior performance of the random walk model in the previous experiment necessitates investigating more formally how well the random walk model can explain the prince changes in the US$ - DM exchange rate. If the price changes in the US$ - DM exchange rate do in fact follow a random walk, the differences between one day's rate and the next should be random. In more precise terms, if the daily changes in the exchange rate can be explained by the random walk model, the residuals left over from fitting the random walk model should be random noise. The residuals are defined as Y(t) - M(t) where Y(t) is the actual value of the time series at time t and M(t) is the value at time t given by the model.

In this experiment, the random walk model is used to fit the US$ - DM exchange rate. As mentioned before, this model assumes that a value at a given time t + N is equal to the value at time t plus some noise. If this is true, then the k - day differenced US$ - DM exchange rate series should be random noise as these values would simply be the random series left over from fitting the random walk model to the k - day price changes. It is also good practice to look at the squares of the differenced series as this helps prevent any cyclic component of the series from cancelling out and making the series appear random when in fact it is not.

Two tests for randomness were run to test the fit of random walk model on the differenced and differenced squared US$ - DM exchange rate data. These tests consisted of the difference-sign test and a serial correlation test. The difference sign test simply looks at the differenced series and counts the number of times a positive change is found in the series. It can be shown (see Kendall and Stuart (1968)) that a truly random series will have (n - 1)/2 positive changes in value and a variance of (n + 1)/12 with the resulting distribution tending rapidly towards normality (Moore and Wallis (1943)). The serial correlation test simply tries to find a correlation between successive values. If a given series has structure beyond random

Copyright © 1995 Software Engineering Press

fluctuations, there will be some degree of correlation between one value and the next (Kendall and Stuart (1968)).

RANDOM WALK MODEL RESIDUALS

	Lag Times				
	1 Day	2 Day	3 Day	4 Day	5 Day
DIFFERENCE SIGN					
exp p.d.	575.0	288.0	192.0	144.0	115.0
exp s.d.	9.80	6.94	5.67	4.92	4.40
diff	559,	278,	185,	144,	116,
p=	0.10	0.15	0.22	1.0	0.82
diff sq.	544,	298,	194,	141,	117,
p=	0.002	0.15	0.73	0.54	8.82
SERIAL CORRELATION					
diff	0.030	0.005	-0.036	-0.049	-0.056
diff sq.	0.063	0.112	0.208	-0.005	0.183
$\pm_{sig}$ @ 0.95%[3]	0.059	0.083	0.102	0.118	0.132

Table 1: Random walk model residuals.

In this experiment the entire data set used in this work was run through these tests at time lags of 1, 2, 3, 4, and 5 days. In other words, the adequacy of the random walk model is being tested for the changes in the US$ - DM exchange rate for periods of 1 to 5 days. It should be noted that when testing at time lags greater than 1 day the series must be differenced such that $d_n = y_{tk} - y_{(t-1)k}$ where d is the differenced value and k is the lag. The reason for this is that if one were to simply difference every value in the series from the value k time steps in the past, one would artificially induce correlation in the series that did not initially exist as the various values resulting from the subtraction process would contain common terms. Therefore, a series of N values will produce a differenced series of size N/k. For this reason, only lags of up to 5 days were tested for the random walk model as lags of more than 5 would produce too small of samples. The results are displayed in table 1.

[3]This is the probability that the correlation found is significantly different from 0 using the general heuristic of $2/\sqrt{N}$ to determine 95% certainty (Chatfield, (1975)).

Looking first at the difference-sign test, the top part of table gives the expected number of turning points and the expected standard deviation for each lag time. The next four rows give the results of the differenced and differenced squared exchange rates along with p which indicates the probability that the number of positive sign changes found in the exchange rate is indicative of it being a random series using a simple z test.

In short, these results are quite marginal except in two cases. A 1 day lag it can be said with greater than 95% certainty ($p = 0.002$) that the difference squared series in not random. Conversely, at 4 days it can be said that the differenced series is random with greater than 95% certainty ($p = 1.0$). The rest of the results fail the 95% percent certainty criteria for either accepting or rejecting that the exchange rate is random. In these cases the probability that the number of positive changes observed indicate that the exchange rate is random ranged from 0.10% ($p = 0.1$) and 82% ($p = 0.82$). A less stringent criteria for accepting the hypothesis that the changes in the US$ - DM are random could be adopted in which any number of positive changes found within one standard deviation (i.e. $p <= 68\%$) of the expected number of positive changes would be accepted as indicative of randomness. In this case, the results are still mixed with the differences series indicating non randomness at lags of 1 ($p = 0.01$), 2 ($p = 0.15$) and 3 ($p = 0.22$) and the differenced squared series indicating non randomness at lags of 1 ($p = 0.002$), 2 ($p = 0.15$) and 4 ($p = 0.54$).

The serial correlation test resulted in slightly more consistent results. For the differenced series, all five time lags indicated randomness $p >= 95\%$ while the differenced squared series indicated significant non randomness at 95% certainty at all lags except 4.

As much as these results are mixed, they do seem to indicate that the US$ - DM exchange rate is not strictly random. In other words, there is some structure to be found in the 1 - 5 day price changes albeit small

Copyright © 1995 Software Engineering Press

and probably very subtle. The random walk model can probably be rejected as the most appropriate model explaining the changes in the US$ - DM exchange rate. Nonetheless, because the structure that exists in the changes is so small (and possibly complex) forecasting these changes will most probably be anything but trivial.

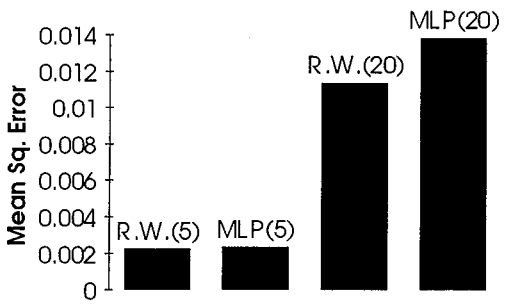


Fig. 3: Results of the 5 and 20 day forecasts.

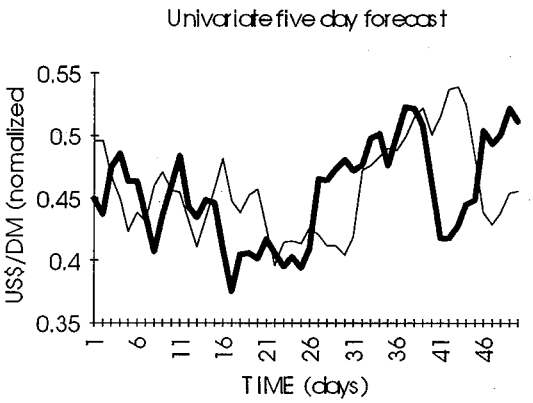


Fig. 4: The 5 day univariate forecast. The thick line is the actual US$ - DM and the thin line is the forecast.

Experiment 3: 5 and 20 day univariate forecasting

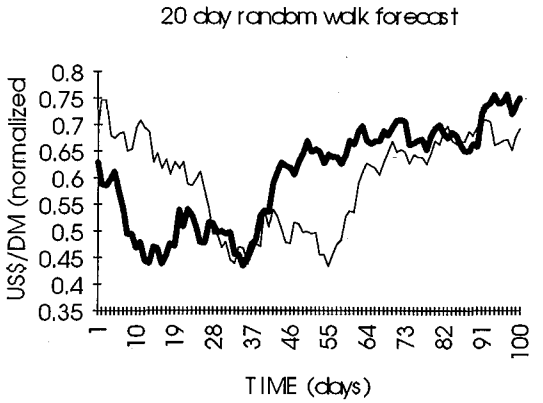
The previous experiment attempted to forecast the US$ - DM exchange rate at time t + 1 with a moving window of univariate data up to and including the rate at time t. Given the results of experiment 2 which demonstrated that there may be some subltle structure in the US$ - DM exchange rate, this next experiment attempted to use a network to find a relationship between a segment of the time series consisting of data up to and including the rate at time t and the value of the exchange rate at time t + 5 and t + 20 that can be used for a forecast.

In regards to the 5 day forecasts, various sizes of input window were attempted. An input window size of 20 trading days W^i = y(1), y(t - 1) ... y(t - 20) is displayed in the results here as originally it was thought that a month's worth of trading days would be sufficient for the network to derive a weekly forecast. Other window sizes did not yield any better results. The number of hidden units was also varied from 1 - 30 none of which lead to an improved performance. The results for the 5 day forecast are displayed in fig. 3.

Again, the random walk model is producing the lowest error performance. As with the one day forecasts, the networks seem to be implementing a random walk type of forecast which can be seen in fig. 4. Again, none of the networks have improved on the mean error performance of the random walk model. Also, looking at the graphs of the actual output of the networks (fig. 5) it can be seen that the networks forecasting 20 days in advance are simply outputting previous input.

Fig. 3 also gives the results of the 20 day forecasts in which a 60 trading day (3 month) input window was used. Fig. 5 displays a typical result from the networks. Clearly, the networks are simply giving the last seen input value (albeit somewhat degraded) as a forecast of the future course of the exchange rate. As with the 5 day forecasts, the number of hidden units was systematically varied. Due to long learning times, though, the input window size was pegged at a value of 60 trading days.

Copyright © 1995 Software Engineering Press



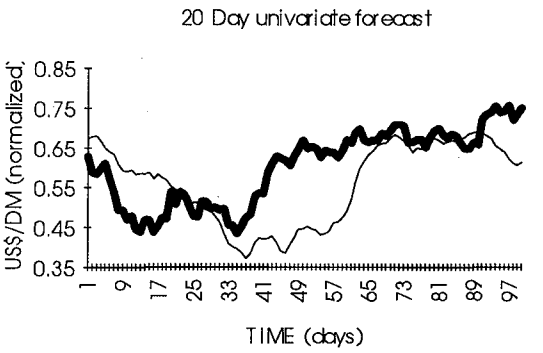


Fig. 5: 20 day random walk forecast and the univariate 20 day forecast.

Experiment 4: Multistep forecasting

An attempt was also made to look at the multistep predictive abilities of the networks used in experiment 1. Initially the results looked surprisingly promising given the previous results. Fig. 6 shows the results of the single day multistep forecast made over the same part of the data set as was used before. The network does seem to have forecast three of the major turning points. Attempts to replicate this on other parts of the data set were unsuccessful though. Generally, these other attempts produced results such as can be seen in fig. 7 which displays a single day multistep forecast using 550 days training staring after the first 200 days and the next 50 days as a test set. Apparently, the networks are modelling the data as being cyclical in nature whose dynamics are largely determined by the in the previous input although what exactly the networks are modelling in the data is unclear. All that can be said, though, is that the initial "positive" results in fig. 6 were probably spurious. It should also be noted that a similar result was found on the 20 day forecasting where a single positive result could not be replicated on other parts of the data set. Nonetheless, these results underscore the need for more care to be taken when analysing results to ensure they are not spurious.

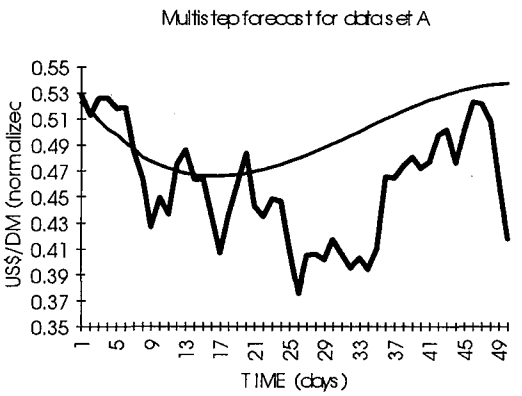


Fig. 6: Multistep forecast using first 850 days as training.

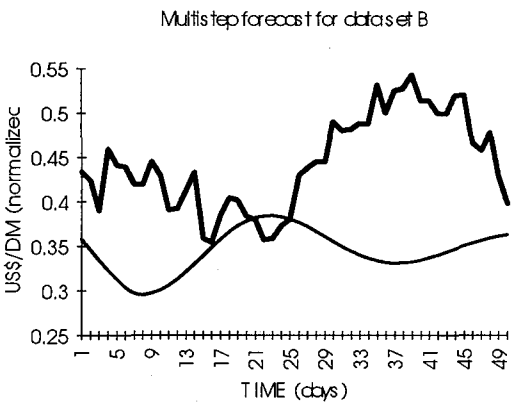


Fig. 7: Multistep forecast using days 200 - 750 as training.

Experiment 5: Multivariate forecasting

Copyright © 1995 Software Engineering Press

In this experiment, the data used to forecast the US$ - DM one and five days in advance was expanded to include the US$ - DM exchange rate, the US$ - Brit. Pound exchange rate and the US$ - Yen exchange rate. In addition, one month and one year Eurocurrency interest rates for each of the above currencies were also given as input to the networks along with the one month and one year London Interbank interest rates.

For the single day forecasts the networks were given 10 days of each of the above variables. Thus the networks had a total of 130 input units with the hidden units being varied from 0 to 30. The networks were trained on the first 850 days of the data set and tested on the following 100. The results of the single day multivariate forecasts with a 5 hidden unit network are displayed in fig. 8.

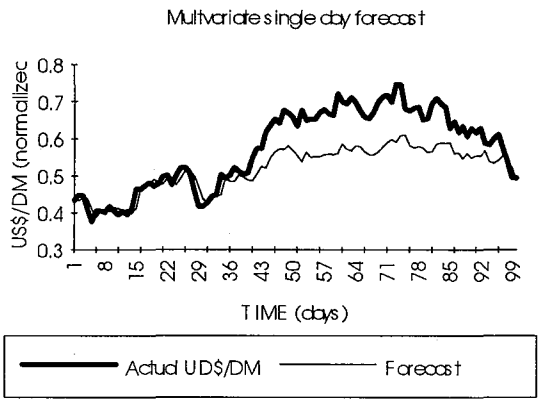


Fig:8: Multivariate single day forecast.

The results here are similar to the univariate 1 day forecasts. Increasing the number of hidden units did not lead to improved performance. As can be seen the networks have not been able to outperform the random walk model. Similar to the univariate one day forecasts, the networks here were very roughly approximating a random walk type performance.

For the 5 day multivariate forecasts the networks were given identical information as before except that a 20 day window of each variable was used giving a total of 260 inputs to the networks. The first 850 days were used for training and the following 100 days for testing. The results are displayed in fig. 9.

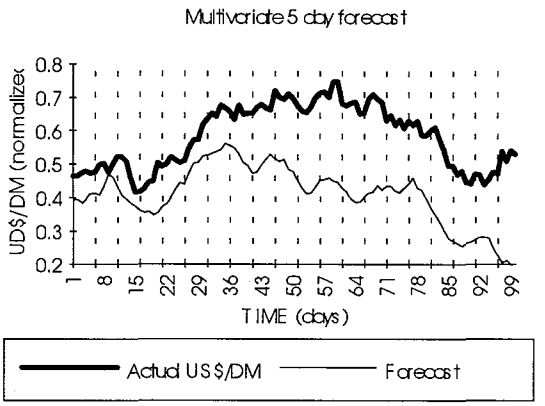


Fig. 9: 5 day multivariate forecast.

The best performance was found with the 20 hidden unit MLP although none of the networks used were able to outperform the random walk model in terms of mean squared error. Nonetheless, the network is not strictly outputting previous input and in fact does to be picking up some degree of the general direction of change (fig. 9). Similar results were found on other parts of the data set.

DISCUSSION

Although the US$ - DM price changes where shown to be not strictly random in a statistical sense, from a forecasting point of view what little structure actually is present may well be too negligible to be of much use. Given the performance of the network models in univariate portion of this study, the random walk model appears to be the more accurate for daily, weekly and monthly forecasting of the US$ - DM exchange rate. Does this mean that the random walk

Copyright © 1995 Software Engineering Press

model is the optimal technical forecasting technique for the US$ - DM exchange rate? Certainly not. However, in forecasting the US$ - DM exchange rate using only daily US$ - DM data, the random walk model has been the most effective of the models examined here at all three forecast lead times.

This has some implications regarding the work of Refenes *et al* (1993). Essentially, the only difference between their study of single day forecasting and the univariate experiments conducted here, is that they used hourly data. It seems reasonable to argue that if their model was truly robust, the use of daily data should have worked just as well. If anything, the hourly data would have been even more noisy than the daily data thus making forecasting even more difficult. Refenes *et al* also looked at multistep forecasting which was also attempted here. The initial positive results we found with multistep forecasting in this work turned out to be spurious. In future work, the use of hourly data will be investigated to see if the dynamics of hourly price changes are more conducive to univariate forecasting than daily changes.

The multivariate 1 day experiment was not much different than the univariate single day forecasting. Why this is so is unclear but perhaps the use of additional currency and interest rate data here was not sufficient to capture the dynamics of the daily price changes.

The 5 day forecast were a bit more interesting if only because the networks did seem to pick up on some of the general trend information. Future work will explore the use of a larger variety of input variables and network architectures.

CONCLUSIONS

The conclusions of this work are three fold: First, previous work claiming good forecasting performance of the US$ - DM using univariate hourly input to feedforward networks could not be replicated with daily data. This suggests that either the results of previous work require some reevaluation or that there is something special about the nature of hourly changes in the US$ - DM prices changes that can be exploited for forecasting that cannot be found in the daily changes. Second, more research could be conducted examining the use of additional types of input and architectures in neural network based financial market forecasting systems. Finally, given the lack of cross model comparisons in previous research, more work needs to be done examining the relative forecasting abilities of connectionist and more standard financial modelling techniques. An experimental framework incorporating cross comparisons between different forecasting models combined with multiple simulation runs is recommended in research claiming the superiority of neural networks in financial forecasting. This will ensure that the advantages of connectionist forecasting methods cannot be simply written off as data mining.

REFERENCES

Binks, D. L. and Allinson, N. M. (1991) "Financial data recognition and prediction using neural networks", *Artificial Neural Networks*, T. Kohonen, K. Makisara and J. Kangas (editors), Elsevier Science Publishers, B.V. (North-Holland), 1709 - 12.

Chatfield, C. (1975) *The analysis of time series: Theory and practice*, Chapman and Hall, London, pg. 25.

Collard, J. E. (1991) "A B-P commodity trader", *Advances in Neural Information Processing Systems III*, B. M. Spatz (editor), Morgan Kaufmann Publishers, San Mateo, CA., 551 - 56.

Diebold, F. X. and Nason, J. A. (1990) "Nonparametric exchange rate prediction?", *Journal of International Economies*, 28, 315 - 32.

Copyright © 1995 Software Engineering Press

Hutchinson, J. M. (1994) *A radial basis function approach to financial time series*, Masters Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Mass, USA.

Kendall, M. G. and Stuart, A. (1969) *The advanced theory of statistics*, Vol 3 Charles Griffin and Company Ltd., London, 355 - 6.

Kimoto, T., Asakawa, K., Yoda, M. and Takeoka, M. (1990) "Stock Market Prediction with Modular Neural Networks", *Procedings of the International Joint Conference on Neural Networks, San Diego, CA.*

Lee, C. H., Park, C. P. (1992) "Prediction of monthly transitions of the composite stock price index using recurrent back propagation", *Artificial Neural Networks 2*, I. Aleksander and J. Taylor (editors), Elsevier Science Publishers, B.V. (North-Holland), 1629 - 32.

Moore, G. H. and Wallis, A. (1943) "Time series significance tests based on signs of differences", *Journal of the American Statistical Association*, 32, 153, (cited from Kendall and Stuart (1969)).

Pindyck, R. S. and Rubinfeld, D. L. (1991) *Econometric Models and Economic Forecasts (third ed)*, McGraw - Hill, Inc., New York, pg. 441.

Refenes, A. N., Azema-Barac, M., Chen, L. and Karoussos, S. A. (1993) "Currency exchange rate prediction and neural network design strategies", *Neural Computing and Applications*, Vol. 1, 46 - 58.

Weigend, A. S., Huberman, B. A. and Rumelhart, D. E. (1992) "Predicting sunspots and exchange rates with connectionist networks", *Nonlinear Modelling and Forecasting, SFI Studies in the Sciences of Complexity*, M. Casdagli and S. Eubank (editors), Addison-Wesley, 395 - 432.

Copyright © 1995 Software Engineering Press

Trading S&P 500 Stock Index Futures Using a Neural Network

Jae Hwa Choi
Dankook University
Department of Management
Cheonan, Korea 330-714

Myung Kee Lee
Tong Yang Futures America
125 S. Wacker Dr. #1080
Chicago, IL 60606

Moon-Whoan Rhee
Towson State University
Department of Finance
Towson, MD 21204

Abstract

Neural networks have found an important niche in financial applications. We apply neural networks to Standard and Poor's (S&P) 500 stock index futures trading to predict the futures market behavior. The results through experiments with a commercial neural network software do support future use of neural networks in S&P 500 stock index futures trading.

1. Introduction

The value of applying a new modeling technique to the prediction of stock prices and thus using it for abnormal profits has long been hotly debated in finance. The efficient market hypothesis [2] argues that stock markets are so competitive that no one could beat the market systematically with publicly available information. Although this hypothesis is very compelling in theory and has in fact shaped the way the finance subject has been taught in classroom for years, a series of recent empirical studies employing sophisticated methods [7] evidence that stock prices can be predicted. In light of this new evidence, we re-investigate the possibility of making abnormal profits using a neural network, a new and superior technology capable of detecting regularities existing in historical financial data. In this paper, neural network models are developed for Standard & Poor's (S&P) 500 stock index futures contracts.

The reasons for the selection of the futures markets in general and of the S&P 500 stock index futures in particular are multi-fold. One is that the futures market requires a substantially smaller margin and as a result is conducive to larger profit making opportunities. Another reason is that futures exchanges have been known to play a price discovery role, which makes any apprehensible price regularities even less likely. In other words, one could make a very strong case against the efficient market hypothesis with evidence for arbitrage opportunities in futures markets. Lastly, stock index futures seems to be a good choice to start with among futures contracts considering a recent controversial issue of program trading. The S&P 500 index futures contract is the most popular stock index futures.

As conventional analytic techniques reach their limit in recognizing data patterns, financial firms and institutions find neural network techniques to solve this complex task. Neural networks have found an important niche in financial applications and have recently been applied to finance and investment domain issues [12]. To date, however, only one published work by [11] has attempted to aid traders in stock index futures trading. We are in the process of developing an intelligent futures trading system. Among other components of the system, neural network models of the system are discussed in this paper.

The rest of this paper is organized as follows. Section 2 introduces the fundamental concepts of neural networks. Section 3 describes the process of developing neural network models to predict the market behavior of S&P 500 stock index futures trading. Results of simulated trading with the

Copyright © 1995 Software Engineering Press

neural network model are summarized in Section 4. A brief discussion of the intelligent futures trading system is given in Section 5. Conclusions and future research issues are discussed in Section 6.

2. Neural Networks

A neural network is a knowledge induction technique in which knowledge is constructed from learning cases and represented over the multilayer network. The field of neural network is inspired by studies of the brain and nervous system. Many neural network algorithms have been developed for different applications [4, 6]. Although the algorithms and networks built from them differ both structurally and mathematically, neural networks are usually specified in terms of the (neuron) node characteristics, network topology, and a learning algorithm. For simplicity, a common three layer neural network is selected and discussed as a representative model in this section.

A neural network consists of a set of interconnected processing nodes. Each node, arranged in layers, is a computational unit that acts on input and produces a result. In addition to the input and output layer, one or more hidden layers are introduced to enhance the network's ability to model complex functions. Nodes are connected to the nodes in the preceding layer for input and the next layer for output. Each connection between nodes has an associated weight. Data enters the network through nodes in the input layer (called input nodes). Input nodes simply pass input data to nodes in the next layer. Nodes in the hidden and output layers receive all input and process them. Figure 1 shows a three layer network architecture and the behavior of a node.

Nodes in the hidden and output layer process their inputs in two steps. First, each node multiplies every input value by its weight, calculates the total of the products, and then passes the sum through a function to produce its output. For example, in Figure 1 a node j in the hidden layer, shown in the box, receives input values from input nodes, aggregates these values based on an activation function, n_j (t), and converts to the corresponding output value by a transfer function, Y_j(t). A commonly used nonlinear transfer function is the Sigmoid function, which generates an output value between 0 and 1. The activation and transfer functions are mathematically represented as follows:

$$n_j(t) = \sum_i w_{ij} x_i(t) + B_j \quad \text{and}$$

$$Y_j(t) = \frac{1}{1 + e^{-n_j(t)}},$$

where

n_j (t) = aggregate input of node j at time t,
x_i(t) = input value from node i at time t,
w_{ij} = weight for connection between node i and j,
B_j = bias of node j,
and Y_j (t) = output of node j at time t.

Network topology refers to the configuration of a neural network. The number of possible interconnecting and grouping nodes into layers is enormous. Although network topology and node behavior are independent, learning algorithms are often tied to specific network architectures. A number of different learning algorithms have been developed. Figure 1 is layered with feedforward connections from the input layer, the hidden layer, to the output layer. This neural network topology, called feedforward-backpropogation (abbreviated as backpropogation), is the most important and most widely used algorithm [3, 10]. Feedforward-backpropagation stands for output feedforward and error backpropogation. A neural network learns through this error backpropagation. The key to a neural network is its learning algorithm.

The neural network is trained with sample cases. Sample cases are presented repeatedly and errors are corrected by adjusting the weights after each erroneous output. In a backpropagation neural network, the output

Copyright © 1995 Software Engineering Press

layer errors are determined by subtracting the actual result from the target result. Then, the derivatives of the output layer errors are passed back to the hidden layer. After each node in the output layer and the hidden layer finds its error value, the node adjusts its weights to reduce its error. The weights are adjusted after the presentation of each case. The goal of minimizing the sum of the network's squared errors is achieved by applying the gradient descent method that minimizes the mean squared error of the system by moving down the gradient of the error curve. The error surface is multidimensional and may contain many local minimas the backpropogation algorithm may not escape. Practical training of backpropogation neural network involves with finding a set of weights that process data accurately enough for the given application rather than finding a global minimum of the error curve. The process of developing neural networks consists of trying several configurations to see which has the least error. Backpropagation neural networks are trained by selecting training parameters, including the learning rate and momentum, to adjust connection weights in the learning process so that the sum of the squared errors can be reduced. Learning rate and momentum are coefficients that determine the portions of the current and previous discrepancies between actual outputs and desired outputs that are to be compensated (0 for no compensation and 1 for full compensation), respectively. High learning rate means that the adjustment of the weights is primarily determined by the current discrepancy. High momentum means that the adjustment of the connection weights is primarily determined by the previous discrepancy. Learning is complete when no further reduction in the sum of squared errors is possible. The resulting neural network can be used as a decision tool when new cases occur.

3. A Neural Network Model for S&P Stock Index Futures Trading

3.1 Selection of Variables

Inputs to the network are price information, technical and statistics derived from price information, and one subjective market indicator. Price information includes Open, High, Low, Close price data. Statistical indicators include Moving Average (MA), Rate Of Change (ROC), and Relative Strength Index (RSI) which are derived from the past price information for the one or two week period prior to the trading day. We also include the Market Breakdown (MB) which classifies the trading day's market into one of three categories. A limited selection of the variables makes possible to compare the performance of our model against others which use a similar set of variables.

Neural networks typically work with inputs in the range 0 to 1 or -1 to +1. When input data are loaded into a neural network, it must be scaled into a numeric range that is comparable with the neural network algorithm. In this study, the input data scaling is performed via the linear scaling function which converts a range of values into [-1, 1]. The neural network then produces an output of value between 0 and 1.

To validate the learning model, the four year data of 1,013 trading days is divided into a learning (original) and test (holdout) sample. The data for the year 1991 through 1993 is used as a learning sample to construct neural network models. The test sample, the data for the year 1994, is used for validation of the neural network.

3.2 Configuring Network

For this complex and noisy problem an experiment with three-layer backpropagation networks is performed to identify the network architecture. The Sigmoid logistic function described earlier is used as the transfer function for each node since it is known that this function is particularly effective when the outputs are categories [9]. For a three-layer network, it is suggested to start with the number of hidden neurons by computing the following formula in [9]:

Copyright © 1995 Software Engineering Press

# of Hidden neurons = 1/2 (Inputs + Outputs) + Sqrt (# of Patterns).

After experimenting with different numbers of hidden neurons, a three-layer network with 8 input nodes, 8 hidden nodes and one output node was selected. Figure 2 shows the topology of the neural network used in the study.

3.3 Learning

Learning consists of presenting the learning data set to the network so that the weights can be adjusted to produce the desired output for each input data set. Weights are adjusted by the backpropogation algorithm after each input vector is presented. Typically, a large number of iterations of the learning data are required to produce a stable set of weights that can properly categorize the learning sample. In this study, the neural network was trained over 760 trading days from 1991 to 1993 for S&P500 futures market.

The proper setting of the learning parameters is part of the art of neural networks. Learning ceases to make any progress if the learning rate and momentum are too high or the network has too few hidden nodes. A neural network continually works to improve the learning model's categorization of the learning sample inputs. Generally, learning improvement is continuous, but eventually there will be a point where the forward progress is too slow to be practical or observable. The experiment examined each neural network architecture with different values for these parameters and stopped with the learning rate and momentum factors of 0.1 and 0.1 respectively.

3.4 Testing the Model

Once the neural network is trained by the learning sample, the learned weights on the connections between nodes are kept constant during the testing phase. The neural network is tested with 240 trading events of the year 1994. The predicted outcome of each trading day has been examined to find a cutting point for classifying either Long(buy) or Short(sell). The final decision rule for the output is set to 0.57 and can be stated as:

output unit $> 0.57 \rightarrow$ Long(buy)
output unit $\leq 0.57 \rightarrow$ Short(sell).

The accuracy of the trained network came out to be 62.5%, which made 158 right out of 253 trading decisions in the test data set at 0.57 cut-off point. However, the network generated 58.5% of accuracy at 0.50 cut-off point. With the learning data set the trained network has 63.8% of accuracy (485 out of 760 trading decisions) at 0.57 cut-off point. At 0.50 cut-off point it generated 61.2%.

4. Simulated Trading Results

A simulated futures trading is designed such that the buy order is issued if the closing price of each trading day is expected to be greater than the open price. And the opposite is the case when the relationships of the two prices are reversed. Trading each day is set to take place in the morning immediately after observing the open price. The ex post near optimized network generated the simulation performance as in Panel A of Exhibit 1. The cut-off value of 0.57 was chosen for ex ante simulation. The trained neural network model has been checked over the test period which covers from January 3, 1994 through December 30, 1994.

Exhibit 2 shows the long/short positions for the network for 253 trading days in 1994. The top box indicates the time to buy and its holding period and the bottom box indicates the time to sell and its holding period. Exhibit 3 shows the ex ante trading performance resulting from these positions. The profit step function shows strong upward slope very consistently.

Exhibit 4 shows the trading simulation performance based on tested

Copyright © 1995 Software Engineering Press

decisions for 1994 S&P500 futures market. It was assumed that order placement be executed at market closing time and the slippage will be 1 tick (5 points) for each transaction. The commission was assumed to be $5.50 per side. As indicated in Exhibit 4, its ex ante performance shows a cumulative profit of $63,308, maximum drawdown of $7,375, and the reward/risk ratio was 8.5. For the seventy-eight days that trading took place, there were gains on fifty-two days, or 67.5% of the time. The average gain was $2,027.5 and the average loss was $1,653.2.

The neural network model performance has been compared with the perfect hindsight information, and five-day and ten-day moving average system. Panel B of Exhibit 2 shows the performance comparison.

5. The Intelligent Futures Trading System

Financial markets are interrelated in increasingly complex ways and operating 24 hours a day throughout the world. Telecommunications and computer networks tie together markets in the form of electronic entities. Financial practitioners are inundated with an ever larger stream of data, produced by the rise of sophisticated database technologies, on the rising number of market instruments. To cope with this information explosion, intelligent systems with quantitative analyses are considered to be the best tool for financial professionals and traders.

The information sources for financial professionals and traders include current market data, historical information, financial reports, bond analyses, quantitative models, technical indicators, etc. The intelligent system should employ techniques to integrate various information from different sources and provide quick recommendation to the decision maker. Each information source may be a subsystem of the intelligent system. The subsystem may be a conventional program or a modeling system. The best way to connect and integrate individual systems in the intelligent system is through the knowledge-based systems approach.

To provide such an intelligent system for the futures trading market, we are in the process of developing an intelligent futures trading system. The system integrates market data with modeling and analytical tools to support trader strategies. Modeling and analytical tools include simple analytical techniques, statistical models and neural network models. The integration will be implemented through the rule-based expert system approach with the object technology. Neural network models of the system are discussed in the current paper.

6. Conclusions and Future Research

This article describes the development and performance of neural networks in trading S&P 500 stock index futures contracts. This model outperforms conventional technical trading systems such as oscillator of moving average systems, which was also one of the input in the neural network. As a point of reference, the best network in [1] produced a gain of $10,301 over a year of trading, and the best network in [11] produced an annualized gain of $60,000 per contract. However, our best network achieved a gain of $63,308 over 1994 trading period per contract.

The current work has some limitations and lays ground for future extensions. More extensive input data including fundamental information are to be used to examine the possibility of performance improvements, to show how much improvement can be made with other variables, and to find out major dominant information affecting the stock index futures market. The robustness of the performance of our model could also be checked by splitting up the data sample into many different training and testing periods. For example, a moving simulation approach [5] is performed at various lengths of periods. This is the approach requiring multiple

Copyright © 1995 Software Engineering Press

repetitive simulations of learning and prediction exercises as time advances. Training and testing periods can even be switched to assess the degree of the stationarity of the index futures prices.

More realistic assumption and strategies of trading simulation should be devised for testing system performance and designing real-time trading systems. We plan to look into the possibility of a pseudo-arbitrage opportunity in the stock index futures. It is well known in the finance literature that arbitrage profits are possible when the actual stock index futures prices differ from the so-called cost-of-carry fair prices by more than transactions costs. We could feed in the fair prices to the neural network system and conduct this experiment.

7. References

[1] J.E. Collard, "Commodity Trading with a Three Years Old," in *Neural Networks in Finance and Investment: Using Artificial Intelligence to Improve Real-World Performance*, Trippi, R., and E. Turban (eds.), Chicago: Probus Publishing Co. 1993

[2] Fama, E. F., "Efficient Capital Markets: A Review of Theory and Empirical Work," *Journal of Finance*, 25, 1970, pp.383-417

[3] Gallant, S. I., *Neural Network Learning and Expert Systems*, MIT Press: 1993, Chapter 11.

[4] Hush, D. R. and B. G. Horne, Progress in Supervised Neural Networks: What's New Since Lippmann, *IEEE Signal Processing Magazine* (January 1993), pp.8-39

[5] Kimoto T., K. Asakawa, M., Yoda, and M. Takeoka, "Stock Market Prediction System with Modular Neural Networks," in *Neural Networks in Finance and Investment: Using Artificial Intelligence to Improve Real-World Performance*, Trippi, R., and E. Turban (eds.), Chicago: Probus Publishing Co., 1990

[6] Lippman, R. P., An Introduction to Computing with Neural Nets, *IEEE Acoustics, Speech and Signal Processing Magazine* (April 1987), pp.4-22

[7] Lo, A., and A. MacKinlay, "Stock Prices Do Not Follow Random Walks: Evidence From a Simple Specification Test," *Review of Financial Studies*, 1988

[8] Lo, A., and A. MacKinlay, "When Are Contrarian Profits Due To Stock Market Overreaction?," *Review of Financial Studies*, 3, 1990, pp.175-205

[9] *NeuroShell 2 User's Manual*, Ward Systems Group, Inc., 1993

[10] Rumelhart, D. E., B. Widrow, and M. A. Lehr, The Basic Ideas in Neural Networks, *Communications of the ACM* (March 1994), pp.87-92

[11] Trippi, R., and D. DeSieno, "Trading Equity Index Futures with a Neural Network," *The Journal of Portfolio Management*, 1992, pp.27-33

[12] Trippi, R., and E. Turban (eds.), *Neural Networks in Finance and Investment: Using Artificial Intelligence to Improve Real-World Performance*, Chicago: Probus Publishing Co., 1993

Copyright © 1995 Software Engineering Press

Exhibit 1

Panel A : Performance of trained Network for 0.50 and 0.57 of cut-off point

Cut-off Point	Cumulative Profit	% Gain	Max. Drawdown	Profit/ Risk	Average Gain	Average Loss
0.50	40558	69	2003	20.25	432.30	245.07
0.57	49280	72	1943	22.79	395.97	259.08

Panel B : Performance Comparison

TYPE	Profit	% Gain	Max. Drawdown	Profit/ Risk	Average Gain	Average Loss
Perfect Information	58088	98	0.0	infinite	433.41	0.00
Neural Network	12831	78	1475.0	4.9	405.50	330.64
Moving Average	-12008	34	12153.0	0.98	288.69	657.17

Figure 1 : A Neural Network

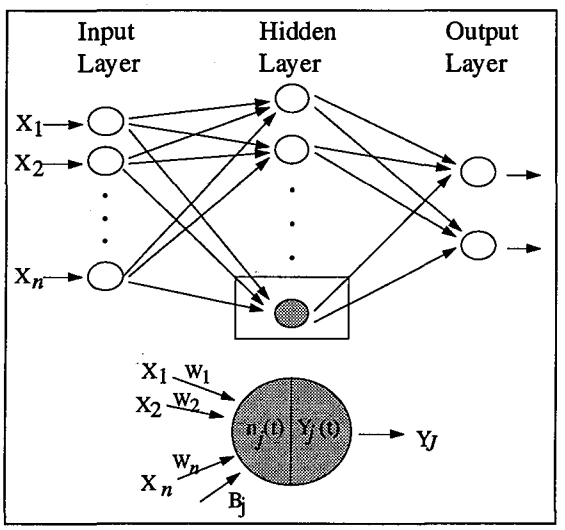
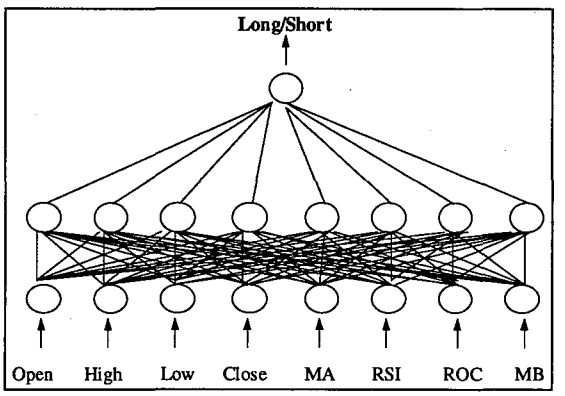
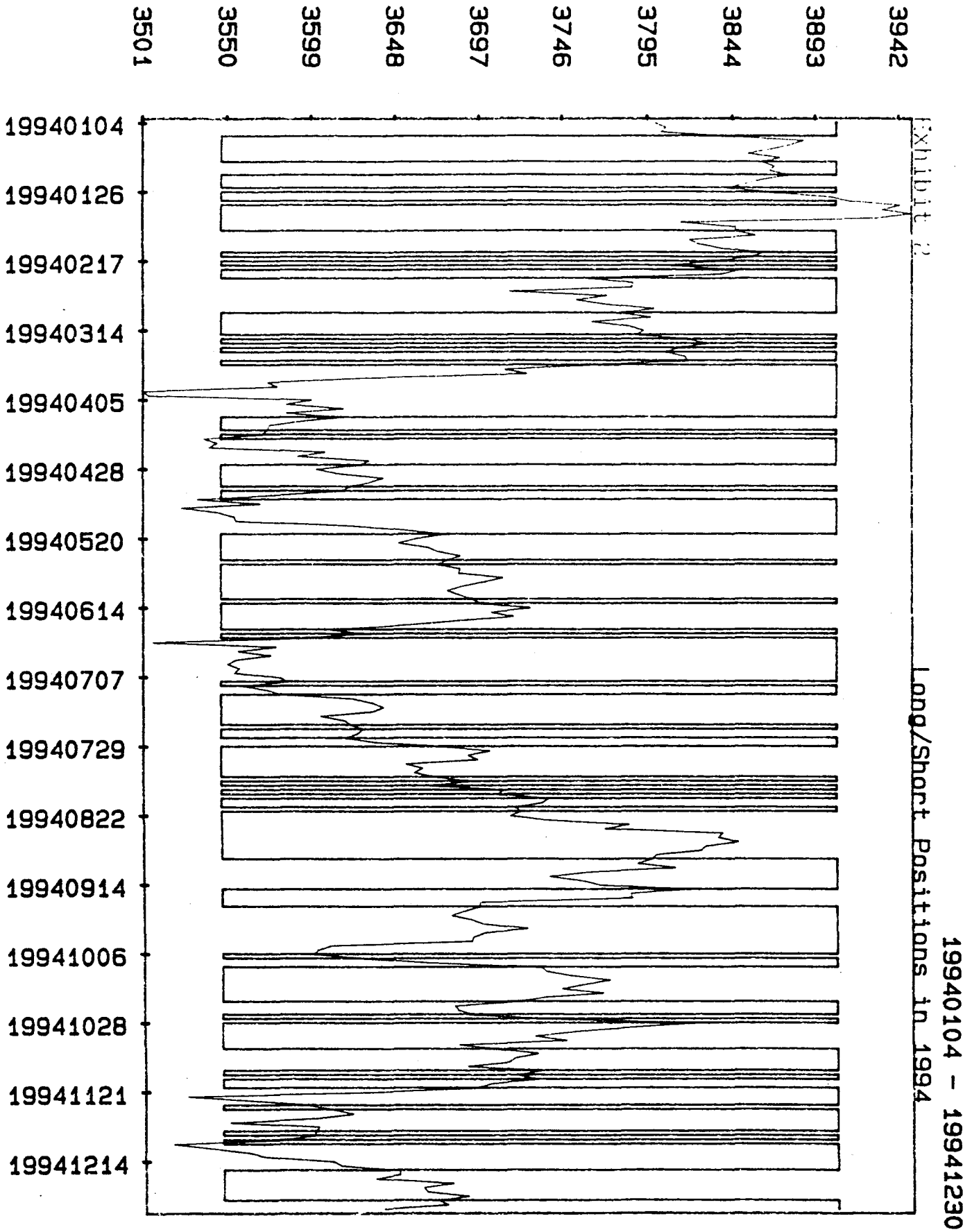


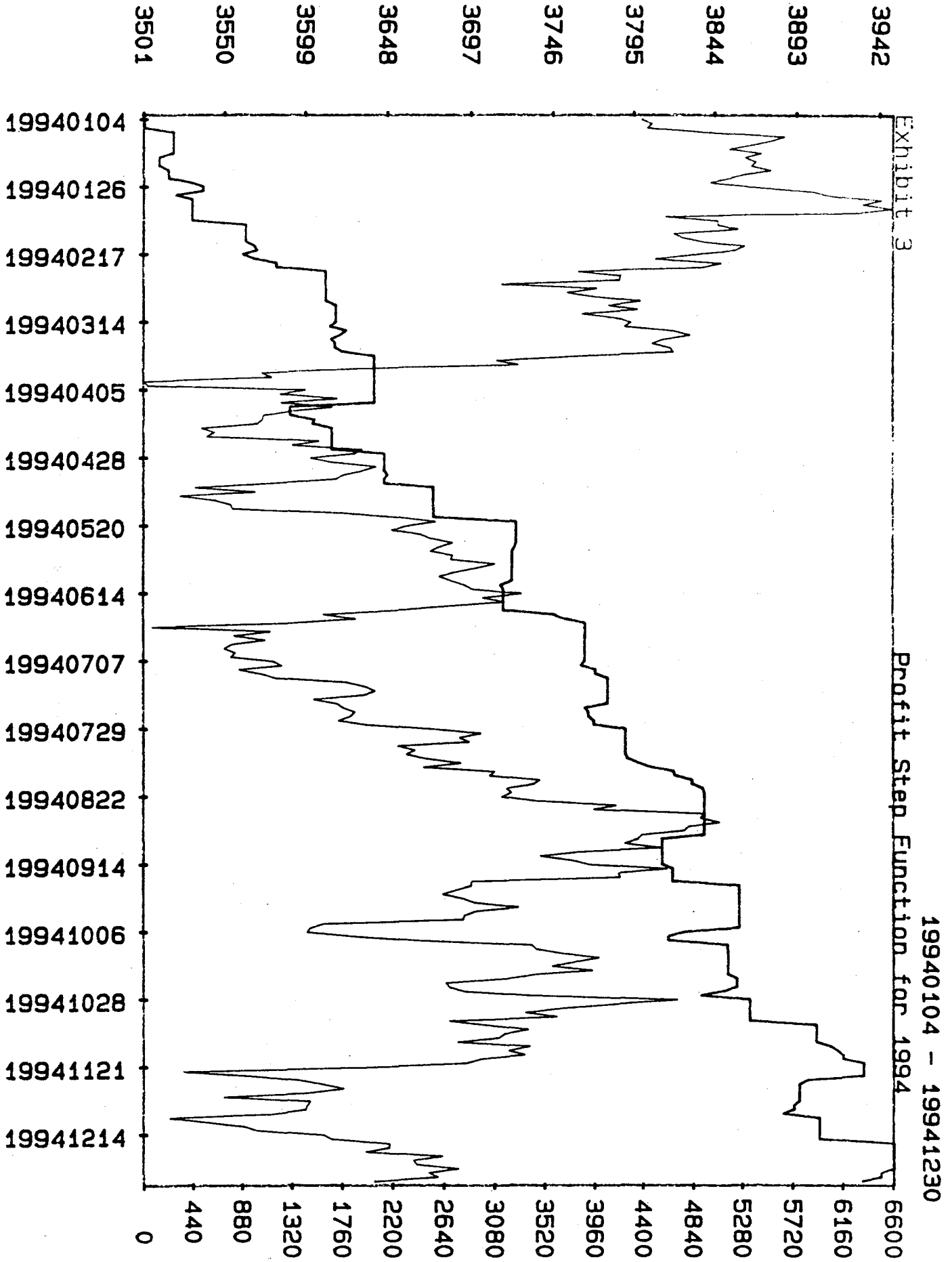
Figure 2 : A Neural Network for S&P Stock Index Futures



Copyright © 1995 Software Engineering Press



Copyright © 1995 Software Engineering Press



Copyright © 1995 Software Engineering Press

S&P neural network review program
sp_input

19940103 - 19941230

Based on close price.
Buy/Sell cutoff at 0.5[illegible]000.
Includes 1 tic of slippage.

Date	Transaction	Price	Profit	Cum.profit	Commsn	Dollars
19940103	Open	37859	0.00	0.00	$ 5.50	$ -5.50
19940107	Sell	38379	520.00	520.00	$11.00	$ 2583.50
19940117	Buy	38624	-245.00	275.00	$11.00	$ 1347.50
19940120	Sell	38789	165.00	440.00	$11.00	$ 2161.50
19940125	Buy	38414	375.00	815.00	$11.00	$ 4025.50
19940126	Sell	38654	240.00	1055.00	$11.00	$ 5214.50
19940128	Buy	39139	-485.00	570.00	$11.00	$ 2778.50
19940131	Sell	39433	294.00	864.00	$11.00	$ 4237.50
19940208	Buy	38464	969.00	1833.00	$11.00	$ 9071.50
19940215	Sell	38624	160.00	1993.00	$11.00	$ 9860.50
19940216	Buy	38568	56.00	2049.00	$11.00	$10129.50
19940217	Sell	38304	-264.00	1785.00	$11.00	$ 8798.50
19940218	Buy	38089	215.00	2000.00	$11.00	$ 9862.50
19940222	Sell	38489	400.00	2400.00	$11.00	$11851.50
19940224	Buy	37618	871.00	3271.00	$11.00	$16195.50
19940308	Sell	37794	176.00	3447.00	$11.00	$17064.50
19940315	Buy	37899	-105.00	3342.00	$11.00	$16528.50
19940316	Sell	38194	295.00	3637.00	$11.00	$17992.50
19940317	Buy	38299	-105.00	3532.00	$11.00	$17456.50
19940318	Sell	38139	-160.00	3372.00	$11.00	$16645.50
19940321	Buy	38069	70.00	3442.00	$11.00	$16984.50
19940323	Sell	38199	130.00	3572.00	$11.00	$17623.50
19940324	Buy	37634	565.00	4137.00	$11.00	$20437.50
19940411	Sell	36159	-1475.00	2662.00	$11.00	$13051.50
19940414	Buy	35734	425.00	3087.00	$11.00	$15165.50
19940415	Sell	35709	-25.00	3062.00	$11.00	$15029.50
19940418	Buy	35369	340.00	3402.00	$11.00	$16718.50
19940426	Sell	36289	920.00	4322.00	$11.00	$21307.50
19940504	Buy	36229	60.00	4382.00	$11.00	$21596.50
19940505	Sell	36174	-55.00	4327.00	$11.00	$21310.50
19940509	Buy	35324	850.00	5177.00	$11.00	$25549.50
19940519	Sell	36764	1440.00	6617.00	$11.00	$32738.50
19940527	Buy	36784	-20.00	6597.00	$11.00	$32627.50
19940531	Sell	36739	-45.00	6552.00	$11.00	$32391.50
19940610	Buy	36939	-200.00	6352.00	$11.00	$31380.50
19940613	Sell	36989	50.00	6402.00	$11.00	$31619.50
19940621	Buy	36109	880.00	7282.00	$11.00	$36008.50
19940622	Sell	36299	190.00	7472.00	$11.00	$36947.50
19940623	Buy	35924	375.00	7847.00	$11.00	$38811.50
19940708	Sell	35859	-65.00	7782.00	$11.00	$38475.50
19940711	Buy	35599	260.00	8042.00	$11.00	$39764.50
19940713	Sell	35824	225.00	8267.00	$11.00	$40878.50
19940722	Buy	36229	-405.00	7862.00	$11.00	$38842.50
19940725	Sell	36294	65.00	7927.00	$11.00	$39156.50
19940727	Buy	36194	100.00	8027.00	$11.00	$39645.50
19940729	Sell	36759	565.00	8592.00	$11.00	$42459.50
19940809	Buy	36719	40.00	8632.00	$11.00	$42648.50
19940810	Sell	36919	200.00	8832.00	$11.00	$43637.50
19940811	Buy	36699	220.00	9052.00	$11.00	$44726.50
19940812	Sell	37114	415.00	9467.00	$11.00	$46790.50
19940815	Buy	37089	25.00	9492.00	$11.00	$46904.50
19940816	Sell	37389	300.00	9792.00	$11.00	$48393.50
19940818	Buy	37189	200.00	9992.00	$11.00	$49382.50
19940819	Sell	37219	30.00	10022.00	$11.00	$49521.50
19940906	Buy	37970	-751.00	9271.00	$11.00	$45755.50
19940915	Sell	38154	184.00	9455.00	$11.00	$46664.50
19940921	Buy	36970	1184.00	10639.00	$11.00	$52573.50
19941006	Sell	36000	-970.00	9669.00	$11.00	$47712.50
19941007	Buy	36280	-280.00	9389.00	$11.00	$46301.50
19941011	Sell	37335	1055.00	10444.00	$11.00	$51565.50
19941021	Buy	37174	161.00	10605.00	$11.00	$52359.50
19941026	Sell	36939	-235.00	10370.00	$11.00	$51173.50
19941027	Buy	37339	-400.00	9970.00	$11.00	$49162.50
19941028	Sell	38215	876.00	10846.00	$11.00	$53531.50
19941107	Buy	37060	1155.00	12001.00	$11.00	$59295.50
19941114	Sell	37320	260.00	12261.00	$11.00	$60584.50
19941115	Buy	37190	130.00	12391.00	$11.00	$61223.50
19941116	Sell	37290	100.00	12491.00	$11.00	$61712.50
19941118	Buy	36940	350.00	12841.00	$11.00	$63451.50
19941125	Sell	35960	-980.00	11861.00	$11.00	$58540.50
19941128	Buy	36100	-140.00	11721.00	$11.00	$57829.50
19941205	Sell	36005	-95.00	11626.00	$11.00	$57343.50
19941206	Buy	35995	10.00	11636.00	$11.00	$57382.50
19941207	Sell	35805	-190.00	11446.00	$11.00	$56421.50
19941208	Buy	35175	630.00	12076.00	$11.00	$59560.50
19941216	Sell	36490	1315.00	13391.00	$11.00	$66124.50
19941228	Buy	36725	-235.00	13156.00	$11.00	$64938.50
19941230	Close	36400	-336.00	12831.00	$ 5.50	$63308.00

Accum. Profit	Draw Down	Maximum Dip	Prof/ Dip	Maximum Exp	Num Trans	%gain/ %trans	TGain/ TLoss	TGain/ #Gain	Tloss/ #Loss	Maximum Gain
12831.00	-1475.00	-2617.00	-4.90	-2617.00	78	0.67	-2.55	405.50	-330.64	1440.00

Copyright © 1995 Software Engineering Press

Paper Session: Expert Systems and Hybrid Approaches

Chair: Stephen Slade New York University

A Multi-Component Approach to Stock Market Predictions

Tim Chenoweth[1,2,3] Zoran Obradović[1]
[1] School of Electrical Engineering and Computer Science
[2] Department of Management and Systems
[3] Department of Economics
Washington State University, Pullman WA 99164-2752

Abstract

The multi-component system proposed in this paper is comprised of a preprocessing component, two neural networks, and a decision rule base. First, the preprocessing component determines the most relevant features for stock market prediction. Next, the two neural networks predict the market's rate of return, with one network trained to recognize large positive and the other large negative returns. Finally, the decision rule base takes the return prediction and determines a buy/sell recommendation. Various experiments using this system to predict S&P 500 index returns were conducted and performance measured by computing the annual rate of return and the return per trade. Comparing the results achieved by themulti-network system to that of the single neural network shows that in general the multi-network system gives a higher return with fewer trades. In addition, some multi-network experiments managed to achieve an annual rate of return greater than that of the buy and hold strategy.

1 Introduction

In general, most quantitative methods that attempt to predict stock market movements are based on statistical time series models [1, 8, 10]. These paradigms are largely unsuccessful due to the inherent complexity of financial markets in general and the stock market in particular. The efficient market hypotheses says that stock prices adjust to new information very rapidly, usually by the time the information becomes public knowledge, making it impossible for statistical paradigms based on this information to make accurate predictions [7].

While the efficient market hypotheses seems to be correct for static and linear relationships between stock prices and historical information, it is possible that dynamic or nonlinear relationships exist that traditional statistical time series methods are incapable of modeling [7]. If this is true, it may be possible to capture these relationships using a nonparametric machine learning approach of multilayer artificial neural networks (NN). Such NN's are powerful computational systems that can approximate any nonlinear continuous function on a compact domain to any desired degree of accuracy [4]. In addition, a NN can account for fundamental changes in the underlying function through incremental retraining using the back-propagation learning algorithm [9].

This paper proposes a hybrid multi-component nonlinear system for S&P 500 stock market predictions. The system consists of statistical feature selection component for identification of the most relevant data, two specialized NN's for extraction of nonlinear relationships from the selected data, and high level decision rules for determining buy/sell recommendations. The system goals are to earn a higher annual return than the buy and hold strategy and to keep the number of trades low to reduce transaction costs. The system details are explained in Section 2 followed by results and analysis in Section 3, and conclusions in Section 4.

2 Methodology

2.1 Feature Selection

The objective of the feature selection component (see Fig. 1) is to identify a small subset of the most relevant features from a larger pool for designing the system in a manner that preserves as much information as possible. This issue is important because fewer features per pattern lead to faster computation and require less training patterns for successful generalization.

For feature selection, the stock market prediction problem is considered to be a two-class problem with one class corresponding to a positive move in the S&P 500 index and the other corresponding to a negative move [3]. The feature selection process performs a number of feature selection techniques utilizing various selection criteria. For each technique and criteria combination the top s features

This work was supported in part by the National Science Foundation under grant IRI-9308523 to Z. Obradović.

Copyright © 1995 Software Engineering Press

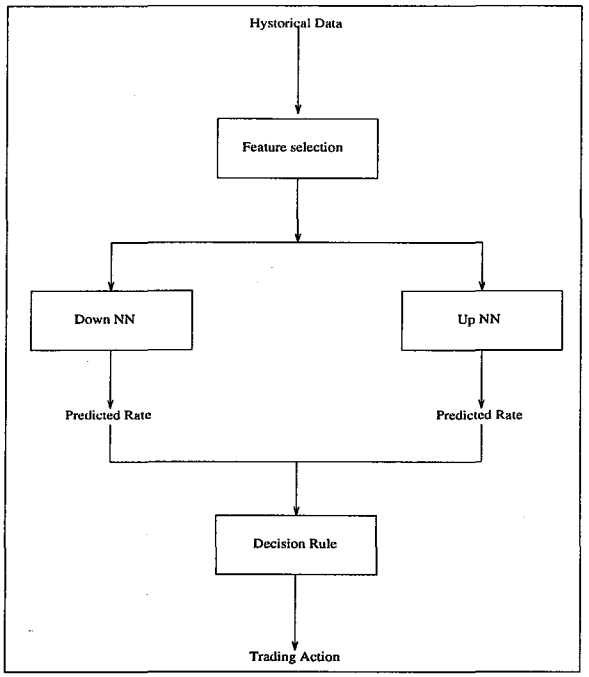


Figure 1: System Architecture

Technique	*Criteria*
Best Feature	Mahalanobis Distance
Best Feature	Estimated Minimal Error
Best Feature	Bhattacharyya Distance
Best Feature	Patrick-Fisher
Best Feature	Euclidean Distance
Best Feature	Univariate Chebychev
Forward Search	Mahalanobis Distance
Forward Search	Estimated Minimal Error
Forward Search	Patrick-Fisher
Forward Search	Bhattacharyya Distance
Backward Search	Mahalanobis Distance
Backward Search	Estimated Minimal Error
Backward Search	Patrick-Fisher
Backward Search	Bhattacharyya Distance

Table 1: Feature Selection Techniques and Criteria

are determined and points awarded based on feature importance (i.e., s points for the most informative feature, $s-1$ points for the second most informative feature, etc.). Then the final scores for the features are analyzed and a determination is made concerning which features to include in the final set. Selection techniques and criteria used in this process are shown in Table 1, with each row corresponding to a technique/criteria combination. Techniques and criteria are explained in [6].

2.2 Return Rate Prediction

The return rate prediction component, shown in Fig. 1, consists of two NN's that are trained using the back-propagation algorithm and an on-line learning scheme. The objective is to train the "up NN" on patterns with a large positive target return and the "down NN" on patterns with a large negative target return. Once both NN's are trained, the test pattern is presented to each of them and the corresponding predictions are collected. A decision rule base is applied to these predictions and a buy/sell recommendation made as explained in Section 2.3.

The on-line learning scheme consists of a sequence of training/prediction session where the NN's are retrained after each session using more resent information. This is achieved by training the NN's using patterns from a fixed size window covering a continuous time segment of historic data. The target return for the time unit immediately following the window is predicted by both NN's and the predictions used by the rule base. Then the training window is shifted forward one time unit (i.e., one trading day), the patterns from the new window used to retrain the NN's, and a prediction made for the next time unit. This process is repeated until the data set is exhausted.

For each training session the target return corresponding to each pattern in the window is compared to a threshold value h. If the return is greater than h the corresponding pattern is added to the "up NN" training set, if the return is less than $-h$ the pattern is added to the "down NN" training set. Any pattern with a target return between $-h$ and h is discarded.

For example, suppose that the training window size is m and that at time t the test pattern is p_t, which means that the training window contains patterns p_{t-m} through p_{t-1}. First, the patterns in the training window (p_{t-m} through p_{t-1}) are separated into "up NN" and "down NN" training sets using the threshold value h as described. Next, both NN's are trained using their respective training sets, and asked to predict the target return for the test pattern p_t. Once the predictions are collected and sent to the rule base, the training window is shifted forward one time unit so that the new test pattern is p_{t+1} and the new training window contains patterns p_{t-m+1} through p_t, and the process repeated. This continuous until the end of the ordered data set is reached.

Copyright © 1995 Software Engineering Press

2.3 Decision Rule Base

The predicted returns from both NN components are used as input to the decision rule base component (see Fig. 1). This component analyzes the predicted returns and outputs a buy/sell recommendation that is used to establish either a long or short position in the market. A long position means purchasing an asset for later resale, while a short position means selling a borrowed asset now and purchasing it later.

This study examines three different decision rule bases. For each rule base the "up NN" prediction r_u is compared to the "down NN" prediction r_d. Each rule base recommends a long position in the market if $r_u > 0$ and $r_d \geq 0$, and a short position if $r_u \leq 0$ and $r_d < 0$. Otherwise the rule base computes the normalized difference *diff* as

$$diff = \frac{\max\{r_u, |r_d|\} - \min\{r_u, |r_d|\}}{\max\{r_u, |r_d|\}},$$

compares this ratio to a predefined threshold value y, and determines a buy/sell recommendation as follows:

- **Rule Base 1: Maintain Current Position Until a Clear Buy/Sell Recommendation is Received.**
  This rule base specifies that if the system is unsure as to what recommendation to make, the action is to do nothing and maintain the old position. Under these rules, if $r_u \leq 0$ and $r_d \geq 0$ the system recommends maintaining the current position (i.e., do nothing). If $r_u > 0$, $r_d < 0$, and $diff > y$ the rule base recommends a long position if $r_u > |r_d|$, and a short position if $r_u < |r_d|$. Otherwise $diff \leq y$ and the recommendation is to maintain the current market position.
- **Rule Base 2: Stay Out of the Market Unless a Clear Buy/Sell Recommendation is Received.**
  The difference between rule base one and rule base two is the action taken when the system is uncertain as to what recommendation to make. In case of uncertainty, the rule base two action is to exit the market. More precisely, if $r_u \leq 0$ and $r_d \geq 0$ the system recommends exiting the market (i.e., if the current position is long then sell, if it is short then buy). If $r_u > 0$, $r_d < 0$, and $diff > y$ the system recommends a long position if $r_u > |r_d|$, and a short position if $r_u < |r_d|$. Otherwise $diff \leq y$ the recommendation is to exit the market.
- **Rule Base 3: Hold a Long Position in the Market Unless a Clear Sell Recommendation is Received.**
  This decision rule base takes advantage of the common *a priori* knowledge that over the past 65 years the market has increased at an average annual rate greater than 10%. Stated another way, this means that given no other information the odds are that the market will increase. This is, in fact, the whole premise behind the buy and hold strategy. Again, the difference between rule base three and the previous rules is the actions taken under uncertainty. In this instance the action is to take a long position in the market. Under these rules, if $r_u > 0$, $r_d < 0$, $diff > y$, and $r_u < |r_d|$, the system recommends a short position. Otherwise the recommendation is to take a long position.

2.4 Performance Measures

The most important criteria when measuring the performance of a stock market prediction model is whether it will make money and how much. Therefore the model's annual rate of return (ARR) is computed as follows

$$ARR = \frac{k}{n}\sum_{i=1}^{n} r_i,$$

where:

n is the total number of trading time units for the experiment;
k is the number of trading time units per year (i.e., 253 for daily trading);
r_i is the rate of return for time unit i.

The sum, $\sum_{i=1}^{n} r_i$, is computed by either adding, subtracting, or discarding the actual daily returns for the S&P 500 index. If the system recommends a long position, the actual return is added to the sum; if a short position is recommended, the return is subtracted; or if the recommendation is to exit the market, the return is discarded.

It is also important to minimize transaction costs by controlling excessive trading (i.e., a 10% return with 50 trades is more profitable than a 10% return with 100 trades). Therefore the break even transaction cost ($BETC$), which may be viewed as the return per trade, is computed as follows:

$$BETC = \frac{1}{m}\sum_{i=1}^{n} r_i,$$

where m is the total number of trading transactions, while r_i and n are defined as previously. A trade is defined as any action that changes a market position. For example, exiting the market constitutes a single trade (i.e., a buy trade to cover a

Copyright © 1995 Software Engineering Press

S&P 500 index return
S&P 500 index return lagged one day
S&P 500 index return lagged two days
U.S Treasure Rate lagged 2 months
U.S Treasure Rate lagged 3 months
30 Year Government Bond Rate

Table 2: Selected Features

short position or a sell trade to cover a long position), while switching from a short position to a long position constitutes two trades (i.e., one buy trade to cover the short position and another buy to establish the long position).

3 Results and Analysis

The system described in Section 2 is used for S&P 500 stock market buy/sell recommendations. The historic data used in this experiment is ordered daily financial time series patterns from the period January 1, 1985 to December 31, 1993. Patterns from January 1, 1985 to December 31, 1988 comprised the initial training window, whereas actual predictions were made for patterns from January 1, 1989 to December 31, 1993. Each pattern in the initial data set contained 24 monthly and 8 daily features. The feature selection process described in Section 2 (with $s = 7$) showed the 6 features with the highest scores clustered together with a significant drop between the sixth and the seventh feature. Based on this, the feature set was reduced from the original 32 features to the 6 features listed in Table 2.

The single NN system trained with patterns composed of the six features from the reduced set obtained an *ARR* and a *BETC* of 2.86% and 0.01% respectively, using 957 trades. For comparison, the *ARR* and *BETC* for a single NN trained using all 32 of the original features are -2.16% and -0.01% respectively using 905 trades, which justifies the feature selection process. The system parameters for both single NN's are shown in Table 3.

Parameter	*Value*
Activation Function	Tangent Hyperbolic
Network Topology	6-4-1
Network Topology (Single NN, All Features)	32-4-1
Learning Rate	0.03
Tolerance	0.00001
Number of Iterations	5000
Training Window Size Size (Multi NN)	1000
Training Window Size Size (Single NN)	250

Table 3: System Parameter Values

Several experiments with the multi-network system described in Section 2 were conducted using the reduced feature set from Table 2 and the system parameters from Table 3. Note that the training window size for the multi-network experiments is larger than for the single NN. All patterns in the training window for the single NN are used in the training process, while the training window for the multi-network is split into 3 disjoint sets. The first set, consisting of all patterns with a target rate greater than h, is used to train the "up NN". The second set, consisting of all patterns with a target rate less than -h, is used to train the "down NN." Finally, the third set, consisting of all patterns with a target return between -h and h, is discarded. Consequently, to ensure an adequately sized training set for both NN components in the multi-network system, it is necessary to have a larger window size.

For the multi-network system, experiments are conducted varying the thresholds h and y. Threshold h is varied from 0.5% to 1.25% in increments of 0.25% and y from 0 to 0.80 in increments of 0.05. For decision rule bases one, two, and three the experiments using the fixed values of h resulting in the largest *ARR* are shown in Tables 4, 5, and 6 respectively. The best annual rate of return was 13.35%

y	*ARR*	Trades	*BETC*
0	-6.75%	330	-0.10%
0.05	-5.70%	290	-0.10%
0.10	-1.24%	238	-0.02%
0.15	-0.01%	198	-0.00%
0.20	2.76%	178	0.08%
0.25	1.61%	156	0.05%
0.30	1.61%	136	0.10%
0.35	-1.17%	108	-0.05%
0.40	4.95%	88	0.28%
0.45	6.56%	64	0.52%
0.50	7.65%	48	0.80%
0.55	4.98%	40	0.62%
0.60	5.70%	36	0.79%
0.65	5.17%	28	0.93%
0.70	10.09%	20	2.54%
0.75	10.09%	20	2.54%
0.80	**10.37%**	**16**	**3.26%**

Table 4: Returns for Rule Base 1 with Threshold h equal to 0.75%

Copyright © 1995 Software Engineering Press

y	*ARR*	Trades	*BETC*
0	5.40%	391	0.07%
0.05	**6.50%**	**391**	**0.08%**
0.10	2.49%	385	0.03%
0.15	0.17%	383	0.00%
0.20	1.75%	381	0.02%
0.25	2.31%	355	0.03%
0.30	3.23%	319	0.05%
0.35	3.14%	295	0.05%
0.40	1.13%	275	0.02%
0.45	0.88%	243	0.02%
0.50	-1.17%	217	-0.03%
0.55	-2.58%	203	-0.06%
0.60	-2.25%	210	-0.05%
0.65	-2.54%	188	-0.07%
0.70	-2.43%	180	-0.07%
0.75	-1.39%	164	-0.04%
0.80	-2.23%	150	-0.08%

Table 5: Returns for Rule Base 2 with Threshold h equal to 1.0%

y	*ARR*	Trades	*BETC*
0	-6.35%	404	-0.08%
0.05	-10.31%	400	-0.13%
0.10	-6.93%	404	-0.08%
0.15	-8.12%	406	-0.10%
0.20	-0.69%	370	-0.01%
0.25	1.95%	370	0.03%
0.30	6.41%	326	0.10%
0.35	6.46%	248	0.11%
0.40	6.59%	282	0.12%
0.45	5.36%	250	0.11%
0.50	7.98%	246	0.16%
0.55	8.67%	218	0.20%
0.60	8.79%	194	0.23%
0.65	9.61%	166	0.29%
0.70	11.34%	162	0.35%
0.75	12.08%	134	0.45%
0.80	**13.35%**	**126**	**0.53%**

Table 6: Returns for Rule Base 3 with Threshold h equal to 0.5%

and was obtained using rule base three with thresholds $h = 0.5\%$ and $y = 0.80$. In comparison, the annual rate of return for the buy and hold strategy was 11.23% and the best return for the single NN was only 2.86%.

4 Conclusions and Future Research

The system proposed in this paper is comprised of a preprocessing component for feature selection, two NN components that use the selected features for return predictions, and a decision rule component that takes the return predictions and determines a buy/sell recommendation. Various experiments using this system to predict S&P 500 index movements were conducted and associated annual rates of return and returns per transaction computed.

By comparing the results achieved by the multi-network system to that of the single NN it can be observed that in general the multi-network system gives a higher return with fewer trades. In addition, some multi-network experiments managed to achieve an annual rate of return greater than that of the buy and hold strategy.

Although these preliminary results are promising, research in progress might lead to further improvements. For instance, the current feature pool is quite limited. As a next research step this feature pool will be considerably extended by incorporating additional features such as daily trading volume and inter-day index highs and lows. It is likely that more informative features will be selected from a larger pool, possibly leading to improved results. In addition, in this study no attempt was made to optimize all the system parameters. It is possible that optimized learning parameters such as the learning rate and the number of hidden units may lead to better results. Further improvements may be obtained by incorporating prior knowledge and constructive NN learning [5], or a recurrent network topology [2]. Finally, the current rule bases are fairly simplistic. Possible improvements include incorporating technical information like moving averages and exponential averages into the system. It may also be possible to use another NN, an expert system, or some combination of the two to analyze the existing system information and determine a market direction.

References

[1] Black, F. and Scholes, M., (May-June 1973) "The pricing of Options and Corporate Liabilities," *Journal of Political Economy, Vol. 81.*

[2] Burgess, A. and Bunn, D., (1994) "The Use of Error Feedback Terms in Neural Network Modelling of Financial Time Series," *Proceedings of the 1994 Neural Networks in the Capital Markets Conference*, Pasadena, CA.

[3] Chenoweth, T. and Obradovic, Z., (1994) "Feature Selection for Predictive Models of the

Copyright © 1995 Software Engineering Press

Stock Market," *Proceedings of the 1994 Neural Networks in the Capital Markets Conference*, Pasadena, CA.

[4] Cybenko, G., (1989) "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems,* Vol. 2, pp. 303-314.

[5] Fletcher, J. and Obradovic, Z., (1993) "Combining Prior Symbolic Knowledge and Constructive Neural Networks," *Connection Science: Journal of Neural Computing, Artificial Intelligence and Cognitive Research,* Vol. 5, no. 3 & 4, pp. 365-375.

[6] Fukunaga, K., (1990) *Introduction to Statistical Pattern Recognition,* Academic Press, San Diego, CA.

[7] Hutchinson, J., (1993) *A Radial Basis Function Approach to Financial Time Series Analysis,* PhD Thesis, Massachusetts Institute of Technology.

[8] Markowitz, J., (1959) *Portfolio Selection: Efficient Diversification of Investments*, John Wiley & Sons, New York.

[9] Rumelhart, etc. (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vols. 1 and 2,* MIT Press, Cambridge, MA.

[10] Sharpe, W., (September 1964) "Capital Asset Prices: A Theory of Market Equilibrium," *Journal of Finance.*

Tim Chenoweth (tchenowe@eecs.wsu.edu) received a B.S. degree in Mathematics in 1981 from the Coast Guard Academy, a M.B.A in Finance from Washington State University in 1991, and is currently completing a M.S. in Computer Science and an Individual Interdisciplinary Ph.D. combining Business and Computer Science, both from Washington State University. He was an active duty officer in the Coast Guard from 1981 to 1989. The objective of his current research is to use advanced technologies to model financial markets.

Zoran Obradovic (zoran@eecs.wsu.edu) received the B.S. degree in Applied Mathematics, Information and Computer Sciences in 1985; the M.S. degree in Mathematics and Computer Science in 1987, both from the University of Belgrade; and the Ph.D. degree in Computer Science from the Pennsylvania State University in 1991. He was a systems programmer at the Department for Computer Design at the Vinca Institute, Belgrade, from 1984 to 1986, and has been a research scientist at the Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, since then. At present, he is an Assistant Professor in the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, USA. The objective of his current research is to explore applicability of neural networks technology to large scale classification and time series prediction problems in very noisy domains.

Copyright © 1995 Software Engineering Press

Intelligent Model Discovery for Financial Time Series Prediction Using Non-Linear Dynamical Systems Theory and Statistical Methods

Oscar Castillo
Instituto Tecnologico de Tijuana
P.O. Box 4207 Chula Vista CA
91909-4207 USA

Patricia Melin
CETYS Universidad Tijuana
P.O. Box 4207 Chula Vista CA
91909-4207 USA

Abstract

We describe a computer program that can be considered an intelligent system for the domain of financial time series prediction. The computer program is an implementation of a new algorithm for discovering mathematical models for financial time series prediction, combining artificial intelligence methodology with Dynamical Systems Theory, Fractal Theory and Statistical methods. Given a financial time series for an specific problem, the intelligent system develops mathematical models for the problem based on the geometry of the data, using three different approaches. First, the computer program develops regression models for the time series using traditional statistical methods, then the program develops non-linear mathematical models based on Dynamical Systems Theory and Chaos Theory, and finally the program develops fractal mathematical models based on the theory of Fractal Geometry. The Intelligent System then analyzes all of the mathematical models obtained before to make a selection of the model that will give us the "best" prediction for the financial time series. This selection is done by the intelligent system using a combination of heuristics and calculations that are contained in the knowledge base. An Intelligent System that can learn models from financial data would be very useful in practice in making the task of prediction more easy and less time consuming.

1. Introduction

We describe a new algorithm (called IDIMM, for Intelligent Discovery of Mathematical Models) for financial time series prediction combining artificial intelligence methodology with Dynamical Systems Theory, Fractal Theory and Statistical methods. The idea of using Dynamical Systems Theory and Fractal Theory as alternative approaches for prediction can be justified if we consider that traditional statistical methods only have limited success in real world financial applications, and this is mainly because financial problems show very complicated dynamics in time. Traditional statistical methods assume that the erratic behavior of a time series is mainly due to a external random error (that cannot be explained). However, a Dynamical Systems approach, using non-linear mathematical models, can explain this erratic behavior because "chaos" as intrinsic part of this type of models. It is a well known fact from Dynamical Systems [4] that even very simple non-linear mathematical models can exhibit the behavior known as "chaos" for certain parameter values, and therefore are good candidates to use as equations for prediction. Fractal Theory also offers a way to explain the erratic behavior of a time series, but the method is geometrical in the sense that the fractal dimension is used to describe the complexity of the distribution of the data points.

We describe a prototype implementation of the algorithm IDIMM as a computer program written in the programming language PROLOG. This computer program can be considered an intelligent system for the domain of financial time series prediction. Given a financial time series the intelligent system develops mathematical models based on the geometry of the data. The mathematical models are constructed using three different approaches: Dynamical Systems Theory, Fractal Theory and traditional Statistical Methods. First the computer program develops regression models for the time series using traditional

Copyright © 1995 Software Engineering Press

statistical methods, then the program develops non-linear mathematical models based on Dynamical Systems Theory and Chaos Theory, and finally the program develops fractal mathematical models based on the theory of Fractal Geometry.

Traditional methods for model discovery succeed when the relationships to be discovered are easy, that is, when we have small data sets with standard structure and variables of one type [7]. However, many real-life financial problems are more complicated than this and cannot always be modeled by traditional techniques. This is the main reason why we think that Artificial Intelligence techniques can help in making the task of developing mathematical models more efficient and accurate. The main idea is that an intelligent system can use heuristics to limit the combinatorially explosive search space of possible mathematical models for a given financial problem. Also the intelligent system can be flexible enough to discover models of varying precision and comprehensibility, depending on the user's problem-specific goals.

The intelligent system develops only the kind of mathematical models that are more likely to give a "good" prediction based on the knowledge that human experts have about this matter. This knowledge is contained in the knowledge base of the intelligent system, and is the main factor in limiting the number of models that the system explores. The intelligent system also has some generalized knowledge about the mathematical models that we expect to discover in the financial domain. This knowledge is expressed as families of parametrized mathematical models. At the end the intelligent system analyzes all the mathematical models obtained in the first part, to make a decision about which one is the "best" model for the given problem. This decision is done using a combination of statistical calculations and heuristics from human experts about this matter. In order to developed our intelligent system we needed to the knowledge extraction from human financial experts. The knowledge acquisition was done by one of the authors while working in Economics Research Department of a University in Mexico with financial and economical experts.

2. Discovering Mathematical Models for Time Series Prediction

The problem of discovering mathematical models from a given time series can be defined as follows:

Given: A data set (time series) with n data points, $D = \{d_1, d_2, ..., d_n\}$. Each data point d has a real-valued and continuos "response" (or dependent) attribute "y", and P "predictor" (or independent) attributes $\mathbf{X} = (X_1, X_2, ..., X_p)$, where one or more of the Xi's can be the time t.

Goal: From the data set D, develop a mathematical model Mb, that is the "best" model to predict "y".

The above problem is not a simple one, because there exists an infinite number of mathematical models that can be build for a given data set. So the problem lies in knowing which models to try for a data set and then to select the "best" one. More formally we can state the problem in the following way:

Let M be the infinite dimensional space of mathematical models defined for a given data set D. Let $MS = \{M_1, ..., M_q\}$ be the set of selected models that are considered to be appropriate for the geometry of the data set D. Let M_b be a model in MS that is considered the "best" one for prediction for the corresponding time series. We consider mathematical models of the following form, for the statistical methods:

$$Y = F(\mathbf{X}) + \varepsilon(0,\sigma)$$

where $\varepsilon(0,\sigma)$ represents a 0-mean Gaussian noise-process with standard deviation σ, this is the random error. $F(\mathbf{X})$ is a polynomial equation in $\mathbf{X}$, where the predictor variables are contained in the vector: $\mathbf{X} = (X_1, X_2, ..., X_p)$.

We consider mathematical models as "dynamical systems" of the following form:

$$dY/dt = F(Y)$$

where Y is a vector of variables of the form: $Y = (Y_1, Y_2, ..., Y_p)$ and F(Y) is a non-linear function of Y. Note that in this case we have deterministic models expressed as differential equations. Other kind of mathematical models are the discrete "dynamical systems" of the following form: $Y_t = F(X)$

Copyright © 1995 Software Engineering Press

where X, in this case, is of the form:
$X = (Y_{t-1}, Y_{t-2}, ..., Y_{t-p})$ and F(X) is a non-linear function of X. Note that in this case we have deterministic models expressed as discrete difference equations.

The mathematical models for the statistical methods can be linear as well as non-linear equations. We show below some sample statistical models that the intelligent system explores:

linear_regression: $Y_t = a + bt$
quadratic_regression: $Y_t = a + bt + ct^2$
logarithmic_regression: $\ln Y_t = a + b\ln t$
first_order_autoregression: $Y_t = a + bY_{t-1}$

The mathematical models for continuous dynamical systems can be one-dimension, two-dimensional or three-dimensional. We show below some sample models that the intelligent system explores:

logistic_differential_equation:

$$dY_1/dt = aY_1(1-Y_1)$$

lotka_volterra_two_dimensional:

$$dY_1/dt = aY_1 - bY_1Y_2$$
$$dY_2/dt = bY_1Y_2 - cY_2$$

lotka_volterra_three_dimensional:

$$dY_1/dt = Y_1(1 - Y_1 - aY_2 - bY_3)$$
$$dY_2/dt = Y_2(1 - bY_1 - Y_2 - aY_3)$$
$$dY_3/dt = Y_3(1 - aY_1 - bY_2 - Y_3)$$

lorenz_three_dimensional:

$$dY_1/dt = aY_2 - aY_1$$
$$dY_2/dt = - Y_1Y_3 + bY_1 - Y_2$$
$$dY_3/dt = Y_1Y_2 - cY_3$$

The mathematical models for discrete dynamical systems can also be one, two, or three dimensional. We show below some sample models that the intelligent system explores:

logistic_difference_equation:

$$Y_{t+1} = aY_t(1-Y_t)$$

logistic_two_dimensional_difference_equation:

$$Y_{t+1} = X_t$$
$$X_{t+1} = aX_t(1-X_t)$$

lotka_volterra_two_dimensional:

$$Y_{t+1} = aY_t - bY_tX_t$$
$$X_{t+1} = bY_tX_t - cX_t$$

henon_map_two_dimensional:

$$Y_{t+1} = X_t$$
$$X_{t+1} = a - X_t^2 + bY_t$$

In all of the above mathematical models a, b and c are parameters that need to be estimated using the corresponding numerical methods. For example, for the regression models we can use the least squares method for parameter estimation, but for the differential equations we need to use the Gauss-Newton method.

The algorithm for discovering the best mathematical model for prediction can be stated as follows:

1.- Read the data set $D = \{d_1, d_2, ..., d_n\}$.
2.- Analyze the data set D to find the components of the time series.
3.- Find the set of selected models:
$$MS = \{M1, ..., Mq\}$$
using the properties of the components of the time series. To complete this task the knowledge base of the intelligent system makes the decision of what models have to be developed. For each model do the following:
a) Determine the parameters of the models based on the methods corresponding to the type of equation.
b) Create the corresponding equation F.
c) Calculate the measures of "goodness" of the model.
4.- Find the "best" mathematical model Mb from the set Ms using the measures of "goodness" of each of the models of the set Ms. To complete this task the knowledge base of the intelligent system makes the decision based on the heuristics of the experts incorporated in the computer program.

Copyright © 1995 Software Engineering Press

We call this algorithm IDIMM (for Intelligent Discovery of Mathematical Models) and is an integration of Artificial Intelligence techniques with Dynamical Systems Theory, Fractal Theory, and statistical methods, to obtain mathematical models for prediction of time series in the financial domain.

3. Description of the Intelligent System

3.1 Architecture of the Intelligent System

In figure 1 we describe the general architecture of the system.

In figure 2 we describe the architecture of "Expert Module 2", which does the selection of the mathematical models that the intelligent system will explore. This module selects the type of statistical models more appropriate for the data, then selects the dynamical systems more appropriate for the data, and finally selects the fractal theory models (fractal dimensions) to describe the data. After this selection is finish, this expert module calls the Numerical Module to do the parameter estimation for each corresponding mathematical model.

3.2 Description of the Knowledge base of the Intelligent System

The knowledge base consists of three parts corresponding respectively to the three expert modules shown in figure 1. The first part contains the knowledge to analyze the time series, i.e., the knowledge to obtain from the data the components of the time series. The second part contains the knowledge to select the kind of mathematical models more appropriate for the type of data given, i.e., given the components of the time series decide which models are more likely to give a good prediction. The third part contains the knowledge to select the "best" mathematical model for prediction.

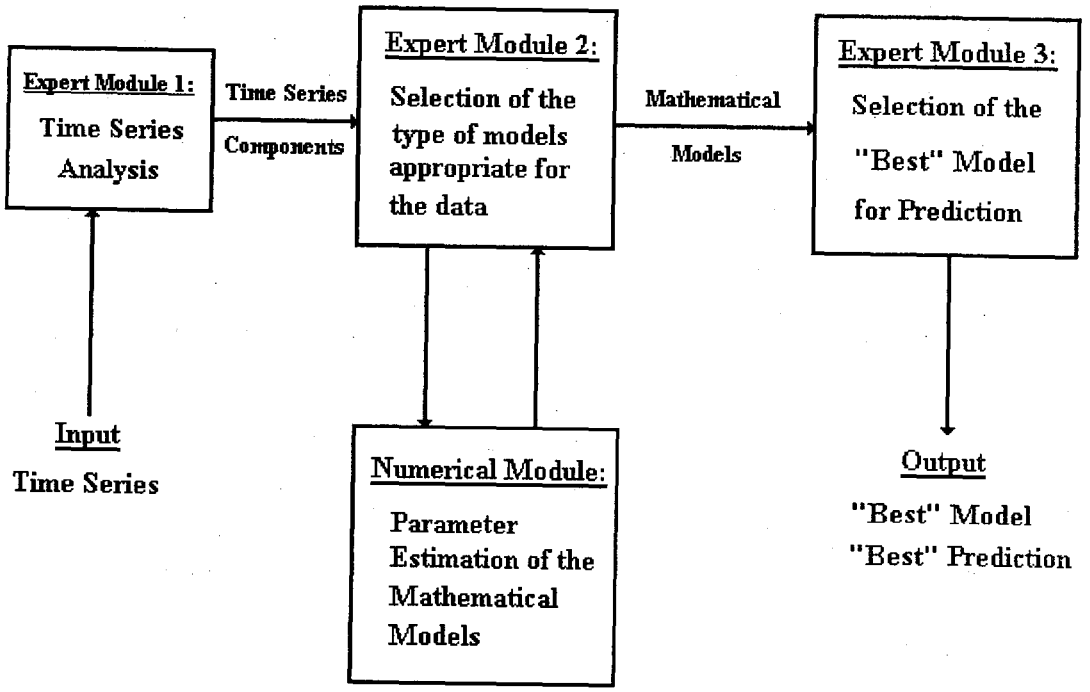


Figure 1.- Architecture of the Intelligent System

Copyright © 1995 Software Engineering Press

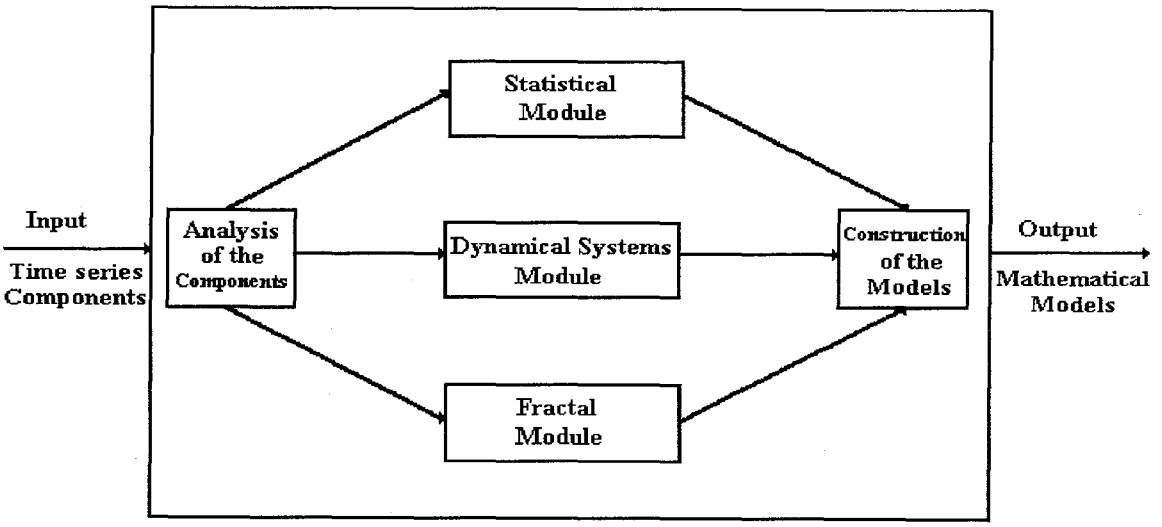


Figure 2.- Architecture of Expert Module 2

i.e., given the models of the second part, decide which one is the "best" to predict the time series.

To give an idea of the way the knowledge base is structured, we show some sample rules of "Expert Module 2". Remember that this module selects the type of mathematical models more appropriate for the data, using as input the components of the time series extracted in module 1. We consider the following types of statistical methods to obtain the models [5]:

1.- Linear regression
2.- Quadratic regression
3.- Logarithmic regression
4.- Trigonometric Least-Squares
5.- First order autoregression
6.- Second order autoregression
7.- Non-linear autoregression
8.- Brown method
9.- Weighted exponential moving averages

We consider the following types of non-linear mathematical models as dynamical systems [4]:

1.- Logistic differential equations
2.- Logistic difference equations
3.- Lotka Volterra two-dimensional differential equations
4.- Lotka Volterra two-dimensional difference equations
5.- Lorenz three-dimensional differential equations
6.- Henon two-dimensional difference equations

We consider the following fractal models (dimensions) as measures of the complexity of the time series [4]:

1.- Correlation dimension
2.- Box dimension

We show in figure 3 some sample rules for deciding, using the properties of the time series, which of the above statistical methods is more likely to give a good mathematical model for the problem.

We show in figure 4 some sample rules for selecting which of the above dynamical systems is more likely to give us a good mathematical model for the given time series.

We also give in figure 5 some sample rules of "Expert Module 3". This is the part of the knowledge base that decides which is the "best" model for prediction using as input the set of models generated by Module 2. In this figure we

Copyright © 1995 Software Engineering Press

Rule 1:
IF Time_series = smooth
THEN Type_Method = regression

Rule 2:
IF Time_series = cyclic
THEN Type_Method = complex

Rule 3:
IF Type_Method = regression
AND Tendency = linear
THEN Type_Model = linear_regression

Rule 4:
IF Type_Method = regression
AND Tendency = non_linear
THEN Type_Model = quadratic_regression

Rule 5:
IF Type_Method = regression
AND Tendency = exponential
THEN Type_Model = logarithmic_regression

Rule 6:
IF Type_Method = complex
AND Seasonal_Part = simple
THEN Type_Model = brown

Rule 7:
IF Type_Method = complex
AND Seasonal_Part = simple
AND Explanation = ad_hoc
THEN Type_Model = trigonometric_least_squares

Rule 8:
IF Type_Method = complex
AND Seasonal_Part = regular
THEN Type_Model = pmep

Rule 9:
IF Type_Method = complex
AND Seasonal_Part = difficult
THEN Type_Model = autoregression

Figure 3.- Sample rules for the selection of the statistical mathematical models

Copyright © 1995 Software Engineering Press

Rule 1:
IF Time_series = smooth_all **THEN** Type_Method = continuous_one_dim

Rule 2:
IF Time_series = cyclic_part **THEN** Type_Method = discrete_one_dim

Rule 3:
IF Time_series = cyclic_all **THEN** Type_Method = continuous_two_dim

Rule 4:
IF Time_series = cyclic_chaotic **AND** Number_variables = three
THEN Type_Method = continuous_three_dim

Rule 5:
IF Time_series = cyclic_chaotic **AND** Number_variables = two
THEN Type_Method = discrete_two_dim

Rule 6:
IF Time_series = cyclic_chaotic **AND** Number_variables = one
THEN Type_Method = discrete_one_dim

Rule 7:
IF Type_Method = continuous_one_dim **AND** Tendency = non_linear
THEN Type_Model = logistic_differential_equation

Rule 8:
IF Type_Method = discrete_one_dim **AND** Tendency = non_linear
AND Seasonal_part = simple **THEN** Type_Model = logistic_difference_equation

Rule 9:
IF Type_Method = continuous_two_dim **AND** Tendency = non_linear
AND Seasonal_part = simple **THEN** Type_Model = lotka_volterra_differential_eq

Rule 10:
IF Type_Method = discrete_two_dim **AND** Tendency = non_linear
AND Seasonal_part = simple **THEN** Type_Model = logistic_delay_difference_eq

Rule 11:
IF Type_Method = discrete_two_dim **AND** Tendency = non_linear
AND Seasonal_part = regular **THEN** Type_Model = lotka_volterra_difference_eq

Rule 12:
IF Type_Method = continuous_three_dim **AND** Tendency = non_linear
AND Seasonal_part = regular **THEN** Type_Model = lorenz_differential_equation

Figure 4.- Sample rules for the selection of the dynamical systems mathematical models

Copyright © 1995 Software Engineering Press

Rule 1:

IF $r_2^2 > r_1^2$ AND $t_{b2} \geq t_\alpha$ AND $t_{c2} \geq t_\alpha$
THEN Result = M_2

Rule 2:

IF $r_1^2 > r_2^2$ AND $t_{b1} \geq t_\alpha$
THEN Result = M_1

Rule 3:

IF $r_3^2 > r_2^2$ AND $t_{b3} \geq t_\alpha$
THEN Result = M_3

Where the mathematical models and the statistics are:

M_1: $Y = a_1 + b_1 t$

r_1^2 = coefficient of regression of M_1
$t_{b1} = b_1/s_{b1}$ = t student value of b_1

M_2: $Y = a_2 + b_2 t + c_2 t^2$

r_2^2 = coefficient of regression of M_2
$t_{b2} = b_2/s_{b2}$ = t student value of b_2
$t_{c2} = c_2/s_{c2}$ = t student value of c_2

M3: $\ln Y = \ln a_3 + b_3 \ln t$

r_3^2 = coefficient of regression of M_3
$t_{b3} = b_3/s_{b3}$ = t student value of b_3
s = standard deviation
t_α = critical value of the t student distribution

Figure 5.- Sample rules of Expert Module 3 for deciding which is the best regression model

show rules to decide only between regression models, but Module 3 has many more rules to consider all the kinds of models mentioned above.

4. Use of the Intelligent System

We show in figure 6 a sample input/output of the use of Expert Module 2 to give an idea of the performance of the Intelligent System. In this figure we show the result of applying the new algorithm for discovery of the best model, for a particular example of a time series of oil prices for Mexico.

5. Comparison with Related Work

There has been some work recently in the area of numerical law discovery, but much of the research in Machine Learning is in other areas such as induction [8]. We think that this is mainly because "discovery" is a more difficult kind of "learning". However, we can state that automated mathematical modelling is very important for many domains of application for obvious reasons. For example in the engineering and financial domains is critical to obtain mathematical models for the problems, to be able to understand them and also to

Copyright © 1995 Software Engineering Press

IDIMM2: AN INTELLIGENT SYSTEM FOR DISCOVERING MATHEMATICAL MODELS FOR FINANCIAL TIME SERIES PREDICTION COMBINING DYNAMICAL SYSTEMS AND FRACTALS WITH STATISTICS

What is the name of the file containing the time series?
>oil_prices.txt
What is the general form of the graph of the time series?
(s) smooth
(c) cyclic
(d) don't know
>c

What is the tendency of the graph of the time series?
(l) linear
(n) non-linear
(e) exponential
(d) don't know
>n

Do you consider the seasonal part of the time series?
(s) simple
(r) regular
(f) difficult
(d) don't know
>f

What type of explanation do you want for the model?
(a) ad_hoc
(t) theoretical
(d) don't matter
>t
THE BEST MATHEMATICAL MODEL FOR PREDICTION CONSIDERING
THE CONDITIONS AND PROPERTIES GIVEN AS INPUT IS:
" Y(T) = 18.74 + 1.03Y(T-1) + 0.28Y(T-1)^2 "
THIS IS A FIRST ORDER DIFFERENCE EQUATION THAT CAN BE CONSIDERED A NON-LINEAR DYNAMICAL MODEL FOR THE GIVEN PROBLEM
Do you want an explanation of why this is considered to be the best mathematical model?
(y) yes
(n) no
>n
Do you want to make another consultation?
(y) yes
(n) no
>n

Figure 6.- Sample input/output of the use of Expert Module 2

Copyright © 1995 Software Engineering Press

be able to predict their future behavior. This is why we consider that more research in this area is very important.

Similar work with respect to Machine Learning can be found in a paper by Moulet [6], however the approach to model discovery is different to ours (this can be seen from the heuristic method by Moulet). Also in a paper by Rao [7] we can see a method for model discovery for engineering domains, but also with a different approach to ours (his approach is similar to "clustering"). Also, there is another very important difference with other authors, in the kind of mathematical models that we are considering for our intelligent system. We are considering non-linear mathematical models from the theory of Dynamical Systems and not only linear regression models like other authors. In this paper we have successfully generalized our previous work on this matter [2], by considering this type of non-linear models.

6. Conclusions

We have developed an intelligent system for the domain of financial time series prediction. The system discovers mathematical models for a given financial time series using a combination of techniques from AI, Dynamical Systems, Fractal Theory and Statistics. This intelligent system can be used to find the best mathematical model for a financial time series, and then the model can be used to predict future values of the time series. Accurate prediction is of great importance in the areas of finance, economics, management and accounting for obvious reasons. Our intelligent system can make more easy the job of mathematical modelling and prediction in all of these areas.

The intelligent system can be improved in the following ways:

1.- Build a better user interface so that it can be used more easily
2.- Provide a larger class of mathematical models that the system can explore to find the best model.

We are planning to work along this lines in the near future.

7. References

[1] Bratko, I., *Prolog Programming for Artificial Intelligence*, Addison Wesley, 1990.

[2] Castillo, O., Melin, P., "An Intelligent System for Discovering Mathematical Models for Financial Time Series Prediction" *Proceedings of the IEEE Region 10's Ninth Annual International Conference*, pp. 217-221, IEEE Singapore Section, August 1994.

[3] Covington, M.A., Nute, D., Vellino, A., *Prolog Programming in Depth*, Scott Foresman & Co. Computer Books, 1988.

[4] Devaney, R., *An Introduction to Chaotic Dynamical Systems*, Addison Wesley Publishing Company, 1989.

[5] Mendenhall, W., Reinmuth, J.E., *Statistics for Management and Economics*, Wadsworth International, 1981.

[6] Moulet, M., "A Symbolic Algorithm for Computing Coefficients Accuracy in Regression", *Proceedings of the Ninth International Workshop on Machine Learning*, pp. 332-337, Morgan Kauffman Publishers, San Mateo, CA, 1992.

[7] Rao, R.B., Lu, S., "A Knowledge-Base Equation Discovery System for Engineering Domains", *IEEE Expert*, pp. 37-42, August 1993.

[8] Sleeman, D., Edwards, P., Editors, *Proceedings of the Ninth International Workshop on Machine Learning*, Morgan Kauffman Publishers, San Mateo, CA, 1992.

[9] Sterling, L., Shapiro, E., *The Art of Prolog*, MIT Press Cambridge Mass., 1987.

Copyright © 1995 Software Engineering Press

Paper Session: Risk Management

Chair: Susan Garavaglia, Dun & Bradstreet Information Services

OPTIMAL MIXTURES OF CLASSIFIERS FOR FINANCIAL DISTRESS PREDICTION

Ignacio Olmeda*
and
Eugenio Fernández*,**

*Dpto. de Fundamentos de Economía e Historia Económica
**Dpto. de Matemáticas

Universidad de Alcalá
Alcalá de Henares 28802 Madrid SPAIN

Ph: 34-1-8854202
Fax: 34-1-8854239
e-mail: eholmeda@alcala.es - ehefv@alcala.es

Copyright © 1995 Software Engineering Press

OPTIMAL MIXTURES OF CLASSIFIERS FOR FINANCIAL DISTRESS PREDICTION

Ignacio Olmeda
Dpto. de Fundamentos de Economía e
Historia Económica.
Universidad de Alcalá
Alcalá de Henares 28802 (Madrid) - SPAIN

Eugenio Fernández
Dpto. de Fundamentos de Economía e
Historia Económica - Dpto. Matemáticas.
Universidad de Alcalá
Alcalá de Henares 28802 (Madrid) - SPAIN

Abstract

In this paper we propose a method for combining classifiers in an optimal way. We pose the problem of finding the optimal combination as an optimization problem (solved by an Evolutionary Programming algorithm) in which one desires to minimize the expected cost of misclassification. We show that the mixtures obtained are superior against any of its constituents on the problem of bankruptcy prediction.

1. Introduction.

Financial agents are increasingly interested in the use of Advanced Computing Technologies (ACT's) such as Artificial Neural Networks, Genetic Algorithms or Machine Learning, for modelling and forecasting purposes. The reason for this is quite obvious, if these "high-tech" tools were truly more powerful, the competitive advantage from using them would be decisive, at least until these technologies were used by any agent so that differential benefits were fully arbitraged. The number of reported successful applications of these technologies has been so high that a "folk-theorem" asserts their universality and superiority against any other procedures.

Comparisons on the forecasting accuracy of a particular ACT against alternatives in classification problems are relatively common in the literature. Most of these comparisons consider only a single competing model (for example a statistical one) and not the combination of two or more of them so that their appropriateness in a general forecasting context is not resolved. In this paper we propose to consider combinations of several methods and to formulate the choice of the optimal mixture as an optimization problem which can be solved by means of appropriate algorithms (such as Evolutionary Programming). We will also show that although a particular technique can be near optimal (under a forecasting criterion) when compared against the others, a combination of them generally provides better results.

2. Method proposed.

From a Decision Support Systems (DSS) perspective an optimal system may not be an individual model but the combination of several of them. In fact, this is the usual way to proceed in order to evaluate projects in many financial contexts: the opinions of a comitee of human experts (each of them representing a particular and relevant aspect of the problem) are aggregated to give an optimal decision. As humans, different quantitative

Copyright © 1995 Software Engineering Press

models are more sensitive to specific aspects of information, and the problem is to exploit these asymmetries in an optimal manner.

A simple way to combine forecasts is to take the opinion of the majority, for example predicting bankruptcy if three out of five models do. Though useful in certain settings (see Fernández and Olmeda, 1995) this method has several drawbacks. For example, if the mixture is composed of too many inefficient and uncorrelated methods, it would perform worse than any single method. Also, it does not account for the different expected performance of the techniques when employed in particular circumstances (such as the plausibility of outliers). Finally, this method can not be used when the classes are continous. For these reasons, an additive procedure that permits a continous aggregation of forecasts should be preferred. Here we propose the basic framework for such a method. It can be extended in a number of ways but for reasons of brevity we only provide a general description.

Let us suppose that we have n examples $E_i = (a_{i1}, a_{i2}, \ldots, a_{ik}, c_i)$ completely characterised by the values of their k attributes and which belong to class c_i. Also, suppose that we have m classifiers and let P_{ij} be the prediction that classifier j makes of example i. Whithout loss of generality we can assume only two classes (failed and non-failed banks) so that $c_i, P_{ij} \in \{0,1\}$ for all i,j. Let $P_i = I(\Sigma^m_{j=1}\alpha_j P_{ij} - \theta)$ be the combined prediction of example i, where α_j is the weight assigned to method j, θ is a confidence level and I is the Heaviside function. Let δ_i be the associated cost of prediction:

$$\delta_i = \begin{cases} e_1 & \text{if } P_i > c_i \\ e_2 & \text{if } c_i > P_i \\ 0 & \text{otherwise} \end{cases}$$

where e_1 and e_2 can be considered the costs of Type I and II errors, respectively (here we will suppose $e_1=e_2=1$). The problem of finding an optimal mixture of classifiers can be formulated as finding the optimal combination of weigths α_j that minimizes the total costs of misclassification. Obviously, since this combination should be established *ex-ante*, the most we can do is to find an optimal mixture which minimizes the expected costs of misclassification. A very convenient way to estimate the expected performance of any particular method consists on dividing the training data (N) into v equal subsamples of size N, estimating the model using $N-n_v$ examples and predicting the remaining n_v ones (*v-fold cross validation*) [Stone, 1974]. This procedure is repeated for each of the v subsamples, and the mean prediction error is computed. Instead of using the cross validated error as a proxy to the expected error, we propose to consider the total error along the whole cross-validated set ($N \cdot v$ examples), since this assures not only an adecuate generalization but also an acceptable learning. For this reason, the mixture model could be suboptimal for some of the cross validated subsets though its performance should be globally optimal. With the above notation the problem becomes:

$$\min_\alpha \ \delta = \Sigma^n_{i=1} \ \delta_i$$
$$\text{s.t.} \quad P_i = I(\Sigma^m_{j=1}\alpha_j P_{ij} - \theta), \ i=1,2,\ldots N \cdot v \quad [1]$$

Other extensions (for example, a subjective measure of confidence) can be easily introduced in [1] as restrictions, leading to the minimization of the cost function with a penalty term for violation of the constraints.

Since P_i is a nonlinear (threshold) function, it is expected that gradient based methods may have some problems. There are many different

Copyright © 1995 Software Engineering Press

ways to solve [1] but in this paper we propose to employ a classical Evolutionary Programming (EP) algorithm (see Fogel, 1992 and references therein).

Specifically, let $\alpha = (\alpha_1,\alpha_2,\ldots,\alpha_m)$ be a vector of weigths, a standard EP algorithm follows the steps:

1.- Take an initial population of s vectors α_k randomly taken on an interval $[a,b]^m$.
2.- Evaluate the fitness of each of these individuals (δ_k).
3.- Add a multivariate gaussian vector $\zeta_k \sim N(0,G(\delta_k))$ to each α_k, being $G(\delta_k)$ proportional to δ_j. The vector $\alpha_{k+s} = \alpha_k + \zeta_k$ is called an offspring of individual α_k.
4.- Evaluate the fitness of each offspring.
5.- For every α_k, $k = 1, 2, \ldots, 2s$, select randomly h competitors from the population and compare their fitness. If the fitness of α_k is smaller than its competitor's assign it a win.
6.- Select the s individuals with more number of wins to generate a new population.
7.- If the termination criteria are not fulfilled go to step 3.

It should be noted that since the α coefficients are not bounded it is possible that the optimal mixture could be a "corner" solution including only a single (best) method. Also, the best decision could consist on doing the opposite as a particular method dictamines (a negative α_j). In this paper, the search process is stopped when the number of generations exceeds one hundred (this takes a couple of minutes on a 486 66Mhz).

Resuming, the proposed method follows the steps:

1.- Construct v samples of the training set. Estimate each of the models on $N-n_v$ examples and predict the remaining n_v ones. Select the model that minimizes the mean cross-validated error.
2.- Using the EP algorithm select the optimal combination of models that minimizes the cost of misclassification along the $N \cdot v$ set of examples .
3.- Re-estimate the models on the whole training set.
4.- Use the weights obtained in 2 to construct the combined forecast.

3. Application to the Bankruptcy Prediction problem.

The problem of bankruptcy prediction is a clasical one in the financial literature. Since the seminal work of Beaver [1966] many different techniques have been used: regresion analysis [Meyer and Pifer, 1970], multivariate Z-score [Altman, 1968], multivariate logit [Martin, 1977], recursive partitioning [Frydman et al., 1985], etc. Recently, tools taken from the Artificial Intelligence area such as Artificial Neural Networks [Odom and Sharda, 1990] or Machine Learning [Messier and Hansen 1988] have been also employed. Here we propose to integrate some of the mentioned techniques to produce an optimal forecast.

The models considered in this paper include a standard feedforward neural network with a single hidden layer trained with backpropagation (NN), two classical statistical techniques: Discriminant Analysis (DA) [Fisher, 1936] and Logit (Logit), and two recent extensions of the CART algorithm of Breiman et al. (1984): Multivariate Adaptive Regression Splines (MARS) [Friedman, 1991] and C4.5 (C4.5) [Quinlan, 1993]. Though the

Copyright © 1995 Software Engineering Press

last two methods are conceptually very similar, they differ both in their structure (the MARS algorithm uses truncated cubic polynomials as basis functions while C4.5 uses step functions) as well as in their performance criterion (the MARS algorithm minimizes a cross-validated error while C4.5 maximizes an information criterion), consequently they can provide different conclusions. We tried a variety of especifications for each of the models (number of basis functions for MARS, number of leaves for C4.5, number of hidden nodes for the NN, etc.), always using all the attributes. For reasons of brevity we give only the results for the best model found (the one which minimizes the mean cross validated error).

From 1977 to 1985 the Spanish banking system suffered the worst crisis of its whole history, affecting 52% of the 110 banks that were operative at the begining of this period. The total cost of this crisis has been estimated in 12 billion dollars. In our first application we employ a database consisting on 66 of these banks (see Pina, 1989). We consider 9 financial and economic ratios (working capital/total assets, sales/total assets, etc.) to evaluate the financial health. The ratios used for the failed banks are from the last financial statements issued before bankruptcy was declared while the data for non failed banks is from the 1982 statements. This database was randomly splitted into two sets, the training set consisted on 34 banks (15 failed and 19 non-failed) and the testing set on 32 banks (14 failed and 18 non-failed). The training set was divided into 6 different non-overlapping training and testing subsets, consisting on 28 and 6 banks, respectively. The whole cross-validated set contains 204 banks and it is used to find the optimal mixtures of forecasts.

In Table 1 we have computed the number of correct predictions, as well as the relative percentage of successes of each of the single methods and of the mixture model. We also indicate when a model is at least as good as any other on a particular set by a small asterisk (*). As one can see, the mixture is the best model both in terms of in-sample fitting and out-of-sample prediction even though, by construction, it should only minimize the total error (last row). The second best model is NN, closely followed by logit and MARS.

We will now employ all the examples of the training set to estimate the models and use them to make the predictions. Then, we use the optimal α values to combine these predictions. The best model (Table 2) is again the hybrid one, performing slightly better than NN (note that the improvement is obtained on the testing set). These models are followed by logit, MARS, C4.5 and finally DA.

Now we will consider the data used in the well-known study of Odom and Sharda (1990). The training set consists on 74 banks while the testing set consists on 55 banks. The training set was divided into 9 different non-overlapping training and testing sets of 66 and 8 banks, respectively. The whole cross-validated set contains 666 banks. As above, we employ these sets to choose the optimal configuration of each of the models as well as to find the optimal mixture. As one can see in Table 3, the mixture and the NN are the best models, exhibiting identical behavior both in-sample fitting and out-of-sample forecasting. This leads to suspect that the method has chosen a mixture that only considers the predictions of the neural net but, as we shall see, this is not the case.

Using all the examples of the training set to construct the models leads to some striking

Copyright © 1995 Software Engineering Press

results (Table 4). First of all it is noticeable that the expected performance of the models (estimated by the cross-validated error) is significantly higher (around 8% for every model) than the actual one. This seems to indicate that the training and testing sets could be different. Second, and more important, this bias could have induced to finding suboptimal structures for some of the models. For example, the results obtained with the NN are slightly worse than the ones reported in the literature (see Odom and Sharda, 1990). In any case, the hybrid model is superior to any single method (in particular to the NN) and is capable of obtaining identical results as the best reported in other studies (Rahimian et al., 1993).

4. Final Remarks.

The method proposed is quite general and flexible enough to accomodate to a variety of situations. It is possible, though, that in certain situations it can be dominated by a single method. For example, when the distribution of the values of a certain attribute is different along the training, cross-validated and testing sets then it is expected that a particular method less sensible to that attribute would perform better. Other possible situations in which its performance can be poor are when there are outliers in the data or the testing set is linearly separable while the training set is not. Several extensions of the method for its application to time series prediction are under progress.

Copyright © 1995 Software Engineering Press

References.

Altman, E.L. (1968): Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. *The Journal of Finance*, **23**, pp. 589-609.

Beaver, W.H. (1966): Financial Ratios as Predictors of Failure. *Empirical Research in Accounting: Selected Studies 1966, Journal of Accounting Research*, Supplement to Volume 4, pp. 71-111.

Breiman, L.: Friedman, J.; Olsen, J. and Stone, C. (1984): Classification and Regression Trees. Wadsworth International, CA.

Fernández, E. and Olmeda, I. (1995): Bankruptcy prediction with Artificial Neural Networks. *Proc. of the Intl. Workshop on Artificial Neural Networks '95 (forthcomming)*.

Fisher, R.A. (1936): The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, **7**, pp. 179-188.

Fogel, D.B. (1992): Evolving Artificial Intelligence. PhD. Diss. University of California San Diego.

Friedman, J.H. (1991): Multivariate Adaptive Regression Splines. *The Annals of Statistics*, **19**, pp. 1-141 (with discussion).

Frydman, H., Altman, E.I. and Kao, D. (1985): Introducing Recursive Partitioning for Financial Classification: The Case of Financial Distress. *The Journal of Finance*, **40**, pp. 269-291.

Martin, D. (1977): Early warning of Bank Failure, A Logit Regression Approach. *Journal of Banking and Finance*, **1**, pp. 249-276.

Messier, W.F. and Hansen, J.V. (1988): Inducing Rules for Expert System Development: an Example Using Default and Bankruptcy Data. *Management Science*, **34**, pp. 1403-1415.

Meyer, P.A. and Pifer, H. (1970): Prediction of Bank Failures. *The Journal of Finance*, **25**, pp. 853-868.

Odom, M. and Sharda, R. (1990): Neural networks for bankruptcy prediction. *Intl. Joint Conf. on Neural Networks*, II pp. 163-168. San Diego, CA.

Pina, V. (1989): La información contable en la predicción de la crisis bancaria: 1977-1985. *Revista Española de Contabilidad y Finanzas*, **18**, pp. 309-338.

Quinlan, J.R. (1993): *C4.5: Programs for Machine Learning*. Morgan Kaufmann, CA.

Rahimian, E.; Singh, S.; Thammachote, T. and Virmani, R. (1993): Bankruptcy prediction by Neural Network, in R.R. Trippi and E. Turban (Eds.): *Neural Networks in Finance and Investing*. Probus, Chicago, IL.

Stone, M. (1974): Cross-validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society B*, **2**, pp. 111-147 (with discussion).

Copyright © 1995 Software Engineering Press

Table 1 - Cross-validated performance (Spanish Banks)

Method	DA	Logit	MARS	C4.5	NN (8 hidd)	Hybrid
Training	90.00%	95.29%	94.11%	84.70%	97.05%	97.64%*
Test	61.76%	76.47%	79.41%	79.41%	79.41%	82.35%*
Overall	85.29%	92.15%	91.66%	83.82%	94.11%	95.10%*

Table 2 - Performance of alternative models (Spanish Banks)

Method	DA	Logit	MARS	C4.5	NN (8 hidd)	Hybrid
Training	30 (88.23%)	32* (94.12%)	31 (91.18%)	29 (85.29%)	31 (91.18%)	31 (91.18%)
Test	26 (81.25%)	28 (87.50%)	24 (75.00%)	28 (87.50%)	29 (90.62%)	30* (93.75%)
Overall	56 (84.85%)	60 (90.91%)	55 (83.33%)	57 (86.36%)	60 (90.91%)	61* (92.42%)

Table 3 - Cross-validated performance (American Banks)

Method	DA	Logit	MARS	C4.5	NN (3 hidd)	Hybrid
Training	92.39%	100%	98.65%	95.27%	100%*	100%*
Test	89.19%	94.59%	93.24%	91.89%	97.30%*	97.30%*
Overall	92.04%	99.39%	98.05%	94.89%	99.70%*	99.70%*

Table 4 - Performance of alternative models (American Banks)

Method	D.A.	Logit	MARS	C4.5	NN (3 hidd)	Hybrid
Training	69 (93.24%)	74* (100%)	73 (98.64%)	70 (94.59%)	73 (98.64%)	74* (100%)
Test	40 (72.72%)	43 (78.18%)	42 (76.36%)	43 (78.18%)	44 (80.00%)	45* (81.81%)
Overall	109 (84.49%)	117 (90.69%)	115 (89.14%)	113 (87.59%)	117 (90.69%)	119* (92.24%)

Copyright © 1995 Software Engineering Press

AN EXPERT SYSTEM FOR ADJUSTING MARINE UNDERWRITING AT CLAIM POINT

Suzanne S. SHAFIK,
MISR INSURANCE
Computer Center
44 A Dokki st,CAIRO,EGYPT

Mohamed.R HASSAN
MISR INSURANCE
Computer Center
44 A Dokki st CAIRO,EGYPT

Ahmed.RAFEA
Head of computer section
ISSR,CAIRO UNIVERSITY
5 Sarwat st,Orman CAIRO,EGYPT

Abstract

The main objective of this paper is to investigate how expert system technology can be applied to insurance applications to handle the knowledge intensive area of risk assessment. . We choose "Adjusting of Marine Insurance Underwriting At Claim Point" AMUACP to be applied. The investigation includes the following issue: A methodology to acquire knowledge from different sources, a knowledge representation, reasoning scheme and implementation of this determined scheme. During knowledge acquisition phase a model of expertise following KADS methodology was built to check the consistency and completeness of elicited data, and to facilitate the mapping of these data of expertise on some structure. In order to describe how the conceptual model is realised in the underlying hardware and software, a design model was built by using the structure preserving design method. The system is implemented in LPA Prolog running on PS/VP under DOS 5.

1- INTRODUCTION :

Insurance underwriting is a combination of business and risk assessment operations. Risk assessment in insurance underwriting has long been a profession speciality.Some common problems affect the insurance industry in the underwriting domain, this problems can be: (1) Shortage of skilled underwriting staff. (2) More numerous guidelines changes in response to external pressures and to rapid business cycles. (3) Decreasing underwriting expertise in different company locations as response to decentralisation approach. (4) Additional data requirement and increasing underwriting complexity.

Insurance companies have found that expert-system technology provides very significant leverage in building systems to handle the knowledge-intensive area of risk assessment [5]. Many underwriting expert systems has been built. For general insurance underwriting [5],[10],[11]and life insurance[1],[4],[8].

The first section of this paper includes introduction. The second section includes the domain problem description. In third section are described how KADS methodology is applied. The fourth section shows how the conceptual model is implemented .

2- PROBLEM DESCRIPTION :

Marine insurance covers marine transportation risks. This high risk affects importation of goods. For this reason the marine underwriting at claim point was chosen. The marine assurance consists of two main phases:underwriting phase and claim handling phase.

2.1 Underwriting Phase .

The underwriter assess the exposure of risks for each marine cargo case. The risk assessment requires a highly qualified expert to judge the accuracy of given information and consider default values for missing information, he takes into consideration the following factors :

.The cargo type and its risk of vice inheritance, the suitability of packing to cargo type.

.The vessel building year, flag and its suitability to cargo type and cargo packing.

.Any potential geographic dangereuos area, war and strike risks which may arise during the

Copyright © 1995 Software Engineering Press

voyage depending on the voyage route.
.Condition of the policy which includes : applied general clause type A or B or C [7],[9] to define the covered risks, the additional coverage clauses as war, or strike clause, reshipment clause, and extension of cover until the installation of cargo in case of machinary.
.Cover limit to define the duration of the policy, it can be warehouse to warehouse, warehouse to port of destination ,port of shipment to warehouse,port of shipment to port of destination.
.The loss ratio of the client and his position.

Depending on the study of these factors the underwriter takes a decision which can be accepting risk with normal rate, or accepting isk with extra rate or refuse to cover the risk.

2.2 Claim handling phase.

If an accident occurs, the client send a claim notice to the insurance company, a claim adjuster expert examines the accident circumstances, all related documents (port authorities report, vessel manifesto, bill of landing, cargo invoices), and damage appearance to define the claim direct loss cause and write a claim survey report .The claim survey report is checked against the policy terms and conditions to apply marine assurance norms, and define the discrepancies between the policy conditions and the circumstances of the accident as described in the survey report . These discrepancies may be one or more of the following: (1) Cargo type or Cargo packing or both are not identical. (2) Vessel information is not identical. (3) The accident place is not in the ordinary route of the voyage. (4) The accident date is not within the duration of the policy(5) Cause of accident is not covered by the policy.
Depending on this study the claim adjuster take one of the following claim decisions :
.The cause of the accident is covered by the policy conditions and the claim must be paid.
.The cause of the accident is excluded by the conditions of marine policy and the claim is refused .
.Pending on the client degree and the discrepancies discovered the claim adjuster expert may refuse the claim or send the case to the underwriter to adjust the policy information, complete any missing information and a new cycle of risk assessment is performed.

In order to acquire this specific knowledge, we examine the existing application systems, manuals, regulations, policy statements and other written material representing compiled expertise of the organisation, other knowledge are acquired by structured interviews with marine underwriter expert.

3 AMUACP Model of Expertise.

Model of Expertise of **KADS**(Knowledge Acquisition and Design Structuring [2],[6] method is applied . The role of model of expertise is defining the functional specification of the problem solving part in the construction of knowledge base systems, it consists of the following layers: Domain knowledge,inference knowledge,and task knowledge:

3.1 Domain knowledge.

It represent the static knowledge which describes a declarative theory on application domain, it consists of concepts,properties,and relations. Figure (1) represent a part of domain kmowledge schema of AMUACP system .

3.1.1 Concepts and properties.

As a result of the analysis of AMUACP system domain, 22 concepts has been identified related to risk assessment and claim handling operations. Each concept has properties which are defined by their names, value types and possible values which can be a single value or a list containing enumeration of values. These concepts are divided into three categories, as follows:
-Concepts holding data extracted from company marine database, such as policy concept, vessel concept, voyage concept, client position concept, and vessel route table concept. These concepts have 43 properties.
-Concept holding claim data captured by the user during consultation session. This concept has 23 properties.

Copyright © 1995 Software Engineering Press

-Concepts holding some acquired marine underwriting knowledge which are used by the system to define the decision taken , as war zone concept, geographic dagereous areas concept, cargo' group concept, cargo concept, cargo packing rate concept, vessel-cargo-extra-rate and coverage condition concept. These concepts have 37 properties.
-Concepts holding intermediate results generated by the system during consultation sesion as cargo-pack claim decision concept, duration-place-cover-claim decision concept, loss-direct-cause-claim decision concept, client-position-claim decision concept, adjusted-policy-information concept and marine-norms concept. These concepts have 59 properties.

3.1.2 Relations :

In AMUACP system, realtions consists of one relation between concepts which relates cargo with its cargo group and 12 relations between property expressions. Relations between expressions are semantically divided into:

.**Comparison relations** which compare the vessel , cargo , packing , cover limit, terms and conditions properties of the policy and voyage concepts against claim concept properties in order to discover the discrepancies.
.**Adjusting policy information relation** by the claim information.
.**Specification of underwriting marine norms relations** to be applied on the adjusted policy information properties based on vessel properties concept, cargo properties concept, route table properties concept and clause application properties concept.

.**Final risk assessment decision relation.**

This relation match the norms of marine underwriting with properties concept of adjusting policy information and policy information. The result of this match is one of the different available system decision.

Part of Schema of Domain Knowledge of AMUCACP system

	Primitive	Name	Description definition
1	Concept	policy	main information of policy
2	Concept	vessel	description of vessel
3	Concept	voyage	defintion of voyage of vessel
4	Concept	route table	defintion of different paths of the vessel time table
5	Cconcept	war-zones	defintion of the war-zone in the world
6	Concept	Geo.-route table	defintion of maritime geographic dangerous areas in the world
7	Concept	wm route table	working concept to hold the alternative route paths of the vessel for the voyage
8	relation between Concept	cargo is-a-kind-of cargo group	cargo inherent all properties of cargo group
9	relation between expressions	determination of claim decision based on cargo,packing information	compare cargo packing in policy against claim if not equal compare packing in claim properties with properties of cargo and cargo group concepts.
10	relation between expressions	determination of claim decision based on vessel information	compare vessel in policy against claim if not equal compare vessel in claim with properties of vessel concepts

Figure 1 Part of Schema of Domain Knowledge of AMUCACP system

Copyright © 1995 Software Engineering Press

3.2 Inference Knowledge:

Inference Knowledge is defined in terms of inference steps and roles.An inference step is defined through its name, an input/output specification and a reference to the domain knowledge that it uses.
Figure(2) depicts the inferance structure of AMUACP system. IT consists of four inference steps namely: Compare, change, match and specify. The following paragraphes describe the function of each inferance steps.

3.2.1 Compare Inference Step:

The main function of this inference step is to compare between policy information against claim information in order to determin descripancies. This comparison includes :
-comparison between policy vessel and claim vessel
-comparison between policy cargo, packing and claim cargo pack.
-comparison of voyage, cover limit policy information and claim place of accident,date of accident .
-comparison between the term and condition of the policy, inherent vice of policy cargo type and the direct cause of the claim.
-Definition of final claim decision depending of found discrepancies,policy exclusion and client position.

3.2.2 Change inference step

The main function of this inference step is to adjust the policy information by the detected descripances.

3.2.3 Specify inference step

The main function of this inference step is to specify marine norms for vessel category, cargo packing , application clause, coverage condition, additional coverage condition to be applied on adjusted policy information.

3.2.4 Match inference step

The main function of this inference step is to match the norm of marine underwrting specified with the policy information. The result of this match, is one of the following decision refuse claim or accept the claim with paying extra premium related to determined descripances.

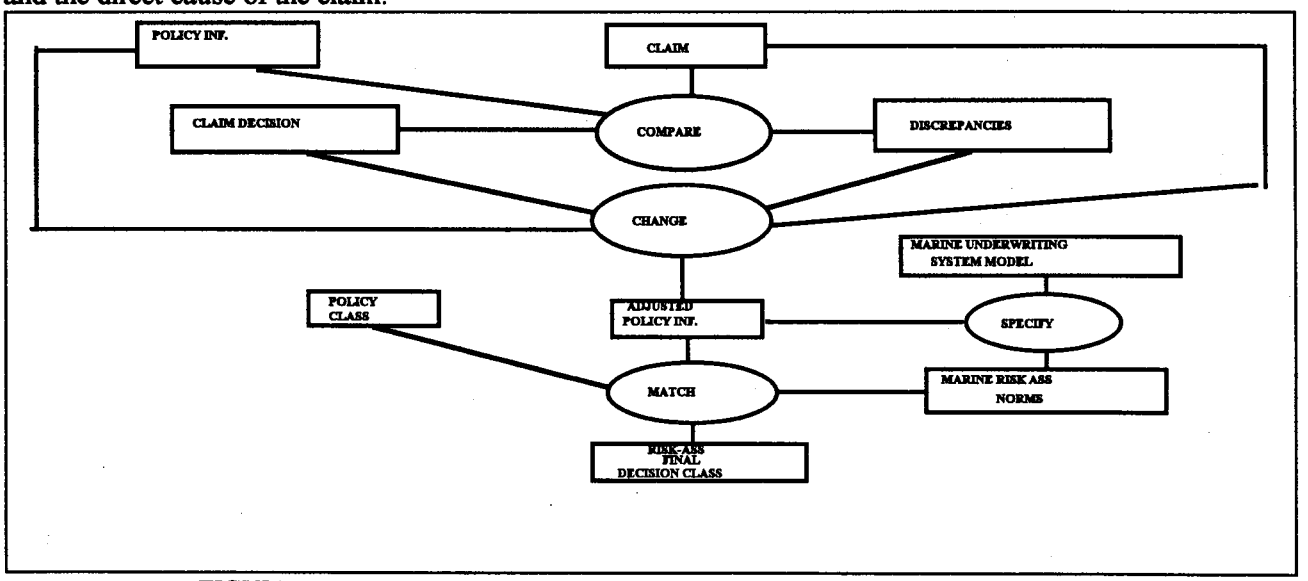


FIGURE 2 INFERENCE STRUCTURE OF AMUACP SYSTEM

Copyright © 1995 Software Engineering Press

3.3 Task knowledge :

It contains knowledge about how elementary inferences can be combined to achieve a certain goal. The tasks knowledge specification of AMUACP system is shown in Figure (3).

Task ADJUST MARINE UNDERWRITING AT CLAIM POINT
Goal define claim decision , final assessment decision in case of reassessment of risk.
Control terms claim decision, reassessment final decision
Task structure
Obtain claim information.
if policy <> claim exit
Generate Wm route
Compare_entities(policyinf,claim=====>discrepancies , claim decision)
if final_decision <> 1 and client degree=1
then claim decision= reassessment of risk.
If claim decision <> reassessment of risk
then display decision
else
Adjust_policy(policyinf,claim,discrepancies=====> adjusted_policy_inf)
Specify_norms(adjusted_policy_inf, marine underwriting system model =====>
marine risk assessment norms).
Match(adjusted_policy_inf,marine risk assessment norms =====>risk-ass- final decision).

Figure 3 AMUACP task structure

4- Implementation of AMUACP expert system.

Conceptual model is transformed to design model by using structure preserving design method which transform the elements of conceptual model to identifiable computational constructs. The following subsection map the conceptual model to our design model.

4.1 AMUACP Knowledge base representation.

Concepts as represented in domain layer are mapped to Prolog Database and relations are mapped to Prolog rules.

4.1.1 Concepts:

The representation of concepts is defined depending on the computation requirements of the input/output part of the inference step and for serving the explanation module. In

Copyright © 1995 Software Engineering Press

AMUACP system we have two types of representation as follows :

Static instances: It is used to represent some kind of knowledge in a form of table.

Concept (property 1, property 2, property3, ..., propertyn).

This representation is used for :

(1) Concepts whose properties should be extracted from the company data base, for example the policy concept, it is represented as follows:

Policy (Policy_no, client, cargo, packing,issue _ date ,cover_limit, app_coverage, ins_rate, ext_ves_rate, war_strike_rate,ext_rate, insured_amount).

(2) Concept holding some kind of knowledge representing marine underwriting norms to be used in order to define a certain decision from a number of decisions. for example cargo group concept is represented as follows :

Cargo group (group name, inherent vice list, suitable packing, unsuitable_packing) ;

Dynamic instances : This representation is used for concepts whose properties are obtained from the user or generated by the system and asserted in the dynamic data base. These dynamic instances have two presentations:

Presentation 1: Implementation for concepts whose its property are obtained from the user.

Property(Concept name,Value)

Presentation 2 : for concept generated by the system and whose its properties are accessed dependently in same time are presented as follows :

Concept (property1, property2, propert... propertyn) example of concept presented by this implementation is vessel_claim_decision (policy_no, claim_dec, ass_cause, refuse cause, accept cause, final)

4-1-2 Relations.

Relation between concepts.

Relation between concepts is implemented in Prolog data base as follows:

Is_a_kind_of (concep1, concept2), which means concept1 inherits all properties and relations from concept 2

Relation between expressions.

Relation between expressions are grouped according to the semantic of the relation and the conclusion part . The conclusion part of a relation is the properties of one concept, whereas the condition part of a relation may be the properties of more than one concepts. This grouping of relations between expressions is performed in order to establish a clear mapping between inference step and the relation between expression . Examining AMUACP relations, it was found that these relations can be classified into two types:

- the first type of relations is the one in which properties of the concepts in the conclusion part are instantiatied with defined values. Example of this type cargopack_claim_decision relation.

The following Prolog coding represents an example of this type :

```
('Refuse_claim','Unsuitable_packing',0,1):-
cargo_pack(Cargo_c,Cargo_p,Cgroup_c,
Cgroup_p ), Cargo_c == Cargo_p ,
\+(Packing_p == Packing_c),
cargo_group(Cgroup_c,Grsuitable_p,
Grunsuitab_p),
memb(Packing_c,[Cunsuitab_p]);
(cargo_group(Cgroup_c,_,Grsuitable_p,
Grunuitab_p),
memb(Packing_c,[Grunsuitab_p]));
(memb('Other_Packing',[Grunsuitab_p])),
\+(memb(Packing_c,[Grsuitab_p])).
```

-The second type of relations is the one in which the properties of the concepts in the conclusion part are instantiated with values of properties of other concepts .

The following Prolog coding represents an example of this type :

```
cargo_packing_clause_rate (marine_norms,
Value) :-
cargo_pack_claim_decision (P1,_,_,_,
Refuse_cause,_,_), client_pos_claim_decision
(P1,_,Final_as_cause,_),
(Refuse_cause == 'different_cargo_sam_grp');
(Refuse_cause_ =='different_packing')),
Final_as_cause == 'reasssessment' ,
cargo_pack_spec_rate(Adj_cargo,Adj_packing,
App,Cunsuitab_p),
```

Copyright © 1995 Software Engineering Press

```
/* check adjusted packing not a member of
unsuitable packing    */
\+(memb(Adj_packing,[Cunsuitab_p])),
App =='A'  ->
/* Specfy cargo_packing_clause_rate_A  */
cargo_packing_rate(Adj_cargo,Adj_packing,
Value,_,_,_),
```

4-2 Inference layer implementation.

The inference layer describes the relation between roles and inference steps, for each inference step we need :
-Role to represent the input/output data structure,
-A representation of domain knowledge,
-An algorithm embodies the method for realising the inference and specifies the control [12].

4-2-1 Role implementation.

Each role is mapped into a subset of the domain layer , so they are implemented as a Prolog data base corresponding to its domain layer subset .

4-2-2 Inference step implementation.

Each inference step is implemented as a Prolog procedures which gets data from input role and asserts result in the output role . An inference step applies a set of relations in order to achieve its inference function.

4-2-3 Task layer implementation

The task layer is implemented as a Prolog code which maps exactly to the task knowledge depicted on figure(3).

4-2-4 AMUACP user interface :

The user interface of AMUACP system has the following characteristics:
-Ease of use of the system without previous computer knowledge. The user starts the consultation session by entring the claim description data. This data arranged depending on the relative inportance in defining system decisions. The user interface is tailored to use the user terminology. Used codes are selected from a prompt menu in interactive mode.
.The system makes logical validation on the claim description data to avoid possible errors. If an error occurs the system display an error message and reaccept the errorneous data.
.The system operations are made explicit to give the field expert the ability to trace the used system components. The system user can trace the reasoning path of the system. Intermidiate results are displayed and the user can redirect the path of the system by changing some parameters. For example if the system decision is reassesment of risk the user can change the client degree to change the taken decision.
.The system has an explanation module. It helps the user to know why the system asks such questions,and how the system reach this conclusion. This happens by displaying a log file containing the path of the consultation session.

5 Conclusion

During the different phases of the application a verification process takes place to demonstrate the consistency, completeness and correctness of implemented system, this verification is established by testing of legal values for data entered by the user , verify the system by using some hypothetical cases that cover all the decision generated by the system and sample of data for 100 cargo types classified into twenty cargo groups extracted from the company database to be used by the system . The system proves its feasibility, optimality and success as a prototype. The implementation of this system proves the impact of applying modelling techniques specially for large system development process . The reusability of Kads interpertation models allow to build this model which can be used as an extention to the interpretation library for insurance risk assessment model elements .
This expert system can be extended functionally to cover other cargo groups and cargo types, Enhancement of user interface is required to arabise the input screens of the system, displayed

Copyright © 1995 Software Engineering Press

conclusion of the claim decision, final assessment decision and the explanation given by the system.

An interface must be established between the expert system and the company relational data base running on company mainframe in order to avoid redundancy and provides to the expert system an up to date container of factual data.

As additional to the function of this expert system a claim handling expert system should be build to define the direct cause of the claim depending on cargo type, packing, vessel type, and loss or damage appearence.

This claim handling expert system should be integrated with Adjust Marine Underwriting At Claim Point AMUACP expert system.

References.

[1]- Bolger. F., Wright g., Rowe J., Gammack,J., and Wood. B., LUST for life : Developing expert systems for life assurance underwriting. Research and Development in expert system edited by Shadbot Nigel, 1989.

[2] - Breuker,87,and Wielinga,Use of models in the interpretation of verbal data, knowledge acquisation for expert systems,a practical handbook, edited by Alison L,Kidd 1987.

[3] - Brouglon.Steve w.,The underwriting advisory system in Sun Alliance International inc. Insurance International Executive conference. La Hulpe Belgium October 1990 .

[4]- Chamberlin. G., Neal. I., and Khan M., Aries at City, University London Ec1v0hb Research and Development in Expert System edited by Shadbot Nigel prentice hall publisher, 1989.

[5]- Chapnick. P., From Abacus to computer, Finance drives technology, AI expert system May 1990.

[6]-Linster,92 ,and Musen, Use of KADS to create a conceptual model of the ONCON Task Knowledge Acquisition, Special issue, The KADS approach to knowledge engineering, Academic press March 1992.

[7]- LLOYDS Survey Handbook, Fourth edition edited by K.G Knight,1985.

[8]-Rafeh. M. A., S. Shafik and W. Reyad, ESLAU: Expert System for life assurance underwriting the implementation and evaluation issues. Second National expert system and development workshop Cairo-Egypt May 1993.

[9]- Regulations of Egyptian Insurance federation,1994.

10]- Rekart.Thomas C., Knowledge Engineering for Insurance Application Processes and Methodologies, Artificial Intelligence Conference, the 3rd annual IBC conference on expert systems in Insurance. October 1991.

[11]- Ulrich. Monika., Expert Systems in Health Insurance, Insurance International executive conference. La Hulpe, Belgium, October 1991.

[12]-Wielinga. B. J., A. TH. Scheiber and J. A. Breuker, KADS: a modelling approach to knowledge engineering,Knowledge Acquisition, Special issue, The KADS approach to knowledge engineering, Academic press March 1992.

Copyright © 1995 Software Engineering Press

Cost-Effective Classification for Credit Decision Making

Grigoris J. Karakoulas

Knowledge Systems Laboratory,
Institute for Information Technology,
National Research Council Canada,
Ottawa, Ontario, Canada K1A 0R6

grigoris@ai.iit.nrc.ca

Abstract

There is an increasing need for credit decision making systems that can dynamically analyze historical data and learn complex relations among the most important attributes for loan evaluation. In this paper we propose the application of a new machine learning algorithm, QLC, to the credit analysis of consumer loans. The algorithm learns how to classify a loan by minimizing the expected cost due to both credit investigation expenses and possible misclassification. QLC is built upon reinforcement learning. A dataset of actual consumer loans is used for evaluating the algorithm. The experiments reported show that QLC performs better than other cost-sensitive algorithms on this dataset.

1. Introduction

According to a recent U.S. Banker survey amongst the 113 top U.S. banks [15], the most popular approaches for automated decision-making for all types of credit products are application scoring and on-line credit bureau scoring. These credit-scoring procedures refer to the evaluation of each applicant by models that are derived from statistical discriminant analysis of the credit history of past applicants [12]. The main drawback of this type of evaluation stems from the reliance of discriminant analysis on a subjective assignment of scores to the credit attributes of an applicant's profile.

As also came out from this survey, more than 60% of the surveyed banks used judgemental — i.e. non-automated — scoring. The most important factors in the adoption of credit decision-making software by banks are: understanding system requirements and understanding credit management needs. In addition, a hindering factor in the deployment of current credit decision systems is their limitation in generating explanations when credit decisions are made. In contrast, the generation of explanations is a relatively easy task when judgemental scoring is used.

Artificial intelligence technologies have been employed for the development of credit-scoring software systems that can meet the emerging needs and requirements [6, 12]. On the one hand, expert systems have the advantage of representing and reasoning about relations amongst symbolic objects. This facilitates the task of generating explanations about objects and about inferences on the relations amongst objects. The disadvantage of expert systems is that the relations embedded in their knowledge base are pre-defined and their maintenance can become a tedious task. The increasing complexity of loan instruments, the volatility of the economic conditions and the importance of risk management in minimizing losses of loan portfolios impose the need for software with learning capabilities for dynamically analyzing various sources of historical data and capturing complex relations amongst the most important attributes for loan evaluation.

On the other hand, neural networks are good for learning complex relations by using non-parametric modeling. However, neural networks are usually considered as black boxes because it is

Copyright © 1995 Software Engineering Press

difficult to understand how learning occurs within their architecture and it is hard to explain how particular decisions are made through the networks once they are trained. Furthermore, neural networks may suffer from slow learning rates. The limitation of neural networks in explanatory capabilities is critical to their adoption for credit scoring. This is because there are regulatory constraints that require explanations to be given to consumers whose applications for a credit product have been rejected [4].

In this paper we propose the application of a new machine learning method for the credit analysis of consumer loans. Most classifiers in machine learning are built with the aim of minimizing errors made when predicting the classification of unseen examples. In contrast, our method is based on the general idea that it is worse to classify a bad customer as good than it is to classify a good customer as bad. Thus, classification errors may ensue different costs depending on the type of error. These costs can be in nominal values or if they are not known they can reflect constraints on the percentage of cases erroneously identified to belong to a particular class. This asymmetry in costs is of particular importance to applications like credit analysis where one class is comparatively rare and of special interest like loan defaults. Asymmetric misclassification costs may act as a focus mechanism for exploring the areas of the attribute space where the rare class is comparatively more common.

In a classification process, in addition to the costs of classification errors there are also the costs of tests which are incurred as information about the attributes of an object is acquired for making a classification decision. For example, credit investigation expenses are involved in the acquisition of information about the credit attributes[1] regarding an applicant. When both types of costs are considered the problem of cost-effective classification amounts to identifying for each state of the classification process an optimal sequence of tests (i.e. an optimal plan) for deciding among competing alternatives (i.e. classifications or additional tests).

Our approach to cost-effective classification is built upon reinforcement learning. The latter is a dynamic optimization paradigm within machine learning [13]. It is used for learning optimal policies in state-space problem-solving tasks. A policy specifies for each state what action to perform next. During learning, the system receives a reinforcement signal (a penalty or reward) after each action. The goal of the system is to find a policy that minimizes/maximizes the expected reinforcement over all future actions. Although reinforcement learning is quite different from typical concept learning, when test and misclassification costs are taken into account credit analysis becomes a stochastic optimization task. The goal of the task is to minimize the total cost of classification of each applicant.

The remainder of this paper is organized as follows. Section 2 proposes a problem formulation that makes reinforcement learning applicable to the cost-effective classification task. Section 3 develops a clustering technique for enhancing the scaleability of reinforcement learning for this complex task. The whole algorithm is presented in Section 4. Section 5 reports on experiments for evaluating the performance of the proposed algorithm. A sample of 1000 actual consumer loans granted is used for the experiments. Related and future work are discussed in Section 6. Conclusions are given in Section 7.

2. Problem Formulation

Consider a task where a case k is to be classified among m classes. There are n tests available each of which can be selected at any time but only once during a trial. The latter is defined as the sequence of tests ended by a classification. At each time t the set of possible actions A_t contains the m classifications and the tests not yet selected in the current trial. At the start of each trial this set has size $m + n$. When the agent selects a test it pays a cost which may be a function not only of the selected test but of prior tests as well. In medical diagnosis

[1] In the sequel, the term *test* will be used for denoting a *credit attribute.*

Copyright © 1995 Software Engineering Press

for example, a set of blood tests shares the common cost of collecting blood from the patient. This common cost is charged only once, when the decision is made to do the first blood test. The result of each test i is denoted by x^i. Having selected test i for case k, the agent observes the value of the test, $x_k^i \in X^i$, which has a distribution conditional on the history of observations prior to time t. The agent must then decide which action to perform next. It may choose to stop further testing and make a classification of the case to class j, $j \in [1, \ldots, m]$. However, if the predicted class is not equal to the actual class of the case, the agent is penalized by the cost of the error made. This cost is defined in the classification cost matrix. Each element $c_{i,j}$ of that $m \times m$ matrix gives the cost of predicting that a case belongs in class j, when it actually belongs in class i. The agent repeatedly goes through cases in order to learn a policy that minimizes in the long run the cumulative cost over all cases.

This problem is characterized by imperfect state information since the state variables referring to the actual classes cannot be observed directly. Instead, the agent gets information about them through the process of testing. For each case k we define the vector of observable history at time t as

$$h_t = \left(x_k^i, x_k^j, \ldots\right) \tag{1}$$

The vector consists of the observed values of the tests performed prior to time t for case k. At the start of each trial (i.e. new case) the dimension of the vector is initialized to zero. As a new observation is added at each stage of a trial, the dimension of the vector increases accordingly. The probability distribution of the history vector can serve as a sufficient statistic that can reformulate the original problem with imperfect state information into a problem with perfect state information. Thus, the state of the reformulated control problem is defined as

$$z_t = P\{h_t|\theta\} \tag{2}$$

where θ is the unknown parameter of the distribution. The performance criterion of the control problem is:

$$V_\pi = E^\pi\left\{\sum_{t=0}^{\infty} \gamma^t c(h_t, a_t)\right\} \tag{3}$$

where γ with $0 < \gamma < 1$ is the discount factor. The cost function in (3) shows the dependence of the cost on prior tests as well as on the currently selected test. The policy is defined as $\pi: H_t \rightarrow P(A_t)$ mapping the space of observable histories into probability distributions of actions. The stochastic nature of the policy enables exploration of the state and action space for overcoming the problem of simultaneous identification of θ and control via π. We defer further analysis on how the policy probabilities $P^\pi\{a|h_t\}$ are calculated until the next section where a generalization scheme is developed. The probabilities will then be defined upon the generalization space.

The agent's objective is to choose a policy π^* such that:

$$\pi^* = \arg min_\pi\left[E^\pi\left\{\sum_{t=0}^{\infty} \gamma^t c(h_t, a_t)\right\}\right] \tag{4}$$

Although, Dynamic Programming (DP) equations can theoretically be written for the optimization problem in (4), the assumptions for prior knowledge and the computational intractability of a DP algorithm, leads us to examine Q-learning as an alternative for this problem.

Q-learning is a reinforcement learning algorithm that is based on an asynchronous, stochastic approximation version of the DP equations [16]. Thus, in our problem the Q-learning equation can be written as:

$$Q_{t+1}(z_t, a_t) = (1 - \beta_t) Q_t(z_t, a_t) + \beta_t[c(h_t, a_t) + \gamma V_t(z_{t+1})] \tag{5}$$

where

Copyright © 1995 Software Engineering Press

$$V_t(z_{t+1}) = \min_b Q_t(z_{t+1}, b) \quad (6)$$

It should be noted that almost all of the theory of Q-learning assumes look-up table representations of the Q-value function. Such representation is not suitable for our problem for two reasons. First, the state of the system in (2) is a vector of real-valued variables. The learning algorithm should be able to generalize over the continuous state-action space and over the training dataset in order to perform well on previous unseen cases in the testing dataset. Second, the policy rules of our problem represent a mapping more complicated than the one of the policy rules in typical Q-learning. The generalization scheme should be able to accommodate such mapping. In the next section we develop a clustering technique suitable for tackling the issues associated with generalization in our problem.

3. A Clustering Technique for Generalization

The technique is based on the idea that as the agent explores the input $(Z_t \times A_t)$ and output $(\Re)$ spaces of the task, clusters are formed for each action from instances of points on the Q-surface. Each time a new instance is created from a history vector h_t the clusters of each action $a \in A_t$ are searched in order to estimate the conditional probabilities of selecting each of the clusters of action a given h_t. The Q-value of (z_t, a) can then be estimated from the Q-values of the clusters of action a using the conditional probabilities as weights. The action with the minimum Q-value is selected for the instance. After updating the Q-value of the instance via the Q-learning equations (5) and (6), the agent should also update its memory with the instance accordingly. We next give definitions of cluster and instance and then formalize the above procedures.

3.1 Definition of Cluster and Instance

A cluster i of action a denoted as c_i^a is represented as a 3-tuple:

$$c_i^a = \langle Z_i, Q_i, n_i \rangle \quad (7)$$

where Z_i is a vector $Z_i = [z_{i1}, z_{i2}, \ldots, z_{in}]$ with each $z_{ij}, j \in [1, \ldots, n]$, defined as

$$z_{ij} = P\{x^j = x_i^j \mid a, \theta_j\} \quad (8)$$

That is, z_{ij} is the probability of test j displaying the value x_i^j given action a and the parameter of the distribution θ_j. In (7) Q_i is the Q-value of the cluster and n_i is the number of instances that have been aggregated in the cluster.

An instance at time t denoted as s_t, is represented as

$$s_t = \langle h_t, Z_t, a_t, Q_t \rangle \quad (9)$$

where h_t is the history vector at t, a_t is the action that is selected for the instance and Q_t is the Q-value of the instance. Z_t is a vector of probabilities defined similarly to (8), i.e.

$$z_{tj} = P\{x^j = x_t^j \mid a_t, \theta_j\} \quad (10)$$

Suppose for the moment that each x^j is a discrete variable with r_j number of values. Also, assume that the agent has beliefs in the form of a prior probability density on $\Theta = [\theta_1, \ldots \theta_n]$. A prior density that is usually assumed in Bayesian analysis is the Dirichlet density [1,2]. The posterior distribution of the probability z_{tj} in (10) is also a Dirichlet density. Omitting some theoretical details we can estimate the distribution in (10) from

$$P\{x^j = x_t^j \mid a_t, \theta_j\} = \frac{N_{tj}^{a_t} + 1}{N_j^{a_t} + r_j} \quad (11)$$

where $N_{tj}^{a_t}$ is the number of times that when action a_t is selected, x^j has the particular value x_t^j and $N_j^{a_t}$ is the number of times that when action a_t is selected, x^j has a value.

Copyright © 1995 Software Engineering Press

In the case that x^j is a continuous variable, it can either be discretized and treated as above, or one can apply Bayesian analysis for continuous distributions (for example, see [1]).

3.2 Q-value Estimation, Matching and Merging

Suppose that instance s_t is created from the current history vector h_t. Selecting an action for s_t requires estimating for each possible action $a \in A_t$ the value of $Q_t(s_t, a)$ from the Q-values of the clusters of action a. Since averaging over the Q-values of all clusters of an action involves a considerable amount of computation without necessarily a payoff in learning, we choose to average only over the k-nearest neighbors. The latter are determined according to the Euclidean distance between the vector Z_t of s_t and the vector Z_i of cluster c_i^a, i.e.

$$d(Z_t, Z_i) = \sqrt{\sum_{\tau=1}^{n} (z_{t\tau} - z_{i\tau})^2} \qquad (12)$$

where the $z_{t\tau}$ are estimated from (11) and $z_{i\tau}$ are the values stored in the cluster c_i^a. The formula for Q-value estimation is:

$$Q_t(s_t, a) = \sum_{i=1}^{k} P\{c_i^a | s_t\} \cdot Q_t(c_i^a, a) \qquad (13)$$

where $Q_t(c_i^a, a)$ is the value stored in the Q_i field of cluster c_i^a. The first term in the sum is the probability that cluster c_i^a is selected given instance s_t. This probability denotes the policy for selecting an action in the space of clusters C^{A_t}, namely $\pi_t: H_t \rightarrow P\{C^{A_t}\}$. It is given by

$$P\{c_i^a | s_t\} = \frac{P\{s_t | c_i^a\} \cdot P\{c_i^a\}}{\sum_j P\{s_t | c_j^a\} \cdot P\{c_j^a\}} \qquad (14)$$

The first term in the numerator of (14) is the probability of s_t having the particular history vector h_t given cluster c_i^a. This probability can be considered as a measure of how probable the values of h_t in s_t are, given cluster c_i^a. It is approximated by

$$P\{s_t | c_i^a\} = \frac{\sqrt{n} - d(Z_t, Z_i)}{\sqrt{n}} \qquad (15)$$

The second term in the numerator of (14) is the prior probability of any instance coming from cluster c_i^a. This probability is estimated by using a formula suggested by Anderson and Matessa in their work on Bayesian analysis of categorization [1]. Thus, we have

$$P\{c_i^a\} = \frac{c \cdot n_i}{(1-c) + cn_a} \qquad (16)$$

where c is the fixed probability that an instance comes from a cluster, n_i is the number of instances aggregated in cluster c_i^a and n_a is the number of instances aggregated in all clusters of action a.

Merging of an instance with a cluster requires the following two conditions to be satisfied: (i) $d(Z_t, Z_i) < \varepsilon_1$ and (ii) $|Q_t - Q_i| < \varepsilon_2$. If the two conditions are met then the instance is aggregated in the cluster by updating the fields of the cluster:
$z_{ij} = z_{ij} \cdot n_i / (n_i + 1) + z_{tj} / (n_i + 1)$,
$Q_i = Q_i \cdot n_i / (n_i + 1) + Q_t / (n_i + 1)$ and $n_i = n_i + 1$.

Similar conditions and operations apply when merging two clusters together.

4. The Proposed Algorithm

We assume that the dataset of the classification task has been split into a training set and a testing set. During learning the agent picks a case from the training set randomly without replacement and initiates a sequential decision process for the case, i.e a trial. During the trial the agent selects actions for making new estimates of the probabilities in (11) and updating the Q-values of its generalization space accordingly. When the agent selects a classi-

Copyright © 1995 Software Engineering Press

fication the current trial ends and a new one starts for the next case. Whenever all the cases of the training dataset have been processed this marks the end of an epoch. A new epoch is created by repeating the above procedure for the whole training set. Learning stops when the average cost of classification in the training set — total cost for the set divided by the number of cases in the set — is within ε between two consecutive epochs. The steps of the Q-learning with clustering (QLC) algorithm for one trial are shown in Figure 1.

Do:

(i) Create an instance s_t from the current history h_t;

(ii) For each possible action $a \in A_t$ estimate $Q_t(s_t, a)$ from its clusters;

(iii)Choose with probability ξ the action $a_t = \arg min_b [Q_t(s_t, b)]$;

(iv) Apply action a_t and pay the cost $c(s_t, a_t)$;

(v) If a_t is a test, update the history and probability vectors to h_{t+1} and z_{t+1} respectively;

(vi) Update the Q-value of s_t by

$$Q_{t+1}(s_t, a_t) = (1-\beta_t) Q_t(s_t, a_t) + \beta_t [c(s_t, a) + \gamma V_{t+e}(s_{t+1})]$$

where $V_{t+e} = min_b Q_{t+e}(s_{t+e}, b)$ is the e-step lookahead value of s_{t+1};

(vii) Update the memory either by merging s_t with a cluster of a_t or by creating a new cluster with only one instance s_t; check whether any clusters of a_t can be merged;

Until a_t is a classification action.

Figure 1: The steps of the QLC algorithm for one learning trial.

Step (iii) defines the exploration scheme of the algorithm. A value is randomly sampled from a uniform distribution in (0,1). If this value is less than ξ then the action with the minimum Q-value is chosen. Otherwise, any action is randomly selected. This scheme enables the algorithm to sufficiently explore the state and action space before converging to a good local optimum. In step (vi) the lookahead value $V_{t+e}(s_{t+1})$ is calculated by iterating over steps (i)-(vi) e times. We introduced this lookahead scheme due to empirical evidence from our experiments that this scheme improves the efficiency of the above algorithm.

5. Experiments

The experiments reported in this section were performed on a sample of 1000 actual consumer loans granted by a German bank. There are 20 attributes in the dataset that take symbolic or real values. There are also two classes of loans: good loans (70% of the dataset) and bad loans (30% of the dataset). The dataset was retrieved from the University of California at Irvine collection of datasets [7]. It was donated to the Irvine collection by Hans Hofmann[2].

Two experiments were performed. The purpose of the first experiment was to compare the performance of the QLC algorithm against the performance of other statistical and neural network algorithms on this dataset as reported in [11]. No test costs were assumed in this experiment. The purpose of the second experiment was to demonstrate the performance of QLC when both test and classification error costs are considered. Due to lack of information about actual credit investigation expenses we assumed a cost of one unit for each test. QLC is compared with Nunez's cost-sensitive algorithm EG2 [8]. This algorithm takes into account only the cost of testing. In both experiments the misclassification cost matrix had the form of Table 1.

It should be mentioned that part of the implementation of the QLC algorithm involves a discretization procedure. In both experiments each real-valued attribute of the dataset was discretized by dividing its range of values in the training set into five inter-

[2] The dataset has the URL ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/german/german.data.

Copyright © 1995 Software Engineering Press

vals of approximately equal size. We also used 4-nearest neighbor for estimating the Q-values by (13). The coupling probability in (16) was set to 0.3. The exploration probability of the QLC algorithm was set to $\xi = 0.9$. The learning rate β_t in the Q-value equation (5) had initial value 0.3 and was decayed as a function of learning experience. For each action the Q-value of state-action pairs was initialized to zero. The threshold ε for stopping training was set to 0.001.

Actual Class	Guess Class	Classification Error Cost
class 1	class 1	$0.0
class 1	class 2	positive error cost
class 2	class 1	negative error cost
class 2	class 2	$0.0

Table 1: The matrix of classification error costs.

In the first experiment we used the same procedure as in [11] for splitting the dataset into a training and a testing set. The training set consisted of 200 good and 200 bad loans randomly chosen from the initial dataset. The testing set consisted of the remaining cases, i.e. 500 good loans and 100 bad loans. We adopted this splitting procedure in order to ensure comparability of our results with those in [11]. For the same reason, the positive error cost was set to 1.0 and the negative error cost to 13.3.

The results are shown in Table 2. LDA is linear discriminant analysis; QDA is quadratic discriminant analysis; CART is a statistical method for building decision trees [3]; NN1 is a neural network with two hidden layers, 45 nodes in the first and 5 nodes in the second layer; and NN2 is a neural network with two hidden layers, 40 nodes in the first and 5 nodes in the second layer. The results of these five methods are taken from [11]. %N.E. denotes the rate of negative errors in the testing set, i.e. the fraction of bad loans that the classifier judges positive. %P.E. denotes the rate of positive errors in the testing set, i.e. the fraction of good loans that the classifier judges negative. The average cost is computed as the total cost of classifying the cases in the testing set divided by the number of cases.

Algorithms	No. Attr.	%N.E.	%P.E.	Avg Cost
LDA	20	28.7	29.1	0.88
QDA	20	28.3	34.0	0.91
CART	15	27.7	28.9	0.85
NN1	20	38.0	24.0	1.04
NN2	20	24.0	31.2	0.79
QLC	20	15.7	25.2	0.56

Table 2: Performance with cost ratio=13.3.

The results of LDA and QDA were derived by leave-one-out cross-validation. The results of CART, NN1 and NN2 were computed by using only one testing set. For the training of the CART algorithm 15 attributes were selected from the 20 attributes of the dataset. The QLC algorithm was run on 10 pairs of training and testing sets. Each pair was formed by randomly splitting the initial dataset according to the aforementioned procedure. The results reported on QLC are averages over the 10 testing sets. Although the algorithms have not been evaluated in exactly the same way, QLC shows a better performance than the other algorithms in terms of both average cost and error rates.

The above splitting procedure creates a training set with equally sized classes in order to enhance learning of the rare class of bad loans. In the respective testing set, however, the ratio of the size of the two classes is different from the ratio in the initial dataset. This disparity may be biasing the results of Table 2. In the second experiment we used a different splitting procedure. The initial dataset was randomly split into 10 pairs of training and testing sets. Each training set consisted of two thirds of the dataset and each testing set consisted of the remaining one third. A cost of one unit was assumed for each test. To enable sufficient testing we set the positive error cost to 40.0, i.e. a value greater than the total test cost. The negative error cost was set according to the negative-to-positive error cost ratio. We experimented with two values

Copyright © 1995 Software Engineering Press

of the error cost ratio: 5.0 and 13.3. In [11] these two values are suggested as the lower and upper limits of the error cost ratio.

The results of this experiment are shown in Tables 3 and 4. QLC performs better than EG2. It should be noted that because EG2 considers only test costs the different values of the error cost ratio do not affect the performance of the algorithm in terms of accuracy. QLC has better performance with cost ratio equal to 5.0 than with cost ratio equal to 13.3.

Algorithms	%N.E.	%P.E.	Avg Cost
QLC	18.2	22.6	32.84
EG2	60.9	14.9	42.56

Table 3: Performance with cost ratio=5.0.

Algorithms	%N.E.	%P.E.	Avg Cost
QLC	16.4	27.5	54.67
EG2	60.9	14.9	102.38

Table 4: Performance with cost ratio=13.3.

6. Discussion

There has been an increasing interest within the machine learning community for devising classification algorithms that are sensitive to either the costs of tests, e.g. [8], or to the costs of classification errors, e.g. [9] (see [5] for an extensive list of references). Turney [14] has recently proposed the ICET algorithm that takes both types of costs into account. The aforementioned research has focused on extending typical decision-tree and rule induction algorithms by either incorporating heuristic cost-sensitive attribute selection metrics or by building a two-tiered method for selecting among decision trees or rule-sets based on their cost-effectiveness.

In the statistics field, the CART algorithm [3] allows misclassification costs to be incorporated into the test selection process of a decision tree. A limitation of the CART algorithm is that it requires converting a cost matrix to a cost vector. This conversion results in having a single quantity to represent the importance of avoiding a particular type of error. The accuracy of the conversion depends on the accuracy of two estimates: (i) the frequency of examples of each class and (ii) the frequency that an example of one class might be mistaken for another.

In this paper we have introduced a new strategy for test selection given the goal of minimizing the expected cost due to both testing and classification errors. The strategy is realized through a single incremental learning algorithm. A particular advantage of our approach is that since the algorithm is incremental, after the learning system is deployed new cases of customers' loans can be incorporated in the system's memory depending on how informative these cases are with respect to the classification model already learned. In other work [5], we have empirically shown using three datasets from the domain of medical diagnosis that QLC performs better than related cost-sensitive classification algorithms. In that work actual costs were used for the medical tests. Future work should, therefore, examine the performance of QLC on credit decision making when actual credit investigation expenses are considered for the test costs.

Due to its stochastic optimization context, our algorithm can be extended for developing more sophisticated credit decision making models that take into account additional pragmatic considerations of credit granting decisions such as the risk of cash flows from credit sales [10].

7. Conclusion

This paper examined the problem of minimizing the expected classification cost due to both tests and classification errors in credit decision making. We presented a new cost-effective classification

Copyright © 1995 Software Engineering Press

strategy that is realized through the QLC algorithm. The latter is a single incremental learning algorithm which is based on a stochastic optimization framework. QLC scales up Q-learning for dealing with the intrinsic issues of imperfect state information and of generalization over continuous spaces and over training data.

We empirically evaluated the performance of QLC using a dataset of actual consumer loans granted. Previous work using this dataset focused only on misclassification costs. QLC performed better than the algorithms reported in that work. When test costs are assumed QLC performs better than both the EG2 algorithm that takes only test costs into account.

Further experimentation is needed to analyze the performance of the QLC algorithm especially when actual credit investigation expenses are considered. Other pragmatic considerations of credit decision making should also be investigated.

8. References

[1] Anderson, J.R. & Matessa, M. Explorations of an incremental, bayesian algorithm for categorization. *Machine Learning*, 9, 1992, 275-308.

[2] Berger, J.O. *Statistical decision theory and bayesian analysis*. New York: Springer, 1985.

[3] Breiman, L., Friedman, J., Olshen, R., & Stone, C. *Classification and regression trees*. Wadsworth, 1984.

[4] Borowsky, M. Looking for a Net Gain. *Credit Card Management Europe*, Vol. 2, 1993, 40-42.

[5] Karakoulas, G.J. Q-Learning for Cost-Effective Classification. KSL-IIT Technical Report, National Research Council Canada, May 1995.

[6] Keyes, J. Winning Back Investor's Confidence. Information Strategy: The Executive's Journal, Vol.1, 1992, 42-44.

[7] Murphy, P.M. & Aha, D.W. *UCI Repository of Machine Learning Databases*. University of California at Irvine, Department of Information and Computer Science, 1994.

[8] Nunez, M. The use of background knowledge in decision tree induction. *Machine Learning*, Vol. 6, 1991, 231-250.

[9] Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. Reducing Misclassification Costs: Knowledge-Intensive Approaches to Learning from Noisy Data. *Proceedings of the Eleventh International Conference on Machine Learning, ML-94*, 1994.

[10] Scherr, F.C. Credit-Granting Decisions Under Risk. *The Engineering Economist*, Vol. 37, 1992, 245-262.

[11] Seitz, J. and Stickel, E. Consumer Loan Analysis Using Neural Networks. In: *Adaptive Intelligent Systems*, S.W.I.F.T. (ed.), pp.177-192. Elsevier Science Publishers, 1993.

[12] Srinivasan, V., Kim, Y.H. Credit Granting: A Comparative Analysis of Classification Procedures. *Journal of Finance*. Vol. XLII, 1987, 665-681.

[13] Sutton, R. Introduction: The Challenge of Reinforcement Learning. Special Issue on Reinforcement Learning, *Machine Learning*, Vol. 8, 1992, 225-227.

[14] Turney, P. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* Vol. 2, 1995, 369-409.

[15] U.S. Banker. Lots of Ways to Make Credit Decisions. *U.S. Banker*, Vol. 102, 1992, 57-59.

[16] Watkins, C.J.C.H. *Learning from delayed rewards*. Ph.D. thesis. King's College, cambridge University, UK, 1989.

Copyright © 1995 Software Engineering Press

GECCO : An Expert System for Mining Investment-Quality Loans

Sue Bynum, Robert Noble, Cheri Todd : GE Capital Mortgage Corporation
6601 Six Forks Rd.
Raleigh NC 27615

Ben Bloom* : Inference Corporation

Abstract

The Guideline Eligibility Compliance Criteria Organizer [GECCO], is a knowledge-based application which automates the information-intensive process of *compliance underwriting* for mortgage loan resale in the secondary market. GECCO was originally built and deployed as a tool for third-party underwriting services in one business component of GE Capital Mortgage Corporation [GECMC], and has then been successfully integrated into two other GECMC businesses for internal use - in wholesale loan pricing, and in loan origination, and closing. The latest GECCO project development has resulted in its integration with the GENIUS™ Automated Underwriting System, as a commercial product offered to mortgage lenders. This paper chronicles the evolution of the GECCO tool, and describes the latest effort to combine its compliance underwriting expertise with the evidential reasoning model of risk underwriting provided by GENIUS™.

Successful incorporation of AI methodologies into critical business application software has reaped numerous significant benefits for GECMC, including higher loan throughput, improved consistency of underwriting decisions, and more effective quality control. GECMC has further distinguished its position as a leading provider of mortgage services through use of AI-based systems, and the GENIUS™ product equips GECMC with a competitive edge for extending its top market share in the mortgage insurance business. GECCO is an exemplary case of leveraging corporate knowledge through reuse of a formalized business model of the compliance underwriting process.

Keywords: Mortgage banking, rule-based expert system, automated compliance underwriting

* Please address correspondence to Ben Bloom, 8517 Sleepy Creek Drive, Raleigh, NC 27613. Email address: bloom@inference.com. Phone : (919) 870-0901. Fax : (919) 870-2445.

Copyright © 1995 Software Engineering Press

Background

GE Capital Mortgage Corporation is a leader in the home mortgage banking industry, with substantial portfolios in mortgage insurance, origination, and private-label mortgage-backed securities. A new GECMC business initiative was launched in 1989 with the creation of GE Mortgage Management Systems [GEMMS], to provide third-party processing and underwriting services to mortgage lenders. The Guideline Eligibility Compliance Criteria Organizer **[GECCO]** began in March 1992 as a GEMMS project to design and build a knowledge-based application for internal use as a tool for automating the information-intensive process of underwriting mortgage loans according to investor guidelines. GECCO was put into production in January 1993 at GEMMS for third party loan processing and underwriting.

During this same period when the GECCO compliance checker tool was developed, the GENIUS™ project was in progress at the GE Corporate Research and Development center. AI specialists at the R&D center built a formalized risk assessment model based on training examples provided by mortgage underwriting experts from the GECMC Company [GEMICO]. The GENIUS™ Automated Underwriting System is the software implementation of the example-based evidential reasoning model of mortgage insurance risk assessment. has proven to be a remarkably successful productivity tool for the Mortgage Insurance branch offices since going into production in February 1993.[1]

The natural marriage of the two AI systems has been brought about as the result of the PC GENIUS™ project, began in May 1994. The project focused on incorporating the compliance underwriting function of GECCO into the GENIUS™ risk assessment function. The rearchitecture effort involved reimplementation of GENIUS™ from a mainframe application into a Windows®-based client/server environment, and writing GECCO rulesets for the two largest mortgage investing agencies, Fannie Mae and Freddie Mac. The combined compliance and risk automated underwriting GENIUS™ functionality was made available as a commercial product to mortgage lenders in January 1995.

Business Context: Mortgage Banking

The field of loan underwriting has proven fertile ground for the application of AI technology for mortgage funding and for mortgage insurance. (See, for example,

[1] See [Gol95] for a thorough presentation of the GENIUS™ project.

Copyright © 1995 Software Engineering Press

[Gol95] and [Tal94]). The GECCO project is distinguished from related efforts by focusing on evaluating compliance with the investor eligibility guidelines which control *resale of mortgage loans* in the secondary market. It would serve to establish the business context of the GECCO tool by starting with an overview of the mortgage banking business and an itemization of the steps typically involved in the mortgage loan processing pipeline:

1. *Loan origination* : application by the borrower for a mortgage loan from a lender

2. *Registration*: Creation of a file for the loan in the processing pipeline

3. *Processing* : gathering and validation of loan documents required to complete the loan

4. *Underwriting* : assessment of risk incurred by the lender through evaluation of subject property, and of factors which determine the borrower's ability and willingness to repay.

5. *Mortgage insurance* [*MI*] : mandatory insurance purchased by the borrower to protect the lender against loss in the event of default by the borrower. *MI* is usually required when the borrower is infusing less than 20% of the loan amount into the transaction.

6. *Closing* : signing and recording of loan documents and transfer of mortgage funds

7. *Servicing* : Collecting the monthly mortgage, along with taxes, insurance and other escrows.

8. *Pricing/Repurchasing* : Sale of the loan from the originating lender to a mortgage investor. The two dominant *agencies* in the secondary market for conventional mortgages are Fannie Mae (Federal National Mortgage Association) and Freddie Mac (Federal Home Loan Mortgage Corporation). Mortgages which fall outside of agency eligibility guidelines may be sold to private investors, who would have their own purchase criteria.

9. *Packaging and Contract Servicing* : Collection of investment-grade loans into pools for issue on Wall Street in the form of mortgage-backed securities.

A discussion of mortgage banking terminology is useful to explain the function served by GECCO. It is necessary to understand the use of terms in this paper for the following: loan underwriting, investor guidelines, and lender compliance. Fannie Mae gives the term *investment-quality* to a mortgage when (1) the borrower's ability and willingness to pay the loan has been established and (2) the market value of the subject property provides sufficient collateral to secure the loan. The borrower's ability to repay the loan is based on income, employment history, assets, liabilities and source of funds. Willingness to repay the loan is based on credit history, separated into examination of mortgage/rent and revolving/installment accounts. The appraisal of the property is based on

Copyright © 1995 Software Engineering Press

evaluation of comparable properties, market trends for the neighborhood, and the property size and condition.

Fannie Mae, like other mortgage bankers, provides sets of guidelines which specify the criteria by which a loan can be judged to be investment-quality. A *guideline* may be thought of as a collection of business rules that evaluate the various pieces of applicant and property information. A mortgage loan application which does not violate any aspect of the investor's guideline is said to be *in compliance*, and may be underwritten by a mortgage loan analyst with confidence that the investor (e.g., Fannie Mae, Freddie Mac) would buy the loan.

Investor guidelines are typically extensive and are updated frequently (often quarterly) to adjust to changing housing market conditions. The size and volatility of a guideline is compounded by the fact that many points are open to the underwriter's interpretation. Certain loan application parameters are rigidly set by a guideline, such as dwelling type (single-family, duplex, condo, etc.), LTV (loan-to-value) limit, occupancy type (primary or second home, or investment property), and mortgage type (purchase, refinance, cash out refinance). Consideration of the large remainder of factors however, is flexible. As an example of a soft factor, Fannie Mae's guideline for the applicant's housing debt to income ratio is 28%, yet "we also recognize that some circumstances may justify your exceeding this ratio. If you use a higher ratio, you need to fully document the compensating factors you feel justify your doing so." Several examples of such factors are also provided to the underwriter.[2]

The underlying notion of salability of a loan is crucial to the underwriter's determination that the loan is in or out of compliance with the rules set forth in the investor guidelines. A typical mortgage lender relies heavily on investor capital in order to maintain a profitable and sizable portfolio of loans. If the loan does not meet investor guidelines, the lender must decide whether to fit the borrower into another type of loan program, add the loan to the "in-house" portfolio, or turn away the borrower's business. Strict interpretation of a guideline could result in denial to underwrite a loan which the investor would in fact have purchased (and, more importantly, loss of business), while a less cautious interpretation can result in funding a loan that cannot be sold and must remain in the lender's portfolio.

[2] [FNM93, p.27]

Copyright © 1995 Software Engineering Press

Project Description : GECCO / Third-party underwriting

The original inception of GECCO was within the third-party underwriting and processing facility at GE Mortgage Management Systems [GEMMS], a part of GECMC. GEMMS offered a unique business concept in providing the service of loan underwriting to mortgage lenders. Rather than having to maintain a full staff of underwriters to process loans, a lender could contract with GEMMS to process and underwrite the lender-originated loans. This arrangement would permit lenders to handle the resource strain of a peak market. A technical challenge facing this concept was that each lender has their own investment guidelines against which loans were to be underwritten. After servicing two or three lenders, it became clear that not even a superhuman underwriter would be able to keep up with the customized, voluminous and often-updated investor guidelines from several lenders at once. The idea came out of necessity for an automated compliance checker that could manage the information overload of a large number of investor guidelines for the GEMMS underwriters.

GECCO is embedded in the VISION loan registration and processing system, a client/server application which uses a GUI to capture the loan application data elements and stores the information in a relational database [Figure 1]. The underwriter can invoke GECCO at any point in loan processing and may rerun on the same loan at different stages in the pipeline as new information is entered. GECCO reports a status of either *in compliance* or *out of compliance* and provides explanatory messages when the latter is the case. The messages are divided into two classes - fatals and warnings - to distinguish between hard and soft guideline constraints.

Prior method of operation

Manual compliance checking for loan underwriting generally can be a tedious, time-consuming and information-intensive process. Prior to development of GECCO, the compliance checking process was performed manually at GEMMS. The manual process involved checking compliance of a loan application file by thumbing through a bookshelf of documents that specify the details of the investor guideline for the relevant loan program. On average, this process took between thirty minutes to an hour per loan.

A rule-based approach was adopted as a natural way of modeling the manual process of guideline compliance checking, based on identifying the specific eligibility constraints which apply to a given loan.

Copyright © 1995 Software Engineering Press

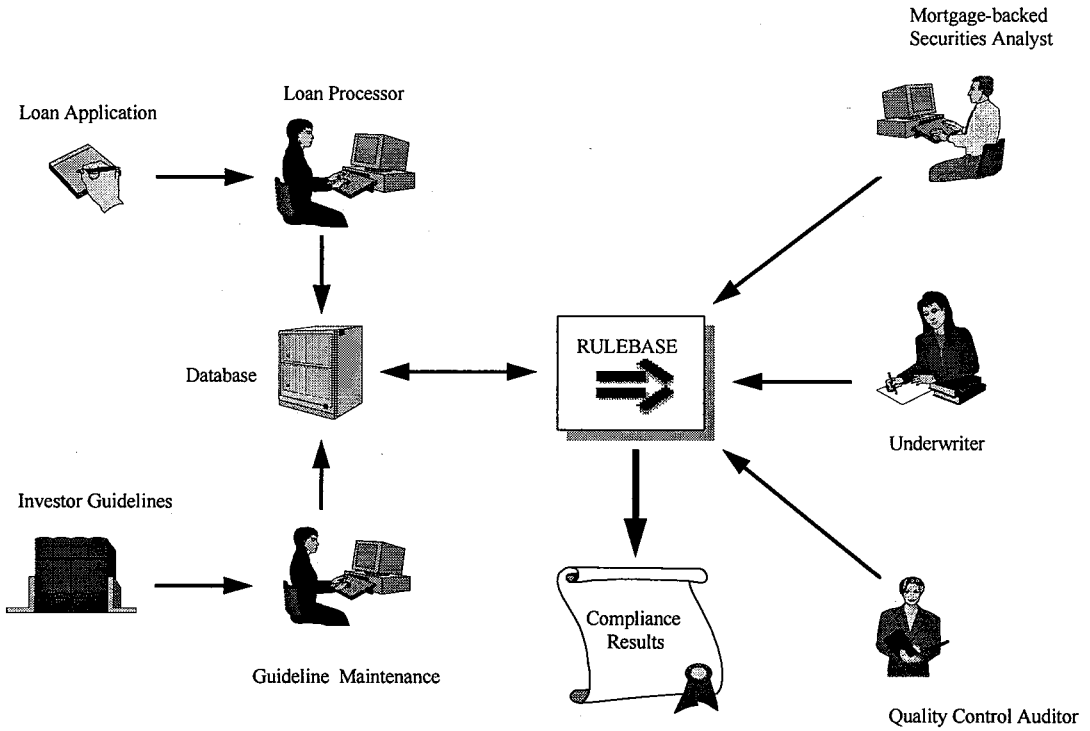


Figure 1 : Compliance Checker Process Overview

Objectives and Benefits of the GECCO Compliance Checker

Several objectives were identified for the original GECCO application within the initial third-party GEMMS application. These included:

- *Produce a flexible investor guideline modeling tool* -- The initial objective of GECCO was to make it possible for a mortgage loan analyst to underwrite a loan for any mortgage lender subject to salability constraints of any investor guideline. This was achieved through an architecture for flexible modeling of an unlimited number of guidelines. The third-party underwriter deployment of GECCO contains over 170 investor guidelines for 70 mortgage lenders.

- *Improve quality of the underwriting process* -- Underwriters tend to look for middle ground within the more flexible aspects of an investor guideline, offsetting a potentially unacceptable factor with other mitigating factors. The goal of GECCO was to provide consistency to the underwriting process, by standardizing both the process of applying the guideline constraints to loan applications as well as the content of the compliance warning messages.

- *Gain a productivity benefit* -- Raise business volume and shorten turn-around time by reducing the time required to process and underwrite a

Copyright © 1995 Software Engineering Press

loan. GECCO runtime of under 30 seconds is a significant time savings compared with the 45 minute manual compliance checking process.

- *Provide a remedial course of action* -- GECCO gives an explanation of which guideline rules are being violated, and provides suggestions of what must be done to correct a variance.

- *Tracking and monitoring capability* -- GECCO provides an audit trail for underwriting, and keeps track of documents required by lenders for closing the loan. This information management reduces the burden of knowledge placed on the analyst.

Versatility of the Application

The early success of GECCO for third-party processing and underwriting raised awareness of other ways in which the rule-based application could be applied to other GECMC businesses in which the questions "Should we buy these loans?" or "Can we sell this loan?" are central to the process.

Several of the possibilities identified for GECCO reuse were:

- *Secondary market/ post-closing (wholesale pricing)*: Batch process a pool of loans to filter out non-compliant loans prior to purchase for repackaging in a mortgage-backed security investment instrument. This filtering process determines that each loan that has already been closed does in fact meet the guidelines of an investor to whom the loan is to be sold. Should a loan fail to comply with investor guidelines at this point, the lender must keep the loan until it is seasoned (i.e., hold and service the loan for a period of time until salable under a different guideline). Each lender may have a customized contract with Fannie Mae and/or Freddie Mac. In addition, lenders have their own group of investors who provide the base of working capital for funding the lender's mortgage originations.

- *Point of sale/prequalification*: At this point in the loan application process, the borrower is volunteering data about employment, assets, liabilities and credit history. Though the information has yet to be verified, it is still useful for a lender at a point-of-sale branch of a mortgage company to run GECCO as an early screening mechanism for fitting the applicant into the best available loan program for which he or she qualifies.

- *In-depth processing and underwriting*: Determine and track receipt of the required documents, and grant conditional approval subject to satisfactory resolution of any pending items.

- *Closing*: As a final quality assurance check. Use GECCO to maintain an audit trail of warning or fatal messages that have been issued at points during loan processing.

- *Quality assurance*: Pull a representative sample of loans from the portfolio to measure the underwriter judgments against the known outcome of the loan. Used in this manner, GECCO can alert an underwriting staff to an undesirable trend if for example loans of a particular type (e.g., condominiums in California) are being rejected by a lender's investors, contrary to what the underwriters had thought to be in compliance with investor guidelines.

Copyright © 1995 Software Engineering Press

From the above list of candidate applications for GECCO reuse, the following have been implemented to date:

GECCO / Wholesale Pricing

A partitioning of the existing GECCO knowledge base was made to facilitate a need for integration of compliance checking capability into the Wholesale Pricing and Registration [WHOPR] project at GE Mortgage Services [GECMSI]. The WHOPR project purpose was to build a rule-based system that would conditionally apply a variety of price adjustment factors to loans delivered to GECMSI for purchase from originating lenders. Determination of compliance with GECMSI resale guidelines was a precondition for loan purchase. The pricing rules would determine the purchase price of a loan once it had passed the compliance check.

The GECCO rules were accordingly divided into checks for lender compliance, used by GEMMS third-party processing, checks for wholesale pricing, and a group of core checks used in both businesses.

GECCO / Loan Closing

The Closing project was the GEMMS follow-on to third-party processing. The primary focus for GECCO in this effort was to expand the processing capabilities with specific functionality needed for delivery of loans to "the closing table." This functionality was achieved with the addition of two new GECCO rule sets: the special conditions rules and the required documents rules.

Special conditions are extra procedures that may be placed on a loan by the lender guidelines (e.g., self-employed borrowers must complete a Fannie Mae form for self-employed, and must provide proof of income for the past two years). A loan may be in compliance subject to special conditions which have been attached automatically by GECCO. The required documents rules generate a list of all documents needed for closing as specified by lender and investor guidelines, and keep track of which documents are still outstanding. Examples of these documents are : proof of flood insurance if the subject property is in a flood zone, or a special form for energy-efficient dwellings in the state of New Mexico.

GECCO Functional Description

GECCO is integrated into a loan entry and processing system, which consists of a GUI component for capturing and updating the loan information and a client/server relational database for efficient data storage. When an underwriter or loan processing agent invokes GECCO, the following process takes place:

Copyright © 1995 Software Engineering Press

1. Retrieval of the relevant data elements for the loan application into structured objects.
2. Retrieval of the investor guideline information for the corresponding loan program (e.g.,, 15-year fixed rate) into structured objects.
3. Activation of the appropriate GECCO rule sets, controlled via command line arguments passed to GECCO from the GUI.
4. Agenda-driven rule processing of the loan and guideline data.
5. Results are posted to the GECCO output screen [Figure 2] and to the database.

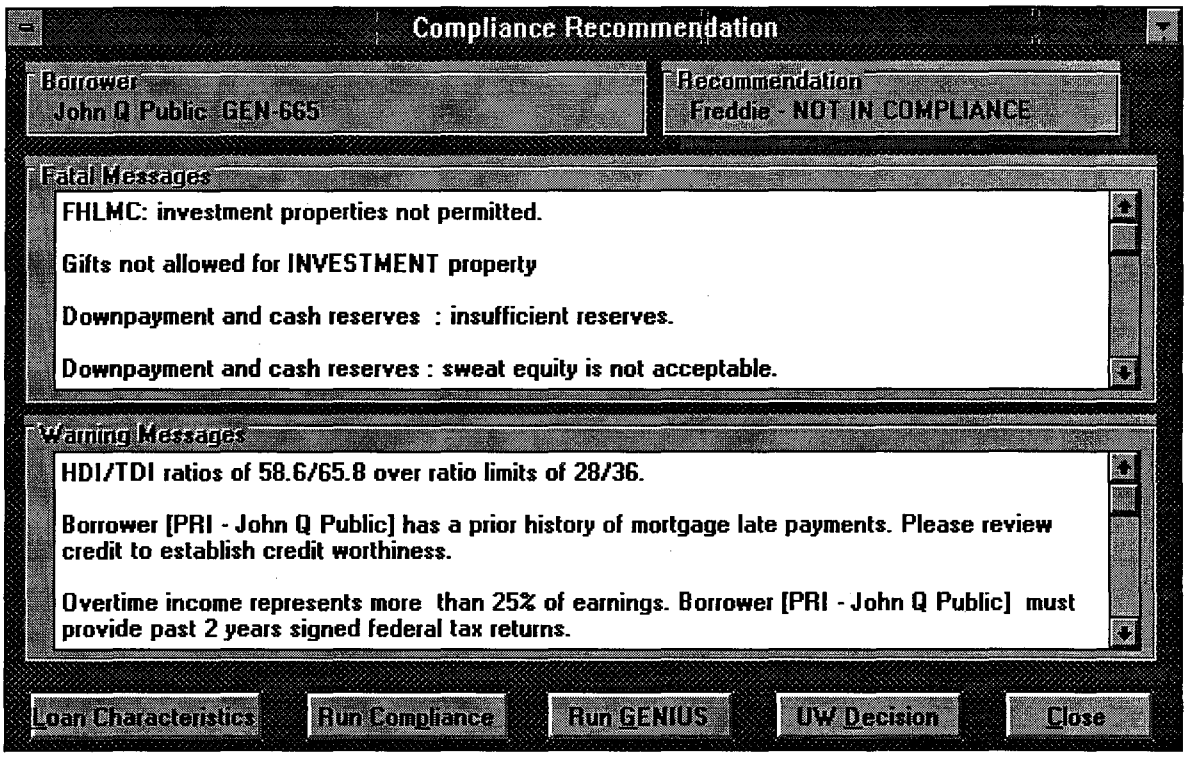


Figure 2 : Compliance Output Screen

The guideline information is input initially by a designated guidelines administrator whose job is to ensure that the GECCO knowledge base is kept current with new guidelines and guideline updates as they are released by investors. A graphical guideline editor screen facilitates this process of defining and modifying guideline information.

Copyright © 1995 Software Engineering Press

The GECCO rulebase contains rules for the following categories of salability checks:

- ARM restrictions
- Bankruptcy / foreclosure procedures
- Downpayment / cash reserve requirements
- Mortgage insurance requirements
- Loan amount limits
- Limits on seller contributions
- Cashout refinance restrictions
- Appraisal requirements
- Non-base income limits
- LTV limits
- Employment-related checks
- Limits on gifted funds
- Qualifying ratios
- Second home/investment prop. restrictions
- US Citizenship status

The GECCO output consists of warning messages that relate soft guideline constraints and fatal messages that correspond to violation of strict guideline rules. In addition, GECCO provides processing packets which give the user suggestions on how to correct specific variances. The user always has the option to override a GECCO *out of compliance* overall result on the basis of compensating factors which may not be accounted for in the knowledge base; in this case, the loan is forced into compliance, and the user is encouraged to provide a justification for the override in a notes area.

System Architecture

Software development for GECCO / Third-party underwriting

The GECCO software development process followed an iterative spiral methodology, building on a series of prototypes to produce the first deployed system. The project began with collection and analysis of user requirements, followed by definition of the object model for investor guidelines and for the necessary loan application data. Next the database access layer was designed, followed by high level design of the compliance rules. The database access layer is the underpinning of the knowledge base, encapsulating the objects and rules from the physical implementation of the database that houses all of the loan application data as well as investor guideline information.

The GECCO knowledge base consists of a collection of objects which hold the relevant data elements referenced in the guideline checks and a collection of discrete rules.

Software development for GECCO / Wholesale Pricing

The project began with an analysis of rules currently enforced by GECCO. These rules were reviewed by business analysts and grouped into distinct sets for third-party underwriting, for wholesale pricing, and for both. The knowledge base

Copyright © 1995 Software Engineering Press

was then partitioned accordingly and the GECCO calling interface was modified so that the activation of individual rule sets could be controlled via external command line arguments.

The reorganized GECCO was regression-tested on a database of test loans and then redeployed to both GEMMS for third-party processing and GECMSI for wholesale pricing.

Software development for GECCO / Loan Closing

The feedback from lenders for enhanced processing capabilities further advanced GECCO functionality. The new development began with analysis of the types of special conditions that lenders wanted to be able to attach to loans and of the types of documents that conceivably could be required. The analysis was followed by a design specification for the new database and knowledge base structures required for storing and modeling both special condition and required document information. Business analysts then gathered all of the specific business rules for both from each GEMMS lender. The rules were then implemented in separate rule sets, one for special conditions and one for required documents, and these new rule sets were then added to the GECCO knowledge base, bringing the number of rules up to 230 in four distinct rule sets.

Hardware and software environment

The identical hardware and software environment applies to the third-party processing and underwriting, wholesale pricing, and loan origination deployments of GECCO:

GECCO is written in ART-IM and is deployed as a C executable on a client/server architecture.

Operating system	: OS/2 2.1
Database	: Sybase 4.9.1
Software	: ART-IM 2.5, Microsoft C 7.0, Sybase DB Library for C
Hardware	: Client : 486 desktop, Server Sun Sparc-20
Network	: Local : Token ring/Novell 3.11, Wide-area : T-3 link

For integration with GENIUS™, the latest development effort has resulted in redeployment of GECCO for Windows, for both client/server and single-user environments.

Operating system	: Windows 3.1
Database	: Watcom SQL Server
Software	: ART-IM 2.5, MS Visual C, Watcom libs, MS Visual Basic
Hardware	: Client/Server or standalone : 486 desktop
Network	: site-specific

Copyright © 1995 Software Engineering Press

Description and significance of AI techniques used

The GE Compliance Checker is an excellent candidate for a rule-based solution because the processing involved fits the model of asynchronous testing of a large number of discrete logical conditions. This approach was facilitated by using a highly optimized inference engine and a powerful pattern-matching rule language. The specific architectural features, and their significance as employed by GECCO, include the following:

- *Agenda-driven inferencing* fires only for rules that apply as opposed to exhaustively testing for each rule sequentially.
- *Partitioned rule sets* provide the flexibility of customizing the behavior of the compliance checker accordingly with the specific eleigility constraints of the selected investor.
- *Ruleset partitions* also serve to make actual distinctions in the business model explicit, so that categories of rules (e.g., credit checks, property checks) may be easily enabled or disabled through the GUI.
- *Selective, optimized database access* is controlled through the rules -- only data required by the given process/ruleset is loaded.
- *Platform independence* is achieved by insulation of the knowledge base from the platform and the database (See Figure 3). A generic ART-IM/SQL interface was developed for GECCO. The interface is based on functions which define mappings between the physical database model and the object model created in ART-IM. The accessor function names are properties of the ART-IM objects and are analogous to create-instance methods, called to generate instances of ART-IM schema. The ART-IM/SQL interface used in GECCO is robust in that it permits arbitrarily complex object mappings to be written and compiled incrementally. The ART-IM deployment compiler which generates compilable "C" code for the complete contents of the knowledge base, thus allowing platform independence.
- *Rule parameterization*, the separation of declarative knowledge, allows the user to customize compliance rules dynamically through the GUI maintenance screens without requiring a change to the executable.
- *Object-oriented inheritance* allows GECCO to exploit the large overlap of information contained in investor guidelines, so that only the differences need to be stored. This method produced a drastic performance improvement as well.
- *High level rule language* permits a straightforward software coding of a compliance business rule; this high level representation not only makes the rules easier to write and debug, but has a positive impact on maintenance of the rulebase as well.

Copyright © 1995 Software Engineering Press

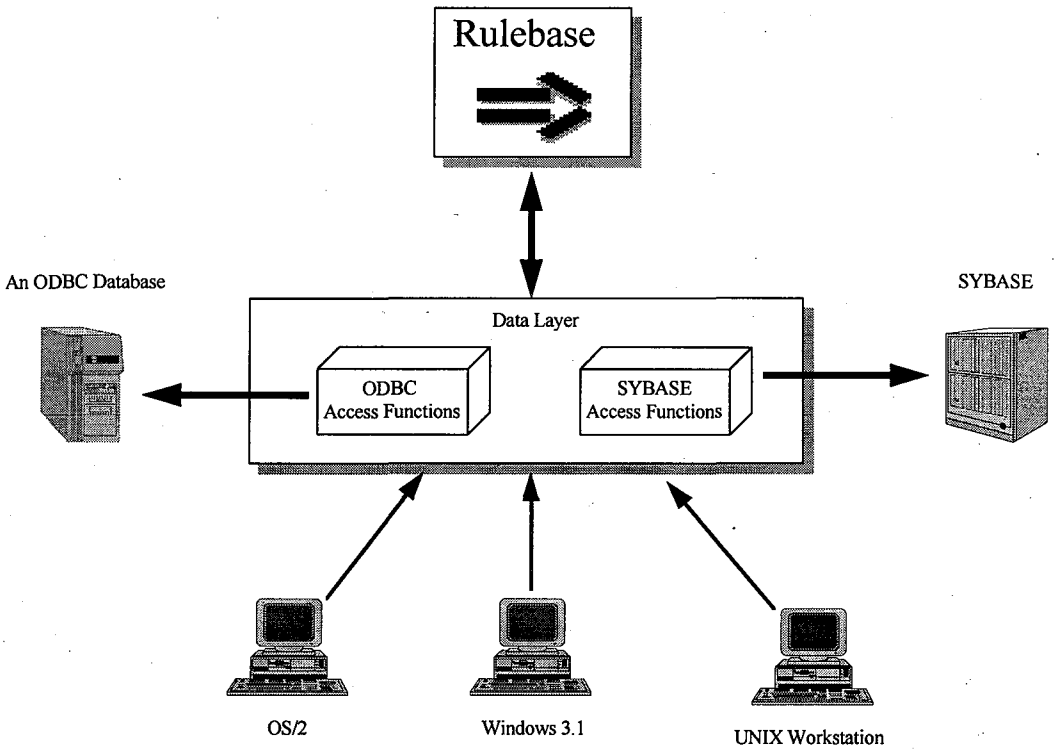


Figure 3: Platform - Independent Architecture

The following code fragment from the knowledge base gives the flavor of a typical compliance checker rule:

```
(defrule check-geographic-non-conforming-loan-amount
  "Check if the loan amount exceeds the guideline limit for the given type of
  dwelling in the given state."
;; Match a property instance
(schema ?
       (instance-of property)
       (prop-dwelling-type ?dwl-code)    ;;  (e.g., single-family, duplex,...)
       (prop-state         ?state))      ;;  (e.g., NY,FL,CA...)

;; Bind the loan amount
(schema ?
       (instance-of loan-application)
       (loan-amount ?loan-amt))

;; Check if the loan amount exceeds the guideline loan amount limit
;; for the matching property state and dwelling type.
(schema ?
       (instance-of geographic-loan-limit)
       (geo-loan-limit-dwelling-type    ?dwl-code)
       (geo-loan-limit-property-state  ?state)
       (geo-loan-limit-amount     ?loan-limit&:(< ?loan-limit ?loan-amt))
       )

=>
```

Copyright © 1995 Software Engineering Press

```
(bind ?msg (sprintf "Loan amount of $%ld over limit of $%ld."
                    ?loan-amt ?loan-limit))
;; Post a fatal message for exceeded loan limit.
(gcc-message FTL ?msg)
)
```

GECCO In Production

GECCO / Third-party processing and underwriting

The GECCO third-party underwriting deployment contains over 170 investor guidelines for 70 mortgage lenders. The knowledge base contained approximately 120 guideline compliance rules. The usage data for the fiscal years 1993-4 shows that GECCO was run on approximately 15,400 mortgage loans, with an average weekly volume of 150 loans. During this period GECCO was run over 52,000 times - over 3 times per loan on average.

Because the loan application information can fluctuate during the 20-30 day window during which a loan file is typically open for processing, it is useful to rerun the Compliance Checker at various points during the process to ensure that important changes are noted, such as with borrower's income, employment status, assets, or credit profile, and their consequences for the salability of the loan taken into account. It is also often customary for GECCO to be run at any point when the loan file passes between mortgage loan processors, as a safeguard against possible oversights.

GECCO / Wholesale Pricing

Usage data for GECCO/Wholesale Pricing reports that for the period from February through December 1994, GECCO has been run on over 7000 loans, of which 1700, or roughly 1/4 were found to violate GECMSI guidelines for purchase.

GECCO / Loan Processing and Closing

The addition of the required documents ruleset and the special conditions ruleset grew the knowledge base to 230 rules. Some database optimizations resulted in a performance improvement which reduced GECCO runtime from 40 to 25 seconds per loan on average.
GECCO/ Loan Processing has been run on over 5000 loans for the six-month period from July through December, 1994.

Copyright © 1995 Software Engineering Press

Application Payoff

Improved throughput and productivity:

The time savings of at least 30 minutes per loan (45 minutes average) directly increased the number of loans processed and underwritten by GEMMS, GECMSI and outside lenders. Comments from underwriters who have used GECCO suggest that they have noted a greater feeling of accomplishment due to increased productivity, that exposure to state-of-the-art information technology makes their job more interesting, and that they have more confidence in the consistency of their work.

Improved risk management

GECCO produces a consistent evaluation of compliance with investor guidelines; this helps to equalize the disparate levels of underwriter experience and gives standardization to the subjective art of underwriting. The result of better quality of underwriting is a stronger portfolio and a fewer number of loans refused by the investor.

Improved customer service:

GECCO for third-party underwriting provided a high-quality and timely level of customer service to mortgage lenders who dispatched loans to GEMMS for processing and underwriting. GECCO is now offered directly to lenders as a commercial product for assessing salability of their mortgage loans.

Improved business process:

GECCO /third-party underwriting made it feasible for GEMMS to pursue the loan processing overflow market.

GECCO /wholesale pricing allowed GECMSI agents to quickly approve or reject purchase of individual loans in a package for resale on the secondary market.

GECCO / PC GENIUS™ allows GECMC to offer their customers a Windows® product for assessing investor requirements for mortgage loan resale.

It was not conceived at the time that the initial effort to build a compliance checker for third-party underwriting would result in a tool that is germane to so many different aspects of the mortgage loan business process. GECCO exploits the overlap of data used across the processing pipeline while providing the flexibility to apply the set of business rules distinct to each of several specific phases, including conditional prequalification at origination, underwriting for salability and for mortgage insurance, processing of required documents, appraisal and credit reports, and quality control.

Copyright © 1995 Software Engineering Press

Present and Future Work

It is expected that the flexibility gained through the GECCO application will continue to have a positive impact on the efficiency of several key GECMC businesses. The GENIUS™ product will continue to be enhanced and supported as driven by external customer needs.

Plans are underway to rearchitect the GECCO data access layer so that it can be available as a plug-in tool for new and existing GECMC software applications with a minimum of integration work Currently, the GECCO guideline maintenance screens are being expanded to allow authorized users to modify existing rules and to define new ones as new mortgage products are offered in the market place. This flexibility to represent any investor's eligibility criteria has generated interest in use of GECCO to build custom portfolios automatically..

The National Processing Center is a major reengineering project now in development at GE Capital Mortgage Insurance; the NPC is focused on streamlining the process so that the mortgage insurance underwriting step may be performed as quickly as possible. The combined capabilities of GECCO and GENIUS™ will be the heart and soul of NPC operations. A new GECCO ruleset for compliance with GECMC Mortgage Insurance guidelines is being defined for this effort. New guidelines are also being defined to increase GECCO functionality for FHA,VA and affordable housing loans.

Summary

GECCO has proven to be an effective tool for the underwriter who must balance the need to bring in new business against the imperative of keeping a portfolio of salable loans. GECCO also gives the benefit of applying standardization to a process that in practice is subjective and prone to variances introduced by different underwriting philosophies.

The reuse of the GECCO as a generic tool for loan salability assessment, prequalification, underwriting, and quality assurance has been a remarkable success for GE Capital Mortgage Corporation.

Copyright © 1995 Software Engineering Press

Acknowledgments

The authors thank Tony Keller, Diane Lecco, and Stan Patterson of GECMC for their useful discussions about the project. In addition, special thanks to Phil Klahr of Inference for his helpful comments on the paper.

List of Figures

References

[Den92] Dennis, Marshall W. *Residential Mortgage Lending, Third Ed.* 1992. Prentice Hall Inc. Englewood Cliffs, NJ.

[FNM93] *Basics of Sound Underwriting.* 1993. Fannie Mae Customer Education Group, Washington, DC.

[FHL94] *FHLMC Underwriting Guidelines Quick Reference.* October, 1994. MRI.

[FNM94] *FNMA Underwriting Guidelines Quick Reference.* October, 1994. MRI.

[GEC93] *GE Capital Mortgage Insurance Underwriting Manual.* July, 1993. GE Capital Mortgage Insurance Corp. Raleigh, NC.

[Gol95] Golibersuch, David et al. *GENIUS, Combining Knowledge Engineering and Machine Learning to Achieve Balanced Risk Assessment in Mortgage Insurance.* Proceedings of the IAAI '95 Conference, to be published in August 1995, Montreal, Quebec.

[Pow90] Powell, Lynn S.. *Residential Mortgage Banking Basics.* 1990. Mortgage Bankers Association of America, Washington, DC.

[Tal94] Talebzadeh, Houman et al. *Countrywide Loan Underwriting Expert System.* 1994. IAAI 1994 Proceedings.

Copyright © 1995 Software Engineering Press

Paper Session: Data Analysis, Modeling, and Representation

Chair: Ypke Hiemstra, Vrije Universiteit Amsterdam

Software for data analysis with graphical models

Wray L. Buntine
RIACS at NASA Ames Research Center
Mail Stop 269–2, Moffett Field, CA 94035–1000, USA
wray@kronos.arc.nasa.gov

H. Scott Roy
Heuristicrats Research, Inc.
1678 Shattuck Avenue, Suite 310, Berkeley, CA 94709-1631, USA
hsr@heuristicrat.com

Abstract

Probabilistic graphical models are being used widely in artificial intelligence and statistics, for instance, in diagnosis and expert systems, as a framework for representing and reasoning with probabilities and independencies. They come with corresponding algorithms for performing statistical inference. This offers a unifying framework for prototyping and/or generating data analysis algorithms from graphical specifications. This paper illustrates the framework with an example and then outlines a software toolkit that allows rapid prototyping of data analysis algorithms. Tools available for this task including methods from Bayesian networks, statistics, and neural networks.

1 Introduction

This paper argues that the data analysis tasks of learning and knowledge discovery can be handled using techniques for graphical models, and therefore rapid prototyping of data analysis algorithms can be done. The ability to adapt data analysis or learning algorithms to the user's application is a key capability for the financial markets, for instance in knowledge discovery[1]. This meta-level use of graphical models for learning was first suggested by Spiegelhalter and Lauritzen [25] in the context of learning probabilities for Bayesian networks. An extension of the standard graphical model is used here that allows this kind of learning to be represented. The extension is the notion of a *plate* introduced by Spiegelhalter[2]. Plates allow samples to be represented explicitly on the graphical model, and thus reasoned about. This makes data analysis problems explicit in much the same way that utility and decision nodes are used for decision analysis problems [23].

Consider, for instance, Figure 1. This presents a situation where a mixture model with hidden variable *class* is used for subsequent prediction of var_1 from var_2 and var_3. The part to the left of the parameters ϕ and θ is the graphical representation of a sample. The contents of the *plate* (the box around the nodes for *class*, var_1, var_2 and var_3) on the left indicates that a sample of N cases with variables var_1, var_2 and var_3 are given, while

[1] For instance, Evangelos Simoudis cites this as one of the key selling points of the knowledge discovery tool RECON from Lockheed AI Center.

[2] Personal communication. The notion of a "replicated node" was my version of this developed independently. I have adopted the notation used by Spiegelhalter and others for uniformity.

Copyright © 1995 Software Engineering Press

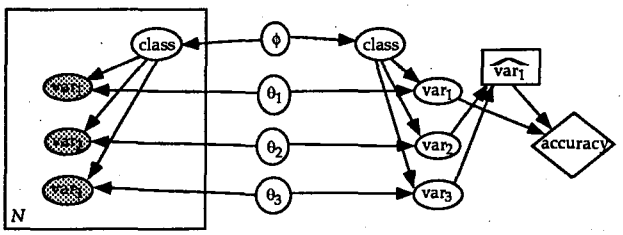


Figure 1: Simple unsupervised learning, with general prediction

class is hidden, being unshaded. The plate indicates that its contained subgraph is replicated N times. The part on the graph to the right of the parameters ϕ and θ represents the prediction task. The value node on the right, the diamond, indicates that subsequent prediction accuracy is the goal of learning. Together, this graph indicates that the utility for the problem is $(var_1 - \widehat{var_1}(var_2, var_3))$, and the joint distribution of the parameters takes the form

$$
\begin{aligned}
&p(\phi, \theta_1, \theta_2, \theta_3, class, var_1, var_2, var_3, \\
&\quad class_i, var_{1,i}, var_{2,i}, var_{3,i} : i = 1, \ldots, N) \; = \\
&\quad p(\phi)\, p(\theta_1)\, p(\theta_2)\, p(\theta_3)\, p(class|\phi) \\
&\quad p(var_1|class, \theta_1)\, p(var_2|class, \theta_2)\, p(var_3|class, \theta_3) \\
&\quad \prod_{i=1}^{N} p(class_i|\phi)\, p(var_{1,i}|class_i, \theta_1) \\
&\qquad\qquad p(var_{2,i}|class_i, \theta_2)\, p(var_{3,i}|class_i, \theta_3) \; .
\end{aligned}
$$

There has been a recent push within the machine learning and neural network communities to dispel the magic and art from the various learning fields and present them more as engineering disciplines. Decision tree methods [4] and feed-forward networks [18, 6] are some examples that show how already popular algorithms can be re-engineered from well understood principles of probability in combination with the knowledge representation and standard search methods. A simple connectionist feed-forward network (using the notation of Hertz, Krogh and Palmer [14]) and its corresponding Bayesian network is given in Figure 2(a) and (b) respectively. Similarly, other

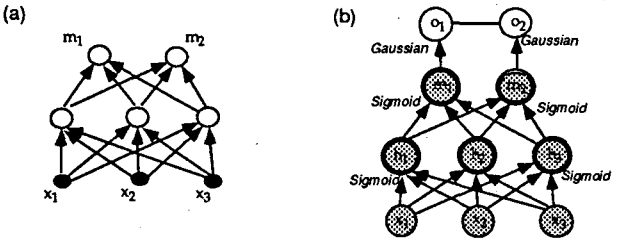


Figure 2: A simple feed-forward network: (a) in native form (b) as a DAG.

neural networks can be modeled with graphical models ("probabilistic networks").

This general approach, engineering using principles of probability, is now becoming widespread. The basic tools of probabilistic (Bayesian) inference used for this process are reviewed, for instance, by Tanner [26], Kass and Raftery [16], Neal [21], and Madigan *et al.* [19]: various exact methods, Markov chain Monte Carlo methods such as Gibbs sampling, the EM algorithm, and the Laplace approximation. With creative combination, these are able to address a wide range of data analysis problems. Gilks, Spiegelhalter and Thomas have taken this process a step further by developing a compiler that generates Gibbs samplers from graphical specifications [10]. This handles a surprisingly broad number of statistical tasks.

It is the thesis of this paper that these techniques are now sufficiently well developed so that software support can be provided for their use in data analysis problems. That is, we are now able to generate components of data

Copyright © 1995 Software Engineering Press

analysis algorithms, and even entire algorithms themselves from high-level specifications. The paper demonstrates the thesis by presenting a framework based around the use of graphical models as a specification language.

We begin with two examples. The first illustrates the intended use of the software we envisage, and the second gives some more mathematical detail. Then we outline in more detail the specification language we are developing. A low-level implementation of these ideas demonstrates the feasibility of our software [22]. More details of the underlying theory for our approach can be found in [3], including results for deterministic nodes and techniques for doing differentiation, both used in modeling neural networks with probabilistic graphical models.

2 Two examples

The software we are developing is intended to be used in an iterative prototype-refine cycle using standard data manipulation and visualization packages such as Matlab, PV-Wave/IDL, or S-Plus. An important observation is that prepackaged data analysis software such as clustering, linear regression, and feed-forward neural networks are sometimes inadequate for the particular task at hand. While these packages are often good for exploratory data analysis, our experience and that of many others indicates that data analysis and knowledge discovery requires more flexibility in general. The first example below illustrates the kind of prototyping our envisaged software is intended to assist, and the second example illustrates some more of the mathematical detail.

2.1 Prototyping data analysis

This example will demonstrate how the system we are developing would operate, reducing a problem that might require weeks of effort into an afternoon's work. Figure 3 plots the raw data for this example. The data give mean

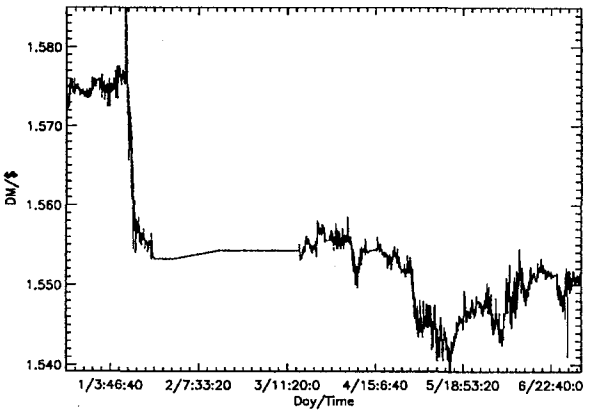


Figure 3: The mean bid-ask price for DM/$.

bid-ask prices posted by banks at various time points over the course of a week for turning dollars into Deutsch marks. The mean bid-ask price (average of the two) is a more stable indicator of the bank's pricing position because the bid or ask price alone also includes effects due to the banks policy on the bid-ask spread. The mean bid-ask price helps reduce the artifacts of "returning to the mean" and "stable patterns" reported in the bid price data or the ask price data alaone [20, 1]. Original data takes the form of a date and time, the bid and asking price, and the bank code.

Date	Bid	Ask	Bank
Sep 1 13:42:40	1.5737	1.5742	CONY
Sep 1 13:42:45	1.5735	1.5745	MGTX
Sep 1 13:43:14	1.5735	1.5740	BBIX

Copyright © 1995 Software Engineering Press

Our goal is to model the time series and to understand individual differences among the banks. The data we have at our disposal consists of the tick data in Figure 3 together with various properties of the banks, such as their geographical location.

We hypothesize that the tick data is effectively a random walk, but where the percentage change at each time point is influenced by the bank posting the price. For example, we might suspect that some banks tend to post larger differences from the previous tick than the average change, or that some banks post more frequently during upswings than downswings, so that the ticks posted by such a bank run contrary to the downward trend. Figure 4 shows the kind of thing we are after, plotting the empirical frequency of percentage changes for all the ticks, for a bank that only posts large changes from the previous tick, and for a large bank that posts many changes.

We sit down at our data analysis system, pull in the raw data, and set up a quick model to do an unsupervised clustering of the banks. Our first pass uses the random so that we can get a basic feel for the different kinds of banks. For each bank we have:

- The mean of the bank's bid-ask spread.
- The bank's geographical location.
- The average number of posts the bank generates per day.
- The massaged tick data giving the bank's relative price change over the immediately preceding price (probably posted by a different bank).

The graphical model, shown in Figure 5, is created using a drawing tool. In this model, the relative change that a bank will offer is assumed to be determined by its class, but is

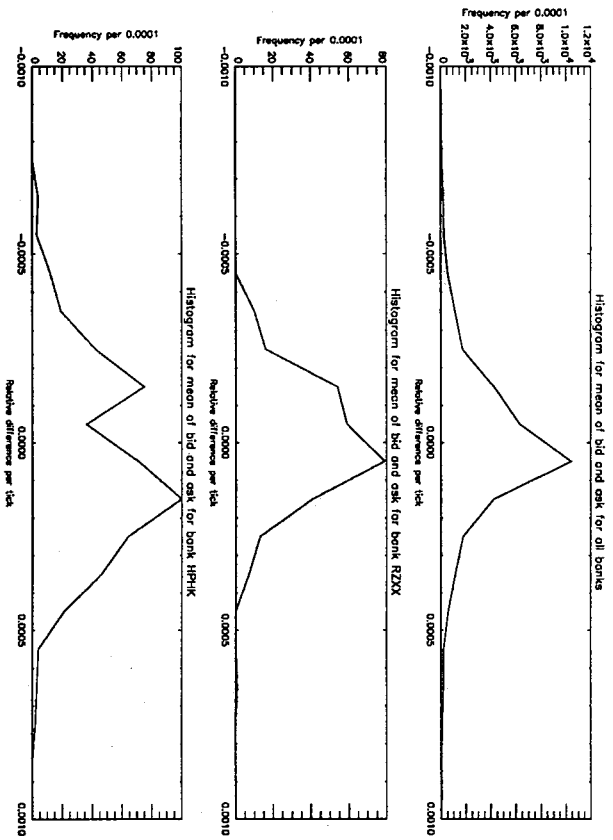


Figure 4: Frequencies for different relative price changes.

Copyright © 1995 Software Engineering Press

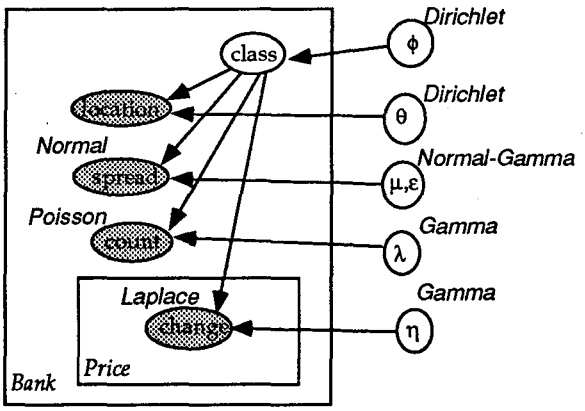


Figure 5: Basic clustered random work model for price changes.

otherwise a random walk. Notice this model has multiple banks, and each bank gives multiple prices, so this model has multiple plates indicated a double product occurs in the full joint distribution for the data (over banks and over the bank's price changes). For the current problem we simply drag a prefabricated mixture model and appropriate components from a palette or component library into the work area, and make some modifications to it. The model should include complete specification of all distributions and parameters (e.g., all parameters of the Dirichlets in Figure 5 be supplied). For instance, we would have to set the various parameters for the conjugate priors appearing in the model. The model of Figure 5 is not a standard clustering model so could not be obtained from any standard statistical package. In other problems we could create a free form model by drawing individual nodes, links, and probability annotations. The drawing tool contains the necessary hooks to associate variables on the plate with the fields from the bank database, and the computed fields from the tick database.

We now press the RUN button. At this point the system performs all the drudgery such as mathematical calculations, programming, and validation, previously requiring weeks of effort:

- It performs known symbolic simplifications on the graphical model. For instance, it knows about sufficient statistics and some closed form solutions to expected values.
- It computes all required derivative functions.
- It chooses an optimization algorithm.
- It finds the parameter values of maximum posterior probability, together with the Bayes factor and the Hessian of the posterior evaluated at the final parameters.
- It generates optimized C code to evaluate the model by performing a data flow analysis over the needed computations.

Now we could provide the system with an algorithm scheme, such as EM or Gibbs sampling (additional examples are given later), and have the system come up with the necessary code for the derivatives, expected values, probabilities, and so forth. However, in this case the default algorithm matching the graph is good enough.

We can now analyze the final model to see what it tells us, for instance using available visualization tools. (All this is make believe.) The classes that it finds are natural ones we might expect. The banks are broken into different classes according to whether they are closer to the New York or London markets. Banks that post infrequently tend to have a higher bid ask spread than those that post often. Each of these groupings also has different random

Copyright © 1995 Software Engineering Press

walks for their pricing. Some smaller banks, for instance, tend to make larger changes.

We now go back to the drawing tool and refine the model to account for the context of the tick data. In the previous model, the relative change that a bank will offer is assumed to be determined by its class, but is otherwise a random walk. In this case we also make the price sensitive to the current average trend which is dependent on previous prices. This new model is given in Figure 6. Having drawn the new

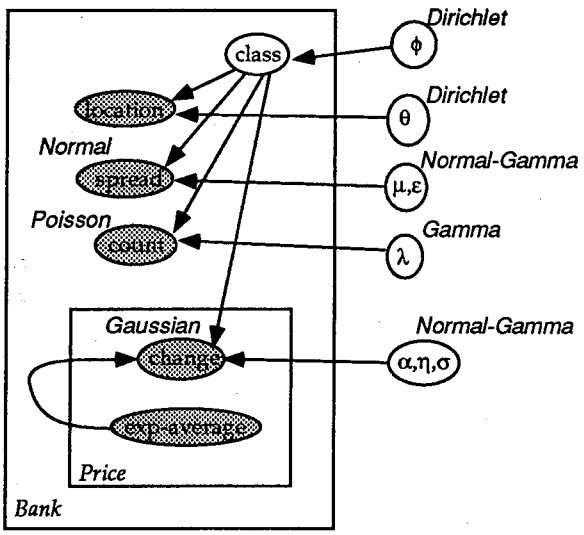


Figure 6: Trend sensitive model for price changes.

model, we simply click the RUN button and let the system re-optimize for the new configuration. This time, we could use the previous classification got as the initial values for the new extended model.

For this more complex model, we would probably have to modify the default algorithm scheme suggested by the system. This is something we expect in general, so the system includes both a graphical model and a general algorithm scheme as inputs. If the algorithm scheme is missing, the system can provide a default using general purpose algorithms such as Gibbs, EM or MAP algorithms.

2.2 A more detailed example

An example of a graphical model for a simple clustering problem is given in Figure 7. To be

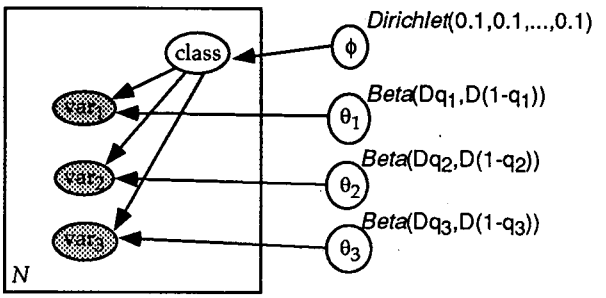


Figure 7: A simple unsupervised learning problem.

explicit, we also have to give the full distributions for the variables in the graph. Assume there are 10 classes, so $class \in \{1, 2, \ldots, 10\}$, and the variables var_i are binary. The graphical model is a mnemonic for the following distributional assumptions for the j-th case being

$$\begin{aligned} var_{i,j} &\sim \text{Bernoulli with prob. success } \theta_{i,class_j}; \\ class_j &\sim \text{10-dimensional Multinomial with} \\ &\qquad \text{probabilities } \phi_1, \phi_2, \ldots, \phi_{10}; \end{aligned}$$

and for the parameters ϕ and θ being

$$\begin{aligned} \phi &\sim Dirichlet(0.1, 0.1, \ldots, 0.1)\ ; \\ \theta_{i,c} &\sim Beta(Dq_i, D(1-q_i))\ ; \end{aligned}$$

for the hyper-parameters D, q_1, q_2, q_3. Using an empirical Bayes approach, we set q_i to be the observed frequency of success for var_i and

Copyright © 1995 Software Engineering Press

set $D = 4$ to yield reasonable prior standard deviation for the possible values of θ_i.

A matching algorithm scheme for this model, an iterated EM algorithm in pseudo-code, goes as follows:

1. Repeat, 5 times.
   (a) Initialize the parameters ϕ, θ randomly according to their prior.
   (b) Repeat until the maximum relative difference in parameter values ϕ, θ is less than $1.0e-5$.
      i. Reassign the sufficient statistics for ϕ as follows:

$$ss(\phi) = \sum_{i=1}^{N} \mathcal{E}_{class_i|var_{1,i},var_{2,i},var_{3,i},\phi,\theta}(class$$

      ii. For $c = 1, \ldots, 10$, reassign the sufficient statistics for $\theta_{1,c}, \theta_{2,c}, \theta_{3,c}$ as follows:

$$ss(\theta_{j,c}) = \sum_{i=1}^{N} \mathcal{E}_{class_i|var_{1,i},var_{2,i},var_{3,i},\phi,\theta}(1_{class_i=c} var_{j,i})$$

      iii. Replace ϕ and θ by their MAP values given the sufficient statistics as above.
   (c) Compute the score for the final parameter values as

$$score = \det\left(\frac{\mathrm{d}^2 \log p(\phi, \theta|sample)}{\mathrm{d}(\phi, \theta)}\right)$$

2. Return the parameters ϕ, θ matching the best score.

The system would automatically compute the derivatives, expected values, MAP calculations, and so forth and insert the code efficiently into the major loops. Notice there are problems with this algorithm scheme, for instance, if a ϕ_i approaches zero the score will become ill-defined because some of the parameters will become redundant. This is irrelevant for the purposes of our illustration.

3 High-level specification of data analysis algorithms

An overview of the basic framework is given in Figure 8. Inputs to the system are a graphical model to specify the variables and their relationships, and an algorithm scheme to specify the algorithm. These two inputs can be patched together from libraries.

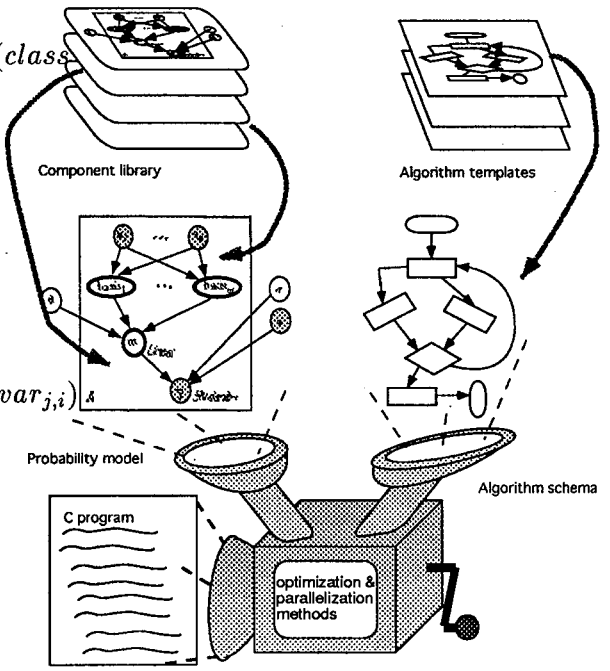


Figure 8: The basic framework for specification.

Copyright © 1995 Software Engineering Press

3.1 A language to specify variables and their relationships

Probabilistic graphical models (chain graphs [27]) extended with plates are used here as a specification language. When augmented with specific functional forms such as the Gaussian and the logistic, this language is sufficient powerful to represent a broad range of problems across several fields: generalized linear models, feed-forward networks, Jordan and Jacobs mixture of experts [15], unsupervised learning of many different kinds, and hybrids of these models. A review of some of the learning problems represented appears in [5].

A graphical model implies a probability model, thus it defines how probabilities, log-probabilities, their derivatives, and some expected values can be computed, and in some cases how sampling can be done. For a Gaussian or discrete Bayesian network, for instance, the usual exact computations are implied by the graph [24]. Techniques for computing these quantities are of course more complex in other cases. In general, exact methods for expected values are not known, and probabilities are not in general easy to compute for chain graphs and Markov networks, although ratios of probabilities are. The automatic calculation of derivatives on structures such as graphs is a well understood problem [11]. In neural networks, this corresponds to the Back-propagation algorithm and its extensions for second derivatives [7]. Likewise, the calculation of derivatives on probabilistic graphical models is an application of the chain rule for differentiation. Details appear in [3].

3.2 Algorithm schemes

Algorithm schemes are high-level algorithms used as input for code generation and compilation. The algorithm scheme refers to variables, probabilities, log-probabilities, derivatives, and expected values for items on the graph whose computation can be determined automatically. Thus the scheme is free of many of the cumbersome equational details found in most languages. Algorithm schemes themselves can be produced automatically from algorithm templates for problems matching the right preconditions, and in other cases may be provided or refined by the user. We do not give a specification of the scheme and template language itself here, but the reader can infer from the examples that the scheme language is a fairly standard procedural language with appropriate hooks into the matching graphical model. Some sources for algorithm templates are as follows:

- Gilks *et al* [10] have developed general algorithms to perform Gibbs sampling on Bayesian networks with plates.
- Other algorithms such as conjugate gradient, Fisher's scoring method, or Laplace approximations [16] can be applied once first and second derivatives are calculated for model parameters.
- Lauritzen describes the application of the EM algorithm [9] to Bayesian networks with a single plate [17] in the context of missing values. The more general application of the EM algorithm for hidden variables is obvious, as for instance done in unsupervised learning [12].

4 General discussion

One task that can never have direct software support is the design of an appropriate model with an appropriate prior. This is a knowledge elicitation problem. Techniques here are varied and range from careful choice of the repre-

Copyright © 1995 Software Engineering Press

sentation to simplify elicitation [13], to techniques for working with components and libraries [2]. But the elicitation task still has to be done afresh with each different problem, except in those prototypical situations that are routinely addressed by standard statistical packages. While one might use a standard package in initial modeling, as the problem becomes better understood specific requirements are needed that canned software may not provide. Of course, tools for software generation alleviate the modeling task greatly by providing rapid prototyping. Nevertheless, it is my view that a sizeable burden in the Bayesian analysis of data is software engineering rather than the statistical analysis itself, and therefore software generators and support tools are both a realistic and important goal.

Naturally, an important part of such a framework is component libraries containing modules for common sub-tasks. Almond *et al.* [2] point out that parts of a graph, *components*, are often shared in a series of applications. Learning and data analysis are no different.

References

[1] Y. Abu-Mostafa. Financial market applications of learning from hints. In *Neural Networks in the Capital Markets*. John Wiley & Co., 1994.

[2] R.G. Almond, J.M. Bradshaw, and D. Madigan. Reuse and sharing of graphical belief network components. In Cheeseman and Oldford [8], pages 113–122.

[3] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[4] W.L. Buntine. Learning classification trees. In D.J. Hand, editor, *Artificial Intelligence Frontiers in Statistics*, pages 182–201. Chapman & Hall, London, 1991.

[5] W.L. Buntine. Representing learning with graphical models. Technical Report FIA-94-14, Artificial Intelligence Research Branch, NASA Ames Research Center, 1994. Submitted.

[6] W.L. Buntine and A.S. Weigend. Bayesian back-propagation. *Complex Systems*, 5(1):603–643, 1991.

[7] W.L. Buntine and A.S. Weigend. Computing second derivatives in feed-forward networks: a review. *IEEE Transactions on Neural Networks*, 5(3), 1994.

[8] P. Cheeseman and R.W. Oldford, editors. *Selecting Models from Data: Artificial Intelligence and Statistics IV*. Springer-Verlag, 1994.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[10] W.R. Gilks, A. Thomas, and D.J. Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 43:169–178, 1993.

[11] Andreas Griewank and George F. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Breckenridge, Colorado, January 6–8 1991. SIAM.

[12] R. Hanson, J. Stutz, and P. Cheeseman. Bayesian classification with correlation and inheritance. In IJCAI91, editor,

Copyright © 1995 Software Engineering Press

International Joint Conference on Artificial Intelligence, Sydney, 1991. Morgan Kaufmann.

[13] David Heckerman. *Probabilistic Similarity Networks.* MIT Press, 1991.

[14] J.A. Hertz, A.S. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[15] M.I. Jordan and R.I. Jacobs. Supervised learning and divide-and-conquer: A statistical approach. In *Machine Learning: Proc. of the Tenth International Conference*, pages 159–166, Amherst, Massachusetts, 1993. Morgan Kaufmann.

[16] R.E. Kass and A.E. Raftery. Bayes factors and model uncertainty. Technical Report #571, Department of Statistics, Carnegie Mellon University, PA, 1993. To appear, *Journal of American Statistical Association.*

[17] S.L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, (19):191–201, 1995.

[18] D.J.C. MacKay. A practical Bayesian framework for backprop networks. *Neural Computation*, 4:448–472, 1992.

[19] D. Madigan, A.E. Raftery, J.C. York, J.M. Bradshaw, and R.G. Almond. Strategies for graphical model selection. In Cheeseman and Oldford [8], pages 91–100.

[20] J. Moody and L. Wu. Statistical analysis and forecasting of high frequency exchange rates. In *Proceedings of Neural Networks in the Capital Markets '94*, Pasadena, CA, 1994.

[21] R.M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

[22] H. Scott Roy. *A Theory of Learning Classification Rules.* PhD thesis, Stanford University, 1995. Pending.

[23] R.D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

[24] R.D. Shachter, S.K. Andersen, and P. Szolovits. Global conditioning for probabilistic inference in belief networks. In R. Lopez de Mantaras and D. Poole, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*, pages 514–522, Seattle, WA, 1994.

[25] D.J. Spiegelhalter and S.L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.

[26] M.A. Tanner. *Tools for Statistical Inference.* Springer-Verlag, New York, second edition, 1993.

[27] N. Wermuth and S.L. Lauritzen. On substantive research hypotheses, conditional independence graphs and graphical chain models. *Journal of the Royal Statistical Society B*, 51(3), 1989.

Copyright © 1995 Software Engineering Press

Summarizing Time Series Data for Optimizing the Settings of Technical Indicators

George K. Georgiou
Queens College and
The Graduate School of
The City Univ. of New York
Dept. of Computer Science
Flushing, NY 11367
(718) 997-3500
georgiou@qcvaxa.acc.qc.edu

Bon K. Sy
Queens College and
The Graduate School of
The City Univ. of New York
Dept. of Computer Science
Flushing, NY 11367
(718) 997-3500
bon@bunny.cs.qc.edu

David B. Sher
Nassau Community College
State Univ. of New York
Dept. of Mathematics
Garden City, NY 11530
(516) 572-7203
sher@amanda.dorsai.org

Abstract

Finding the optimal setting(s) of a technical indicator with respect to the historical data of a given stock is a computationally expensive task because the size of the data is normally large. This paper presents a method for partitioning the data into equivalence classes such that each class contains data blocks of similar patterns. Then, optimizing the settings of a technical indicator with respect to an instance is equivalent (or close to equivalent) to that for the entire class. We discuss the metrics involved in the derivation of equivalence classes and an example illustration of the optimization process.

1. Introduction

Developing intelligent trading systems that signal profitable entry and exit points of the long (or short) trades of a stock (or commodity) is a research area which may produce useful tools for alerting and assisting traders in critical decision making. A "good" intelligent trading system should advise its users taking a position at the optimal (or close to optimal) trading opportunities. By optimal we mean the peak and the bottom of each upward/downward trend.

A critical task related to the development of an intelligent trading system is the choice of a set of technical indicators [1,2,6-10] with optimal parameter settings that maximizes a pre-defined utility function. For example, a utility function can be defined in terms of the net profit (loss) resulted from the trades initiated at a certain period of time [4].

There are two optimization problems involved in the task just mentioned: (i) the optimal setting(s) of a technical indicator with respect to the historical data of a given stock/commodity, (ii) the optimal combination of the technical indicators that maximizes a utility function. In this paper, we focus on the first optimization problem.

Optimizing the parameter setting of a technical indicator with respect to the historical data of a given stock/commodity is very computationally expensive because the size of historical data is normally large. This paper presents a method to deal with the computational problem by "summarizing" the behavior of a stock/commodity. Our approach is based on the partition of the data into equivalence classes such that each class contains data blocks of similar patterns. By doing so, optimizing the parameter setting of a technical indicator with respect to an instance is equivalent to that for the entire class, thus reducing the computational cost. In this paper, the equivalence class is induced by the criterion that the data patterns exhibited by the data blocks of the same class closely resemble each other.

Partitioning a data set into equivalence classes of similar patterns requires the identification of data patterns which exhibit similar behaviors. Patterns that are not similar to any other data pattern in the data set, are called unique patterns or *data signatures* [3] and each forms a class of its own. Therefore, the lower bound of the number of classes that a data set may be partitioned into heavily depends on the number of data signatures.

In section 2 we discuss the concept of similarity between data patterns. Section 3 presents a discussion on class equivalence and a classification algorithm for summarizing a data set. Section 4 describes an alternative method of classification that first identifies the data signatures in

Copyright © 1995 Software Engineering Press

a data set and then derives classes from all the non-signatures. We describe the optimization of the settings of a technical indicator along with an example illustration in section 5. Finally, we conclude the paper in section 6 along with future research directions.

2. Similarity Between Data Patterns

The *similarity* (or *sameness*) between two entities can be thought of as a predicate that returns truth or falsity depending on the criterion that we use. This criterion can be an n-tuple of attributes and two entities are rendered similar if they have matching attributes. For example, if we are interested only in shape, then a green pepper and a yellow pepper are similar objects, but if we are interested in shape and color then they are not. In matching individual attributes in some cases one might only accept exact matches, while in some other cases one might accept an almost exact match, particularly when an exact match is virtually impossible to get due to the subjectivity of the attribute.

In dealing with time series data we are interested in summarizing the "behavior" of the series. One way to derive this summarization is to consider the various subsets of the series (given some subset size) and describe the behavior of each subset in relation to other subsets. The behavior of each subset is an ordered description of the consecutive data points involved, with respect to their respective values. For instance, a subset of three points, P_1, P_2 and P_3 has an *up-down* (or $\{+,-\}$ or a *peak*) behavior, if the value of P_2 is higher than the other two. Moreover, we are interested in the magnitude of this behavior; i.e., the actual shape of the peak relative to the values of the points involved. For these reasons and for the purpose of determining similar behavior among subsets of the series (which we call data patterns), we are using the *linear correlation coefficient*, a standard and widely applied measure of the degree of association between two series of values (also known as *Pearson's r* and *product-moment correlation coefficient*). For pairs of values (x_i, y_i), $i = 1, \cdots, m$, the linear correlation coefficient ρ is given by [5]:

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$$

The coefficient, ρ, is bounded between -1 and $+1$, inclusive. A value of ρ close to 1 denotes positive correlation (i.e., x and y are increasing or decreasing together), while a value of ρ close to -1 denotes negative correlation (i.e., y increases as x decreases or vice versa). A value of ρ near zero indicates that the two series are uncorrelated.

Each pair of terms considered in the statistic is scaled by the expected value (mean) of its corresponding subset series. Therefore, it is not necessary to de-trend a series before attempting to summarize its behavior. In our case, two subsets (of equal size) of consecutive data values are considered similar to each other, if they have a high positive correlation. At this point we can see that the criterion of similarity between two patterns is some predefined correlation threshold, $\bar{\rho}$. Then, the closer $\bar{\rho}$ is to 1, the stricter the acceptance for similarity.

3. Deriving Equivalence Classes

Each data pattern in a time series, $\bar{X}(t)$, is characterized by its size (number of data points), k, and its time stamp (its time of occurrence), t_α (Figure 1). For the rest of the paper we denote a data pattern by $P_X^k(t_\alpha)$. Then, given k, it can be easily seen that in a series $\bar{X}(t)$ there are $n-k+1$ data patterns, where n is the size of $\bar{X}(t)$.

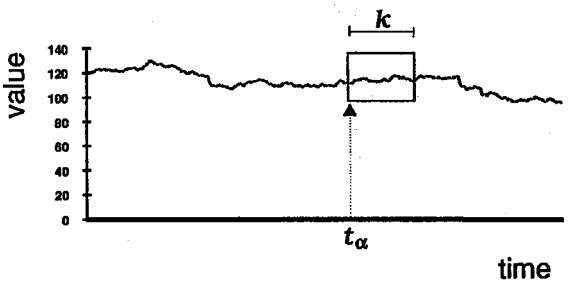


Figure 1: A time series and a pattern

Our task is to partition these $n - k + 1$ patterns into groups (classes, $Cl_X^{\bar{\rho},k}(i)$s) of similar patterns. In each class we include only data patterns whose pairwise correlation (with all other data patterns of the class) is equal or higher than a predefined correlation threshold, $\bar{\rho}$. By doing so, we ensure that the transitivity property holds true within the class. Each derived class is an equivalence class (i.e., a class whose members satisfy the *RST* properties; namely, *R*eflexive, *S*ymmetric and *T*ransitive) and each instance (data pattern) in a class is a summary of the class.

Copyright © 1995 Software Engineering Press

To derive these equivalence classes we first decide on the size of the patterns we wish to analyze (i.e., set the value of k) and the criterion of similarity (i.e., the correlation threshold, $\bar{\rho}$). Then, initially we consider that each possible pattern of size k in the series is in its own class. Subsequently, we start processing pairs of patterns (in an ordered fashion) in an attempt to find pairs whose correlation is above the threshold. For example, if $P_X^k(t_1)$ is found to correlate with $P_X^k(t_5)$ then we know that we can group these two patterns in the same class, say $Cl_X^{\bar{\rho},k}(1)$. Keeping track of a parent-child association between these two patterns, we continue to process $P_X^k(t_5)$ in an attempt to add members to the already established class. So, if say, $P_X^k(t_5)$ correlates with $P_X^k(t_{16})$, we will include $P_X^k(t_{16})$ in the class $Cl_X^{\bar{\rho},k}(1)$ only if it correlates also with $P_X^k(t_1)$. In addition, the parent-child association between $P_X^k(t_5)$ and $P_X^k(t_{16})$ is established only if $P_X^k(t_{16})$ is accepted as a member in the class. For instance, if $P_X^k(t_{16})$ does not correlate with $P_X^k(t_1)$ then $P_X^k(t_{16})$ is not accepted in $Cl_X^{\bar{\rho},k}(1)$ and the search proceeds by testing $P_X^k(t_5)$ with $P_X^k(t_{17})$. The process is an iterative one, and at each iteration it only considers patterns which are not classified by previous iterations.

Following is the algorithm to derive these equivalence classes.

Algorithm 1: Derivation of equivalence classes from set $\bar{X}(t)$.

Input: A time series data set, $\bar{X}(t)$, data window, k, correlation threshold, $\bar{\rho}$.

Output: The set, $Cl_X^{\bar{\rho},k}$, of all equivalence classes, $Cl_X^{\bar{\rho},k}(i)$s, in $\bar{X}(t)$.

Description: Let $Cl_X^{\bar{\rho},k} \leftarrow \{\}$ and set $\bar{\rho}$, k.
Set $\alpha \leftarrow 1$, $\beta \leftarrow 2$, $\gamma \leftarrow 1$, $x \leftarrow 1$ and $n \leftarrow |\bar{X}(t)|$

/ initialize each pattern to be in its own class */*

loop1: **if** $x > n - k + 1$ **then goto** 1
else $P_X^k(t_x).parent \leftarrow x$
$P_X^k(t_x).child \leftarrow x$
$x \leftarrow x + 1$
goto *loop1*

/ compute classes */*

1: **if** $\alpha \leq n - k + 1$ **then goto** 2 **else goto** 7

2: **if** $P_X^k(t_\alpha).parent = P_X^k(t_\alpha).child$ **then goto** 3
else $\alpha \leftarrow \alpha + 1$
goto 1

3: **if** $\beta \leq n - k + 1$ **then goto** 4
else *insert* $P_X^k(t_\alpha)$ *in* $Cl_X^{\bar{\rho},k}(i)$
insert $Cl_X^{\bar{\rho},k}(i)$ *in* $Cl_X^{\bar{\rho},k}$
$i \leftarrow i + 1$
$\gamma \leftarrow \gamma + 1$
$\alpha \leftarrow \gamma$
$\beta \leftarrow \alpha + 1$
goto 1

4: **if** $P_X^k(t_\beta).parent = P_X^k(t_\beta).child$ **then goto** 5
else $\beta \leftarrow \beta + 1$
goto 3

5: **if** $\rho(P_X^k(t_\alpha), P_X^k(t_\beta)) \geq \bar{\rho}$ **then goto** 6
else $\beta \leftarrow \beta + 1$
goto 3

6: **if** $P_X^k(t_\alpha).parent = \alpha$ **then**
insert $P_X^k(t_\alpha)$ *in* $Cl_X^{\bar{\rho},k}(i)$
$P_X^k(t_\alpha).child \leftarrow \beta$
$P_X^k(t_\beta).parent \leftarrow \alpha$
$\alpha \leftarrow \beta$
$\beta \leftarrow \alpha + 1$
goto 3
else $x \leftarrow \alpha$
$\alpha \leftarrow P_X^k(t_\alpha).parent$

loop2: **if** $\rho(P_X^k(t_\alpha), P_X^k(t_\beta)) \geq \bar{\rho}$ **then**
if $P_X^k(t_\alpha).parent \neq \alpha$ **then**
$\alpha \leftarrow P_X^k(t_\alpha).parent)$
goto *loop2*
else *insert* $P_X^k(t_x)$ *in* $Cl_X^{\bar{\rho},k}(i)$
$P_X^k(t_x).child \leftarrow \beta$
$P_X^k(t_\beta).parent \leftarrow x$
$\alpha \leftarrow \beta$
$\beta \leftarrow \alpha + 1$
goto 3
else $\alpha \leftarrow x$
$\beta \leftarrow \beta + 1$
goto 3

7: Return $Cl_X^{\bar{\rho},k}$

4. Data Signatures and Equivalence Classes

In the previous section we discussed an algorithm for deriving the equivalence classes of patterns in a data series. In this section, we describe an alternative two-step method of deriving equivalence classes. First, we discover the unique patterns of the series which we call *data signatures*[3].

Formally, a data signature is a block of consecutive time varying data points which exhibits *low* or *no* statistical correlation to any other block of data in the same data set.

Copyright © 1995 Software Engineering Press

For instance, the two patterns shown in Figure 2a are data signatures since they have low correlation. In Figure 2b the middle pattern is a data signature, since it does not correlate highly with either one of the extreme blocks. The two extreme blocks are not data signatures because they are almost identical (highly correlated).

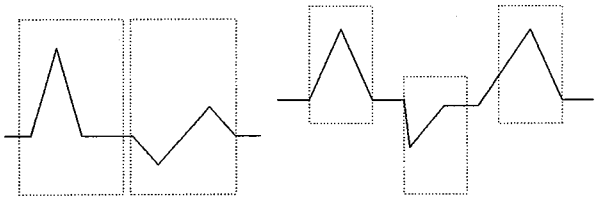


Figure 2a **Figure 2b**

A data signature is found if its correlation with all other data windows in $\bar{X}(t)$ is less than a predefined *correlation threshold*, $\bar{\rho}$.

Once we derive the data signatures for a particular correlation threshold, we know the lower bound of the number of equivalence classes we would deal with, which is *# of data signatures* +1. In other words we know that a data signature forms a class by itself. In addition, we do know that if a data pattern is not a data signature, then there exists at least one other data pattern (possibly more) with which it is highly correlated, and therefore, together they may form an equivalence class. At this point we proceed to the second step. The data patterns which were found to be non-signatures are further considered in the derivation of the remaining equivalence classes using the algorithm described in section 3.

Although for the purpose of deriving equivalence classes this approach is never better (computationally) than the straightforward approach described in the previous section, by first identifying the unique patterns of a series, we do get a better insight on the number of true single element classes. A unique pattern is guaranteed to be its own class regardless of which method we use. But, with the straightforward approach we do not really know whether a particular one-element class contains a unique pattern or a pattern which correlates with a few but not all of the patterns of some other class.

5. Setting a Technical Indicator

The basic function of a technical indicator is to assist a user in making trading decisions (i.e., when to buy and when to sell). The indicated trading actions are dictated by a set of parameters incorporated in the formula of the indicator. A popular such indicator is the *Wilder Relative Strength Index*, (*RSI*) [9,10]. RSI is a momentum oscillator that measures the velocity of directional price movement by comparing a stock's highest highs and lowest lows over a period of time, thus able to indicate trend reversals at an early point. The formula for RSI is:

$$RSI = 100 - \frac{100}{1 + RS}$$

where

$$RS = \frac{Average\ of\ Up\ Closes\ for\ X\ Number\ of\ Days}{Average\ of\ Down\ Closes\ for\ X\ Number\ of\ Days}$$

The number of days considered in RSI is thus a parameter. Moreover, a second pair of settings involves the thresholds for a buy and/or a sell signal. For example, setting these two parameters at 30 and 80 respectively, then a buy signal is triggered when RSI goes from a value of less than 30 to greater than 30 and a sell signal is triggered when RSI goes from a value of greater than 80 to less than 80. Figure 3 visually displays the series of the daily closing price of a stock (upper graph) for a two-year period and its corresponding 9-day RSI (lower graph) with 30 and 80 trading thresholds.

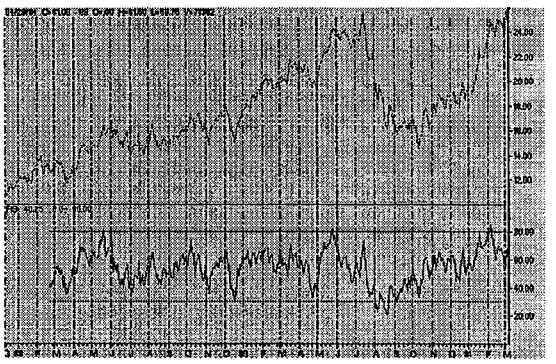


Figure 3: A stock and its 9-day RSI

The optimization of a technical indicator is the setting of the aforementioned parameters such that a utility function, e.g., net profit, is maximized. Generally speaking, optimizing the parameters of the indicator is specific to the stock and in particular to a specific time period. In other words, the optimized setting of the indicator for some period of the stock may be different than the optimized setting of the same indicator for some other period.

Copyright © 1995 Software Engineering Press

Traders are usually interested in the settings of an indicator which historically have a "good" performance, with the belief that they will work effectively in the future as well. In finding those settings, one might decide to consider a fixed length time period, say 100 days, and pull several 100 day periods from the historical data and check for a consistent performance of the setting of the indicator for these periods. By consistent performance we mean that the chosen settings return a value (or values) of the utility function which are approximately close. While this is a common and practical approach, theoretically, there are $n - 100 + 1$ different 100 day periods that the user would have to consider when optimizing the indicator. Immediately we can see that this raises computational concerns.

By summarizing the stock data in classes of similar patterns we want to avoid optimizing the parameters of an indicator for patterns that are found to be similar, since setting these parameters for similar patterns would be approximately the same. The question that arises is how consistent those settings are for members of the same class under different values for the pattern size and correlation threshold.

To address this question we experimented with the daily closing price of International Business Machines stock (IBM). We applied our classification algorithm on the data using different correlation thresholds (0.5, 0.6, 0.7, 0.8, 0.9 and 0.95) at different pattern lengths (20, 50 and 100). Our preliminary results indicate that the larger the pattern size the smaller the number of classes we partition the series into (i.e., an increased ratio of number of possible patterns over the number of classes found which translates to more patterns per class) regardless of the correlation threshold. Our criterion for setting the trading thresholds of the 9-day RSI indicator was set to be the value crossing the bottom (buy signal) and the value crossing the peak (sell signal) of the indicator's pattern. We found that these values for patterns in the same class for higher correlation thresholds such as 0.9 and 0.95 were significantly more consistent than for lower correlation thresholds. Also, analysis with patterns of length 100 resulted in more consistent results than for the smaller lengths.

6. Conclusions

A theoretical optimization of the parameters of a technical indicator on a time series data requires the processing of all the patterns from the data of some length k. We presented a method to deal with this computational complexity, partitioning the series in patterns of similar patterns and then optimizing the parameters using only one pattern per class. In our preliminary experimental results we found that the level of consistency for settings of the 9-day RSI indicator for a particular stock, increased with higher correlation thresholds and larger data patterns.

In our future work, we will conduct experiments with various technical indicators and different utility functions in an attempt to derive an optimal combination of indicators over a wide range of stocks.

Acknowledgments

We would like to thank the anonymous referees for their valuable suggestions and comments. This work has been supported in part by a grant from the City University of New York: (PSC-CUNY Research Award Program).

References

[1] R. Amacher, H. Ulbrich, *Principles of Economics*, South-Western Publishing, 1986.

[2] R. W. Colby, T. A. Meyers, *The Encyclopedia of Technical Market Indicators*, Homewood, Ill. : Dow-Jones-Irwin, 1988.

[3] G. K. Georgiou, B. K. Sy, D. B. Sher, Data Signatures for Validation and Evaluation of Temporal Associations, in *Proceedings of the 1995 Florida AI Research Symposium*, Melbourne Beach, Florida, April 1995.

[4] J. E. Granville, *New Strategy of Daily Stock Market Timing for Maximum Profit*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[5] M. Hamburg, *Statistical Analysis and Decision Making*, HBJ, 1991.

[6] P. J. Kaufman, *The New Commodity Trading Systems and Methods*, New York:Wiley, 1987.

[7] G. C. Lane, Stochastics, *Trading Strategies*, Futures Sym Intl, 1984.

[8] G. C. Lane, Lane's Stochastics, *Technical Analysis of Stocks and Commodities*, June, 1984.

[9] Telescan Inc. *Telescan Reference Manual*, Houston, Texas, 1993.

[10] J. W. Wilder, Jr., *New Concepts in Technical Trading Systems*, (Trend Research, Box 128, McLeansville, NC 17301), 1978.

Copyright © 1995 Software Engineering Press

A NEWS CATEGORISATION SYSTEM FOR TRADERS AND ANALYSTS

L.Gilardoni, P.Prunotto, G.Rocca
Quinary SpA
via Crivelli, 15/1
20122 MILANO, Italy
Tel: +39-2-58302712 ; Fax: +39-2-58305374
E-mail {lg,pp2,gr}%quinary@iunet.it

F.Deotto, A.Di Cresce
Euromobiliare S.I.M S.p.A
via Turati, 9
20121 MILANO, Italy
Tel:+39-2-62041

Abstract

In this paper we discuss how different techniques have been integrated in a system designed to perform fine grain text categorisation, involving information extraction, of natural language texts. The texts analysed are agency news (Reuters news), while the target users are traders and analysts of an Italian merchant bank, which requested the possibility of having both a broad and a fine grain text categorisation. The need for a flexible text categorisation system which could be able to perform a rich categorisation lead to the development of a new system integrating different text analysis techniques. The techniques used range from pattern based text analysis, used for a first 'shallow' categorisation, to full NLP, including full text parsing and semantic analysis of the text.

The work reported has been done in the framework of the LRE (Linguistic Research and Engineering) project COBALT (LRE 61-011); LRE projects are partially funded by CEC. The COBALT prototype was developed by a team composed by Quinary SpA, Italy, UMIST-CCL, UK, Cril, France, Reuters SpA, Italy, and Euromobiliare, Italy.

1. Introduction

Managing the overwhelming amount of unstructured, textual information made available in electronic form, either via on-line information sources or via e-mail or in CD-ROM is becoming a crucial task. The main requirements to attack this problem are the ability of recognizing concepts mentioned in the texts and to relate a portion of text to a set of conceptual categories.

When dealing with financial news, several systems have been built performing a broad categorisation, recognizing the main topics the news deal with, but a more powerful approach is needed to perform a richer categorisation, able to discriminate also on details reported in the agency news. The requirements of performing a precise categorisation, with a high degree of recall and precision, and of managing big amounts of texts, impose severe constraints on the possible approaches to the problem. Keyword search and pattern matching techniques on their own are fast and cost effective, but can't achieve the accuracy users need from an automated system. On the other hand, more complex systems, based only on natural language understanding technologies, are still much too slow and not robust enough for practical uses.

The basic idea behind the Cobalt approach is therefore to integrate different technologies such as shallow pattern matching and robust syntactic and semantic analysis in order to get the advantages of each while overcoming their weaknesses. The main challenge is obviously to manage the integration, both from the point of view of effectively being able to obtain the best result from each component and from the point of view of building a robust, extendible system.

The main glue among the different components is given by the use of knowledge representation techniques to describe the main concepts in the application domain and the relationships among the different kinds of information that can be extracted by the different modules. This allows us to define a 'common substrate' for each component specific knowledge and therefore to manage integration defining opportunistic strategies depending on 1) partial results of each component 2) domain knowledge available to the system and 3) known capabilities of each component. Moreover it becomes possible to exploit domain knowledge to manage the complexity of patterns and analysis rules, making it easier to maintain and extend the system.

Copyright © 1995 Software Engineering Press

2. The Cobalt Project

The COBALT project ([1]) was concerned with the problem of capturing factual and definitional knowledge from textual sources. The main aim of the project was to demonstrate that different text analysis techniques can be used together to provide a system able to perform a fine grain classification, placing it at the boundary between text categorisation systems and text understanding systems (which are generally characterised by tasks such as being able to extract information from texts or to summarise or abstract texts [2]).

Research and development activities in COBALT resulted in the production of an experimental "empty categorisation shell", which has been used for the real-world prototype described in this paper. The main aim of the prototype is to be able to recognise news which are interesting for the end user from the set of news distributed by Reuters' datafeed and to make them available for routing and retrieval. The possibility of filtering important news at a particular moment will greatly aid end users in their work since it will make it possible for them to concentrate only on the potentially important news without being diverted by the big amount of uninteresting ones distributed by the datafeed.

Both end users of this prototype, the analysts and traders, are mainly interested in financial news, related to specific companies and stock markets as well as to the economic and political situation of the country. Although the two different types of users are interested in the same kind of news, they however have quite different requirements with respect to the kind of categorisation that must be performed.

Traders, who are in charge of buying and selling stocks on the market need help in quickly discriminating between news; for this type of user, a broad categorisation identifying the main topics pertaining to the financial news types is sufficient since they need to keep an eye on the overall market situation without being overwhelmed by the great amount of incoming news. Their major requirements are a high speed of the system (since the arrival of a relevant news may imply an immediate reaction in the market) and a high recall, which is preferred to a high precision since they do not have to miss out any of the possibly relevant news.

On the other side, analysts are usually assigned to specific sectors and they use news as a source of information for making analyses and writing reports regarding their sector and the companies belonging to it. Unlike the traders, time is not so much a constraint, since they do not have to react immediately to a news but have instead plenty of time to reflect on its contents and on the longer term impact it may have. Moreover, this kind of user needs a fine grain news categorisation since they are usually interested in filtering news depending on particular information regarding the main topics identified by the broader categorisation; for example, they might be interested in filtering news depending on who is the vendor in a stake modification story.

While the first users mainly need a broad categorisation, supplied by what we will refer to in this article as "base categories", the second users' need implies the necessity for the system to have some capabilities of information extraction which is to be used for the identification of the fine grain categories. In the following we will refer to this fine grain categorisation as "compound category" assignment, since it implies the identification and extraction of information which will fill some slots (what we call 'category relevant slots') that have been associated to the base categories thus rendering them "compound".

While base category assignment may be achieved with quite a high accuracy through a shallow analysis of text (in Cobalt done using pattern matching techniques), the identification of compound categories requires a deep syntactic and semantic analysis of the contents of the text. Information to be inserted in the relevant slots to build compound categories could then be extracted from the text representation that results from the deep analysis.

The mechanism that controls text analysis is based therefore from one side on knowledge about how to perform categorisation and on how to extract relevant information from texts, and from the other side on user needs, explicitly stated in terms of filters, expressed as boolean combinations of the categories that have been defined for the system.

3. Categories and Concepts defined for the Cobalt Prototype

The end users are interested in financial news both related to specific companies listed in the Milan stock exchange as well as to news related to the behaviour of certain foreign stock markets

Copyright © 1995 Software Engineering Press

and to the political and economic situation of the country.

In the prototype, a set of categories has been defined (in a hierarchical manner) for the company news and for the foreign stock market behaviour news; the political and macroeconomics news related to the country (Italy) have not been treated.

The total number of base categories defined is 240; where 15 pertain to the stock market news and 225 to the company news. Out of the defined categories, 180 are explicitly assigned by categorisation rules and the rest act as containers for grouping the different categories. Approximately 155 of the explicitly assigned categories actually correspond to either company or stock market instances. Approximately 100 concepts (i.e. generic concepts, such as "stake_selling", having patterns defining them) and 150 instances (i.e. concept instances, such as "Fiat" which have associated specific patterns) have been defined to treat the company news; 22 concepts and 14 instances have been defined for the stock market news for a total number of approximately 550 patterns.

Regarding the hierarchical structure of the category KB, there is a root category named category_ which has two main subtrees respectively for the company news and for the stock market news.

For the company news, the further identified categories correspond to the single company instances the user is interested in (companies listed in the Milan stock exchange) defined hierarchically reflecting their sectorial subdivision as already used by the users, and categories identifying topics pertaining to company news. Such topics are for example: share capital changes (capital increase and decrease), stake selling and buying, share offerings, initial public offerings, quarterly, yearly and half yearly economic results, changes in the management of the companies, rating of companies.

For the stock market news, the single foreign stock markets the user is interested in, and the different behaviour of the markets (we have identified them as up, down and steady) constitute the further classification.

For some of the categories, a set of relevant slots has been defined thus allowing the users (mainly the analyst) to specify compound categories. Fine grain categorisation rules have been initially defined for: stake_modification, share_offer and form_joint_venture. For these, we will give therefore a more detailed description both regarding the base category and its relevant slots.

stake_modification: should be assigned if the news speaks about the sale/acquisition of a stake or shares of a company. In a stake modification event there are different roles: the vendor of the stake, the buyer of the stake and the object of the sale, i.e. shares or a stake of the company whose share/stake is being traded. Other information such as the amount of the stake/shares, and the price paid are relevant for the creation of compound categories. Attributes have therefore been defined for this base category to take into account this information and in particular they are: ":vendor", ":buyer", ":company_name", ":goods", ":number-of-shares", ":stake-percentage", ":unit-price", ":total-price" and ":currency". An example of compound category that could be formulated is :stake_modification in which the stake-percentage being traded is greater than 50%.

share_offer: should be assigned if the news speaks about either an offer to buy or to sell shares of a company. The relevant information for this base category are: whose shares are being offered and by whom, the amount of shares and the price for the offer, which are identified respectively by the following relevant slots: ":company-name", ":offerer", ":amount", ":unit-price", ":total-price" and ":currency".

form_joint_venture: should be assigned if the news speaks about two or more companies joining up to form a joint venture. The outcome of a joint venture is usually a new company or the modification of one of the participating companies with an exchange of shares among them. Information that has been associated to this base category are represented by the following relevant slots: ":companies-involved:" identifying the companies involved in the venture; ":venture-outcome:" which is the new company formed as a results of the venture; ":investment:" indicating the quantity invested in the venture; ":currency" which identifies the currency the investment is expressed in; ":activity" of the new joint venture; (for example: production, sale, design, marketing, services); ":phase:" identifying in what phase of the venture agreement the venture is in (whether we are in a plan, agreement or in the actual activation of the venture); ":sales" which identifies the expected sales or sales volume of the new venture; ":stake-

Copyright © 1995 Software Engineering Press

partitioning:" which identifies the stake division in the new venture of the participating companies.

Figure 1 below reports a portion of the hierarchy as shown through the system interface; bold nodes represent base categories, with underlying text reporting available relevant slots; boxed nodes mean that a subhierarchy exist below the node.

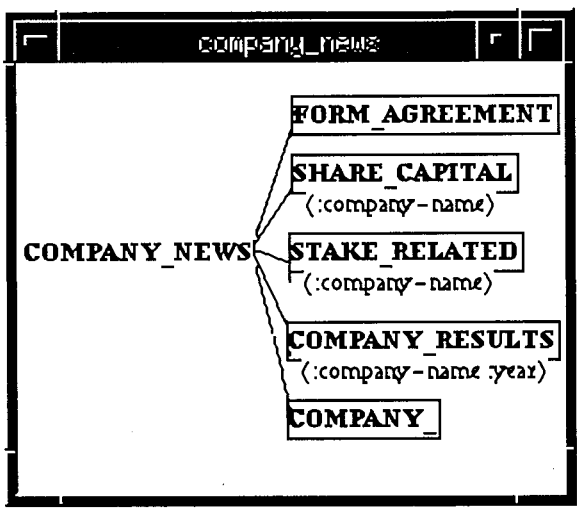


Fig. 1: a portion of the category hierarchy

In order to assign the categories to the texts, a knowledge base of concepts has been defined in which the domain knowledge is expressed both in terms of properties and relations existing between concepts, and in terms of patterns which are used to perform the shallow analysis. Concepts are organised in a hierarchy which can be exploited for score inheritance by the shallow categorisation rules [3].

Entities describing the domain range from the single instances identifying the stock markets, and the corresponding market indexes, the user is interested in; the companies listed in the Milan stock exchange organised in a hierarchy reflecting the sectorial division used by the user; entities appearing in the balance sheet and results statements; currency instances; personal positions within an enterprise. Such entities are obviously described by properties (for example, :company-name for the concept company_) and relations linking these entities (for example, the relation :stock-market-index and its inverse :related-bourse which link instances of the concept stock_market to instances of the concept stock_market_index). The KB contains also concepts describing the possible actions that are relevant in assigning the categories identified, such as, for example, the buying/selling of shares/stakes in a company, the increase or decrease of the share capital, the upward and downward behaviour of shares and stock market indexes, the forming of a joint venture, the election to or dismissal from the board of directors and so on.

4. Prototype Functionalities

The main functional requirement for the prototype is that of being able to filter news coming from a datafeed according to user specified interests. User interests vary both from user to user, and, for each user, from moment to moment according to the specific situation or work he is doing in a particular moment. Thus the necessity of a user configurable system which allows him to specify from moment to moment his interests in terms of categories (base or compound) to be satisfied by news in order for them to be reputed interesting. The news which satisfy the user's interests will then be made available to him and will displayed on the screen.

Besides specifying the meaning of interesting news, the users wanted to have the possibility of displaying the different kinds of news in different windows on the screen thus aiding them in further visually classifying among the filtered relevant news.

For this reason the idea of "user profile" has been introduced in the prototype and has been defined as a set of windows to which a set of filters is associated. Profile execution should then consist in the display on the screen of the windows defining it, and in the display of the headlines of the incoming news in the windows which have associated at least one filter which is satisfied by the category assignment that has been performed on the news by the system. The user could then read or print the entire text of the news whose headlines appear in the profile windows.

Filters are defined by specifying both a name, a level of importance and a definition in terms of boolean combinations of categories (be them base or compound). The level of importance for a filter defines the action to be performed when the filter is satisfied. Simple actions involve evidencing the title using different colours, or directly displaying the news contents, or sending alert messages to the user via e-mail. Straightforward extensions could include integration in workgroup environments and

Copyright © 1995 Software Engineering Press

storing of classified texts in specialized data bases.

Following is an example of filter definition containing the base category form_joint_venture and the compound category defined by specifying the slot :vendor for the base category stake_modification:

"(or form_joint_venture
(stake_modification where :vendor Fiat))

5. Cobalt Architecture

The COBALT system is composed of four main modules; one which performs shallow analysis, two devoted to deep analysis, performing syntactic and semantic analysis of text, and a control module.

The shallow categorisation component of COBALT (TCSM, Text Categorisation System Module) is based on a commercial product, and is aimed at identifying in each considered text portion references to known concepts. Such references are recognised thanks to the presence of defined combinations of keywords ("patterns") that are associated with the concepts' definitions.

The basic text categorisation system has been enhanced by integrating it with a knowledge representation language that allows to describe the domain of interest, expressing entities' properties and relations. Domain concepts are represented as frame-like objects organised in a hierarchy, and patterns exploit domain knowledge represented in the concepts knowledge base; moreover, the concepts hierarchy allows inheritance of patterns as well as of match results. This work, which is described in detail in [3], represents an attempt to increase the capability of the pattern match based analysis and to ease the construction of the search patterns.

This first level shallow analysis sets up a rough representation of the conceptual content of each considered portion of text, which can then be used to evaluate its "relevance". "Relevance" in our context is both application dependent and user dependent, in that different users are generally interested in different topics and at different levels of detail. Therefore, the "relevance" associated to different news may involve the number of retrieved concepts as well as the presence of references to some "important" concept, and will also take into account the ranking defined by the different users.

The deep text analysis is based on NLP techniques. Relevant portions of text can then be submitted to the syntactic analysis module (SCAM, Surface COBALT Analyser Module) which is based on a sort of unification-based "categorial grammar", see [4].

The output of the module is the input to the semantic parser (CLAM, COBALT Language Analysis Module) which produces, by applying a set of substitution and triggering rules, a formal representation of the content of the original text in terms of NKRL ([5]) a specialised Knowledge Representation Language.

The semantic analysis is in charge of producing a 'normalised' representation of the text in terms of concepts and relations among them, resolving syntactic/semantic ambiguities and anaphora. The representation is necessarily rich, to take into account the real text contents and will directly reflect the text structure.

A specific module, the COBALT Control Module (CCM), provides the application developer with a general approach and languages for defining, in each specific application, which categories have to be handled, which information is carried with them, and how to perform categorisation. The category definition language enables the application developer to define the categories for the application, which are characterised by a name a definition and a set of attributes. Attributes, defined as slots of specific types, could contain either control information (such as references to mechanisms needed by the control structure to perform categorisation) or domain information, such as additional attributes that could be relevant to the specific category (e.g. the fact that for "stake_selling" category a "buyer" could be a relevant information potentially associated to the category). A rule based control language allows to specify the control strategy.

In the prototype developed, the categorisation flow is dependent on both the results obtained from the textual analysis components (TCSM, SCAM and CLAM) and on the user preferences set through the profile definition. Both the rules to assign the base categories, the rules that fill the relevant slots of the base categories, and the control rules that activate deep analysis when needed and decide which slot filling rulesets to fire are written in the COBALT Rule Language

First of all, news texts are read into the system from an external source via a reader program which is in charge of reading the news items from the news feed and transforming them into

Copyright © 1995 Software Engineering Press

their internal format. Each text then undergoes the surface analysis performed by the TCSM module. Then, a first set of CCM application dependant rules is fired. This set of rules is in charge of analysing TCSM pattern matching results and performing base categories assignment.

After base categories assignment, filters defined in the current profile(s) running are verified; for each compound category mentioned in a filter for which an assignment of the corresponding base category has been made, a specific ruleset is activated.

The activation rulesets that are fired whenever a compound category has to be matched are in charge of actually trying to provide the information requested in the compound category or to fail. In the prototype, each ruleset, different for each base category, will activate deep analysis (SCAM and CLAM in sequence) and then fire a set of rules in charge of analysing the results and perform compound category assignment. These rules will try to fill the category relevant slots with the results obtained by the deep analysis (i.e. on the NKRL representation of the text). The category with the relevant slots filled will then be the compound category assigned to the text.

If the categories (base and/or compound) that have been assigned to the text satisfy any of the filters defined in the currently running profile, then the action corresponding to the filter relevance will be executed, namely to display the news' headline or contents in a window.

It's worthwhile noticing that

- only shallow analysis is always performed on the text; deep analysis is performed only on demand depending on the user needs (usually analysts' needs);
- by partitioning slot filling rules depending on specific categories, the system will only look for requested information, not trying to extract from deep analysis results information which is not needed

an obvious advantage of this approach is the gain in speed; on the other side however, if categorisation results need to be saved for later retrieval this implies that deep analysis need to be done again to retrieve further compound categories.

6. Evaluation

A complete evaluation of the prototype has already been performed on a set of unseen texts. The evaluation involved shallow categorization (i.e. assignment of base categories) of about 500 texts, automatically computing accuracy measures on the results. A subset of 50 randomly chosen texts then underwent full analysis, involving information extraction and recognition of compound categories,.

A full analysis of evaluation results has already been completed concerning shallow analysis results, while the analysis of compound categorization results is still ongoing, due to the higher complexity of evaluation data.

Shallow categorization results showed an overall accuracy of over 85%; in figure 2 the results obtained on all the categories for which at least 10 texts in the evaluation corpora were relevant are reported. Most of the categories reached over 90% in both recall and precision, and the few that fall below these figures were also the ones less recurring in texts and to which lower attention was paid during development.

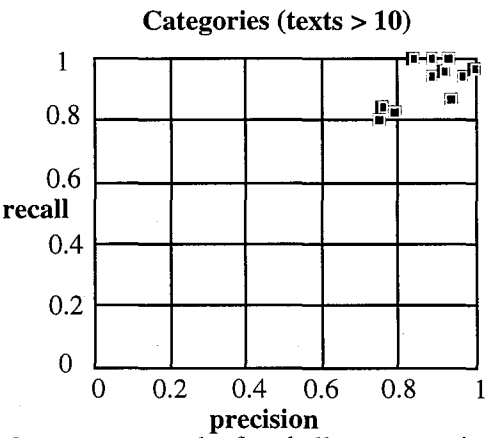


Fig 2: accuracy results for shallow categorization

Preliminary results concerning deep analysis showed that only about 50% of the information available in texts could be extracted, but also showed that the error rate (i.e. wrong conclusions) is quite low.

Processing time is about 1 sec for shallow categorization and ranging from 10 to 30 seconds for deep analysis (timings on a SparcStation 2 with 32 Mb of main memory), proving effective to satisfy user requirements. Moreover, it must be noted that the prototype has been developed with no special attention on speed optimization

Copyright © 1995 Software Engineering Press

7. Conclusions

The Cobalt prototype was built with the aim of demonstrating how the integration of shallow and deep text analysis techniques could be successfully exploited in performing the kind of categorisation of financial news needed by a real end user.

The integration of shallow (pattern based) and deep (syntactic and semantic) analysis was envisaged to overcome the conflicting requirements of a rich categorisation to be performed quickly. From this point of view, the integration of the two technologies proved to be able to deliver expected functionalities, while the control mechanism developed for the prototype (and the general idea implemented in the Cobalt shell of providing a powerful control language) showed that necessary flexibility could be achieved.

Although a detailed analysis of the raw results is still ongoing, some conclusion could already be drawn. Shallow and deep analysis capabilities proved themselves effective and complementary in performing the task. Preliminary results from the evaluation confirmed the capability of shallow analysis to perform a quick and accurate enough categorisation, but also confirmed the impossibility of doing too much precise category assignment using it alone. In particular, while good results could be obtained in performing generic category assignment (i.e. determining the main topic the news deal with, e.g. stock trading), it proved impossible (or not cost effective) trying to perform richer categorisation involving information extraction (e.g. who actually traded the shares) using shallow analysis directly. Syntactic and semantic analysis, on the other side, while being still too inefficient or not robust enough to perform in a reliable way simple categorisation on the whole set of possible incoming texts, proved to be useful in analyzing specific texts to extract interesting information.

The shallow analysis results confirmed the high accuracy that could be obtained with the employed techniques; alternative approaches for performing a 'pre filtering' stage before a richer categorisation still deliver lower accuracy. The known problem related with knowledge based pattern analysis, namely that of the development effort needed, were partially mitigated by exploiting domain hierarchies [3]; moreover, the use of explicit patterns (instead of, for example relying on statistical techniques or on semantic dictionary based techniques [6]) make easy to understand and justify (and thus modify and extend) system behaviour. The use of statistical techniques as a support for the pattern definition activity, although not exploited within the Cobalt project, is envisaged to further minimise development effort and will be considered as prioritary for further development.

Concerning deep analysis results for relevant slot filling, while the overall results showed that only about 50% of the information available in texts could be reliably extracted, the chosen approach proved anyway to be robust enough to process any agency news. Improvements need to be done to extend the domain specific lexicon and to enhance the treatment of unknown words, mainly for proper name recognition and proper treatment of abbreviations or acronyms. However, even if the syntactic and semantic analyses still need further work to enhance the obtainable results, they proved to be effective in successfully integrating shallow techniques, thus justifying the chosen architecture.

7. References

[1] Rocca, G., Spampinato, L., Zarri, G.P., Black, W., and Celnik, P. "COBALT: COnstruction, augmentation and use of knowledge BAses from natural Language documenTs" in *AI-94, 14th International Avignon Conference*, Paris June 1994.

[2] Lewis, D.D. "*Representation and Learning in Information Retrieval*" PhD Thesis, Comp. Sc. Dept.; Univ. of Mass. Amherst, MA Technical Report 91-93, 1992

[3] Gilardoni, L., Prunotto, P., and Rocca, G. "Hierarchical Pattern Matching For Knowledge Based News Categorization" in *RIAO-94*, New York, October 1994.

[4] Wood, M.M. "*Categorial Grammars*". London: Routledge, 1993.

[5] Zarri, G.P. "Semantic Modelling of the Content of (Normative) Natural Language Documents", in *Avignon '92 - Proc. of the Specialized Conference on Natural Language Processing and Its Applications*. Nanterre: EC2, 1992.

[6] Liddy E.D., Paik W. and Yu.S. "Text Categorization for Multiple Users Based on Semantic Features from a Machine-Readable Dictionary" in *ACM Transactions on Information Systems*, 1994-12-3

Copyright © 1995 Software Engineering Press

A Knowledge-based System for Early Warning of Balance of Payments Crises in Emerging Market Countries

Theodore D. Raphael, Ph.D.
Mystech Associates, Inc.
5205 Leesburg Pike, Suite 1200
Falls Church, Virginia 22041

John Varley
Nathan Associates, Inc.
2101 Wilson Blvd., Suite 1200
Arlington, Virginia 22201

Reference Aid for Economic Research (REFER) is a computer-based support environment for balance of payments (BoP) analysis. REFER currently provides three services. First, it displays key economic indicators for BoP analysis (calculated from the International Monetary Fund's International Financial Statistics (IFS) database) and warns the user when anomalous changes occur. Second, REFER assists analysts in assessing the effects of domestic and international events on a country's BoP by applying a body of BoP expert knowledge. Third, it allows the user to browse through the IFS database, conduct statistical analyses, and generate graphs of the IFS data and the results of the statistical procedures.

INTRODUCTION

Mystech has developed knowledge-based systems in domains representative of both the physical sciences and the behavioral sciences. In the former, for example, Mystech has modeled, under a wide range of conditions, the multistep process by which torpedoes are preset for targeting prior to launch. In the latter domain, Mystech has modeled, in the field of political psychology, the decision-making behavior and effectiveness of political leaders.

Key differences between the physical and behavioral sciences have substantial implications for the knowledge base system design and development strategies in their respective domains. The difficulty with which the knowledge is acquired and represented increases with the degree to which the expert's task is cognitive rather than mechanical and probabilistic rather than deterministic. One cannot rely on the immutable laws of the physical sciences to build highly effective explanatory and predictive models of human behavior. All of the variables that affect decision processes cannot be controlled in an experimental setting and manipulated precisely, as one endeavors to develop and test behavioral models. This problem is compounded when one attempts to apply these models to completely uncontrolled operational environments. The difficulty in modeling human behavior

Copyright © 1995 Software Engineering Press

intensifies further as one considers the size and complexity of the models required to account for the relevant variables and the relationships among them. A number of research efforts have addressed this problem (Anderson, 1987; Hudson, 1987; Thorson, 1984).

In Mystech's experience, knowledge-based systems have demonstrated promise in modeling human behavior, as they excel at handling numerous, complex combinations of variables and their interrelationships when they are not subject to *a priori* determination. They also have the capability to integrate probability and confidence factors in the model, a necessity in the behavioral sciences.

SYSTEM DESCRIPTION

Reference Aid for Economic Research (REFER) is a computer-based support environment for balance of payments (BoP) analysis. It supports two major analytic functions. First, it supports the proactive search for information and the detection of significant trends and patterns for analysis. Second, it supports the identification of early warning indicators of potential problems and the assessment of the effects of discrete events that occur with little or no warning.

REFER currently provides three services. First, it displays key economic indicators for BoP analysis (calculated from the International Monetary Fund's International Financial Statistics (IFS) database) and warns the user when anomalous changes occur. Second, REFER assists analysts in assessing the effects of domestic and international events on a country's BoP by applying a body of BoP expert knowledge. Third, it allows the user to browse through the IFS database, conduct statistical analyses, and generate graphs of the IFS data and the results of the statistical procedures.

Statistical Data Analysis

With regard to the first function, the user may apply sophisticated statistical procedures to the IFS or user-created databases, browse through these databases, and dump subsets of data to a spreadsheet for further analysis and fast, easy graphing. The combination of a statistical package, spreadsheet, and the IFS and user-defined databases provides a powerful environment for quantitative analyses of economic data.

Knowledge-based Data Analysis

With regard to the second function, a knowledge base of international economics expertise is employed to provide two additional services, key economic indicators and key economic events.

Key Economic Indicators. Predefined key economic indicators are calculated based on IFS data updated monthly; their levels and trends are measured; anomalous changes and key warning threshold violations are identified based on expert rules; and the results are reported to the user.

A default set of indicator threshold values is customized for each country in the system. The user may also create an optional, user-defined set of threshold values for each country and, thereafter, select which set will be invoked when the IFS database is analyzed. This feature allows the user to adjust the sensitivity of the warning thresholds as desired.

The current key indicators are:

- Budget Deficit/Surplus, Level of (expressed as a percentage of GDP);
- Exchange Rate (nominal), Percentage Change in;
- Exchange Rate (real), Percentage Change in;
- Exports Less Imports;
- Exports Less Imports Trend;
- Foreign Exchange Reserves, Level of (expressed as months of imports (MOI));
- Foreign Exchange Reserves, Percentage Change in (in terms of MOI);
- Interest Rate, Real; and
- Wholesale Prices, Percentage Change in.

Copyright © 1995 Software Engineering Press

Additional indicators are displayed that should be monitored, but for which no thresholds are specified; they are:

- Exports Less Imports, Change in (to be monitored-no specific threshold);
- International Terms of Trade (to be monitored-no specific threshold); and
- Investment Less Savings (to be monitored-no specific threshold).

Line graphs of historical trends are provided for each indicator and for three indicator pairs that represent the terms of the basic balance of payments equation (import/exports, investment/ savings, and government revenues/ expenditures).

By this process, a massive database is analyzed, key economic indicators are evaluated, and the results are returned to the user in seconds.

Significant Economic Events. In addition, an analyst who is confronted with the occurrence of one or more events may enter information about those events, and REFER will assess the magnitude and direction of their effects on the country's balance of payments. Approximately three dozen event types are available for selection (domestic and foreign; financial and nonfinancial), ranging from interest and exchange rate changes to trade regulation and price changes. Event definition data entry forms are provided for each event type, so that the user may define the particular event in terms of key characteristics of analytic value, such as direction of change, specific commodity or instrument involved, temporal boundaries, etc.

The events in the current typology are:

Domestic Financial

- Central Bank Regulation or Operation Change
- Exchange Controls Change
- Exchange Rate Change
- Exchange Reserve Change
- Financial Institution Change
- Inflation/Deflation Change
- Interest Rate Change
- Investment Earnings Change
- Other Capital Flows Change

Domestic Nonfinancial

- Economic Performance (GDP) Change
- Economic Regime Change
- Government Fiscal/Budget Change
- Price Change
- Supply Change
- Quota Change
- Subsidy Change
- Tariff Change
- Unusual Population Shift

Foreign Financial

- Exchange Rate Change
- Exchange Reserve Change
- Foreign Direct Investment Change
- Inflation/Deflation Change, World
- Interest Rate Change
- Investment Earnings Change
- Net Credit (Loan) Availability Change
- Official Assistance Change
- Other Capital Flows Change
- Worker Remittances Change

Copyright © 1995 Software Engineering Press

Foreign Nonfinancial

- Economic Performance (GDP) Change
- Price Change, World
- Supply Change, World
- Quota Change
- Tariff Change
- Trading Arrangement Change

A country profile database was also developed for two purposes in conjunction with the development of the event analysis knowledge base. First, the maintenance of country profiles allows the inference engine to generate analyses, based on general principles of balance of payments theory, that are modified appropriately for each country based on differing country profile factors. For most events, no single analysis is appropriate for all developing countries, according to REFER's domain experts. The analysis should, and does, vary from country to country depending upon one or more country profile factors.

Second, the country profile database supports the display of country-specific information in the user interface. For example, when the user wishes to enter information about an event involving a major world price change for a commodity, he/she selects the event from the event typology above and an event definition data-entry template is displayed. A list of the country's major commodities, called from the country profile database, is then displayed for selection. If the commodity in question is not listed, then the user is to infer that it is not an important commodity for the country under study (as only major traded and nontraded commodities are listed in the country profile).

For example, in the commodity price change event described above, the user would select the appropriate event type from the event typology and indicate the direction of change (increase/decrease) on the event definition data-entry template. REFER would then check the country profile to determine if the commodity is a major export, import, or nontraded (but tradeable) item, note demand and/or supply elasticities, as appropriate, and draw appropriate inferences accordingly regarding the impact of the price change on the country's balance of payments.

The country profile also contains characteristics that are fundamentally relevant to a broad range of events. For example, exchange rate regime type (fixed/crawling peg/managed float/free float) affects the analyses of many events, particularly the domestic and foreign financial events.

The current country profile factors are:

- economy type (agriculture/industry/ service/mixed);
- elasticities, demand (for each major commodity-elastic/inelastic);
- elasticities, supply (for each major commodity-elastic/inelastic);
- exports (major);
- export competitors (major);
- export customers (major);
- imports (major);
- import suppliers (major);
- level of import controls (low/high);
- monoculture (yes/no);
- nontraded (major);
- raw materials; and
- stock market (yes/no).

A world profile database has also been developed which currently maintains lists of:

- major industrialized countries;
- international financial centers; and
- large developing countries.

Copyright © 1995 Software Engineering Press

For example, if a foreign country (i.e. a country other than the one being analyzed) experiences a major adjustment in its interest rate, then the analysis generated of the impact of the event on the focal country's balance of payments will vary depending on whether or not the foreign country is also an international financial center.

Both the country and world profiles are easily modified and updated through the user interface by the system administrator.

Thus, REFER employs a body of expert knowledge and conducts event analyses based on user-entered event-definition information and input from the country and world profiles. The analyses focus on estimates of the effects inferred from these data on the country's balance of payments (current and capital accounts). The analyses also provide guidance in assessing the importance of the event for the target country's balance of payments and other economic effects, the temporal factors involved (time lag and duration of effects), and common analytic traps to be avoided.

REFER Session Report

At the end of a REFER session, the user may request a written report that is sent to a word processor and displayed for review. The report contains three major sections:

- Key Economic Indicator Status Report-based on analysis of the IFS database and the selected set of indicator threshold values;
- Event Analyses-report of the analyses of entered events generated by the inference engine and knowledge base; and
- Balance of Payments Data-table of the principal balance of payments accounts, based on IFS data.

Adding New Countries to REFER

New countries may be incorporated into the REFER system literally within a matter of minutes. The system administrator need only add the name of the country to the Master List (with IFS country code), create a country profile and set indicator warning thresholds through the country profile and indicator threshold editing utilities, and execute the provided REFER software to download the new country's IFS data from IMF's CD ROM to REFER's SQL Server database. The new country is then available for BoP analysis with the full range of REFER's suite of tools.

CRITICAL SYSTEM DESIGN PROBLEMS AND ISSUES

Perfect design solutions that obviate the need to accept tradeoffs in pursuit of multiple goals are rare. The design of REFER is no exception. Embedded in a key functional requirement of the system are three goals:

> **REFER must possess the ability to treat *any* domestic or foreign economic event that may occur at *any* time in the future with regard to its impact on the BoP of *any* developing country.**

This requirement raises four technical challenges. How can REFER be designed:

- to assess information about an almost infinite number of future events in a manner that is general enough to encompass the universe of possible cases, but specific enough to differentiate among them sufficiently to provide assessments relevant to the circumstances of a particular case?
- to assess the BoP impacts of an event in a manner that is broad enough to apply to a wide range of developing countries, but is tailored sufficiently to the focal country to be valid and accurate for a particular case;

Copyright © 1995 Software Engineering Press

- to reflect accurately the characteristics of the countries and the international system on which its inferences are partially based, as those characteristics change over time, without requiring onerous maintenance of the system or, even worse, time-consuming and costly continual iterative development (in an effort to modify the knowledge base to reflect accurately the inexorably changing economic and political systems of the world); and
- to assess the magnitude of expected change for a specified event with sufficient precision to provide a meaningful assessment on the effects on BoP.

To address the first problem, Mystech and Nathan developed the event typology and event definition data-entry templates to provide a compromise between an attempt to accommodate an infinite number of specific events or force a choice among highly generalized events that may be marginally relevant to any specific event.

To be effective, the typology must reflect the universe of events that can affect the balance of payments. To meet this requirement, the typology was systematically designed to reflect the inputs to all of the elements of the BoP accounting structure (current account, capital account, and their respective subaccounts) and reflect as well the terms of the fundamental relationships expressed in the BoP identity: $x-m = (i-s) + (t-g)$, (where x = exports, m = imports, i = investment, s = savings, t = government revenues, and g = government expenditures).

The event data-entry template enables the user to provide REFER with information about a particular event that goes beyond defining it as simply one of the general event types. On the template, the user can specify the direction of change (characterized appropriately for the event type, e.g. price change-increase/decrease or exchange controls change-tighten/relax) and other factors that define the event. In this manner, REFER obtains sufficient information to provide an assessment that is relevant to the user's particular case, without requiring the impossible, near impossible, or at least impractical-maintenance of an exhaustive (and exhausting) catalogue of potential specific cases.

Another problem with the catalogue method, of course, is the maintenance burden of adding potential cases as the future reveals situations previously unanticipated, e.g. the oil price shocks of the 1970s.

With regard to the second and third problems, if a decision is made to implement knowledge that is tailored to each country and its contemporary problems in order to maximize policy relevance, then the knowledge base may be valid for only a short period of time, since circumstances in an economy and political system can change rapidly. It also makes the addition of new countries a labor-intensive process, since reliance on country-specific knowledge requires the acquisition, representation, and implementation of a new body of knowledge for each additional country.

On the other hand, if general models of developing countries and international economic theory form the core of the knowledge base to maximize the useful life of REFER and to minimize maintenance burdens, then one runs the risk of providing a system that, in any specific situation, offers guidance that is too simple, too general, and of marginal relevance to be of much analytic value.

To address this dilemma, Mystech and Nathan structured the knowledge by relating a core knowledge base of BoP theory for each event to the country profile of the focal country and the world profile, so that for any single situation (i.e. need to assess the BoP effects of a specified event for a specified focal country) the analysis is general enough to be robust over time and circumstances, but sufficiently tailored to the focal country, through use of the country profile, to be relevant to the specific case.

By this approach, a flexible, robust, but relevant analytic aid is maintained with little effort as the world changes and the system is expanded to accommodate additional countries.

Copyright © 1995 Software Engineering Press

The fourth problem concerns the determination of sufficient precision in expressing the magnitude of BoP effects that result from the occurrence of a given event. This issue was considered at length during the design phase, and, for three principal reasons, the decision was made to eschew econometric modeling in favor of a general nonparametric approach.

First, the risk of failure was deemed to be unacceptably high, especially considering the cost of developing econometric models. While there was high confidence in our ability to develop a knowledge base that would support forecasting of the direction of effects and nonparametric measures of their magnitude (e.g. very large/large/moderate/low/nil), there was considerably less confidence in our ability to develop a knowledge base that could generate forecasts with the level of precision associated with econometric models, while meeting simultaneously an acceptable standard of accuracy.

Second, an econometric approach requires very substantial database support. Missing or substantially lagged data can cause major problems that may render the system inoperable from time to time; this is a significant and unacceptable vulnerability.

After an exhaustive survey, we found that the IMF/IFS database was the most complete and relevant database available for BoP analysis; but even IFS contains missing data and, in some cases, significantly lagged data. Moreover, we found that data for certain critical variables, such as those that measure indebtedness, are not available reliably from any sourcein electronic form for developing countries. For example, the most detailed and comprehensive source that we found, the World Bank's World Debt Tables are updated only annually.

Third, for an analytic tool to be useful for BoP analysis, its forecasts do not need to be expressed in parametric terms. One purpose of REFER is to warn against analytic traps that lead to erroneous conclusions in analyzing the BoP effects of events. This purpose is well-served, therefore, when REFER provides accurate guidance regarding the expected direction of change and qualitative estimates of its magnitude.

For these reasons, we concluded that, although an econometric approach would raise forecasting precision, the result would yield marginal analytic utility at substantially increased technical risk and development costs.

KNOWLEDGE BASE DEVELOPMENT METHODOLOGY

Mystech employs a proprietary methodology for knowledge acquisition, representation, and implementation. Structured interviews are conducted with the domain experts, they are audio and video recorded, verbatim transcripts are produced, the transcripts are analyzed by knowledge engineers, and the knowledge is extracted from the transcripts and structured in an object-oriented, hyper-text knowledge representation environment. The represented knowledge is then implemented in software code.

The system is then exercised with test scenarios and the output is validated by the original domain experts working with knowledge engineers in an iterative fashion until acceptance is achieved.

For REFER, Mystech and Nathan have conducted 46 knowledge acquisition sessions with leading international economists, yielding over 200 interview hours of acquired, represented, implemented, and validated knowledge.

SOFTWARE DEVELOPMENT TECHNOLOGY

REFER is currently under iterative development and employs an object-oriented approach. Inference Corporation's Automated Reasoning Tool for Information Management (ART-IM) is being employed in the development of the event-assessment knowledge base. The knowledge is being implemented in frames and production rules with a forward chaining approach.

Copyright © 1995 Software Engineering Press

The graphical user interface is being developed with Glockenspiel's CommonView interface development tool (now published by Computer Associates, Inc.) and C++.

A key feature of the software architecture is the separation of the knowledge base and user interface; the information that appears on the display is controlled almost entirely by the knowledge base. As the knowledge base expands (with additional countries, indicators, events, etc.), necessary changes to the interface will be handled automatically by the knowledge base and will require little or no modification of the interface software.

The ART-IM/C++ combination and software module separation also maximize REFER's cross-platform capability (e.g. MS DOS/Microsoft Windows, UNIX/X-Widows (OSF Motif), etc).

REFER runs under OS/2 on IBM-compatible personal computers and is a Presentation Manager application.

FUTURE RESEARCH AND DEVELOPMENT

The following enhancements are currently undergoing research and development:

- To this point in its development, the knowledge base focuses on estimating the effects of economic events on the BoP of developing countries. Questions remain to be addressed. What policies/instruments are available to governments to respond to BoP problems? How are particular governments (perhaps according to economic regime type) likely to respond to particular problem types? In the view of expert economists, how should they respond? If there is a discrepancy between how they are likely to respond and how they should respond, what are the implications for the likely, as opposed to the expected, effects of the anticipated policy choice?;
- In the wake of the 1994 balance of payments crisis in Mexico, we have begun an effort to develop a BoP Crisis Early Warning (BoP/CEW) module for REFER that complements the event assessment module. The latter reflects a forward chaining approach, in which economic events are assessed for their unknown impacts on the BoP of developing countries. Here the stimulus (i.e. an event) is known but the outcome (i.e. effects on BoP) is unknown; thus the reasoning process employed is characterized as forward chaining. For the BoP/CEW, however, backward chaining is appropriate. In this case, the outcome is known or postulated (i.e. the occurrence of a BoP crisis), but the antecedents to it (i.e. early warning indicators) are unknown. Initial knowledge acquisition has been conducted to address the issue of how expert international economists attempt to anticipate BoP crises;
- A Case-based Reasoning (CBR) tool is currently under development. This tool will provide users with the ability to identify and compare historical cases with known outcomes (in terms of BoP effects) to a current situation with similar characteristics for which the outcome is yet unknown. This comparative approach provides a framework for drawing inferences about the likelihood of possible outcomes of the current situation based on an analysis of similar historical cases; and
- REFER currently uses the IFS database as supplied by IMF. Unfortunately, as noted above, the data may be missing or delayed in reporting. Under development is the capability of extending IFS data with user-supplied data, as desired, to provide the most complete and current database possible.

REFERENCES

Anderson, P.A. Using Artificial Intelligence to Understand Decision Making in Foreign Affairs: The Problem of Finding an Appropriate Technology. In S.J. Cimbala (Ed.), *Artificial intelligence and national security.* Lexington, MA: Lexington Books, 1987.

Copyright © 1995 Software Engineering Press

Hudson, V.M. Using a Rule-Based Production System to Estimate Foreign Policy Behavior: Conceptual Issues and Practical Concerns. In S.J. Cimbala (Ed.), *Artificial intelligence and national security.* Lexington, MA: Lexington Books, 1987.

Thorson, S.J. Intentional Inferencing in Foreign Policy: An AI Approach. In D.A. Sylvan & S. Chan (Eds.), *Foreign policy decision making: perception, cognition, and artificial intelligence.* New York: Praeger, 1984.

Copyright © 1995 Software Engineering Press

Paper Session: Optimization: Portfolios and Profit

Chair: Ken Kleinberg, Gartner Group

Copyright © 1995 Software Engineering Press

A Genetic Algorithm Approach to Optimizing Portfolio Merging Problems

William Edelson
Computer Science Dep't
Long Island University
University Plaza
Brooklyn, N.Y. 11201

Michael L. Gargano
Computer Science Dep't
Pace University
Pace Plaza
New York, N.Y. 10038

Abstract

The portfolio merging problem can be viewed as finding the optimal mix of k different categories of portfolios in a combined aggregate portfolio to maximize expected profit or minimize risk subject to numerous constraints. This optimization problem is important in asset allocation applications. Conventional optimization techniques have been used effectively in the past on problems involving the merging of portfolios. However, there are many real world portfolio merging problems whose solution do not lend themselves readily to conventional techniques. A genetic algorithm (GA), a biologically inspired optimizing search procedure, is more suitable for solving these types of problems.

We apply a GA to various portfolio merging problems leading up to the problem of maximizing the return/risk ratio with the added constraint of a satisficing expected return. We make use of goal programming techniques to recast the problem into one that is more suitable for solution by a GA. This is shown to have advantages in generating initial feasible solutions quickly, without compromising the effectiveness of the GA.

1 Introduction

The portfolio merging problem can be viewed as finding the optimal mix of k different categories of portfolios in a combined aggregate portfolio to maximize expected profit or minimize risk subject to numerous constraints. The portfolio categories might be equity sectors such as banking, energy, healthcare, utilities, etc., or they might be asset categories such as corporate bonds, equities, money market instruments, municipal bonds, etc.

Conventional optimization techniques have been used effectively in the past on problems involving the merging of portfolios. However, there are many real world portfolio merging problems whose solution do not lend themselves readily to conventional techniques. These more complicated problems are usually characterized by a solution space which is unstructured (eg, multimodal), discontinuous, or poorly understood. A genetic algorithm (GA), a biologically inspired optimizing search procedure, is more suitable for solving these types of problems.

The genetic algorithm paradigm [1,4] is an adaptive method based on Darwinian natural selection. It applies operations of reproduction (based on survival of the fittest),

Copyright © 1995 Software Engineering Press

crossover, and mutation, to a given population of potential solutions to generate a new, more fit, population of potential solutions. The process repeats itself until it converges to a stable optimal (or near optimal) solution. A GA is particularly suitable for multi-parameter optimization problems with an objective function subject to numerous hard and soft constraints.

We apply a GA to three portfolio merging optimization problems. We first investigate a mean-variance optimization problem solving for the optimal allocation mix which minimizes risk subject to an equality constraint for the expected return. Next, we maximize the expected return subject to an inequality constraint (upper bound) for the risk . Lastly, we maximize the return/risk ratio subject to the additional constraint of a satisficing expected return. We make use of goal programming techniques in these problems to recast them to ones that are more suitable for solution by a GA. This is shown to have advantages in generating initial feasible solutions quickly, without compromising the effectiveness of the GA.

2 Model

An investment is allocated among k portfolios

$$P_1, P_2, \ldots, P_k$$

in accordance with corresponding portfolio allocation weights

$$C_1, C_2, \ldots, C_k \, .$$

The Expected Total Return is:

$$F(C_1, \ldots, C_k) = C_1 * \overline{P}_1 + \cdots + C_k * \overline{P}_k$$

where $\overline{P}_j$ is the expected return of the jth portfolio and must be measured statistically. The Total Risk is:

$$R(C_1, \ldots, C_k) = C_1^2 * R_1 + \cdots + C_k^2 * R_k$$

where R_j is the risk in the jth portfolio and must be measured statistically.[1]

The following additional constraints are applied to the model:

$$C_1 + C_2 + \cdots + C_k = 1$$

$$0 < L_j \leq C_j \leq U_j < 1$$

where L_j and U_j define the range of the portfolio allocation weights C_j and generally depend on market fundamentals and/or investor preferences.

3 Optimization Problems

Problem #1

Minimize portfolio risk subject to an equality constraint for the expected return. That is:

$$minimize \; R(C_1, C_2, \ldots, C_k)$$

subject to the constraints:

$$F(C_1, C_2, \ldots, C_k) = D$$

$$C_1 + C_2 + \cdots + C_k = 1$$

$$0 < L_j \leq C_j \leq U_j < 1 \, .$$

This is a mean-variance optimization problem producing an efficient portfolio (minimum risk) for a given expected return (D).

[1] For simplicity we are assuming there is no statistical correlation between portfolios; however, the results should generalize to the case of correlated portfolios.

Copyright © 1995 Software Engineering Press

Problem #2

Maximize the expected return subject to an inequality constraint for the risk. That is:

$$maximize\ F(C_1, C_2, \ldots, C_k)$$

subject to constraints:

$$R(C_1, C_2, \ldots, C_k) \leq b$$

$$C_1 + C_2 + \cdots + C_k = 1$$

$$0 < L_j \leq C_j \leq U_j < 1\ .$$

This optimization problem can be considered a risk averter and is treated in [6,10].

Problem #3

Maximize the return/risk ratio subject to the constraint of achieving at least an expected return (D). That is:

$$Maximize\ \left[\frac{F(C_1, C_2, \ldots, C_k)}{R(C_1, C_2, \ldots, C_k)}\right]$$

subject to the constraints

$$F(C_1, C_2, \ldots, C_k) \geq D$$

$$C_1 + C_2 + \cdots + C_k = 1$$

$$0 < L_j \leq C_j \leq U_j < 1\ .$$

In contrast to the model of problem #1, this model permits the acceptance of a higher risk to realize a higher profit. This optimization problem is a basic underpinning of modern portfolio theory and is similar to a problem investigated in [8].

4 Goal Programming

We borrow from the technique of goal programming [2,3], which stresses the satisfaction of multiple objectives, to recast the optimization problems in section 3.0 into ones that are more suitable for solution by a GA. Goal programming indirectly determines the unknown variables in an optimization problem by directly minimizing positive and negative deviations from the goal constraints' right hand side values. In effect, it uses slack variables. This process of minimizing deviations from a prespecified level (rather than satisfying this level absolutely) relaxes the rigidity of the constraint and results in a larger space of feasible initial populations for the GA. This suggests the possibility of generating initial feasible solutions quickly, even for problems with numerous and complicated constraints.

Recasting Problem #1, we have a new objective function to minimize:

$$minimize\ \{R(C_1, C_2, \ldots, C_k) + \beta\,(d^+ + d^-)\}$$

subject to new constraints:

$$F(C_1, C_2, \ldots, C_k) - d^+ \leq D$$

$$F(C_1, C_2, \ldots, C_k) + d^- \geq D$$

$$C_1 + C_2 + \cdots + C_k = 1$$

$$0 < L_j \leq C_j \leq U_j < 1$$

$$d^+ \geq 0,\ d^- \geq 0$$

where d^+ and d^- are the positive and negative deviations (ie, the slack), respectively, from the threshold expected return D. Here, $\beta = \beta(d^+, d^-)$ scales the deviations d^+ and d^- in the new objective function in accordance with their importance[2].

[2]Large values of the deviations were scaled quadratically while small ones were scaled linearly.

Copyright © 1995 Software Engineering Press

5 GA Methodology

Our genetic algorithm requires an initial population of feasible members (potential solutions satisfying the constraints), an evaluation function to score each member of the population, conventions for creating new members of the population by mating and random mutation, and a grim reaper mechanism to discard low scoring members of the population to make room for new ones.

In these optimization problems, the population $POP = \{(C_1, C_2, ..., C_k)\}$ is a large subset of all the feasible members. POP is initially chosen randomly and consists of a large but finite number of members. Each C_j is encoded with s bits. Thus, the encoding is a bit string of $k * s$ bits. The evaluation function (objective function) scores the performance or worth of individual members of the population. (In our optimization problems where we minimize risk, the closer the score is to zero, the higher is the score). The mating convention is such that only high scoring members will preserve and propagate their "worthy" characteristics from generation to generation and thereby help in continuing the search for an optimal solution.

Selection of parents for mating involves choosing one member from the high scorers by a "roulette wheel" approach and choosing the other member randomly. The reproductive process is a simple crossover operation where two selected parent members (bit strings) are cut into head and tail sections at some randomly chosen position and then have their tails swapped to create two offspring members. The crossover operation is repeated on the parents until at least one offspring is feasible. A grim reaper mechanism replaces two low scoring members in the population with two newly created feasible offsprings or one newly created feasible offspring and one parent. Mutation is a GA mechanism where we randomly choose a member of the population and change one randomly chosen bit in its bit string representation. If the mutant member is feasible, it replaces the member which was mutated in the population. This process is done infrequently and is useful in creating new areas of search.

We can now state the genetic algorithm which we used:

Step 1: Initialize a large feasible population.

Step 2: Evaluate any member which has not yet been evaluated.

Step 3: Sort the members of the population by their scores.

Step 4: Select parents for mating from the upper three quartiles of the population; one using a "roulette wheel" approach and one randomly.

Step 5: Generate offsprings using simple crossover. Replace the lower quartile of members of the population with feasible offsprings.

Step 6: Mutate a randomly selected member of the population at a randomly selected bit once every generation.

Step 7: If time is up then return best solution found else go to Step 2.

6 Results

We generated a number of computer solutions for these optimiztion problems using the GA package GENESIS [7]. and a customized GA software package developed in C++ by the authors. The input data for these problems [9]

Copyright © 1995 Software Engineering Press

consists of the percentage annual expected returns and risks for a three-asset class portfolio as shown in the table below:

Class	*Return*	*Risk*
Stocks	.13	.034200
Bonds	.08	.003600
Treasury Bills	.06	.000016

Estimates of the weights for an efficient portfolio producing a yield of 8% (Problem #1) using a GA with a population of 32 without Goal Programming techniques are: $C_1 = 0.1666$, $C_2 = 0.4166$, and $C_3 = 0.4166$. Convergence ocurred after 12 generations. This is a near-optimal result which compares favorably to the true efficient portfolio of $C_1 = 0.16$, $C_2 = 0.44$, and $C_3 = 0.40$ calculated by quadratic programming techniques [5]. The number of tries to generate an initial feasible population is 6,527. Solving the same optimization problem using a GA with a population of 32 with Goal Programming techniques described in section 4.0 also converges to $C_1 = 0.1666$, $C_2 = 0.4166$, and $C_3 = 0.4166$ after 12 generations. However, the number of tries to generate an initial feasible population is reduced to 780, an improvement of about a factor of 8.

Similar results were observed for Problem #2 and Problem#3.

7 Conclusions

Genetic Algorithm solutions of portfolio merging problems which we generated consistently compare favorably to known solutions. Recasting the optimization problem using goal programming techniques (slack variables) is shown to have advantages in generating initial feasible solutions quickly, without compromising the effectiveness of the GA. This suggests the posibility of significantly reducing the time to generate an initial feasible population for larger portfolios problems with complicated and numerous constraints.

We feel that the genetic algorithm paradigm is both a powerful and flexible tool for solving portfolio merging problems.

8 References

References

[1] Goldberg, D., Genetic Algorithms, Addison Wesley Publishing Co., 1989.

[2] Ignizio, James P., Goal Programming and Extensions, D.C. Heath & Co., 1976.

[3] Schniederjans, Marc J., Linear Goal Programming, Petrocelli Books Inc. , 1984.

[4] Davis, L., Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.

[5] Hillier, F. and Lieberman, G. , Operations Research, second edition, Holden - Day Inc., 1974.

[6] Gargano, M.L., Chamoun, P., von Kleeck, D.L., Using Genetic Algorithms to Solve Financial Portfolio Problems Related to Optimal Allocation, Portfolio Insurance, and Performance Prediction in Second Int'l. Conference on AI Applications on Wall St., New York., 1993.

[7] Grefenstette, J.J., Davis, L., Cerys, D., GENESIS and OOGA: Two Genetic Algorithm Systems. TSP, Melrose, Calif., 1991.

[8] Freedman, Roy S., Digiorgio, Rinaldo, A Comparison of Stochastic Heuristics For Portfolio Optimization in Second Int'l. Conference on AI Applications on Wall St, New York. , 1993.

Copyright © 1995 Software Engineering Press

[9] Arnott, R., Fabozzi, F., Active Asset Allocation, Probus Publishing Co., Chicago., 1992.

[10] Haugen, R., Modern Investment Theory, Prentice-Hall, N.J., 1986

Copyright © 1995 Software Engineering Press

Genetic Algorithms for Predicting Individual Stock Performance

Sam Mahfoud & Ganesh Mani

LBS Capital Management, Inc.
311 Park Place Blvd., Suite 330
Clearwater, FL 34619
Phone: (813) 726-5656
E-mail: {sam, ganesh}@lbs.com

Abstract

Genetic algorithms are applied to predicting the performance of individual stocks. A method is introduced that extends genetic algorithms from optimization problems to classification and prediction problems. The resulting genetic algorithm system is compared to a neural network system.

Introduction

Artificial Intelligence (AI) techniques have been used for a number of tasks ranging from playing grandmaster-level chess to predicting protein secondary structures. Expert systems, neural networks, and genetic algorithms are three popular AI paradigms. Typically, characteristics of the application domain have determined the particular AI method that is employed.

The investment management domain is particularly challenging because of the abundance of noisy, numeric data and the lack of strong theories of how stock prices move. Even though the efficient market hypothesis (Fama, 1970) has been challenged from a number of directions, finding patterns of persistent predictability is difficult, and typically requires high-speed computers and clever algorithms. AI techniques can facilitate the quest for these patterns of predictability.

In the capital markets and other complex, real-world domains, it is hard to specify *a priori* a good set of rules, making knowledge engineering an extremely difficult task. It is preferable to *learn* (automatically acquire from data) a good set of rules. Expert systems typically do not learn. Both genetic algorithms and neural networks, on the other hand, have the ability to learn from the vast amount of data available in the financial domain.

While expert systems and neural networks are seeing increasing use in finance (Barr & Mani, 1993; Fishman, Barr, & Loick, 1991; Hall, 1994; Hutchinson, Lo, & Poggio, 1994; White, 1994), genetic algorithms (GAs) are relative newcomers. We concentrate on GAs in this study and illustrate how they can be applied to the task of predicting the performance of individual stocks. We compare the performance of a GA system that incorporates a *niching method* to that of a neural network system. The two systems predict the movement in the price of a stock, relative to the market.

A criticism of neural-network approaches has been that they are black boxes, and that the user can not readily comprehend the final rules that these systems acquire and subsequently use to make decisions. An advantage GAs offer is that, like expert systems, they are capable of producing user-readable rules, along with the reasoning underlying each particular rule. The key to such explanation capabilities is choosing a rule format for the GA that the end user can easily understand.

Copyright © 1995 Software Engineering Press

Genetic Algorithms

Genetic algorithms are general-purpose search techniques for solving complex problems. Based upon genetic and evolutionary principles, GAs work by repeatedly modifying a population of artificial structures through the application of selection, crossover, and mutation operators. GAs have traditionally been used for optimization, but with a few enhancements can be applied to classification and prediction as well.

The choice of an appropriate structure for a particular problem is a major factor determining a GA's success. GAs are capable of operating upon a variety of structures, including binary strings (Goldberg, 1989), computer programs (Koza, 1992), neural networks (Whitley, Starkweather, & Bogart, 1990), and *if-then* rules (Bauer, 1994).

We first examine how a traditional GA performs optimization. The goal in optimization is ideally to find the best possible solution to a problem. In real-world problem-solving, one does not usually know the *best* possible solution. Therefore, a more realistic objective is to find a *good* solution; or, given a current benchmark, to search for a *better* solution. A GA's *fitness function* measures the quality of a particular solution.

The traditional GA begins with a population of n randomly generated structures, where each structure encodes a solution to the task at hand. The GA proceeds for a fixed number of generations. During each generation, the GA improves the structures in its current population by performing selection, followed by crossover, followed by mutation. After a number of generations, the GA converges, meaning that all structures in the population become identical or nearly identical. The user typically chooses the best structure of the last population as the final solution.

Selection is the population improvement or "survival of the fittest" operator. Basically, it duplicates structures with higher fitnesses and deletes structures with lower fitnesses. A common selection method is to randomly choose two structures from the population and hold a tournament, advancing the fitter structure to the crossover stage. A total of n such tournaments are held to fill the input population of the crossover stage.

Crossover, when combined with selection, results in good components of good structures combining to yield even better structures. Crossover forms $n/2$ pairs from the n elements of its input population. Each pair advances two offspring structures to the mutation stage. The offspring are the results of cutting and splicing the parent structures at various crossover points. The crossover stage advances a total of n elements to the mutation stage.

Mutation creates new structures that are similar to current structures. With a small, prespecified probability, mutation randomly alters each component of each structure. The mutation stage advances n elements to the selection stage of the next generation, completing the cycle.

Figure 1 illustrates one generation of a GA with a population of size $n = 4$. Structures are represented as rectangles, each containing two square components. Components may take on one of two values, represented by the colors, black and white. Assume that structures with two black components have the highest fitness, structures with two white components have the lowest fitness, and mixed structures have intermediate fitness.

The selection stage holds four tournaments between randomly chosen pairs of individuals. The crossover stage then cuts and splices structures at component boundaries. Finally the mutation stage, through random choices, mutates only the leftmost component of Structure 9, yielding Structure 13. Although the initial population has no optimal structures, after the generation shown, an optimal structure emerges (Structure 14).

Copyright © 1995 Software Engineering Press

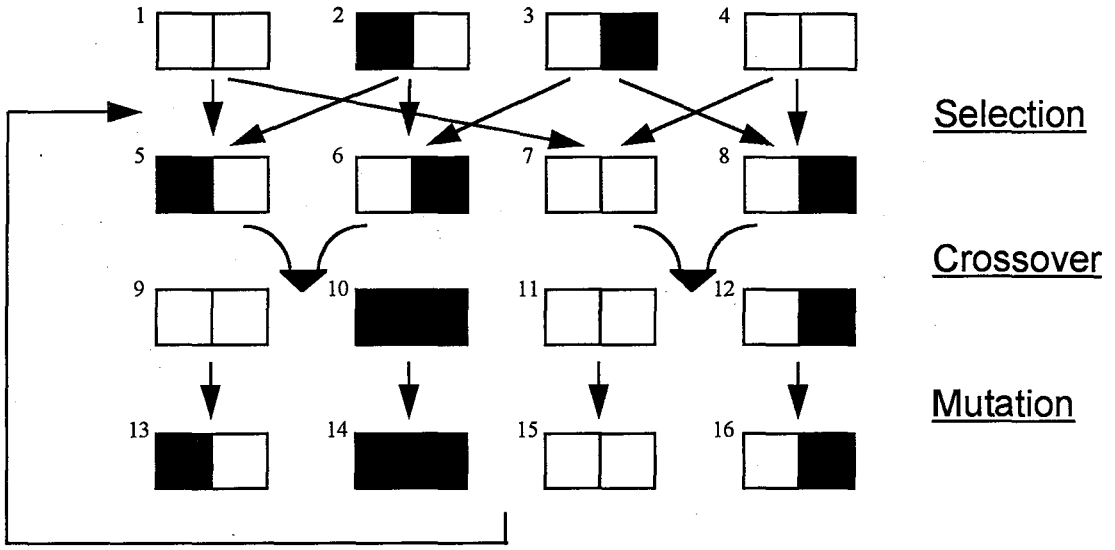


Figure 1. One generation of a genetic algorithm

Compared to traditional parameter optimization techniques, genetic algorithms offer several advantages. The first advantage is general applicability. GAs do not require information such as gradients. If a problem is not differentiable or otherwise well-behaved, many traditional optimization techniques will be of no use. A second advantage is resilience in handling the local optima of difficult problems. While traditional techniques are likely to converge to a local optimum once they are in its vicinity, GAs conduct search from many points simultaneously, and are therefore more likely to find a global optimum. GAs are designed to handle highly nonlinear spaces.

Genetic Algorithms for Financial Prediction

A few prior studies have attempted to apply genetic algorithms to financial prediction and related tasks. Most recently, Richard Bauer's (1994) book has recommended GAs for stock selection. His GA finds thresholds for one or more variables, above or below which a stock is considered attractive. For instance, if the GA's structure consists of two variables representing a particular stock's price and earnings per share (EPS), the final rule the GA returns might look like the following:

IF [*Price* < 15 and *EPS* > 1] THEN Buy

Past performance of a particular rule over some time period serves as a GA's fitness function. The user is responsible for choosing a potentially good rule structure for the GA. While Bauer limits his attention to rules of the above form, other formats are also possible, such as rules with fuzzy variable bounds or outcomes. Beyond stock selection, Bauer's approach also allows a user to test simple relationships among variables when the user has only a vague idea of potentially good variables.

Sikora and Shaw (1994) apply GAs to predicting loan defaults and company bankruptcies. One of their approaches is to run a GA repeatedly, each time generating a different rule. They combine all final rules to form a database of rules. Allen and Karjalainen (1993) utilize genetic programming, a type of GA that operates on computer-program structures, to find trading rules for the S&P 500. Packard (1990) applies GAs to time-series forecasting.

Copyright © 1995 Software Engineering Press

Complex applications such as predicting stock performance are often highly nonlinear tasks. This means, for example, that when a 10% increase in one variable results in a buy signal, a 20% increase in that same variable could result in a sell signal. Often a single, simplistic rule is insufficient to model relationships among financial variables. Sometimes what is required is a combination of rules. For example, consider the following two trading rules based on fundamental analysis.

Rule 1: IF [*P/E* > 30] THEN Sell
Rule 2: IF [*P/E* < 40 and *Growth Rate* > 40%] THEN Buy

Given only the price-to-earnings (P/E) ratio of a particular stock, a good rule of thumb is that stocks with too high a P/E -- in this example greater than 30 -- are unattractive. Rule 1 encodes this rule of thumb. However, if a stock is a high-growth stock, one may wish to make an exception to Rule 1. Rule 2 encodes such an exception. The interacting combination of a general rule (Rule 1) plus an exception (Rule 2) results in a better trading strategy than either rule alone.

Traditional GAs return only one solution. To return interacting combinations of solutions, it is necessary to extend the genetic algorithm through use of a *niching method* (Mahfoud, 1992, 1995a, 1995b). Unlike the traditional GA, which makes the population eventually converge around a single point in the solution space, the GA that uses a niching method converges about multiple solutions or niches. The word *niche* comes from the field of ecology, indicating the particular environmental factors that are favorable for a particular species. The analogy in the financial forecasting case is that different rules within the same GA population can perform forecasting for different sets of market and individual company conditions, contexts, or situations.

The extension of a genetic algorithm using a niching method allows the GA to return a final population that is similar in many ways to the knowledge base of an expert system. In fact, niching GAs represent one method of acquiring knowledge for an expert system -- without requiring a human expert. Note that in an expert system with 100 rules, where each rule contains 15 variables and each variable can take on one of 32 different values, 3.8×10^{22} possible rules exist. This is more rules than the fastest computer could expect to evaluate in a person's lifetime. The number of possible *combinations* of 100 rules is much higher. Most possibilities would undoubtedly produce poor performance. Therefore, it would be nearly impossible to find one of the best sets of rules by trying arbitrary combinations. A financial expert could perhaps develop a good set of rules by weeding out most possibilities using his or her experience and background knowledge. However, that expert would likely produce a set of rules qualitatively different than one produced by a GA, due in part to the expert's *a priori* bias against counterintuitive or contrarian rules.

Neural Networks

A neural network is a general-purpose model for handling pattern recognition or classification tasks. Neural networks were conceived as models of the brain, but are in fact artificial computational models that roughly mimic simple operations of real neurons. For computational finance tasks, neural networks are useful because of their ability to handle large amounts of noisy, numeric data.

A neural network is an appropriately connected set of simple processing elements or nodes. Connections between nodes have a strength or weight associated with them. Each weight is initially set to a random value. As the network "learns" to classify or recognize patterns, the weights change. A popular algorithm for effecting these weight changes is *backpropagation* (Rumelhart, 1986). The central idea behind backpropagation is to use the error -- the difference between the network's current prediction and the actual answer -- to adjust the weight on each link. This process repeats several times, each time reducing the error by a small amount.

Copyright © 1995 Software Engineering Press

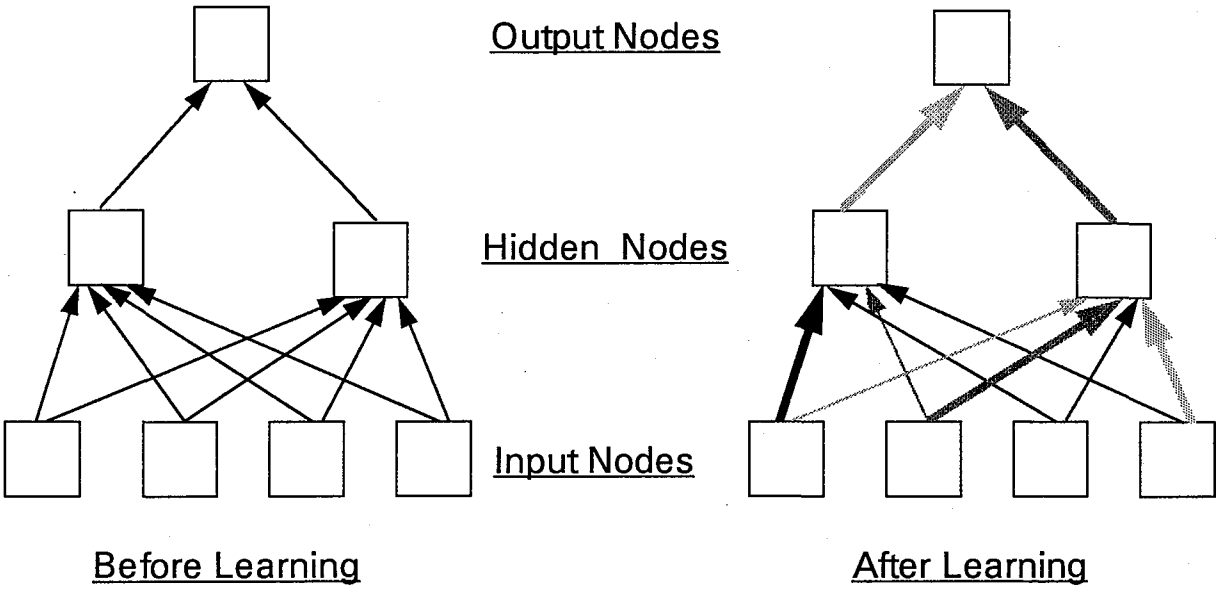


Figure 2. The training of a neural network

Figure 2 shows an example of a simple neural network with four input nodes, two hidden nodes, and one output node. Before learning, all of the weights are set to small random values. After learning, some of the weights become strengthened, and others weakened, as depicted by the various shaded connections.

Preliminary Results

As a preliminary set of experiments, we attempt to predict the return, relative to the market, of a MidCap stock randomly selected from the S&P 400. We call the chosen stock *X*. We employ fifteen proprietary inputs representing technical as well as fundamental information about the stock. The GA operates upon all of the inputs to find favorable or unfavorable combinations of circumstances with respect to the output variable being predicted. The GA, in combination with the niching method it employs, evolves a population of over 100 interacting rules. Each rule indicates, in terms of the fifteen indicators, the particular conditions that produce various price behaviors relative to the market. We look at increases and decreases over 331 past, overlapping, 12-week time periods, as well as the values of the fifteen indicators over these past time periods. Each time period is hence an *example*. 70% of these examples are used to form a fitness function for the GA. (In the parlance of machine learning, this is the *training set*.) 20% of the examples are used to decide when to terminate the GA. (This is the *stopping set*.) Finally, 10% of the examples are used to measure performance. (This is the *out-of-sample* or *test set*.)

We perform 100 different experimental runs of the GA, allocating examples to the three sets randomly for each experiment. The results we report are for the 100 out-of-sample sets. The GA returns one of three predictions for each out-of-sample example: *up*, *down*, or *no prediction*. Averaged over the 100 out-of-sample sets, the GA correctly predicts Stock *X*'s direction relative to the market 47.6% of the time, produces no prediction 45.8% of the time, and incorrectly predicts the direction relative to the market 6.6% of the time. Thus, over half of the time (47.6% + 6.6%), the GA makes a prediction. When it does make a prediction, the GA is right 87.8% of the time. Note that it would be possible to force the GA to make a prediction each time. However, the no-prediction option allows the GA to indicate those times when a stock is nearly equally likely to move in either direction.

Copyright © 1995 Software Engineering Press

We apply a second performance measure, called the *average alpha score*, that takes into account magnitude as well as direction. If the GA predicts a test example correctly, then it receives as a score, the absolute value of the actual return of the stock relative to the market. For an incorrect prediction, the score is the absolute value of the actual return, negated. The score is averaged over all test examples and then over the 100 experiments. The GA achieves an average alpha score of +10.2%. By contrast, random guessing produces an expected average alpha score that is slightly negative. Another naive strategy, choosing the most common direction in the training set, yields a slightly positive score (less than +2%).

A sample rule that the GA generates in one of the experiments is of the following form:

IF [*Earnings Surprise Expectation* > *10%* and *Volatility* > 7% and ...] THEN *Prediction* = *Up*

Such rules can serve as approximate explanations of how the various technical and fundamental input factors relate to future, individual stock returns.

Comparison to Neural Networks

We perform the same set of 100 experiments using a neural network with one layer of hidden nodes. Each experiment involves training the neural network with the backpropagation algorithm, using the same training, stopping, and test sets as in the corresponding GA experiment.

For each experiment, the neural network makes no prediction when the squared correlation on the stopping set is less than 0.5. Note that this decision is made experiment by experiment rather than example by example (as in the GA). The network makes no prediction in only 5 out of 100 experiments (5% of the time). 79.2% of the time, the neural network correctly predicts Stock *X*'s direction relative to the market. 15.8% of the time, it incorrectly predicts direction relative to the market. When it does make a prediction, the neural network is correct 83.4% of the time. The neural network achieves an average alpha score of +9.2%.

In the above experiments, the neural network made many more guesses than did the GA. One way of being more selective with the network is to allow a prediction only when that prediction is more than 0.5 standard deviations away from the mean (across all test examples) of the output variable. This helps in eliminating many near-zero or noisy predictions. Under this new methodology, the neural network makes no prediction 48.6% of the time, a correct prediction 47.4% of the time, and an incorrect prediction 4% of the time. When it does make a prediction, the network is correct 92.2% of the time, achieving an average alpha score of +13.4%.

Discussion and Conclusion

The preliminary results demonstrate that GAs and neural networks are both promising methods for predicting individual stock performance. Their success is due to their ability to learn nonlinear relationships, among the input factors, that result in a stock outperforming or underperforming the market. An advantage of GAs is their ability to output comprehensible rules.

Both methods achieve a high degree of accuracy (83% to 93%) in forecasting the direction of the selected stock relative to the market. Both methods also achieve high average alpha scores (+9% to +13%). The neural network makes more predictions, on the average, except when explicitly restricted. We are currently investigating methods for forcing the GA to make more predictions, without substantial loss of accuracy. With any method, however, there is a tradeoff between number of predictions and overall accuracy.

Although the experiments we conduct are similar for the GA and the neural network, the experiments are not completely standardized. For instance, the GA, like the neural network, could employ a magnitude-based cutoff for deciding when to make a prediction. Likewise, the neural

Copyright © 1995 Software Engineering Press

network could raise its squared correlation threshold to eliminate more cases with poor performance. One area of current research is better standardization of experiments across different AI methods.

Although the neural network and genetic algorithm produce very similar overall results, it is possible that the concepts they learn are qualitatively different. We know that the "rules" the two methods generate *look* different. It is possible that where one method fails, another might succeed, leading to a synergistic, combined approach. This combined approach would be analogous to having multiple human experts independently perform the same task and pool their results afterward. Should the learned concepts be qualitatively the same, the GA could provide explanations of what the neural network is forecasting.

The above comparative results consist of multiple experiments on a single stock. Genetic algorithms and neural networks, however, are widely applicable. LBS Capital Management currently employs over 3000 neural networks (one for each stock) to manage large investment portfolios, totaling over $600 million. We are currently beginning to use GAs for portfolio construction (an optimization task) as well as stock selection (a classification / prediction task). We are also working towards applying the GA prediction method to a broad range of stocks and indices, and towards enhancing the GA's ability to predict magnitude.

Acknowledgments

The authors would like to thank Steve Ward of Ward Systems Group for help with neural network implementations, Roy Stringfellow for graphics, and Kevin Jacobs for miscellaneous programming.

References

Allen, F., & Karjalainen, R. (1993). Using genetic algorithms to find technical trading rules. Unpublished manuscript, Wharton School, University of Pennsylvania.

Barr, D., & Mani, G. (1993). Neural networks in investment management: Multiple uses. *Proceedings of the Second International Conference on AI Applications on Wall Street*, 81-87.

Bauer, R. J. Jr. (1994). *Genetic algorithms and investment strategies*. New York: John Wiley & Sons.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*.

Fishman, M. B., Barr, D. S., & Loick, W. J. (1991). Artificial intelligence and market analysis. *Technical Analysis of Stocks and Commodities, 9*(3), 18-29.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Hall, J. (1994). Using artificial intelligence to track market style. *Presented at the Quantitative Analytic Technology Conference*, New York, NY.

Hutchinson, J., Lo, A. W., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance, 49*(3).

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Mahfoud, S. W. (1992). Crowding and preselection revisited. In R. Manner & B. Manderick (Eds.), *Parallel Problem Solving From Nature, 2* (pp. 27-36). Amsterdam: Elsevier.

Mahfoud, S. W. (1995a). Population sizing for sharing methods. In D. Whitley (Ed.)

Copyright © 1995 Software Engineering Press

Foundations of Genetic Algorithms, 3. San Mateo: Morgan Kaufmann.

Mahfoud, S. W. (1995b). Niching methods for genetic algorithms. (Doctoral dissertation, University of Illinois at Urbana-Champaign). *Dissertation Abstracts International.*

Packard, N. H. (1990). A genetic learning algorithm for the analysis of complex data. *Complex Systems, 4*(5), 543-572.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Volume I: Foundations* (pp. 318-362). Cambridge: MIT Press.

Sikora, R., & Shaw, M. J. (1994). A double-layered learning approach to acquiring rules for classification: Integrating genetic algorithms with similarity-based learning. *ORSA Journal on Computing, 6*(2), 174-187.

White, H. (1994). Using artificial neural networks to develop trading systems for the S&P 500 index futures. *Presented at the 2nd International Workshop on Neural Networks in the Capital Markets*, Pasadena, CA, November 1994.

Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic Algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing, 14*, 347-361.

Copyright © 1995 Software Engineering Press

Two experiments in the stability of stock statistics

Burton Rosenberg
Department of Mathematics
and Computer Science
University of Miami

Abstract

This paper announces support in the form of the Spearman rank correlation test for the hypothesis: stock variance is a stable commodity, but the covariance of stocks varies randomly. Among the consequences of this hypothesis are:

1. *Arbitrage equations involving covariances do not constrain the marketplace.*
2. *Variance is a stable commodity whose price is set by the arbitrage opportunities it presents.*
3. *Portfolio theories depending on estimates of future stock covariances are not at present useful theories.*

The result is not unexpected, however the conclusions challenge some of the existing literature.

1 Introduction

The theory of the efficient portfolio aids the investor to stabilize available capital, as well as provides a justification for risk-return payoff. However, to calculate with the theory, estimates of market variances and covariances are required. The question arises as to how well past variances and covariances predict future variances and covariances. Besides this motivation, whether variances and covariances can be valuable assets depends on whether they are stable. A predictable market behavior might be combined with an arbitrage opportunity thereby pricing the market behavior. If, say, covariances had no predictability, they would also be of no value as an asset.

The theory by which an optimal portfolio is calculated is due to Markowitz [6]. For that theory, variance and covariance information is required. One possibility would be to calculate the historical variances and covariances of a universe of stocks and bring these values forward to the next time step. We decided to question this supposition. Weakening the requirements, we tested only how the rankings of stocks from least to most variant and the rankings of stock pairs from least to most covariant change from time step to time step. In other words, is it true that a high variance stock this year will be a high variance stock next year? Will a high covariance stock pair continue to covary strongly during the next year?

According to the methods of this paper, it is true that the variance of a stock moves with the stock into the next time step. Variance is a property of the stock and in this sense we say it is stable. However, they cannot confirm that covariance is stable, in the following sense: the distribution of the Spearman rank correlation coefficient for the two orderings of stock pairs by covariance during consecutive time periods is essentially the distribution achieved by taking two independent, random orderings.

The problem of prediction of stock price movements has been previously studied from the time-series standpoint. The work of Granger and Morgenstern [4] uses the classical techniques of Fourier Analysis to study the spectral qualities of stock price movements. A very detailed study of stock price variance has been undertaken by Shiller [7] in order to bring into accord observed variance and the

Copyright © 1995 Software Engineering Press

efficient market hypothesis. In addition, there has been much work done using the ARCH model introduced by Engle [2] and the extension GARCH model introduced by Bollerslev [1]. These heteroscedastic models assume that stock price is a gaussian normal random variable with time varying variance, the variance predicted according to the parameters of the model.

This paper attacks the problem from a different angle. We consider it a problem in hypothesis testing, rather than one of model fitting. Furthermore, we use nonparametric methods and thus have no hypotheses on distributions.

2 The experiments

Two experiments are described. The first experiment, summarized in Figures 1 and 2, uses a data set of 212 stocks containing records of at least 280 closing prices since October 30, 1993. The data was taken from MIT's Stock Market Project [8]. We use this data to test quarter and semi-annual data streams running from the third quarter of 1993 until the fourth quarter of 1994. This data is of limited depth in time, but does give us a large population of stocks to work with.

The second experiment, summarized in Figures 3 and 4, uses the CRSP data set on thirteen stocks running from July, 1962 through December, 1992. We corrected this data for splits but not dividend disbursements. This data was used to test a stream of annualized variance and covariances, normalized for means, of prices from 1963 until 1993.

The experiment on the stability of a stock's variance compares two time periods, j_1 and j_2 for a sample R of N stocks, picked from our universe of stock data. In our experiment, j_1 and j_2 are consecutive quarter, semi-annual or annual periods. Sorting by variances during each of the time periods gives us two orderings α and β of the stocks, from least to most variable:

$$\operatorname{Var} {\alpha_1}^{j_1} \leq \operatorname{Var} {\alpha_2}^{j_1} \leq \ldots \leq \operatorname{Var} {\alpha_N}^{j_1}$$

and,

$$\operatorname{Var} {\beta_1}^{j_2} \leq \operatorname{Var} {\beta_2}^{j_2} \leq \ldots \leq \operatorname{Var} {\beta_N}^{j_2},$$

where α_i, β_i are the various stocks in the sample R.

The rank of a stock $r \in R$ under the α order is the i such that $\alpha_i = r$,

$$\operatorname{Rank}_\alpha(r) = \{\, i \,|\, \alpha_i = r \,\}, \quad \text{any } r \in R.$$

Likewise,

$$\operatorname{Rank}_\beta(r) = \{\, i \,|\, \beta_i = r \,\}, \quad \text{any } r \in R.$$

We wish to compare these two rankings in order to reject the possibility that there is no significant influence of the past on the future. Spearman's rank correlation coefficient [3], [5] is the correlation of ranks under the two orders:

$$r_R = 1 - \frac{6 \sum_{r \in R} \big(\operatorname{Rank}_\alpha(r) - \operatorname{Rank}_\beta(r)\big)^2}{(N+1)N(N-1)}.$$

Similarly, consider $S = R^{(2)}$ the collection of all distinct stock pairs, and select a size N subset $R \subset S$. Two orders α and β can be defined for consecutive time periods j_1 and j_2,

$$\operatorname{Cov} {\alpha_1}^{j_1} \leq \operatorname{Cov} {\alpha_2}^{j_1} \leq \ldots \leq \operatorname{Cov} {\alpha_N}^{j_1}$$

and,

$$\operatorname{Cov} {\beta_1}^{j_2} \leq \operatorname{Cov} {\beta_2}^{j_2} \leq \ldots \leq \operatorname{Cov} {\beta_N}^{j_2},$$

and Spearman's coefficient is calculated to compare the two rankings.

For Experiment 1, where N is large, if the two rankings were chosen independently at random, r_R would be approximated as a zero-mean, $1/(N-1)$ variance normally distributed random variable. In Experiment 1, the underlying space of events is the choice of subset R. We calculate,

$$z = r_R \sqrt{N-1}.$$

The event,

$$|\, z \,| \geq 2.575,$$

will occur only 1% of the time if α and β were independently chosen orders.

For small N, such as Experiment 2 where $N = 6$, the Spearman coefficients are compared in a table of theoretically calculated values [5]. The approach here is to consider the set of stocks fixed and the randomized event to be the choice of a pair of years.

Copyright © 1995 Software Engineering Press

In fact, we exhaustively use all consecutive years within the range of our data set.

The first experiment uses the MIT data set and is summarized in Figures 1 and 2. Three subexperiments are cited, each subexperiment had eighteen trials. In Figure 1, the third quarter of 1993 is denoted 93.3, and so on, and the first half of 1994 is denoted 94.1–2, and so on.

For the variance subexperiment, two time periods were selected and forty distinct stocks were picked uniformly at random from the population of 212 stocks. The variance of these stocks were calculated and ranked for the two time periods, and the Spearman correlation coefficient derived. Since N is large, the z value is calculated and shown in Figure 1. This was done for each of eighteen trials, that is, eighteen selections of forty stocks.

The covariance subexperiment was similar, however forty distinct stock pairs were selected rather than forty stocks. The random selection was done by selecting uniformly at random twice from the population of 212 stocks and throwing out the choice if the pair has already been chosen or if the two choices happen to be the same stock.

The subexperiment "Random" consisted of selecting forty pairs of values uniformly at random. That is, if variance or covariance were truly random, it could yield z values as in this subexperiment.

The data shows that the hypothesis of independence is rejected for variance. However, with each time period, the ranking of covariance appears to shuffle almost as unpredictably as Random. This is illustrated in Figure 2, where cumulative probabilities have been totaled and graphed, along side a normal distribution.

The second experiment uses the CRSP data set and is summarized in Figures 3 and 4. Fourteen stocks were selected at random from the CRSP data base, provided that their histories ran from 1962 through 1992. The prices were adjusted for splits, at which time one of the fourteen was rejected because of a long period of missing price information. In one covariance subexperiment, the thirteen data sets were arranged into six pairs, leaving one stock out, and the covariances were rank correlated for years y and $y+1$. The covariances were corrected for stock price by dividing by the mean of each stock for the year, thus yielding a dimensionless quantity. Each y in the range $1962, \ldots, 1991$ was considered a trial, and the cumulative distribution function of the thirty resulting Spearman coefficients is shown in Figure 3, curve *cov2–2*. Additional choices of six pairs were performed and yielded similar results, which are not shown.

Figure 3 also shows the results of two variance subexperiments. The six pairs cited in *cov2–2* were broken into two disjoint sets of six stocks, and Spearman coefficients calculated for rankings of variance divided by mean price squared at time y versus $y+1$, for $y = 1962, \ldots, 1991$. These thirty trials were cumulated to form curves *var–2* and *var–2bis*. Finally, the theoretical null hypothesis curve for $N = 6$ is given as curve *n–6*.

The calculations for this project were done in Perl on a DEC–5000/125 workstation under Ultrix 4.3. Further details of the programs and data sets are included in an extended Technical Report.

3 Conclusions

It appears that stock volatility is stable in time: a high variance stock yesterday will be a high variance stock tomorrow. However, the same is not true for the covariance of two stocks. A strong correlation of two stocks yesterday does not lead to a strong correlation of those stocks tomorrow. To test this idea, we applied nonparametric tests to the ranking of stocks from least to most variant and to the ranking of stock pairs from least to most covariant. For variance, there is this stability. However, for covariance, we cannot distinguish between actual stock data and a purely random shuffle at each period of covariance ranking.

This means that a portfolio adjusted correctly for the previous period, according to the methods of classical portfolio theory, should have no advantage over a neutral portfolio for the next period, since the facts upon which the adjustment is predicated are no more likely to stay put than is a pack of cards to remain unmodified after a thorough shuffling.

Also, this work underlines a subtlety in the theory of portfolio diversification. The stabilizing effect of diversification is not due to deterministic occurrences of negatively correlated industry cycles. Rather, negatively correlated stocks arise haphaz-

Copyright © 1995 Software Engineering Press

ardly, provided that the portfolio is large enough.

Furthermore, since covariance cannot be relied upon to retain its ranking, it cannot be bought and sold. This would lead one to believe, but one cannot conclude, that arbitrage equations involving covariance are unlikely to constrain the marketplace. On the other hand, the stability of variance that is confirmed in this paper concords with current use of variance, for example in the Black-Scholes option pricing formula, as a salable commodity and a source of arbitrage opportunities.

Acknowledgement: The author acknowledges the help of Prof. Thomas Gosnell of the Finance Department. Without his generous offer of assistance, the analysis of the CRSP data sets would not have been accomplished in time for these proceedings.

References

[1] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

[2] Robert F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4):987–1007, 1982.

[3] John E. Freund. *Mathematical Statistics.* Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[4] Clive W. J. Granger and Oskar Morgenstern. *Predictability of Stock Market Prices.* Heath Lexington Books, Lexington, Massachusetts, 1970.

[5] Maurice G. Kendall. *Rank Correlation Methods.* Hafner Publishing Co., 1955.

[6] Harry M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments.* Cowles Foundation Monographs, Yale University Press, 1959.

[7] Robert J. Shiller. *Market Volatility.* The MIT Press, Cambridge, Massachusetts, 1989.

[8] Mark Torrance. Experimental stock market data. http://www.ai.mit.edu/stocks/.

Copyright © 1995 Software Engineering Press

Trial	Time Period	Experiment		
		Variance	Covariance	Random
1	93.3 vs. 93.4	3.42	0.21	1.44
2	93.3 vs. 93.4	4.32	−1.58	−0.55
3	93.3 vs. 93.4	3.63	−0.23	−0.90
4	93.4 vs. 94.1	4.44	0.28	−0.24
5	93.4 vs. 94.1	2.98	1.05	1.48
6	93.4 vs. 94.1	2.53	0.0058	−1.31
7	94.1 vs. 94.2	2.30	−0.76	0.59
8	94.1 vs. 94.2	1.58	1.41	−0.24
9	94.1 vs. 94.2	3.13	0.92	−1.56
10	94.2 vs. 94.3	3.83	2.73	0.58
11	94.2 vs. 94.3	3.74	1.75	0.20
12	94.2 vs. 94.3	2.88	−0.51	1.16
13	94.3 vs. 94.4	4.16	1.29	0.55
14	94.3 vs. 94.4	3.68	0.13	0.90
15	94.3 vs. 94.4	4.32	−0.86	0.58
16	94.1-2 vs. 94.3-4	3.94	0.096	0.060
17	94.1-2 vs. 94.3-4	1.12	−2.50	−0.26
18	94.1-2 vs. 94.3-4	2.03	−0.32	−1.17

Figure 1: Table of experiment one results.

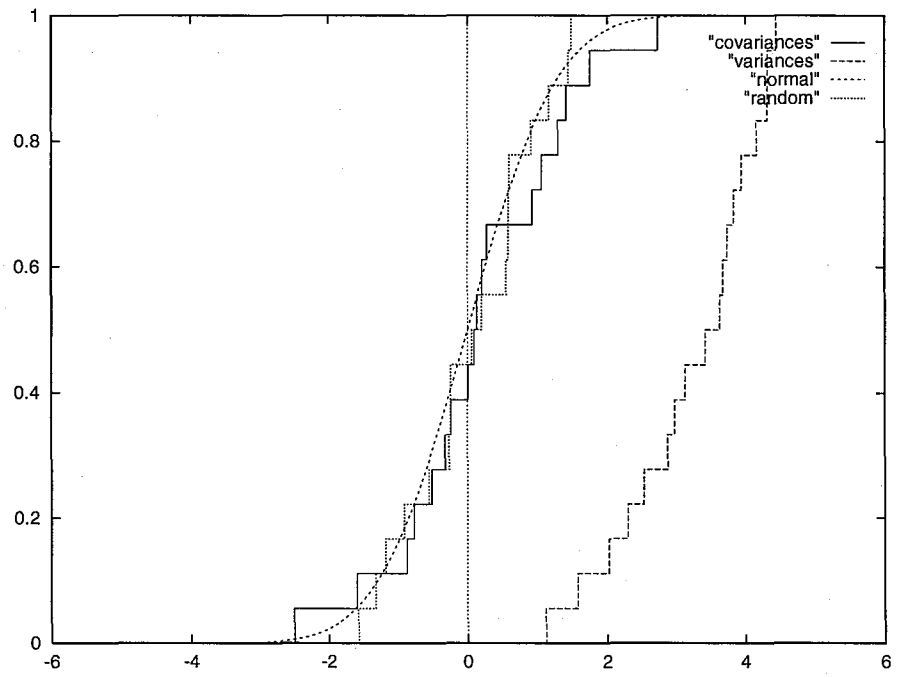


Figure 2: Summary of the first experiment.

Copyright © 1995 Software Engineering Press

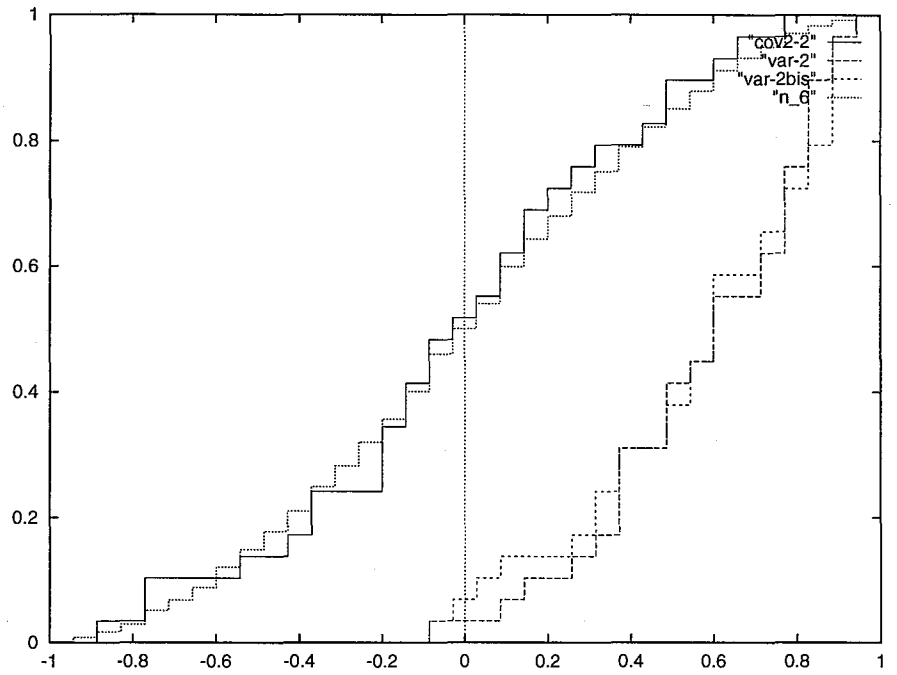


Figure 3: Summary of the second experiment.

Sym	Description	n/g	var-2	var-2bis	cov2-2
ADX	Adams Express Co, NYSE		•		•[1]
AL	Alcan Aluminum Ltd, NYSE		•		•[2]
BHY	Belding Hemingway Inc New , NYSE				
CAN	Continental Can Inc Del, NYSE		•		•[3]
DYA	Dynamics Corp of America, NYSE	*			
FP	Fischer & Porter, AMEX			•	•[1]
GQ	Grumman Corp., NYSE		•		•[4]
IP	International Paper Co, NYSE		•		•[5]
LDR	Landauer Inc, AMEX		•		•[6]
NMK	Niagara Mohawk Pwr Co, NYSE			•	•[4]
PKE	Park Electrochemical Corp, NYSE			•	•[6]
RGS	Rochester Gas & Elec. Corp, NYSE			•	•[3]
SCX	Starrett L. S. 'A', NYSE			•	•[2]
UIS	Unisys Corp, NYSE			•	•[5]

Figure 4: Table of stocks in experiment two.

Copyright © 1995 Software Engineering Press

Paper Session: Improving Neural Network Models

Chair: Gia-Shuh Jang, Springfield (Taiwan)

Neural Network Model Performance: Comparing Results in Photo Finish Situations

Susan Garavaglia
Dun & Bradstreet Information Services, N. A.
Three Sylvan Way
Parsippany, NJ 07054
garavaglia@dbisna.com

Abstract: *When choosing between competing statistical models, neural networks should be compared with more traditional and well-understood statistical methods, such as logistic regression or linear discriminant analysis. Two reasons for this are: 1) there is a greater "installed base" of functioning statistical models using these methods, and, 2) the cost of implementing a neural network model is somewhat higher due to the relatively greater number of connection weights than linear coefficients for the same given application. When the empirical performance of competing models is close enough to be considered a "photo finish," additional analysis should be pursued to uncover indicators of statistical stability and robustness. Herein, examples of this comparative analysis are discussed for four competing business failure prediction models; three are backpropagation neural networks, and the fourth is a logistic regression model. The results show the backpropagation network trained on the confusion matrix criterion to be slightly superior to the logistic regression model.*

1. Introduction and Background

A photo finish in a horse race means that the true winner must be determined by examining a photograph taken at the finish line, because the results are too close to decide by live observation. By analogy, in financial modeling there may be a photo finish between a neural network model and traditional statistical model. Textbook examples, by definition, show clear and unambiguous results of neural network model performance and favorable comparisons to older methods. However, real applications, and especially financial applications, are much more difficult to develop and justify. Some reasons for this are:

1. Financial applications are almost always developed with relatively small samples taken from large populations. Therefore, sampling issues, treatment of outliers, and model stability must be addressed.

2. Financial applications are mostly probabilistic as opposed to deterministic, mainly because of the degree of noise in the data and latent variables. For example, two companies with exactly the same observable financial conditions at one point in time may experience different outcomes later, (e. g., one may file for bankruptcy protection while the other may continue to operate as a solvent business). There are real reasons for one business failing, but the data available to the modeler may not reflect these reasons.

3. Many different measures of stability and performance are available, and the preferences of individual econometricians may lead them to make different decisions regarding the selection of a "best" model. In addition, some measures of performance are purely pragmatic, and more technical measures of goodness of fit may be given a lesser weight in the decision relative to, say, gains chart statistics. Gains charts are used to estimate the relative gains in predictive power of models vs. other predictive methods or random selection.

This case study strives to illustrate all the aforementioned characteristics of financial applications and to demonstrate how different measures of performance may lead to the

Copyright © 1995 Software Engineering Press

selection of different models and modeling techniques. In addition, the contest between models is made more intriguing by the very close competition and the nature of the application itself. The application is prediction of business failures in the health care industry, an industry in which the failure rates are among the lowest for any industry group. The models to be compared are a logistic regression model, and three different neural network backpropagation models.

As background, Dun & Bradstreet has developed 13 industry specific models that combine balance sheet data elements, company "demographics" and company-specific payment performance information to predict business failure over an 18 month performance window. The models were developed using logistic regression and provide a relative rank ordering of risk by a universal score, universal percentiles, and industry specific percentiles. Thus, a health care firm can be compared to all of the scored firms in the D&B database or its industry peers. Although the exact specifications of the logistic regression model are D&B-proprietary, the set of predictors used for all models discussed in this article include:

1. A continuous variable representing payment performance at time of observation
2. A continuous variable representing the age of the firm
3. A continuous variable representing payment variability over the last two years
4. An indicator for a set level of Current Liabilities/Net Worth Ratio
5. An Indicator for Derogatory Payment Notes in File
6. An indicator for a set level of Cash/Total Assets Ratio
7. An indicator for a set level of Sales/Working Capital Ratio
8. An indicator for Derogatory Public Record Items (Open Suits, Liens, or Judgments)

In addition, the neural network models incorporate an additional variable, an indicator for specific 4-digit SIC (Standard Industry Classification) codes within the overall health care industry group of SICs, 8011 to 8099. These higher risk SICs included nursing homes and other nursing care services, psychiatric hospitals, and medical laboratories.

Generally, our experience in trying to improve the predictiveness of financial stress models through the use of neural networks has yielded the kind of photo finish results that are detailed herein. We believe that this is because (1) our method of data collection and representation tends to favor relationships that are either highly monotonic or linear, and (2) because the use of probability by logistic regression compensates for its not having the additional estimators that are found in a backpropagation hidden layer. For additional background on bankruptcy prediction techniques and results, see Altman [1].

In a photo finish situation there could be trade-offs in selecting the neural network model versus the logistic regression model for implementation. The logistic regression model has only as many coefficients as variables, plus an intercept term, and is relatively simpler to implement from a programming standpoint. However, when expected performance is about equal and there is evidence of model stability, the model user has nothing to lose by implementing the neural network model and valuable experience to gain. What is learned by using a neural network model may be transferred to other neural network opportunities.

2. Development and Testing Methodology

The data used for all models comes from D&B's database of health care industry companies that have supplied full fiscal year end financial statements within two years prior to the observation period. In addition, for the selected companies, Total Assets, Total Liabilities and Total Current Liabilities were each required to be positive and the companies' other D&B data had to be complete for full

Copyright © 1995 Software Engineering Press

reporting purposes. The final set of data elements to be included in the models was determined by performing an exploratory data analysis of the relationship between individual data elements and their relationship to the outcome of business failure or continued business solvency. A brief discussion of the appropriate data analysis can be found in Hosmer and Lemeshow [3].

Once a set of predictors was selected, three data sets were created: a development (training) data set, a holdout (testing) data set, and later on, a validation data set to be used when more recent business failure data became available. Table 1 provides a summary of the data sets.

Data Statistics (Unweighted)

	Total	Non-Failures	Failures
Development/Training	2316	2210	106
Holdout/Testing	772	742	30
Validation	2970	2908	62

Table 1

For the original sample, a "snapshot" of the companies' predictive data was taken as of mid-1991. This is possible because D&B regularly archives its databases. For each company, a subsequent "snapshot" of the outcome, either failed or not failed, was taken over a number of calendar quarters up to the end of 1992. Thus, the observation period was at one point in time and the performance period spanned the following 18 months. The validation data set used June and December of 1992 as the observation periods and the entire year of 1993 as the performance period. This still fixed the performance period as 18 months, but allowed the computation of an annual failure rate. In 1992, the business failure rate for the health care segment that met the data criteria was 0.8%. In 1993, it fell to 0.2%.

2.1 The Logistic Model

The challenge from a modeling standpoint is obvious: as the failure rate is very low, there are very few "bads" and model stability is of great concern. Some techniques to work around this problem include the bootstrap and jackknife methods discussed in Amemiya [2]. Other approaches include limiting the number of variables while using as many dichotomous independent variables as possible, or validating the model based on quintiles or deciles rather than percentiles. The approach used for these models was to transform the data to remove as much noise as possible. Transformations included the use of categorical variables and natural logarithms of continuous variables. As a later test of the model with the validation data set showed highly favorable results with regard to performance and stability, it was decided that there would not be a significant benefit from the effort of bootstrapping or jackknifing.

Because the logistic regression method maximizes a likelihood function, probability "matters." Therefore, the modeling data for a logistic regression should reflect the expected proportions of failures and non-failures in the population. All of the failures and a sample of the non-failures were used, but the non-failures were "weighted up" to their expected proportions in the population that would be scored by the resulting model. The reason for using all the failures is simply because there are relatively few of them.

The performance criteria used for the logistic regression model and the neural network models were:

1. The percentage of failures captured in the first percentile.
2. The percentage of failures captured in the first five percentiles
3. The percentage of failures captured in the first decile
4. The D&B Predictiveness Index (PI), which is a geometrical measure of the ratio of the area of the ROC (receiver operating curve) to a hypothetical ROC for a model that screens perfectly, (i. e., every failure scored lower than the lowest scoring non-failure). See Hutton [4] for more

Copyright © 1995 Software Engineering Press

information on the ROC as it relates to neural networks.

5. The Kolmogorov-Smirnov (K-S) statistic, which is a point measure of how well the model "separates" the classes. See Mood, *et al*, [7] for a complete explanation.
6. The Kullback-Liebler (K-L) statistic, which is a more comprehensive measure of class "separation."[1]
7. The Multivariate Wald statistic for the neural network models and the univariate Wald statistic for the logistic regression model (See Wald [9], Hosmer and Lemeshow [3], and Kuan and White [6]).

It should be noted that all of these measures are at least somewhat dependent on the proportion of observations in each class (reflected here by the failure rate), and cannot reasonably be used to compare two different model applications. Thus, absolute measures of "goodness" for each of these criteria are not relevant. For criteria 1-6, the highest value for each statistic determines the "best" model, and in criterion 7, the Wald test, the number of parameters for which the Wald statistic exceeded the critical Chi-squared value is used to determine the "best" model.

2.2 The Neural Network Models

A number of variations on the size and number of hidden layers were tried before a network with a hidden layer of seven units was settled on for the rest of the development work. Using this network architecture, three different versions were trained, varying the optimization criteria, and using the validation data set to test. The rationale for using the validation data set, as opposed to the holdout/testing data set, was that the validation data set had more failures and contained more recent data. Basing performance on more recent data also gives the model a longer "shelf life," in that it is expected keep its level of predictiveness longer. The lowest mean squared error, the highest average correct classification rate, and highest correlation coefficient based on a confusion matrix were used as optimization criteria. The classification rate measures the average percentage of neural network output values that fall in the right place on the 45° diagonal (after any necessary rounding), where the axes are actual value and desired value. The confusion matrix method of optimization uses the correlation coefficient of the points scattered on and around the same diagonal. Information about the neural network software used, NeuralWorks Professional/II®[2], and how to train a network based on best model criteria, can be found in NeuralWare, Inc. [8].

The failures in the training set were duplicated enough times to equal the number of non-failures, which brought the total number of observations to 4,330. This strategy improved the performance by a slight amount, apparently because the neural network had more opportunities to learn from the failures, which enabled it to compensate for its lack of knowledge of the probabilities used by logistic regression.

After this first "round" of selecting an overall architecture, the second round involved selecting the best network from the optimizing criteria. The confusion matrix criteria performed best among the three, but still was not quite as good as the logistic regression model in some criteria. From this point, further improvements on the confusion matrix version of the model were attempted by additional training and jogging the weights periodically to see if the network was stuck in local minima.

After some amount of training, which included several retreats back to a previously best network, it was decided that it was

[1] White [10] illustrates the use of the K-L statistic in terms of comparing an estimated conditional density to a true conditional density, which would make the best model have the lowest K-L value. In this case, the K-L statistic is being used to compare the estimated conditional densities of the two classes which should be as far apart as possible, thus giving a relatively large K-L value for the best model.

[2] NeuralWorks Professional/II is a trademark of NeuralWare, Inc.

Copyright © 1995 Software Engineering Press

unlikely that any additional improvements could be made within the same architecture and optimization parameters.

The full comparison of all the models is in Tables 2a through 2e, with the fourth column showing how many failures ("Val. Bads") were screened by each model at the first, fifth, and tenth percentiles.

Logistic Regression

Criteria	Dev	Hld	Val	Val. Bads
1%	16	13.3	30.6	19
5%	45.3	33.3	51.6	32
10%	62.3	43.3	66.1	41
PI	73.4	53.48	75.5	
K-S	0.61	0.45	0.5	
K-L	1.2	0.88	1.31	

Table 2a

Neural Network - Confusion Matrix Criterion (Typical Result)

Criteria	Dev	Hld	Val	Val. Bads
1%	9.4	12.4	17.7	11
5%	39.6	30	50	31
10%	56.6	40	61.3	38
PI	75.6	51.56	70.8	
K-S	0.62	0.47	0.53	
K-L	1.26	0.92	1.33	

Table 2b

Neural Network - Classification Rate

Criteria	Dev	Hld	Val	Val. Bads
1%	10.4	6.7	14.5	9
5%	41.5	30	46.8	29
10%	67	40	56.5	35
PI	81.44	49.56	66.4	
K-S	0.67	0.48	0.52	
K-L	1.52	0.92	1.17	

Table 2c

Neural Network - Root Mean Squared Error

Criteria	Dev	Hld	Val	Val. Bads
1%	6.6	6.7	7.6	5
5%	35.7	20	37.1	23
10%	65.1	40	59.7	37
PI	82.48	51.96	72.44	
K-S	0.69	0.43	0.55	
K-L	1.49	0.7	1.12	

Table 2d

Neural Network - Best Confusion Matrix

Criteria	Dev	Hld	Val	Val. Bads
1%	10.4	6.7	10.8	6.7
5%	43.2	26.7	51.6	32
10%	66	40	69.4	43
PI	82.24	40.96	73.12	
K-S	0.7	0.31	0.61	
K-L	1.56	0.72	1.42	

Table 2e

An overall comparison of the logistic regression model with the neural network models is in Table 2f.

Best Model Selection by Criteria

Criteria	Holdout	Validation
1%	Logistic	Logistic
5%	Logistic	Logistic/Best CM Tied
10%	Logistic	Best CM
PI	Logistic	Logistic
K-S	Classification Rate	Best CM
K-L	First Confusion Matrix/ Tied with Class. Rate	Best CM

Table 2f

The Best Confusion Matrix version gave a better result than the logistic regression model in three categories and was tied in one. The logistic regression model was superior to the neural network in the first percentile screening rate and the Predictiveness Index.

3. The Wald Test as a Potential Tie Breaker

The Wald Test is a statistical hypothesis test to determine the significance of predictors. The methodology is to propose the Null Hypothesis, (i. e., that the "true" coefficient or connection weight associated with the predictor is zero, therefore rendering the predictor irrelevant to the outcome), and then determine the likelihood that the Null Hypothesis is false and can be rejected. This is essentially accomplished by taking the estimated coefficients or connection weights

Copyright © 1995 Software Engineering Press

from the regression or the neural network models and calculating their relative contribution to a correct result versus the contribution of random "noise" to the correct result. It is related in application to the *t-test* and the *F-test*, but is defined for models that are more complex than linear models. Johnston [5] includes an extensive discussion of tests of significance for linear models; and Amemiya [2] describes several tests of significance for more complex models, including the multivariate Wald Test.

Kuan and White [6] briefly discuss the use of the Wald Test to determine the significance of predictors in a neural network model with one hidden layer and provide the appropriate form of the test. The Chi-squared critical value is determined by the number of hidden units plus one, which should be used in place of the degrees of freedom in looking up the critical value in a table. Most statistics textbooks contain a table of critical values by degrees of freedom (See either Johnston [5] or Mood, et al, [7].). For both the multivariate and the univariate tests, critical values are at the 5% significance level, or a probability of 0.05 that a true Null Hypotheses is being rejected. For one degree of freedom, the critical value is 3.841, and for 8 degrees of freedom (defined for this type of test as 7 hidden units plus one) the critical value is 15.507.

The multivariate form of the Wald Statistic is computationally intensive and is not often available in standard software. For this research the calculations were programmed using Mathematica®[3] (Wolfram [11]) and took several days to run on a standard 386-based PC. The univariate form of the Wald Statistic is supplied as part of the standard output for the SAS®[4] Logistic Regression Procedure. For more information see [9].

As examining the tests of significance is part of the traditional model development process, all eight predictors in the final logistic regression model had Wald Statistics that exceeded the critical value of 3.841, which is consistent with the *p-values* (significance levels) all being below 0.05. The *p-values* are the probabilities that a true Null Hypothesis is being rejected. The Wald Statistics on the neural network models were calculated, after the fact, on models trained to meet other criteria, viz., root mean squared, classification rate, and confusion matrix measures. Therefore, these statistics served not so much to determine if the predictors were relevant, as this was a known result of the logistic regression, but to determine how well the network extracted predictiveness from the data. In addition, since one more input was used (the SIC indicator within the general health care group), it could demonstrated that the neural network was able to make use of other information that was not statistically significant in a purely linear estimation. Therefore, the total number of test statistics that exceeded the critical value is a relative measure among the three neural networks. As the multivariate Wald Statistic determines the significance of inputs or predictors, it is appropriate for the input to hidden layer connections only. Therefore, the percentage of the total is based on: 9 inputs times 7 hidden units = 63 connections.

The results were as follows in Table 3:

[3] Mathematica is a trademark of Wolfram Research, Inc.

[4] The trademarks refer to the products and services of SAS Institute, Inc.

Copyright © 1995 Software Engineering Press

Criterion/Dataset	Wald Statistics > Critical Value	
	Number	Percentage
Classification Rate		
Training/Dev	55	87.30%
Test/Holdout	47	74.60%
Validation	55	87.30%
RMS Error		
Training/Dev	21	33.33%
Test/Holdout	7	11.11%
Validation	22	34.92%
Confusion Matrix		
Training/Dev	36	57.14%
Test/Holdout	18	28.57%
Validation	36	57.14%

Table 3

These initial results, which show that the classification matrix criterion method has the highest proportion of significant connections, are consistent with expectations, given that the average classification rate was 84.29%, based on the failures classification rate of 83.02% and the non-failures classification rate of 85.57%. The confusion matrix criterion method, which uses the correlation coefficient between the outputs and the correct classification, had the next highest proportion of significant connections, with a correlation coefficient of 0.271602 for the best model using that criterion. The lowest root mean squared error obtained was 0.606613, which is consistent with the expectation of the worst relative performance. It is proposed that the confusion matrix method yielded the best rank ordering power, as shown in the percentile and PI statistics, because the rounding used in the classification matrix criterion caused a number of observations to be counted in the wrong class.

The exercise of calculating the multivariate Wald Statistics is also useful to examine the persistence of certain predictors across all models. Table 4 shows the sum of statistics that exceeded the critical values for all neural network models and all data sets (training, testing, and validation) for all hidden units. Payment performance has the highest number of statistically significant connection weights, and is also the strongest predictor in the logistic regression model. This should be expected as well, because, intuitively, the first sign of a business in trouble is usually delinquency in bill-paying. What is interesting about this result is that the special SIC category, (tied with the Current Liabilities/Net Worth ratio), has the next highest number of statistically significant connection weights, but was not statistically significant in the logistic regression model! This supports the selection of a neural network model when the data is very complex.

Predictor	Total Number>Critical Value
Payment Performance	52
Age of Company	23
Payment Variability	14
Current Liabilities/Net Worth	39
Derogatory Payment Notes	31
Cash/Total Assets	30
Sales/Working Capital	34
Derogatory Public Record Items	35
Special SIC Category	39

Table 4

4. Economics of Model Implementation

A pragmatic observer of this discussion might challenge the results in the tables by saying, "Look, the Confusion Matrix Neural Network screens only two more failures than the Logistic Regression. Who cares which model gets implemented?" But, for users of Financial Stress Predictive Scores, any single business failure could mean very large losses, and the difference between screening 43 versus 41 failures at the first decile represents a 5% improvement. Of course, a rational economic decision on which model to use would be based on the expected incremental cost to develop, implement, and maintain the neural network model versus the

Copyright © 1995 Software Engineering Press

expected loss avoidance based on the expected improvement in performance. There may be intangibles as well, such as the model user's reputation and regulatory implications of imprudent risk taking.

Another consideration is how the model will be used. For example, the logistic regression model screened the best of all the alternatives at the first percentile level. A company that seeks to minimize its losses and still approve 99% of its risks would do best with this model. However, a company that is seeking the lowest risk with a 90% approval rate would do better with the Best Confusion Matrix Neural Network Model.

From this set of research results, it appears that the logistic regression model may have slightly better rank ordering power than the best of the neural network models, as indicated by the Predictiveness Index and the screening power in the lowest percentiles. However, the neural network models were able to get predictive information out of an additional input and showed a good proportion of statistically significant connection weights. This gives potential users the confidence that the network has true predictive power and model stability.

5. Summary and Conclusions

Selecting a best version of a statistical model is not always straight-forward. Measures of expected performance and statistical stability must be evaluated in some economic context of how the model will be applied and the relative costs of all of the alternatives under consideration. The nature of the data also determines which modeling paradigm will yield the best result. The results obtained by this study appear to suggest that there may be some upper limit of predictiveness to just about any combination of model and data set because of unobservable factors that influence the outcomes. The neural network paradigm may be able, through the use of hidden units that have the effect of combining subsets of inputs, to provide information that is not immediately observable in strictly linear analysis. However, the presence of this type of information can be detected through data analysis, which often includes the development of a linear model as an initial step.

Neural Network models remain a promising technology, but, as part of the model development process, a significant amount of time should be budgeted for exploratory data analysis, and pre-implementation analysis such as statistical hypothesis testing, as described in this article. If standard programs and procedures are set up for the analytics, the process will run smoothly, and each implemented model will be better understood and accepted by its users.

6. References

[1] Altman, Edward I. 1993. *Corporate Financial Distress and Bankruptcy*. Second Edition. New York, NY: John Wiley & Sons.

[2] Amemiya, T., 1985. *Advanced Econometrics*. Cambridge, MA: Harvard U. Press.

[3] Hosmer, David W. And Stanley Lemeshow. 1989. *Applied Logistic Regression*. New York, NY: John Wiley & Sons.

[4] Hutton, Larrie V. 1992. Using Statistics to Assess the Performance of Neural Network Classifiers. Johns Hopkins *APL Technical Digest*. V. 13. N. 2.

[5] Johnston, J. 1984. *Econometric Methods*. Third Edition. New York, NY: McGraw-Hill.

[6] Kuan, Chung-Ming, and Halbert White. 1991. Artificial Neural Networks: An Econometric Perspective. Working Paper. U. Of California, San Diego.

[7] Mood, Alexander M., Franklin A. Graybill, and Duane C. Boes. 1974. *Introduction to the*

Copyright © 1995 Software Engineering Press

Theory of Statistics. Third Edition. New York, NY: McGraw-Hill.

[8] NeuralWare, Inc. 1993. *Reference Guide: Software Reference for Professional II/Plus ® and NeuralWorks Explorer®*. Pittsburgh, PA: NeuralWare, Inc.

[9] SAS Institute, Inc. 1990. SAS® Technical Report P-200, *SAS/STAT® Software: CALIS and LOGISTIC Procedures, Release 6.04*. Cary, NC: The SAS Institute, Inc.

[9] Wald, Abraham. 1943. Tests of Statistical Hypotheses Concerning Several Parameters when the Number of Observations is Large. *Transactions of the American Mathematical Society*. V. 54, No. 3. pp. 426-482.

[10] White, Halbert, Jr. 1990. Learning in Artificial Neural Networks: A Statistical Perspective. *Neural Computation*, Vol. 1, pp. 425-464. Also in White, Halbert, Jr. 1992. *Artificial Neural Networks: Approximation and Learning Theory*. Oxford: Blackwell.

[11] Wolfram, Stephen. 1991. *Mathematica: A System for Doing Mathematics by Computer*. Second Edition. Redwood City, CA: Addison-Wesley.

The author wishes to acknowledge the help of Jim Markovitch of D&B in editing the final version of this article.

Copyright © 1995 Software Engineering Press

Financial Classification: Performance of Neural Networks in Leptokurtotic Distributions.

Ravi Krovi
Dept. of Accounting & Information Systems
Southern Arkansas University
Magnolia, AR 71753
rakrovi@saumag.edu

Balaji Rajagopalan
Dept. of Management Information Systems
University of Memphis
Memphis, TN 38152
rajagopa@msuvx1.memphis.edu

Akhilesh Chandra
Dept. of Accounting
North Carolina A&T State University
Greensboro, NC 27411
chandraa@ncat.athena.edu

Ned Kumar
Dept. of Management Information Systems
University of Memphis
Memphis, TN 38152
kumarns@msuvx1.memphis.edu

Abstract

This paper presents the results of a performance analysis of two popular techniques of classification: neural networks and FLDA, a statistical approach. It is suggested that neural networks which are not limited by assumptions such as normality and equal variances, would perform better especially for financial applications. Several financial applications have been documented to assume a form of non-normality called leptokurtosis. Our results indicate the conditions under which the neural network might outperform other approaches for financial applications. Comparisons based on real world data as well as simulated data provide strong evidence that the neural network performs better when ranked data is used.

1. Introduction

The classification problem involves assigning data cases based on a set of variables to two or more groups. Classification is a very common problem encountered in the business world and is of considerable interest in the financial community. For example, investors are interested in classifying a firm on the basis of its financial soundness (bankruptcy); Stock analysts are interested in categorizing investments in firms as involving high, medium or low risk based on their financial ratios; the FDIC has an interest in identifying banks or financial institutions that are likely to go bankrupt; credit rating agencies are interested in classifying customers as belonging to high/low risk categories.

Traditional statistical procedures such as the Fisher's Linear Discriminant Analysis (FLDA) have been widely used to find solutions to the classification problem. While such techniques have worked well for some problems, the performance has been unimpressive for most business applications. This is because statistical approaches to the classification problem require some assumptions about the data, the most important of them being multivariate normality and the homogeneity of covariance. However, prior research has consistently shown that financial data in particular violate the above mentioned assumptions. For example, it has been shown that the distribution of changes in daily future prices is not normal but is actually leptokurtic; i.e. a distribution with fat tails (Hudson et al., 1987). An analysis of the empirical distributions of asset and commodity prices revealed similar forms of non-normality (Peters, 1991). Leptokurtotic distributions also seem to be prevalent in other financial data such as stock returns (Brock et al., 1991) and exchange rate changes (Hsieh, 1988).

Recently , neural network approaches have been

Copyright © 1995 Software Engineering Press

used effectively to solve classification problems (Salchenberger et al., 1992; Marquez et al., 1992). However, there has been no study which addresses the issue of why or under what conditions one approach outperforms the other. As a result, no strong generalizations can be made as regards the superiority of any particular approach. In this paper, we present the results of a comparison between the traditional statistical approach (FLDA) and neural networks specifically for leptokurtotic distributions. In our comparisons, we vary two conditions: the assumption of homogeneity of covariances, and the effect of using ranked data versus raw data. Results of a comparison between the two techniques for the liquidation / merger alternative are also presented and discussed vis a vis the results from the theoretical distribution.

2. The Statistical Model (FLDA)

Fisher's linear discriminant analysis (FLDA) is the most frequently used classification rule. The rule works well in situations where the groups to be discriminated can be separated by a straight line. Consider the simplest case where two groups have to be differentiated. Let G_1 and G_2 denote the groups. Further, it is assumed that the two group populations are n-variate (n >= 1) normal and the homogeneity of variance-covariance is valid.

In such a case, an unclassified data case is assigned to group G_1 if
$X_0^T S^{-1}(X_1-X_2) >= 1/2(X_1-X_2)^T S^{-1}(X_1-X_2) + \ln(p_2/p_1)$
and to group G_2 otherwise.

3. Neural Network Classifiers

Neural networks are composed of highly interconnected neurons or processing elements organized in layers. The simplest form of such a network is one that has two layers: input and output. Feedforward networks are characterized by unidirectional flow of signals from the input to the output layer. Connections between the neurons have a numerical weight associated with them that explains the influence of input units on the output units. These weights are learned by the network through training that consists of repeated presentation of examples from a training set. In the backpropogation algorithm, there is a middle layer that transforms and develops internal representations of the inputs. The transfer function that describes the relationship between layers is usually like a logistic continuous function. The ability of such multi-layered networks to represent nonlinear functions is well documented (Kolmogorov, 1963). The present study uses multilayered networks with a variant of the backpropogation learning procedure.

4. Methodology

Using a procedure proposed by Fleishman (1978), a leptokurtotic distribution with a skewness of 0.25 and a kurtosis level of 3.0 was simulated. The number of variables was five. There were two levels of manipulation: homogeneity and heterogeneity of covariance structures. The design is summarized below:

	Equal Covariances	Unequal Covariances
Neural Network	U_1=(0,0,0,0,0) U_2=(1,1,1,1,1) Cov_1=Cov_2=I	U_1=(0,0,0,0,0) U_2=(1,1,1,1,1) Cov_1=I Cov_2=2I
FLDA	U_1=(0,0,0,0,0) U_2=(1,1,1,1,1) Cov_1=Cov_2=I	U_1=(0,0,0,0,0) U_2=(1,1,1,1,1) Cov_1=I Cov_2=2I

Note: U1 andU2 represent the means of 5 variablesand I represents the identity matrix.

Two hundred observations were generated for each of the four conditions with hundred cases per group. Within each cell, hundred observations were used as the training set and the remaining hundred were used as a test or holdout sample. Five simulations were carried out per cell and the results averaged. Using the same distributions, the procedures of NN and FLDA were repeated for ranked data instead of raw data.

5. Network Design

A single hidden layer, feedforward backpropogation network with five input nodes,

Copyright © 1995 Software Engineering Press

four hidden nodes (based on a 75% rule suggested by Salchenbarger et al., 1992), and one output node was developed. The input nodes represent the five discriminating variables and the output node is the classification or group node. The delta rule was used to train the network. A convergence criterion of root mean square error of 0.01 was used for training and a sigmoidal function was used to update weights.
The function used to classify the continuous output was as follows:

f(0) = 1 if f(0) >= 0.5
0 otherwise.

6. Results

The classification rates of the experimental comparisons between the neural network and FLDA for leptokurtotic distributions are presented below.

RAW DATA

	Equal Covariances	Unequal Covariances
NN	80.5%	71.5%
FLDA	80.0%	69.0%

RANKED DATA

	Equal Covariances	Unequal Covariances
NN	94.5%	85.5%
FLDA	89.6%	64.4%

7. A Real World Example

With increasing bankruptcies, security analysts, investors and corporations are paying closer attention to the survival prospects of a firm. There is a need to identify firms that might be strong candidates for being acquired. The variables chosen for this merger-liquidation model were based on failing firm theories well document in prior studies (Palepu, 1986; Eddey, 1991). They were: Return on assets (ROA), Return on stockholder's equity, Price-earnings ratio, Dividend yield, Turnover ratios, Liquidity ratios, Sensitivity to economic conditions, Firm size, and Growth resource mismatch. Data for these variables was collected for one, two, and three years before the actual event with a training sample of 60 and a holdout sample of 60 firms. The results of this comparison are presented below:

	Time=1	Time=2	Time=3
FLDA	67.53%	66.43%	64.24%
NN	73.00%	70.60%	70.15%

8. Discussion

The results of the merger-liquidation classification comparisons confirm some intuitive expectations about the techniques. As the actual event approaches, there is an improvement in the predictive ability. However, the neural network model performs only marginally better than the FLDA. The same conclusion can be reached when looking at the classification rates for raw data generated in the form of a leptokurtotic distribution. This is also irrespective of the violation of homogeneity of covariances. The performance of both approaches is by and large mediocre especially in the case of unequal variances. However, after using ranked data for our comparisons, we can make three generalizations. First, for leptokurtotic distributions such as asset and commodity prices, neural networks are a more accurate classification tool than FLDA. Second, there is a drastic improvement in the neural network's performance when ranked data is used. Third, the disparity between the neural network and the FLDA for ranked data is most obvious under the condition of unequal covariances.

Our results provide strong support for the superiority of the neural network approach in financial applications, but they also show the conditions under which performance can be

Copyright © 1995 Software Engineering Press

improved. That is, it seems that the neural network performs better when the data is ranked and is not affected by the violation of the assumption of unequal variances. We are currently in the process of conducting additional simulations to determine if such improvements are consistent across other distributions (purely guassian, platokurtotic etc.). We are also in the process of designing and testing a genetic algorithm based approach to classification.

9. References

Brock, W. A., Hsieh, D. A., and LeBaron, B. *Nonlinear Dynamics, Chaos, and Instability: Statistical Theory and Economic Evidence,* Cambridge, MA: The MIT Press, 1991.

Eddey, P. H. "Corporate Raiders and Takeover Targets," **Journal of Business Finance and Accounting**, 18(2), January 1991.

Fleishman, A. I. "A Method for Simulating Non-Normal Distributions," **Psychometrika**, Vol. 43, 4, December 1978.

Hsieh, D. A. "The Statistical properties of Daily Foreign Exchange Rates: 1974-1983," **Journal of International Economics**, Vol. 24, 129-145.

Hudson, M. A., Leuthold, R. M., and Sarassoro, G. F. "Commodity Futures Price Changes: Recent Evidence for Wheat, Soybean, and Live Cattle," **The Journal of Futures Markets**, Vol. 7, 287-301.

Kolmogorov, A. N. *On the Representation of Continuous Function of Many Variables by Superposition of Continuous Functions of One Variable and Addition.* American Mathematical Society Translation, 1963.

Marquez, L., Hill, T., O'Connor, M., and Remus, W. "Neural Network Models for Forecast: A Review," **The Twenty-fifth Hawaiian International Conference on Systems Sciences**, Hawaii, Jan 8-10, 1992.

Palepu, K. G. "Predicting Takeover Targets: A Methodological and Empirical Analysis," **Journal of Accounting and Economics,** Vol. 8, 1986.

Peters, E. E. *Chaos and Order in the Capital Markets: A New View of Cycles, Process, and Market Volatility*, New York: John Wiley, 1991.

Salchenberger, L. M., Cinar, M. E., and Lash, N. A. "Neural Networks: A New Tool for Predicting Thrift Failures," **Decision Sciences**, Vol. 23, 899-916.

Ravindra Krovi is an Assistant Professor of Information Systems at Southern Arkansas University. He has published in Journal of Information and Management and presented papers at conferences such as HICSS, IEEE, and DSI. His research interests are in the areas of adaptive systems design and artificial intelligence. He is currently working on a distributed agents model of negotiations.

Balaji Rajagopalan is a doctoral candidate at the University of Memphis. He has a paper accepted for publication in the Journal of Information Processing and Management and has presented papers at conferences such as IRMA and SWDSI. He has also been invited to present at the DSI National Conference. He is currently working on a chaos theory model of information technology diffusion.

Ned Kumar is a doctoral candidate at the University of Memphis. He has published in Information Processing Management and Decision Support Systems and conferences such as Decision Sciences Institute and Information Resources Management Association. He is working in the area of global information systems and outsourcing.

Akhilesh Chandra is an Assistant Professor of Accounting at North Carolina A&T State University. He has published in the Journal of Computer Information Systems and Managerial Finance and conferences such as AAA, DSI, and IRMA. His research interests are in the areas of managerial accounting and decision making.

Copyright © 1995 Software Engineering Press

Training Robust Neural Nets by Minimizing Weights not Errors

Patrick J. Lyons, Ph.D. and Santanu Kar
Department of Management
St. John's University
Jamaica, New York 11439, USA

Abstract

A frequent obstacle to applying neural networks to business is overtraining. The traditional model to train a neural network minimizes the sum of the squares of the deviations of the target and computed output. This paper presents a second model in which acceptable deviations of the target and computed output are specified as constraints and the objective function is to minimize the sum of the squared weights. The resulting weights provide a robust solution. As examples, the experience of using the model in training the XOR and 3-Parity neural networks is presented, along with a copper price forecasting model.

1. Introduction

A frequent obstacle to applying neural networks to business is overtraining. When overtraining occurs, the resulting neural network memorizes the training data very well, but does not evaluate new data satisfactorily. This usually happens when several weights become large in absolute value and relatively small changes in input create large changes in output. The objective of this paper is to demonstrate that by minimizing the weights and not the errors a more robust neural network will be produced. To do this, two models will be introduced and their resulting solutions will be compared.

The first model is based on the traditional model of adjusting the weights of a neural network so as to minimize the sum of the squares of the deviations of the target and computed output. This will be called the Error Minimization Model. Several different algorithms can be used with this model to determine the solution, such as standard error back propagation with fixed learning rate, error back propagation with line minimization (also called steepest descent), quasi-Newton, and conjugate gradient. In the second model, acceptable deviations of the target and computed output are explicitly specified as constraints and the objective function is to minimize the sum of the squared weights. This model is called the Weight Minimization Model. These two models are then applied to the XOR and 3-Parity Problems. The resulting solutions of the Weight Minimization Model are more robust than the Error Minimization Model. The Weight Minimization Model is also applied to a copper price forecasting model to demonstrate the use of the technique on a larger practical problem.

1.1 Review of the Literature

An important factor in applying neural networks to business is the proper training of the network. This is especially true for applications that synergistically use expert systems, neural networks, Lotus compatible worksheets, and/or dBase compatible files. For example, Lyons [3] describes a methodology to integrate neural networks and expert systems for merger & acquisition analysis. The book and software of Lyons [5] contains the PC software which demos the system described in Lyons [3]. Lyons [4] describes how AI technology can be used technology transfer. References such as Freeman [1], Kosko [2], Maren [6] and McClelland [7] discuss the art of training a neural network with the Error Minimization Model and various versions of the back propagation algorithm. In the recent article, Van der Smagt [9] analyzes these

Copyright © 1995 Software Engineering Press

algorithms along with quasi-Newton and conjugate gradient algorithms for the Error Minimization Model. This current paper extends that analysis by applying these algorithms to the Weight Minimization Model.

1.2 Overview

In Section 2 of this paper, the Error Minimization Model is defined. For the XOR problem, the model is encoded using a Lotus compatible worksheet to minimize the sum of the squares of the deviations of the target and computed output. The model is solved with either the Lotus or Excel solver, and the computational experience is discussed. The solution does suffer from the usual problem of over training. In Section 3, the Weight Minimization Model is presented where the acceptable deviations of the target and computed output are specified as constraints and the objective function is to minimize the sum of the squared weights. The resulting weights provide a robust solution. The experience of using the worksheet solvers in training the XOR and 3-Parity neural networks is included. In Section 4, the Weight Minimization Model is applied to a copper price forecasting model. This paper assumes that the reader has familiarity with training neural networks.

2. Error Minimization Model

In this section, the Error Minimization Model for the traditional three-layer feedforward neural network architecture is presented. It is described in precise mathematical terms so as to compare it with the Weight Minimization Model of Section 3.

2.1 Statement of Error Minimization Model

The objective of the Error Minimization Model is to minimize the sum of the squares of the deviations of the target and computed output. To define this mathematically, first the equations to compute the output of the neural network will be specified followed by the expression for the error. Let x_{ip} denote the input value to the i-th input processing element for the p-th training pattern, where i equals 1 to m, the number of input processing elements and p equals 1 to q, the number of training patterns. Thus, the net input for the p-th pattern into the j-th processing element of the hidden layer can be computed as:

$$net_{jp}^{h} = \sum_{i} w_{ji}^{h} x_{ip} - \theta_{j} \qquad (1)$$

where w_{ji}^{h} is the weight of the connection from the I-th input element to the j-th hidden element, and θ_{j}^{h} is the bias term. The h superscript refers to quantities in the hidden layer. Next, the sigmoidal activation function is used to compute the output of the j-th processing element of the hidden layer as:

$$out_{jp}^{h} = \frac{1}{1 + e^{(-net_{jp}^{h})}} \qquad (2)$$

In a similar fashion, the net input into the k-th processing element of the output layer can be computed as:

$$net_{kp}^{o} = \sum_{j} w_{kj}^{o} out_{jp}^{h} - \theta_{k}^{o} \qquad (3)$$

where w_{kj}^{o} is the weight of the connection from the j-th hidden element to the k-th output element, θ_{k}^{o} is a bias term and n is the number of processing elements in the hidden layer. The o superscript refers to quantities in the output layer. Again, the sigmoidal activation function is used to compute the output of the k-th processing element of the output layer as:

$$out_{kp}^{o} = \frac{1}{1 + e^{(-net_{kp}^{o})}} \qquad (4)$$

Now that the equations to compute the output of the neural network are specified, the objective function of the Error Minimization Model may be defined as:

$$minE = \tfrac{1}{2} \sum_{kp} (tar_{kp} - out_{kp}^{o})^2 \qquad (5)$$

where tar_{kp} denotes the target value for the k-th output processing element of the p-th training pattern and out_{kp}^{o} satisfies Equations (1) - (4).

2.2 Implementation of Error Minimization Model

The Error Minimization Model for the XOR problem is encoded in a Lotus compatible worksheet and solved with either the Lotus or Excel solver. The patterns are entered in the worksheet as depicted in Figure 1. An initial set of small random numbers is used as a starting point for the weights and bias terms for the two hidden processing elements and the one output processing element. After the worksheet solver is invoked, these numbers are replaced with the final solution as shown in Figure 2. The results of encoding Equations (1) and (2) for the two hidden elements is shown in Figure 3. The similar results of encoding Equations (3) and (4) for the one output element is given in Figure 4 along with the deviations from the target values and the value

Copyright © 1995 Software Engineering Press

of E as determined by Equation (5).

Concerning the performance of the solvers for different sets of initial random numbers for the weights and bias terms, the Lotus solver frequently gave the message that it could not solve the XOR problem. However, when the same worksheet was input into Excel, the Excel solver frequently solved the XOR problem within 30 seconds on a 486 PC. The weights and bias terms of a typical solution are given in Figure 2, with the corresponding errors shown in Figure 4.

PATTERNS

	Input		Target
p	X1	X2	
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Figure 1 - Patterns for XOR Problem

Layer	Neuron	Weights From Neuron 1	Weights From Neuron 2	Bias
Hidden	j=1	6.771183	-19.4905	-15.1135
Hidden	j=2	-116.55	148.6741	-35.4626
Output	k=1	-22.7804	-25.1157	-35.6946

Figure 2 - Weights for XOR Problem

	Hidden			
p	Net 1	Net 2	Out 1	Out 2
1	15.11354	35.46258	1	1
2	-4.37698	184.1367	0.012407	1
3	21.88472	-81.0872	1	6.1E-36
4	2.394203	67.58689	0.916384	1

Figure 3 - Data for Hidden Elements

Output		Errors	
Net 1	Out 1	Tar-Act	(Tar-Act)^2
-12.2015	5.0E-06	-5.0E-06	2.5230E-11
10.29626	0.999966	0.000034	1.1396E-09
12.91421	0.999998	2.5E-06	6.0654E-12
-10.2967	0.000034	-3.4E-05	1.1386E-09
		Obj Fct:	1.1548E-09

Figure 4 - Data for Output Element

3. Weight Minimization Model

As can be seen in Figure 4, the absolute deviations of the target and output are all less than 10^{-4}. For most business applications, it is not necessary to have such high precision. Many times, achieving higher precision on the training set of data, leads to poorer performance evaluating new data. This is called overtraining. To overcome this with the standard error back propagation method, the training process is monitored and stopped when it is felt that the summed squared error is sufficiently small. However, what frequently happens is that some patterns have small errors while others have unacceptably large errors. This is the motivation behind the second model, the Weight Minimization Model.

The objective of the Weight Minimization Model is to minimize the sum of the squares of the weights and bias terms subject to the constraints that, for each pattern and each output element, the deviation of the target and output is acceptably small. By analyzing each training pattern, the analyst can feel confident that the pattern is appropriate and should be kept or it is inappropriate and should be eliminated. Sometimes the training data has inconsistencies. When this is the case, no training method will work because there is no solution. The old saying, "Garbage in, garbage out," is especially true with neural networks. The training data must be analyzed for inconsistencies. The Weight Minimization Model provides the analyst a method to do this analysis explicitly.

3.1 Statement of Weight Minimization Model

The objective function for the Weight Minimization Model is defined as:

$$\min W = \sum_{ij} (w_{ji}^h)^2 + \sum_{j} (\theta_j^h)^2 + \sum_{jk} (w_{kj}^o)^2 + \sum_{k} (\theta_k^o)^2 \qquad (6)$$

subject to the constraints that

$$| tar_{kp} - out_{kp}^o | \leq err_{kp} \qquad (7)$$

where err_{kp} denotes the acceptable error of the k-th output processing element for the p-th training pattern, tar_{kp} denotes the target value of the k-th output processing element for the p-th training pattern and out_{kp}^o satisfies Equations (1) - (4).

Copyright © 1995 Software Engineering Press

3.2 Implementation of Weight Minimization Model for the XOR Problem

The worksheet implementation of the Weight Minimization Model is a straightforward modification of the worksheet created for the XOR Error Minimization Model. To implement the objective function of Equation (6), the squares of the weights and bias terms are computed as shown in Figure 5. For the solver, this cell is identified as the optimal cell. To implement the constraints of Inequality (7), an additional column is added adjacent to the Errors columns, as shown in Figure 6. Please note that since the actual output cannot exceed 1.0 nor go below 0.0, the absolute value stated in Inequality (7) can be replaced by the one-sided inequalities given in Figure 6. For the solver, these cells are identified as the constraint cells.

Concerning the performance of the solvers for the Weight Minimization Model, here too the Lotus solver usually gave the message that it could not solve the problem. However, the Excel solver frequently solved the problem within 30 seconds on a 486 PC. The weights and bias terms of a typical solution is given in Figure 5, with the corresponding errors shown in Figure 6. Please note how the absolute values of the weights and bias terms are much smaller than for the solution of the Error Minimization Model as given in Figure 2. In fact, the sum of the squared weights and bias terms is 40,023 for the Error Minimization Model versus 244 for the Weight Minimization Model. Also the values of the weights and bias terms of the solution to the Weight Minimization Model (see Figure 5) have a symmetry, which one would suspect from the symmetry of the patterns and network architecture.

The solution of the Weight Minimization Model is more robust than the Error Minimization Model in two respects. This can be verified visually for the XOR problem because the solution is three dimensional. This is not true of the copper forecasting problem presented later, which is six dimensional. For example, Figure 7 shows a three dimensional plot of the neural network output of the Error Minimization Model as a function of x_1 and x_2. Note how sharply changing and asymmetrical the output is. By contrast, Figure 8 shows the output of the Weight Minimization Model. This surface is smooth and symmetrical. In general, one solution is more robust than another solution if small changes in x_1 and x_2 result in small changes in the output. This is not true of the Error Minimization solution for values of x_1 and x_2 in the neighborhood of the sharply dropping cliffs. Here, a little change in x_1 and x_2 can cause the output to jump from the floor of the valley to the top of the plateau. A similar solution for the copper forecasting model would cause the forecast to vary greatly due to small changes in the input variables. Because of this, the solution of the Weight Minimization Model is more robust than the Error Minimization Model.

Another aspect in which the solution of the Weight Minimization Model is more robust than the Error Minimization Model is with respect to the initial random weights. For the Error Minimization Model making small changes in the initial random weights can cause large changes in the location and orientation of the cliffs, whereas the shape of the Weight Minimization solution changes very little.

3.3 Implementation of Weight Minimization Model for the 3-Parity Problem

The worksheet implementation of the Weight Minimization Model for the 3-Parity Problem is a straightforward modification of the worksheet created for the XOR Weight Minimization Model. The new variable is denoted by x_3 and the additional four patterns are appended to the Pattern Matrix. One additional processing element is affixed to the input layer and another processing element is attached to the hidden layer, both fully connected. Figure 9 shows a typical solution.

Concerning the performance of the solvers for the 3-Parity Weight Minimization Model, here too the Lotus solver usually gave the message that it could not solve the problem. However, the Excel solver solved the problem several times within 5 to 8 minutes on a 486 PC. Please note how the values of the weights and bias terms of the solution (see Figure 9) have a symmetry, which one would suspect from the symmetry of the patterns and network architecture. These results also indicate that this solution of the Weight Minimization Model is robust.

Layer	Neuron	Weights From Neuron 1	Weights From Neuron 2	Bias	
Hidden	J=1	4.702212	-4.7022	-2.46581	
Hidden	J=2	-4.70219	4.702182	-2.46581	
Output	K=1	-5.8776	-5.87761	-8.63767	Obj Fct:
Sum of sq. Wts		78.76757	78.7675	86.76968	244.3047

Figure 5 - Modification for Objective Function

Copyright © 1995 Software Engineering Press

Errors		Constraints	
Tar-Act	(Tar-Act)^2		
-0.1	0.01000001	0	Tar-Act >=-0.1
0.1	0.010000021	0	Tar-Act <=0.1
0.1	0.00999999	1	Tar-Act <=0.1
-0.1	0.010000019	0	Tar-Act >=-0.1
	0.020000021	Constraint satisfaction: 1=Yes 0=No	

Figure 6 - Modification for Constraints

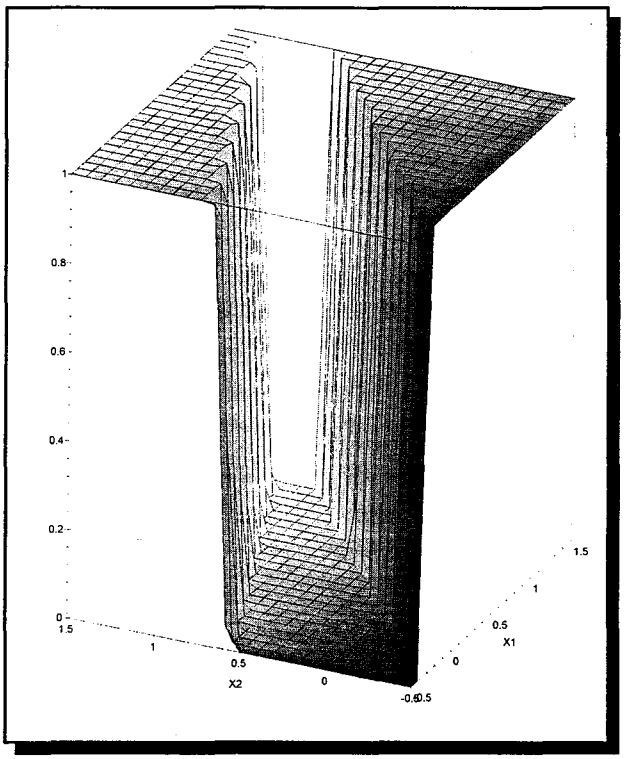


Figure 7 - 3D Plot of Error Minimization Model Solution

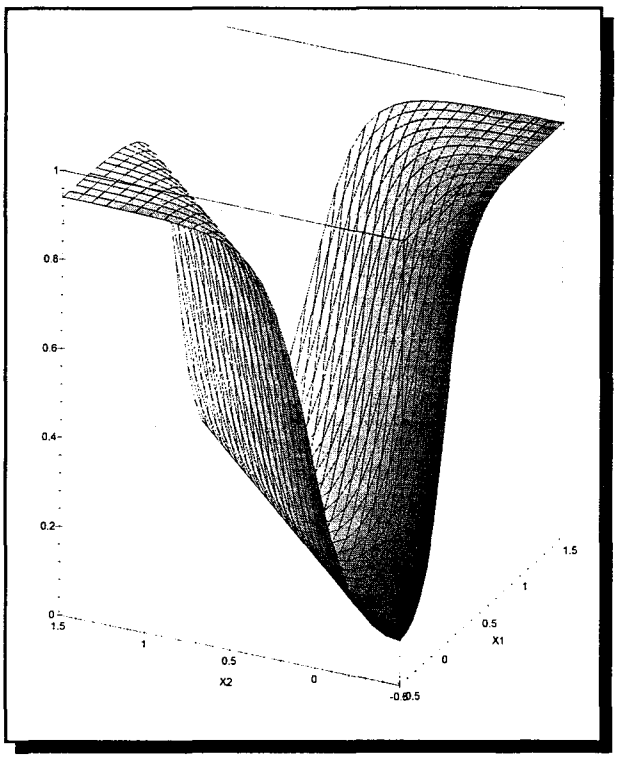


Figure 8 - 3D Plot of Weight Minimization Model Solution

Layer	Neuron	Weights From Neuron 1	2	3	Bias	
Hidden	j=1	4.30269	-4.49167	4.30337	-2.46128	
Hidden	j=2	-4.90067	5.08072	-4.90143	-2.26531	
Hidden	j=3	3.41254	-3.53727	3.41301	5.66029	
Output	k=1	-6.19687	-6.66774	-7.15128	-9.58061	Obj Fct:
Sum of sq. Wts		92.5764	102.9601	105.3326	133.8946	434.7636

Figure 9 - Solution of 3-Parity Problem

4. Time Series Forecasting

A powerful aspect of neural networks is that they can be used to create nonlinear forecasting systems which use more than one time series as input. To illustrate this, a case of predicting the scrap copper spot market price index will be considered. The values of the copper price index for the time period from January 1989 to December 1993 were obtained from the Survey of Current Business [8] and encoded in an Excel worksheet. Figure 10 shows a partial listing of the copper price index for the year 1992. Two other time series which may be related to the copper index are:

- Manufacturers' new orders for consumer goods and materials industries in 1987 dollars (bil. $), and
- Index of new private housing units authorized by local building permits (1967=100).

The values of these two time series for January 1989 to December 1993 were obtained from the Survey of Current Business and included in the worksheet. Figure 10 shows the values of these series for most of 1992.

Copyright © 1995 Software Engineering Press

4.1 Copper Forecasting Model

A reasonable neural network model to forecast the future one month change in the copper index would consist of five input processing elements (PEs), three hidden layer PEs, and one output PE. The five input PEs are associated with the input data as follows:

- PE 1 - Manufacturers' Orders
- PE 2 - New Homes
- PE 3 - Copper Price
- PE 4 - Last One Month Change
- PE 5 - Last Two Month Change.

The data for the fourth PE is found by subtracting the value of the previous month's copper index from the present month. In a similar fashion, the data for the fifth PE is found by subtracting the value of the copper index two months earlier from the present month. These two time series supply the change and rate of change information to the forecast.

To train the neural network, the data from the three year time period March 1989 to February 1992 is selected. The data for each of the five input PEs and the one output PE is normalized to range between -1.0 and 1.0. Figure 11 gives a partial listing of the normalized data. Equations (1) thru (4) are encoded in the worksheet as shown in Figure 12. Note that since the data has been scaled from between -1.0 to 1.0, Equations (2) and (4) are modified to $2/(1+\exp(x))-1$ so that they range from -1.0 to 1.0.

4.2 Solution Procedure

The first step of the solution procedure starts with initializing the weights to a set of random values, whose values vary from -0.3 to 0.3. Figure 13 shows a typical set of initial weights. The range form -0.3 to 0.3 was determined by experimentation, however little difference in the overall solution was noted for the initial random weights varying from the small range of -0.1 to 0.1 to the large range of -3.0 to 3.0.

The second step of the solution procedure is to invoke the Solver so as to find an initial feasible solution. The Solver Parameters are set as follows:

- Changing Cells - the neural network weights
- Constraints
  - absolute value of the error (target - output) for each pattern is less than an acceptable error tolerance (0.5)
  - absolute value of each weight is less than 3.5
- Target Cell - not specified
- Options - max time 3600 seconds, iterations 10,000, precision 0.1, tolerance 10%, use automatic scaling, estimates quadratic, derivatives forward, and search Newton.

It should be noted that the error constraints are truncated as follows:

If target > 0.4 and output > 0.3, then error = 0.0.

If target < -0.4 and output < -0.3, then error = 0.0.

The rationale for this is that if the normalized copper price is up by 0.4 and the forecast is greater than 0.3, such as 0.8, then this is an acceptable forecast and the model should not try to make the error any smaller. Rather it should concentrate on learning the differences between up and down forecasts. Then the solver is invoked and usually runs for 10 minutes or more. The typical response is that the Solver cannot find a feasible solution which satisfies all of the pattern constraints.

The third step is to inspect the results for inconsistencies. It is noted that approximately eight patterns have large deviations with the forecasts. Five of these patterns are not consistent with expectations. These months are May 1989, July 1989, April 1990, June 1990, and November 1991. For example, in May 1989, all the five inputs are positive, yet the future one month change is negative. This is the difficult subjective aspect of developing a forecasting model. If too much data is omitted, then critical relationships are eliminated. However, keeping inconsistent data results in poor model performance.

The fourth step in the solution procedure is to remove these months from the constraints and to use the Solver to minimize the sum of the squared weights with the remaining constraints. The rationale for this is that by minimizing the weights, rather than the errors, a smooth surface will be created as demonstrated by the XOR problem. Figure 14 shows a typical set of final weights.

4.3 Evaluation of Forecast

The above neural network model can be used to forecast the change in the copper price one month in the future. Figure 15 shows the results for the first ten months outside of the training data (March 1989 to February 1992). For the first three months, March, April, and May 1992, the forecasted change in copper price was about 2 or 3¢, and the actual changes ranged from 0 to 5¢, which is acceptable. In June 1992, the forecasted change is -0.6¢ when the actual change is 10.6¢. This is an unacceptably large error. For the next four months, starting in July 1992, the actual changes in copper prices decrease for every month with a cumulative drop of 22.4¢. For this same period, the

Copyright © 1995 Software Engineering Press

forecast also predicts a decrease for every month, but the cumulative drop is only 11.3¢. For the next two months, both the forecast and actual changes are close. However, for the year 1993, the actual change in copper is negative for every month from January to October while the forecasted change in copper is positive for every month. This is unsatisfactory. The overall evaluation of the model is mixed. Perhaps retraining the model every three months on the last three years of data would improve performance. However, the incorrect sign of the forecasts for most of 1993 indicates that it may be wise to try other time series than manufacturers' orders and/or new homes.

Month	Mfrs Oders	New Homes	Copper Price
Dec-92	110.00	93.80	0.834
Nov-92	105.54	89.20	0.793
Oct-92	104.29	90.30	0.841
Sep-92	101.65	88.80	0.895
Aug-92	101.84	85.70	0.986
Jul-92	102.35	86.80	1.017
Jun-92	102.92	84.30	0.911
May-92	101.14	84.20	0.873
Apr-92	102.21	84.10	0.825
Mar-92	100.66	86.30	0.827

Figure 10 - Partial Data for Copper Data

Normalized Data for Training (First 3 Years)

Data normalized between -1 and +1.

Normalized = 2*(actual - min)/(max - min) - 1

Actual						
Min	92.4800	62.7000	0.5720	-0.1170	-0.1940	-0.1170
Max	114.6000	139.4000	1.0840	0.1060	0.2010	0.1060

Constants for normalization - determined manually from the above actual min, max						
Min	90.0000	60.0000	0.5000	-0.1200	-0.2000	-0.1200
Max	115.0000	140.0000	1.2000	0.2000	0.2500	0.2000

	Input PE 1	Input PE 2	Input PE 3	Input PE 4	Input PE 5	Output PE
Month	Mfrs Oders	New Homes	Copper Price	Last 1 Month Change	Last 2 Month Change	Future 1 Month Change
Feb-92	-0.1816	-0.2150	0.0114	-0.0687	0.2000	-0.4188
Jan-92	-0.3088	-0.3525	-0.0714	0.0062	-0.0356	-0.0687
Dec-91	-0.3648	-0.3850	-0.1886	-0.4000	-0.2578	0.0062
Nov-91	-0.0328	-0.5375	-0.1200	-0.3063	-0.1244	-0.4000

Figure 11 - Partial Normalized Data

Hidden						Output	
Net 1	Net 2	Net 3	Out 1	Out 2	Out 3	Net 1	Out 1
0.7211	1.4859	-0.3539	0.3457	0.6309	-0.1751	-0.1610	-0.0803
1.1348	0.9131	-0.6429	0.5134	0.4273	-0.3108	-0.0473	-0.0237
1.0294	0.5834	-0.3372	0.4736	0.2837	-0.1670	-0.0463	-0.0231
1.2046	0.7729	-0.7176	0.5387	0.3683	-0.3442	-0.0843	-0.0421
1.4658	0.3801	-0.9179	0.6248	0.1878	-0.4292	-0.1325	-0.0662
1.2247	0.4352	-0.6510	0.5458	0.2143	-0.3145	-0.1554	-0.0775

Figure 12 - Partial Computations for Hidden and Output PEs

Table of Random Weights Using Formulas						Bias
Neuron	From Neuron					
	1	2	3	4	5	
j=1	0.0427	0.1927	-0.0321	0.1927	0.2497	-0.0245
j=2	0.2738	-0.1984	0.1183	-0.1835	-0.1956	0.1064
j=3	0.2440	0.0029	-0.0233	0.1253	-0.0555	0.2427
k=1	-0.2444	-0.2054	0.0589	none	none	-0.1212
	-0.3 Min weight 0.110302 Formula					
	0.3 Max weight					

Figure 13 - Initial Weights

		Weights					Bias
Layer	Neuron	From Neuron					
		1	2	3	4	5	
Hidden	j=1	0.0282	0.3920	-0.7174	1.3705	0.0007	0.3996
Hidden	j=2	-0.0026	0.4216	0.1370	-0.5820	0.2698	-0.1295
Hidden	j=3	-0.3354	0.9353	-3.2581	-3.2526	1.3697	0.2830
Output	k=1	3.2111	-3.2684	1.4010	none	none	-1.1318
Sum of sq. Wts		10.4245	11.8885	13.1111	12.7962	1.9488	1.5375

Figure 14 - Final Weights

Month	Forecasted Change in Copper Price	Actual Change in Copper Price	$ Error in Forecast
Dec-92	0.040	0.056	0.016
Nov-92	0.026	0.041	0.015
Oct-92	-0.011	-0.048	-0.037
Sep-92	-0.040	-0.054	-0.014
Aug-92	-0.061	-0.091	-0.030
Jul-92	-0.001	-0.031	-0.030
Jun-92	-0.006	0.106	0.112
May-92	0.022	0.038	0.016
Apr-92	0.027	0.048	0.021
Mar-92	0.027	-0.002	-0.029

Figure 15 - Forecast

Copyright © 1995 Software Engineering Press

5. Conclusions

This paper has presented two models for training neural networks. The first conclusion is that, for the basic XOR problem, the Weight Minimization Model produces a more robust solution than the Error Minimization Model. The second conclusion is that the Weight Minimization Model can be successfully extended to the 3-Parity Problem and that the resulting solution is also robust. Future research will investigate the higher order parity problems.

Another conclusion is that the Weight Minimization Model can be applied to practical time series forecasting problems, as demonstrated by the copper price index forecasting model. With this model, the analyst can identify and eliminate inconsistent training patterns and then determine the weights. In general, the Weight Minimization Model encoded in a Lotus compatible worksheet provides the analyst with a flexible environment to develop neural networks. Future research will investigate larger neural networks. The authors welcome test cases.

References

1. Freeman, James A. and David M. Skapura, *Neural Networks: Algorithms, Applications and Programming Techniques*, Reading, MA: Addison-Wesley, 1991.

2. Kosko, Bart. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Englewood Cliffs, NJ: Prentice Hall, 1990.

3. Lyons, Patrick and Stephen C. Persek, "Integrating Neural Networks and Expert Systems for Merger & Acquisition Analysis", *Proceedings of the First International Conference of Artificial Intelligence Applications on Wall Street*, Oct. 10, 1991, IEEE Computer Society Press, Los Alamitos, CA, pp. 200-205.

4. Lyons, Patrick, Thomas Abraham, Larry W. Boone, and Brenda Massetti, "Using AI Technology for Technology Transfer", *Proceedings of the Second International Conference of Artificial Intelligence Applications on Wall Street*, Apr. 22, 1993, Software Engineering Press, Gaithersburg, MD 20879, pp. 242-249.

5. Lyons, Patrick, *Applying Expert System Technology to Business*, Belmont, CA: Wadsworth Publishing Company, 1993.

6. Maren, Aliannna, Craig Harston, and Robert Pap. *Handbook of Neural Computing Applications*, San Diego, CA: Academic Press, 1990.

7. McClelland, James L. and David E. Rumelhart, *Parallel Distributed Processing*, Cambridge, MA:Bradford-MIT Press, 1988.

8. *Survey of Current Business*, October 1994, Bureau of Economic Analysis, U.S. Department of Commerce, Washington, D.C.

9. Van der Smagt, P. Patrick, "Minimization Methods for Training Feedforward Neural Networks," *Neural Networks*, Vol. 7, No. 1, 1994, pp. 1-11.

Copyright © 1995 Software Engineering Press

Paper Session: Fundamental and Value Strategies

Chair: Mike Gargano, Pace University

Predicting Quarterly Excess Returns: Two Multilayer Perceptron Training Strategies

Ypke Hiemstra
Faculty of Economics and Econometrics,
Vrije Universiteit Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam,
The Netherlands
email: yhiemstra@econ.vu.nl

Christian Haefke
Department of Economics
Institute for Advanced Studies
Stumpergasse 56, A-1060 Vienna,
Austria
email: chris@ihssv.wsr.ac.at

Abstract

This paper compares two different training strategies for multilayer perceptrons to predict quarterly stock market excess returns. Finance research suggests that quarterly stock market excess returns are to some extent predictable, but only marginal attention has been paid to possible nonlinearities in the return generating process. The paper discusses input selection, examines the two training strategies, and evaluates multilayer perceptron performance. Several performance measures are calculated, and a test is performed whether the mean squared errors of the various models differ significantly. Strong nonlinear effects appear to be absent, but the multilayer perceptron predictions produce a much higher payoff when applying a straightforward tactical asset allocation policy.

1. Introduction

Finance research suggests that monthly, quarterly and annual excess returns are to some extent predictable assuming a linear model specification. Only marginal attention has been paid to possible nonlinearities in the return generating process. HIEMSTRA (1993,1994B) applies multilayer perceptrons (MLPs) to predict quarterly excess returns. The motivation to consider MLPs is their universal approximation capability (HORNIK STINCHCOMBE WHITE 1989), and robustness when distributions are non-Gaussian (LIPPMANN 1987). This paper compares two MLP training strategies to predict the quarterly excess return on the S&P500 and uses OLS as a simple benchmark. HIEMSTRA (1994B) trains MLPs by backpropagation, varying the number of hidden nodes and using cross-validation to determine optimal stopping (WEIGEND ET AL. 1990). We extend this research by considering an alternative way to train MLPs, and by including a formal test whether the mean squared errors of the various models differ significantly. The additional strategy which we consider seeks to minimize the in-sample error of MLPs with varying numbers of hidden nodes using the Polak-Ribière Conjugated Gradient (PRCG) algorithm. In a second step it selects a net of particular complexity using an estimate for the prediction risk.

First we discuss excess return prediction and input selection. Section 3 discusses the generation of out-of-sample predictions. Section 4 discusses the two MLP training strategies. Section 5 presents the MLP results and compares them to OLS. Section 6 presents the results of a straightforward investment strategy based upon the MLP and OLS predictions, and section 7 contains conclusions.

Copyright © 1995 Software Engineering Press

Table 1: Regression Statistics

Regression Statistics for OLS	
Multiple R	0.45
R^2	0.20
Adjusted R^2	0.16
Standard Error	7.58
Observations	93

2. Predicting Excess Returns

Evidence has been accumulated that a significant part of the variation in stock market returns can be predicted using information known at the time of prediction, e.g., CAMPBELL (1987), FAMA FRENCH (1989), FERSON HARVEY (1991) PESARAN TIMMERMANN (1994). These studies invariably apply linear modelling, with a limited number of independent variables. In particular evidence has been accumulated that ex ante information on inflation, interest rates, the business cycle, and valuation measures like the dividend yield, can be used to predict monthly, quarterly, and annual excess returns.

This study focuses on quarterly excess returns. The linear model we use as a benchmark can be found in HIEMSTRA (1994B), tables 1 and 2 present the in-sample OLS results on the entire data set. The model uses four inputs: dividend yield (YSP), short term interest rate (SIR), inflation rate based on the consumer price index (CPI), and the change in the 12-month moving average of the industrial production index (DIP). YSP and SIR are instantaneously available, and so the latest observations prior to the forecasted period were used to predict, i.e. the values at the end of the preceding month. Macroeconomic information is available typically on a monthly basis with a lag of some 20 days, and so DIP and CPI were used with a 2-month lag. The data set consists of 93 quarterly observations covering the period 1970-1993, of YSPt-1, DIPt-2, SIRt-1, and CPIt-2. The desired output is the S&P500 quarterly excess return, defined as total return (price movement plus dividends related to the initial investment) minus the risk-free rate of return, the 3-month T-bill rate.

Table 2: Coefficients of linear model

	Coefficients	Standard Error	t Statistic
Intercept	-2.87	3.43	-0.84
YSPt-1	4.43	1.26	3.52
CPIt-2	-1.00	0.39	-2.54
SIRt-1	-0.76	0.37	-2.03
DIPt-2	-6.78	2.12	-3.20

The R^2 has a satisfactory value for predictions at this frequency, and all coefficients pass the test for significance at the 95% level. The signs of the coefficients correspond to the findings of other studies, YSP having a positive coefficient, and the other variables having a negative coefficient.

3. Generating Out-of-Sample Predictions

Given the small sample size at the quarterly frequency, and the noisy character of the data, it is crucial to design sufficient out-of-sample results to reliably estimate generalization. PESARAN AND TIMMERMANN (1994) apply a recursive approach to predicting excess returns, in which case all data available at time t is used to forecast the excess return at time t+1. Cross-validation and bootstrapping are examples of resampling techniques (see for example WEISS AND KULIKOWSKI 1991). Cross-validation uses all data for testing, and consumes considerable less resources than recursive prediction or bootstrapping. We apply 10-fold cross validation to estimate generalization. The 10 test sets were combined to form out-of-sample estimations on the entire data set.

4. Estimating the Multilayer Perceptrons

The MLP architecture we use is the standard MLP architecture for nonlinear regression (HAYKIN 1994) with bias weights and one hidden layer. The hidden units have *tanh* activation functions, the output unit has a linear activation function. Inputs were normalized to have mean zero and unit variance. The question of network design concentrates on the number of hidden nodes.

Copyright © 1995 Software Engineering Press

The motivation to use backpropagation is to stop training before the in-sample minimum on the error function is reached (early stopping), based on the assumption that in the initial stages of learning the net picks up the most overt and accessible patterns (THORNTON 1992). The iterative character of backpropagation allows to stop learning if at a certain stage in the learning process the net starts fitting noise, which may be well before the error function reaches its minimum on the training set. WEIGEND ET AL. (1990) suggest to set part of the training data apart, introducing a cross-validation set in addition to the train and test sets, and stop training at the point where the error function on the cross-validation set has its minimum. True generalization is estimated by the performance on the original test set. To apply early stopping, it is necessary to select the number of hidden nodes in advance. The sample size is a constraint on the number of weights in order to produce reliable training. A rule of thumb is that there should be at least five training examples for each weight (KLIMASAUSKAS 1992). HIEMSTRA (1993,1994b) proposes a MLP with 2 hidden neurons for quarterly excess return prediction.

On the other hand, WHITE (1991) points out that overtraining and overfitting should be separated. "*In the statistical context there is no such thing as overtraining, because the closer one gets to* $\hat{\theta}_n$ *the better.*" $\hat{\theta}_n$ denotes the estimator for the true parameter vector θ_n of the underlying process. So, given a network of some complexity, the goal is to find the global minimum of the error function, and from the set of alternative algorithms (e.g., ordinary differential equations, second order algorithms, Levenberg-Marquardt, see for example SARLE (1994)) any algorithm that efficiently minimizes in-sample error is applicable. Overfitting occurs when there are too many free parameters in the MLP, while a network with too few hidden units has no capacity for satisfactory internal representation and will be unable to learn properly. To select the network with the right complexity, we apply *sequential network construction (SNC)* (MOODY UTANS 1994). We start out with 1 hidden unit and estimate the network. Then a hidden unit is added, and the net is retrained. This is repeated until the hidden layer contains a predetermined maximum number of nodes. The fit of each network is evaluated using an estimator of its prediction risk (i.e. the expected out-of-sample performance). We estimate prediction risk for each net using the Schwartz Information Criterion (SIC), (SCHWARTZ 1978, RISSANEN 1978, 1980, 1987) which produces very conservative models, and which can be used for nonlinear and ARCH models (GRANGER KING WHITE 1992). An application of this criterion can be found in HAEFKE HELMENSTEIN (1995). The SIC of model λ is computed as follows

$$SIC = \log MSE + \frac{P}{N}\log N \qquad (1)$$

with MSE being the mean squared error, N the number of observations and P the number of parameters[1]. For each of the 10 cross-validation sets we estimated MLPs using PRCG[2], selecting the appropriate complexity through SIC (SWANSON WHITE 1992).

[1] We used the number of weights, an alternative is Moody's *effective* number of parameters (MOODY 1992).

[2] Every single net was estimated using the PRCG algorithm, as provided by GAUSS. This is a local optimization algorithm, and so training was repeated five times. The net with lowest in-sample error was selected.

Copyright © 1995 Software Engineering Press

5. Evaluation and Comparison of Out-of-Sample Results

Table 4 compares the SIC-optimal complexity approach with the results for backpropagation using 2 hidden neurons (BPN2) and OLS reported by HIEMSTRA (1994B). Error rate refers to the ratio of correct sign predictions to all predictions. Business value calculates the sum of the excess returns for those occasions that the predicted excess return is negative, and indicates the benefits obtained by implementing a timing strategy for the whole period of 93 quarters that exits the stock market when the expected excess return is negative. R^2 is computed as the squared correlation of the predicted and the actual values. Theil's coefficient of inequality considers the case of a no-change forecast and assumes a value of less than one if the forecast outperforms a no-change forecast (THEIL 1966). The MLP models outperform OLS in terms of correlation, error rate and business value. The poor results on MSE (and NMSE) of the SIC-optimal MLP are explained by the high mean of the predictions of this model.

Table 4: Comparison of out-of-sample results of the 3 approaches

	SIC-optimal MLP	BPN2	OLS
MSE	77.8816	64.6619	65.6435
NMSE	1.1451†	0.9507†	0.9652†
R^2	0.0850	0.0841	0.0701
Theil	0.6637	0.5510	0.5594
correlation	0.2915*	0.2899*	0.2648*
Mean	2.5307	1.8868	1.9509
Error rate	0.387	0.419	0.430
Business value	35.5341	53.2983	29.3062

An F-test was performed to test whether the variances of the residuals of the three models are significantly different from each other. If so, one model outperforms the other because a larger variance of residuals is inferior. Assuming under the H_0 that the variances of the residuals are equal[3], the F-test statistic reduces to:

$$F = \frac{MSE_1}{MSE_2} \qquad (2)$$

where F follows an F-distribution with $(N-P_1)$, $(N-P_2)$ degrees of freedom, N denoting the number of observations and P_1 and P_2 the number of parameters for the respective model. The F-test did not reject the null hypotheses of identical MSEs.

6. Asset Allocation

Figure 1 compares the performance of asset allocation strategies using the three models. The left panel of figure 1 shows policy efficient frontiers. A policy efficient frontier (HIEMSTRA 1994A) represents the annualized ex post risk-return properties of a particular tactical investment policy for those strategic portfolios that, given the policy, produced the highest return for the respective risk exposures. The investment policies shown operate on a strategic portfolio consisting of stocks and bonds, and exit the stock market when the excess return prediction is negative. The policies were tested on the period 1976-1993 (net of trading costs), with the Shearson-Lehman Aggregate Bond Index representing bond returns. The bottom line indicates the results of a buy-and-hold policy, a 100% bonds portfolio located at the low end, a 100% stocks portfolio located at the high end. The line in between shows the results of a tactical policy using OLS predictions. The solid upper line and the dashed line show the results using the predictions of the SIC-optimal MLP and BPN2, respectively. The latter two lines basically collapse, reflecting a striking similarity in the two neural net models from this point of view.

† Not significantly different from 1 at the 95% confidence level.

* Significantly different from 0 at the 95% confidence level.

[3] A t-test confirmed that the mean of the residuals is not significantly different from zero at the 95% confidence level.

Copyright © 1995 Software Engineering Press

Figure 1. The left panel shows policy efficient frontiers, the right panel the relative value of a 1976 investment when adopting a strategic stock weight of 1 (net of trading costs). In both cases, buy-and-hold is the bottom line, the line above buy-and-hold represents OLS results. The solid upper line and the dashed line show the results using the predictions of the SIC-optimal MLP and BPN2, respectively.

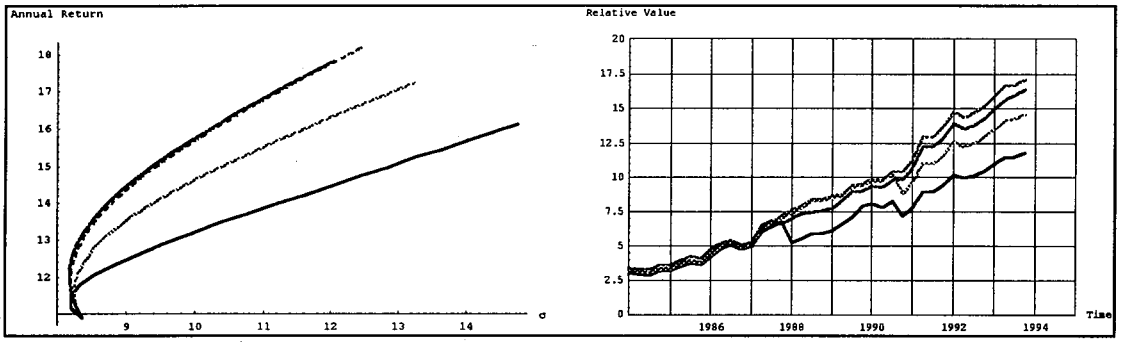


Figure 1 shows that active management has a dramatic payoff. For a particular risk exposure, an OLS-based policy can add well over 100 basis points annually, and both neural nets can generate a similar additional return on top of that. The panel to the right shows the relative value over time of an initial investment in 1976 when the three policies operate on a strategic portfolio of 100% stocks.

7. Conclusion

The MLP models that we estimated improve over OLS in terms of correlation, error rate, and business value. We did not find indications for strong nonlinearities, as OLS captures most of the predictability demonstrated by the MLPs. Also, on the basis of the F-test we cannot reject the H_0 that the out-of-sample MSEs of the two MLP models and OLS are equal. However, in terms of added value when applying a straightforward tactical asset allocation policy, the differences are very significant. The SIC-optimal MLP results are particularly compelling, since this model reflects a purely formal approach to MLP estimation that lets the data determine model complexity.

8. Acknowledgements

We thank Gerhard Ruenstler, Institute for Advanced Studies, for helpful comments on section 5.

9. References

CAMPBELL, J.Y. (1987): Stock Returns and the Term Structure, *Journal of Financial Economics*, pp 373-399.

FAMA, E., FRENCH, K. (1989): Business Conditions and Expected Stock Returns, *Journal of Financial Economics*, pp. 23-49.

FERSON, W.E., HARVEY, C.R. (1991): Sources of Predictability in Portfolio Returns, *Financial Analysts Journal*, May-June, pp. 49-56.

GRANGER, C.W.J., KING, M.L., WHITE, H. (1992): Comments on Testing Economic Theories and the Use of Model Selection Criteria, *University of California, San Diego.*

HAEFKE, C., HELMENSTEIN, C. (1995): Neural Network in the Capital Markets: An Application to Index Forecasting, in GILLI, M. (ed): *Computational Methods in Economics and Finance*, Dordrecht, Boston, London: Kluwer Academic Publishers, *forthcoming.*

HAYKIN, S. (1994): Neural Networks: A Comprehensive Foundation, New York: Macmillan.

HIEMSTRA, Y. (1993): A Neural Net to Predict Quarterly Stock Market Excess Returns Using Business Cycle Turning Points, *Proceedings of The First International Workshop on Neural Networks in the Capital Markets*, London Business School, London.

HIEMSTRA, Y. (1994a), The Application of Intelligent Systems to Tactical Asset Allocation, *Workshop AI in Finance and*

Copyright © 1995 Software Engineering Press

Business, 11th European Conference on AI, Amsterdam.

HIEMSTRA, Y. (1994b): Linear Regression versus Backpropagation Networks to Predict Quarterly Excess Returns, *The Second International Workshop on Neural Networks in the Capital Markets*, California Institute of Technology, Pasadena.

HORNIK, K., STINCHCOMBE, M., WHITE, H. (1989): Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, pp 359-366.

KLIMASAUSKAS, C. (1992): Applying Neural Networks, in: TRIPPI, R.R., TURBAN, E, (eds): *Neural Networks in Finance and Investing*, pp. 47-72 Chicago: Probus Publishing Company.

LIPPMANN, R.P. (1987): An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, April, pp. 4-22.

MOODY, J. (1992): The *Effective* Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems, *Advances in Neural Information Processing Systems 4*, San Mateo: Morgan Kaufman.

MOODY, J., UTANS, J (1994): Architecture Selection Strategies for Neural Networks: Application to Corporate Bond Rating Prediction, in REFENES, A.N. (ed): *Neural Networks in the Capital Market*, London: Wiley & Sons.

PESARAN, M.H., TIMMERMANN, A. (1994): Forecasting Stock Returns, *Journal of Forecasting*, pp 335-367.

RISSANEN, J. (1978): Modelling by Shortest Data Description, *Automatica*, pp 465-471.

RISSANEN, J. (1980): Consistent Order-Estimates of Autoregressive Processes by Shortest Description of Data, in JACOBS, O., DAVIS, M., DEMPSTER, M., HARRIS,C., PARKS, P. (eds.): *Analysis and Optimization of Stochastic Systems*, pp 451-461, New York, Academic Press.

RISSANEN, J. (1987): Stochastic Complexity and the MDL Principle, *Econometric Reviews*, pp 85-102.

SARLE, W.S. (1994): Neural Network Implementation in SAS Software, *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, SAS Institute, Cary, NC.

SCHWARTZ, G. (1978): Estimating the Dimension of a Model, *Ann. Statist.*, pp 461-464.

SWANSON, N., WHITE, H. (1995): A Model Selection Approach to Assessing the Information in the Term Structure Using Linear Models and Artificial Neural Networks, *Journal of Business and Economic Statistics, forthcoming.*

THEIL, H. (1966): Applied Economic Forecasting, Amsterdam: North Holland Publishing Company.

THORNTON, C.J. (1992): Techniques in Computational Learning, London: Chapman and Hall.

WEIGEND, A.S., HUBERMAN, B.A., and RUMELHART D.E., (1990): Predicting the Future: A Connectionist Approach, *International Journal of Neural Systems*, pp. 193-209.

WEIGEND, A.S., LeBARON, B. (1994): Evaluating Neural Network Predictors by Bootstrapping, *CU-CS-725-94 University of Colorado*, submitted to ICONIP-94.

WEISS, S., and KULIKOWSKI, C.(1991): Computer Systems That Learn, Palo Alto: Morgan Kaufman.

WHITE, H. (1991): Parametric Statistical Estimation with Artificial Neural Networks, *mimeo*, Department of Economics and Institute for Neural Computation, University of California, San Diego.

Copyright © 1995 Software Engineering Press

Automated Understanding of Financial Statements Using Neural Networks and Semantic Grammars

James Markovitch
Dun & Bradstreet Information Services, N. A.
150 Mount Airy Road
Basking Ridge, NJ 07920-2015
jmarkov@shell.portal.com

Abstract

This article discusses how neural networks and semantic grammars may be used to locate and understand financial statements embedded in news stories received from on-line news wires. A neural net is used to identify where in the news story a financial statement appears to begin. A grammar then is applied to this text in an effort to extract specific facts from the financial statement. Applying grammars to financial statements presents unique parsing problems since the dollar amounts of financial statements are typically arranged in multiple columns, with small paragraphs of text above each column. Text therefore is meant to be read both vertically and horizontally, in contrast to ordinary news text, which is read only horizontally.

1 Introduction

Each year more information becomes available in electronic form. Some of this comes from well known, general sources such as UPI, Reuters and the Internet; still other data comes from lesser known sources such as the PR NEWSWIRE and BUSINESS WIRE, which supply financial and other information.

However, for information to be useful it must be either indexed or digested to suit each individual's or company's needs, inasmuch as it is inevitable that much of the news collected will turn out be of little or no interest. For an overview of the attempt to solve this problem via indexing techniques, and an account of indexing techniques to large data sets such as MEDLINE, see [3]. This source discusses retrieval methods and ranking algorithms that help make the data that has been archived more accessible. The methods employed take advantage of word stemming, Boolean queries, word-weighting and vectoring schemes. See Addison, *et al,* [1] for a discussion of indexing techniques specifically applied to Real-Time news.

An additional method to aid in the processing of large amounts of news requires that the news be understood. Unfortunately this understanding is not easily arrived at in software. See Shirmer and Kuehn, [5] for a discussion of understanding news via word experts and neural nets; they describe a method whose goals are similar to those described here. Also see Bearden, *et al,* [2] for detailed descriptions of how grammars may effectively mimic human understanding in limited domains.

The present article offers a partial solution to one problem in electronic news understanding: the understanding of financial statements embedded in text. Figure 1 represents the upper portion of a typical financial statement.

TYPICAL COMPANY, INC.
CONSOLIDATED FINANCIAL INFORMATION
UNAUDITED (000's)

	Three Months Ended October 31		Six Months Ended October 31	
	1993	1992	1993	1992
Net sales	$64,314	$63,831	$121,037	$122,180
Gross profit	$32,560	$34,281	$ 61,931	$ 63,963

Figure 1

Copyright © 1995 Software Engineering Press

Such a financial statement might be embedded within an accompanying news story that tells the reasons for the rise or fall of that company's profits, sales, etc. Characteristic of such a financial statement is its columnar presentation of numerical information. Figure 1 has no fewer than four columns of numbers, each representing the financial results for a different time period.

For a human to read such a document presents surprisingly few problems. The column headings, which are isolated paragraphs of text suspended above their respective columns, are easily read as distinct paragraphs. These columns are then readily scanned for the sales and profit information they contain. Unfortunately, for a computer to accomplish this same visual task requires that it have the same visual sense that a human does. It is not clear how to arrive at this visual sense in software, however.

For this reason a method was developed that did not depend on a visual sense of the financial statements, but rather on its *grammatical* sense. Although this method of understanding a financial statement may not accurately reflect how a human reads such a document, it will be shown that the method is a reasonably effective way for a computer to understand financial statements.

2 Using Neural Nets to Find a Financial Statement

Before a financial statement may be examined for its grammatical sense, it is necessary first to locate one. This task is surprisingly difficult to accomplish via programming logic. The first problem is that there is no single recognizable feature that marks the start of a financial statement. And secondly, there are blocks of text, particularly news story headlines announcing earnings results, that look like the start of a financial statement, but are not. Experience has shown that neural networks are effective in recognizing handwriting, speech and visual patterns though a process of statistically based feature extraction [4]. For these reasons a backpropagation network was used to find the start of financial statements in each news story.

In order for the neural net to find the start of the financial statement, a batch program was written to supply it with a "moving window" of fifteen lines of text, where the neural net's single output was trained to produce a score of 1 if the fourth line of this "window" was the start of a financial statement, and to produce a 0 otherwise. For each of the 15 lines the neural net was told about:

1) the line's length,
2) the percentage of letters in the line that were in capitals,
3) the number of leading spaces in the line,
4) the number of embedded spaces in the line,
5) the percentage of the characters in the line that were digits, and,
6) a Boolean value that told whether or not the line was centered.

In addition the neural net was told about two other characteristics of the line: specifically whether certain keywords, such as *inc*, and *company* occurred in the line; and secondly, whether words such as *month*, *quarter*, *year*, *financial*, and *consolidated* were present. In total, the neural net had 120 inputs, representing 8 characteristics, pertaining to 15 lines.

When preparing the training data (i.e., development data) and test data (i.e., holdout data), a line was regarded as containing the start of a financial statement if it had a company name, at least one date, and at least one column of numbers. The requirement that a company name be present was an arbitrary requirement that might not be suitable in all circumstances. The line containing the company name was regarded as the start of the financial statement.

3 Using Semantic Grammars to Parse a Financial Statement

Once the start of a financial statement has been identified, it must be analyzed and understood. When the financial statement of

Copyright © 1995 Software Engineering Press

Figure 1 is rearranged as a stream of text, it appears as in Figure 2.

```
TYPICAL COMPANY, INC. <RETURN> CONSOLIDATED
FINANCIAL INFORMATION <RETURN> UNAUDITED (000's)
<RETURN> Three Months Ended Six Months Ended <RETURN>
October 31 October 31 <RETURN> 1993 1992 1993 1992
<RETURN>  Net sales $64,314 $63,831 $121,037 $122,180
<RETURN> Gross profit $32,560 $34,281 $ 61,931 $ 63,963
<RETURN>
```

Figure 2

A close examination of this stream of text and others similar to it reveals underlying regularities that may be exploited by using a semantic grammar. Semantic grammars, which are described in [2], are an effective means for understanding sentences within a restricted domain. The world of financial statements is clearly such a restricted domain, but can financial statements be viewed as sentences with their own grammar?

To resolve this question several hundred financial statements were analyzed to uncover sentence-like regularities. From this analysis a context-free grammar and a lexicon emerged that allowed a large percentage of financial statements to be processed.

```
HEADING ::= for-the 12-PL end for-the 34-PL end RETURN end
12-PE end 34-PE RETURN 1-Y-E 2-Y-E 3-Y-E 4-Y-E RETURN
12-PL    ::= 12-N-L 12-T-L
12-PE    ::= 12-M-E | 12-M-E 12-D-E
34-PL    ::= 34-N-L 34-T-L
34-PE    ::= 34-M-E | 34-M-E 34-D-E
```

Figure 3

A portion of the context-free grammar is shown in Figure 3, in particular, that portion of the grammar that helps interpret four-column financial statements. In this figure and the remaining figures the symbol *::=* is read as *is defined as*, which follows [2].

```
12-N-L    ::= 3 | 6 | 9 | 12 | three | six | nine | twelve
12-T-L    ::= week | month | year | weeks | months
34-N-L    ::= 3 | 6 | 9 | 12 | three | six | nine | twelve
34-T-L    ::= week | month | year | weeks | months
12-M-E    ::= Jan | Feb | Mar | Apr | May | Jun | etc.
12-D-E    ::= first | second | third | fourth | fifth | etc.
34-M-E    ::= Jan | Feb | Mar | Apr | May | Jun | etc.
34-D-E    ::= first | second | third | fourth | fifth | etc.
1-Y-E     ::= 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | etc.
2-Y-E     ::= 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | etc.
3-Y-E     ::= 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | etc.
4-Y-E     ::= 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | etc.
for-the   ::= for the | for
end       ::= ending | ended
```

Figure 4

Likewise a portion of the lexicon used is shown in Figure 4.

4 One Special Problem

One characteristic of grammars that are designed to understand natural language is that the form of the representation is unimportant. Accordingly, if in a grammar a verb phrase is represented as *VP*, *VerbP* or *V-P*, it does not alter the effectiveness of the grammar in the slightest. This is not entirely the case when parsing

Copyright © 1995 Software Engineering Press

financial statements, however. With financial statements it is necessary, once parsing is complete, to distinguish between a dollar amount found in the first column, and a dollar amount found in the second column. It is also necessary to distinguish between a "period end date" that applies only to column one, and a period end date that applies to columns three and four. The easiest way to do this is to build "cases" into the language that are analogous to the cases used in normal languages, and *to reflect these case differences by using the symbols of the grammar itself.* Accordingly, a period end date that applies only to column one is represented in the grammar as *1-PE*, while a period end date applying to columns three and four appears as *34-PE*. This small restriction on the formation of the grammar has valuable practical consequences when at a later time the meaning of the parsed financial statement must be determined. It allows the components of the parse tree to be processed with relative ease to find what period end dates, and what period lengths, apply to which columns.

5 Neural Network Technical Details

The neural network used had a hidden layer of 16 nodes, in addition to its input layer of 120 nodes, and its single output node. The training data was composed of 26,880 lines of text, which contained 228 financial statements. This data was captured from the PR NEWSWIRE and the BUSINESS WIRE on Nov. 11, 1993 and Nov. 24, 1993. The single output was trained to be a 1 in the presence of a financial statement, and a 0 otherwise. Training on the data was terminated when the worst error for any member of the training data set was no greater than .25. This was accomplished after 135 training iterations.

The test data was composed of 13,736 lines of text, which contained 125 financial statements. This data was captured from the PR NEWSWIRE and the BUSINESS WIRE on Dec 10, 1993 and Dec 17, 1993. When validating using the test data, an output node score greater than .1 was treated as signifying the presence of a financial statement.

6 Neural Network Results

When the network was run against the test data, it proved effective. 118 of 125 financial statements were identified for a rate of 94.4%. Among the 13,611 lines that contained no statement, just twelve were wrongly identified as financial statements, for an error rate of .0881% (less than 1 in a 1,000).

In general the network performed very effectively, with only occasional lapses for "unusual" financial statements. An unusual financial statement might be one whose company name is not centered, but rather appears flush left, or one whose company name lacks the word *incorporated*, *co.*, or *inc.*, etc.

7 Grammar Results

A program employing a more robust version of the grammar and lexicon described earlier proved fairly effective in parsing financial statements. Specifically, the headings of 72 of the 125 financial statements were understood by the grammar (here the word *heading* refers to the lines that tell start and end dates, as well as period lengths). The grammar was slanted towards understanding income statements and it frequently failed in circumstances where it met with some other form of statement, such as a statement of cash flow. Adding new statement types to the grammar can readily expand its comprehension, however. Currently it supports only sixteen.

Problem cases arose as a consequence of the wide variety of financial statements present in news stories. In particular, "unique" financial statements typically appeared at times other than the end of a quarter, and sometimes appeared to be edited by hand for special release.

Still other problems arose from the phrases *in thousands* and *in millions* that sometimes acted as a multiplier on all of the dollar amounts of the financial statement, or sometimes on just a limited portion of it. It is hard to anticipate the variety of ways this multiplier might appear, and

Copyright © 1995 Software Engineering Press

failure to anticipate correctly leads to a very large error. Complex code had to be written to handle these cases, as well as to search the lines of text following the headings for the specific financial information required: e.g. net income, total sales, etc. This task was done using traditional programming methods.

One lesson learned is that the neural network ideally should provide more information than merely where a financial statement starts. For instance, it is useful to know where a statement ends (so as to avoid "falling through" to unrelated text and data). Likewise it is useful to know specifically on which line the grammar should be applied. This line is often many lines after the company name. Lastly, it appeared that a moving window of just 15 lines was too short for a proper understanding of some statements.

8 Conclusions

The above problems notwithstanding, it proved possible to process financial statements embedded in news stories. The method used also proved flexible enough to accommodate new financial statement types as they were discovered. In addition, the grammar used showed a large degree of resistance to misinterpretation. If text other than a financial statement was presented to the grammar, it would readily reject it as unparsable. Currently, the system described has not been developed into a deployable system, but the results achieved here indicate that these methods can be used to create a practical system to automate the understanding of financial statements. The general conclusion is that grammars may prove more effective in a wider array of contexts than is readily apparent.

9 References

[1] Edwin Addison, Judith Feder, Paul Nelson and Tom J. Schwartz, "Extracting and Disseminating Information from Real-Time News." In *Proceedings of the Second International Conference on Artificial Intelligence Applications on Wall Street.* New York, NY, April, 1993, Gaithersburg, MD: Software Engineering Press.

[2] Colin Beardon, David Lumsden, and Geoff Holmes, 1991. *Natural Language and Computational Linguistics An Introduction.* Chichester, England: Ellis Horwood Limited.

[3] W. B. Frakes, and R. Baeza-Yates, 1992. *Information Retrieval: Data Structures and Algorithms.* Englewood Cliffs, NJ, Prentice-Hall.

[4] Marilyn McCord Nelson, and W. T. Illingworth, 1991. *A Practical Guide to Neural Nets.* Reading, MA, Addison-Wesley.

[5] Kai Schirmer and Michael Kuehn, "Fact Extraction from Financial News." In *Proceedings of the Second International Conference on Artificial Intelligence Applications on Wall Street.* New York, NY, April, 1993, Gaithersburg, MD: Software Engineering Press.

Copyright © 1995 Software Engineering Press

Using Neural Networks to Predict the Degree of Underpricing of an Initial Public Offering

Steven Coy, Ravikumar Balasubramanian,
Bruce Golden, Ohseok Kwon, and Heshmat Beirjandi

College of Business & Management
Van Munching Hall
University of Maryland
College Park, MD 20742-1815

Abstract

Underwriters and investors are both interested in knowing, in advance, the price at which an initial public offering (IPO) will trade at the end of the first trading day. In this paper, we propose neural network models to predict initial returns for IPOs. Using data from the mid-1980s, which include 1423 observations, we develop neural network models as well as regression models. The neural network models which include 4 input nodes, 3 hidden nodes, and 1 output node consistently outperform the regression models.

1.0 Introduction

It has been observed that investments in initial public offerings (IPOs) yield positive average initial returns. There is no obvious reason why the issuer does not extract the right price from the investors at the time of issue. Several information-based theories have been proposed in the finance literature to explain this anomaly. However, regression models have not been able to predict the first-day trading price and the expected initial returns with reasonable success.

In this paper, we propose neural network (NN) models to predict the initial returns using a set of ex ante variables. Our purpose in using neural networks is two-fold: (1) To build a NN model which can predict the first-day trading price with reasonable accuracy and (2) to compare the predictive quality of NN models against regression models for this application.

We begin our discussion with a regression analysis of the full data set and several subsets of the full data set. We then use these models to benchmark the performance of the NN models. Next, we build two sets of NN models. The first set of NN models was generated using a commercially available software package, Brainmaker. The second set of models was generated from an original, and more flexible, code. Our results indicate that neural networks can be used with a fair degree of success for predicting initial returns on IPOs. Using Mean Absolute Error (MAE) as our measure of performance, we found that our best model using Brainmaker (from experiment 1; see Table 6) outperformed our best linear regression model by about (1030-968)/1030 = 6.02% and it outperformed our best nonlinear regression model by about (1007-968)/1007 = 3.87%. The best model based on our original code (also from experiment 1) outperformed our best linear regression model by about 8.74%, and it outperformed our best nonlinear regression model by about 6.65%.

2.0 Background

The phenomenon of abnormal initial returns to IPOs has attracted a lot of attention in the finance literature, and several theories have been proposed to explain this puzzle. Benveniste and Spindt [1] argue that information-gathering activities by underwriters during the pre-issue period affect the level of initial returns. They observe that changes in the offer price between

Copyright © 1995 Software Engineering Press

the filing of the prospectus and the offer date are a function of information gathered from investors during the pre-issue period. When positive feedback is revealed through high demand for the issue, the final offer price will exceed the expected offer price. When negative feedback is revealed by low demand, the offer price is set below the expected value. If investors reveal their true demand, the issuer will increase the offer price. To induce the investor to reveal information truthfully, the issuer has to promise an increase in allocation of shares. Thus, investors face a trade-off between increased allocation and underpricing. However, if allocations cannot be increased beyond a certain point, then the investor has to be rewarded for truth-telling mainly through underpricing.

The above theory yields several natural hypotheses which were then tested by Hanley [2]. Hanley identifies several variables, such as offer amount and underwriter's reputation, which may affect the initial returns. The ordinary least squares (OLS) regression between the initial returns and the variables (Table 3 of the Hanley paper) indicates that about 17.8% of the total variation can be explained by the variables. While many predictions made by Benveniste and Spindt have been validated in Hanley [2], the relatively low R^2 value might be explained by one of the following three reasons (or a combination of these): (1) One or more explanatory variables have been omitted from the model; (2) there is a nonlinear relationship between initial returns and one or more of the variables identified; and (3) the data set has a large random noise component.

3.0 Experimental Method

3.1 Overview

In this paper, we first build linear and nonlinear regression models to predict the initial returns used by Hanley. These will serve as benchmarks for our neural network models. We then build neural network models to explain and predict the initial returns using the same variables. Our premise is that the initial returns

Variable	*Definition*
Initial Returns	$(P_f - P_0)/P_0$
Percent Change in the Offer Price	$(P_0 - P_e)/P_e$
Offer Amount	NP_0
Percent Change in the NASDAQ Index	$(M_1 - M_0)/M_0$
Reputation of the Underwriters	Average market share for 1983 - 1987

Key to Abbreviations

P_0 = Offer Price
P_e = Expected Offer Price = midpoint of the offer
P_f = Price at the end of the first trading day
N = Number of shares
M_0 = NASDAQ index at the time of preliminary filing
M_1 = NASDAQ index on offer date

Table 1. Definition of Variables

Copyright © 1995 Software Engineering Press

may not be linearly related to these variables. If significant nonlinearities exist, a neural network model might be able to capture these nonlinearities better than either a linear or nonlinear regression model. If so, this would improve the predictive power of the model. Our results are based on two sets of NN models. The first set of models was built using Brainmaker. The second set of models was generated using our original code. Both Brainmaker and the original code use the backpropagation algorithm for training the network. Brainmaker allows the setting of a variety of parameters such as the sigmoid slope, learning rate, error threshold, number of neurons, and number of hidden layers by the user. However, the original code allows the user more flexibility. For example, it allows the user to set a different sigmoid slope for each neuron and it allows the pruning of a fully connected network.

3.2 Data

For this study, we used data on 1430 firm commitment IPOs from January 1983 through September 1987 compiled from *Investment Dealers' Digest Corporate Database* reports. This represents the entire population for this period. We describe each variable in Table 1. The first variable listed, initial returns, is our dependent variable. Based on Hanley [2], we choose percent change in the actual offer price from the expected offer price, offer amount, percent change in the NASDAQ index, and underwriter's reputation as our independent variables. Although Hanley [2] includes other variables in her explanatory regression model, we omitted these variables in our models since these were ex post variables which cannot be used for prediction.

3.3 Experimental Design

Our investigation involved three experiments using four different models in each experiment. These included a regression model, a nonlinear regression model, a neural network model generated by Brainmaker, and a neural network model generated by the original code. All of the neural network models were composed of multilayer perceptrons which were trained using the backpropagation algorithm. The data were divided into three subsets: A, B, and C. These subsets were obtained by partitioning the data alphabetically. Seven of the 1430 observations were deleted due to missing data. This left 1423 usable observations.

Figure 1 illustrates the experimental procedure. In experiment 1, we trained the models on subset A and tested them on subset B. After we had determined the best model from several iterations of training and testing, we validated the model on subset C. In the case of the regression models, we determined the equation of the fitted line from the training subset. We then predicted the dependent variables in the test set using this equation. In experiment 2, we reversed the order of training and testing. In this case, we trained on subset B and tested on subset A. Again, we validated the model on subset C. In experiment 3, we combined subset A with subset B. We then chose the best parameter settings and training procedures from the previous two experiments. We trained the data using these settings and procedures. After training, we tested on subset C. With these restrictions, we were able to validate the model with the test set.

4.0 Regression Analysis

4.1 Linear Regression Models

Before we developed the regression models for the individual training subsets, we generated the regression analysis for our reduced model (Hanley's model less the ex post variables) using both SPSS and Microsoft Excel over the full data set. The data in Tables 2 and 3 and Figures 2 and 3 were generated from these packages. To begin, the overall fit of the first regression model looks good. As can be seen from Table 2, the F-test for overall fit and the t-test for the individual variables are statistically significant at $\alpha = 0.05$.

This model also has reasonably low multicolinearity and little autocorrelation. As was

Copyright © 1995 Software Engineering Press

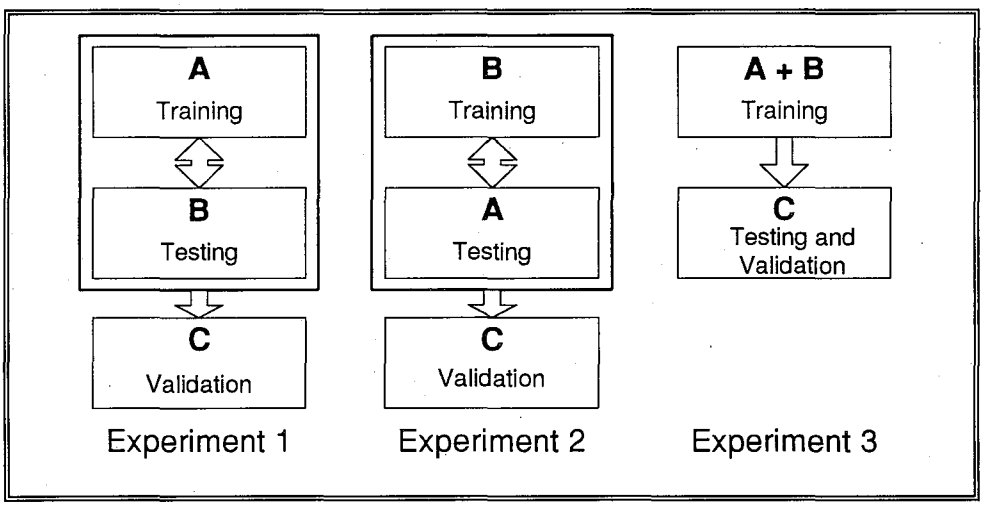


Figure 1. Visualization of Experimental Design

Regression Statistics for Dependent Variable, Initial Returns			
Multiple R	0.4190	Durbin-Watson	1.9495
R Square	0.1756	Mean VIF	1.2043
ADJ. R Square	0.1732		
Standard Error	0.1578		
Observations	1423		

ANOVA

	df	*SS*	*MS*	*F - statistic*
Regression	4	7.5200	1.8800	75.4958
Residual	1418	35.3110	0.0249	
Total	1422	42.8310		

	Coefficients	*Standard Error*	*t - statistic*	*P-value*
Intercept	0.1206	0.0060	20.2669	0.0000
Percent Change	0.3853	0.0304	12.6669	0.0000
Offer Amount	-0.0007	0.0002	-3.0148	0.0026
NASDAQ	0.3772	0.0653	5.7720	0.0000
Average Market	-0.3525	0.1059	-3.3288	0.0009

Key to Abbreviations

Initial Returns: Percent increase in the selling price of an IPO at the end of the first trading day over its offer price

Percent Change: Percent change in the actual offer price from the expected offer price quoted in the preliminary prospectus

Offer Amount: Offer amount

NASDAQ: Percent change in the NASDAQ index from file date to offer date

Average Market: Average market share of the lead underwriters

Table 2. Simple Linear Regression over the Full Data Set

Copyright © 1995 Software Engineering Press

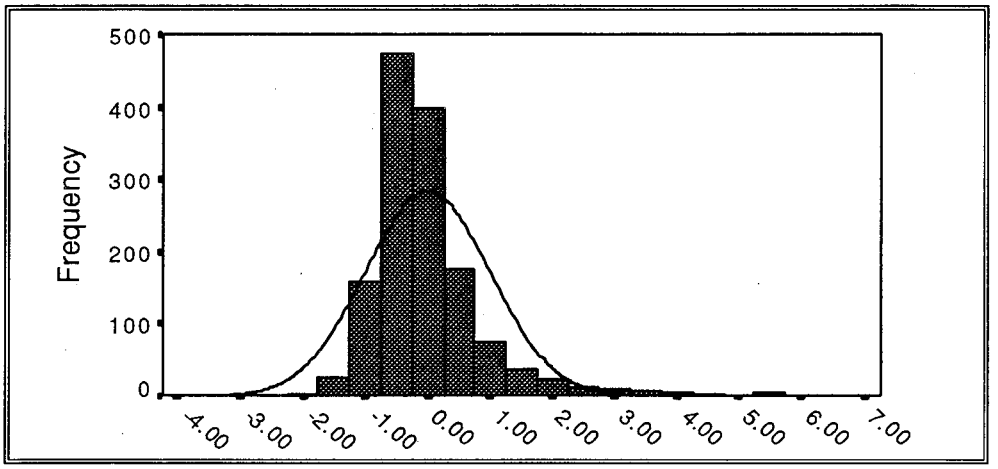


Figure 2. Histogram of Standardized Residuals from the Simple Linear Regression Model

expected, we found that the removal of the ex post variables from the full model, described by Hanley, slightly reduced the adjusted R^2 value. In the first of our regression models, we found that the amount of unexplained variation is quite high. This fact alone makes the use of regression for prediction of the dependent variable problematic.

The simple linear regression model also appears to violate the normality assumption. Figure 2 clearly indicates that the residuals are not normally distributed. In support of this graphical evidence, we determined that there are a total of 30 outlying residuals (2.1% beyond $\pm 3\sigma$). Nineteen of these are beyond $\pm 4\sigma$ from the mean. We could not justify the removal of any of these outliers. If the residuals were normally distributed, the expected percentage of outliers would be less than 0.26%.

4.2 Nonlinear Regression Models

After studying the partial residual plots of the individual variables, we concluded that performing nonlinear transforms on the dependent variable—initial returns—and on the independent variable—offer amount—were appropriate. Using the Box-Cox procedure (for details, see Johnson and Wichern [3], pp. 164-166), we determined that the natural log transformation was appropriate for both. In the case of initial returns, we added 1 to each value to ensure that all of the values were positive before taking the log. We then regressed over the new set of variables. While the R^2 value improved, we discovered that the variable—average market—did not pass the t-test. We attempted a transformation on this variable using the Box-Cox procedure, but the t-statistic did not improve to the extent that we could include average market in the model. Hence, we limited the final regression model to three independent variables.

The resulting model was the best that we tested. The F-statistic improved by 38.8%, little multicolinearity remained in the model, and the adjusted R^2 value improved to over 20.5%. The regression results for this model are given in Table 3. Although the resulting model has improved, the histogram of standardized residuals (Figure 3) indicates that this model still probably violates the normality assumption. Based on this analysis and the low R^2 value, we hope that the neural network models will outperform regression.

4.3 Regression over the Training Subsets

In Table 4, we present a summary of the regression models over the training subsets. While not shown, the F-tests and t-tests in these models were consistent with the full data set models. Overall, these models are similar to the full data set models. In particular, we see that the adjusted R^2 value improves in each model that uses nonlinear transforms. There is, however, one significant difference between these models

Copyright © 1995 Software Engineering Press

Regression Statistics for Dependent Variable, T-Initial Returns			
Multiple R	0.4549	Durbin-Watson	1.9495
R Square	0.2069	Mean VIF	1.0791
Adjusted R Square	0.2052		
Standard Error	0.1235		
Observations	1423		

ANOVA

	df	*SS*	*MS*	*F-statistic*
Regression	3.0000	5.6503	1.8834	123.3890
Residual	1419.0000	21.6599	0.0153	
Total	1422.0000	27.3103		

	Coefficients	*Standard Error*	*t -statistic*	*P-value*
Intercept	0.1453	0.0089	16.2566	0.0000
Percent Change	0.3421	0.0234	14.6494	0.0000
T-Offer Amount	-0.0275	0.0035	-7.7990	0.0000
NASDAQ	0.3410	0.0511	6.6672	0.0000

Key to Abbreviations

T-Initial Returns: Percent increase in the selling price of an IPO at the end of the first trading day over its offer price; transformed with the function ln(Initial Returns + 1)

Percent Change: Percent change in the actual offer price from the expected offer price quoted in the preliminary prospectus

T-Offer Amount: Natural log of the offer amount

NASDAQ: Percent change in the NASDAQ index from file date to offer date

Table 3. Regression over the Full Data Set using Nonlinear Transforms

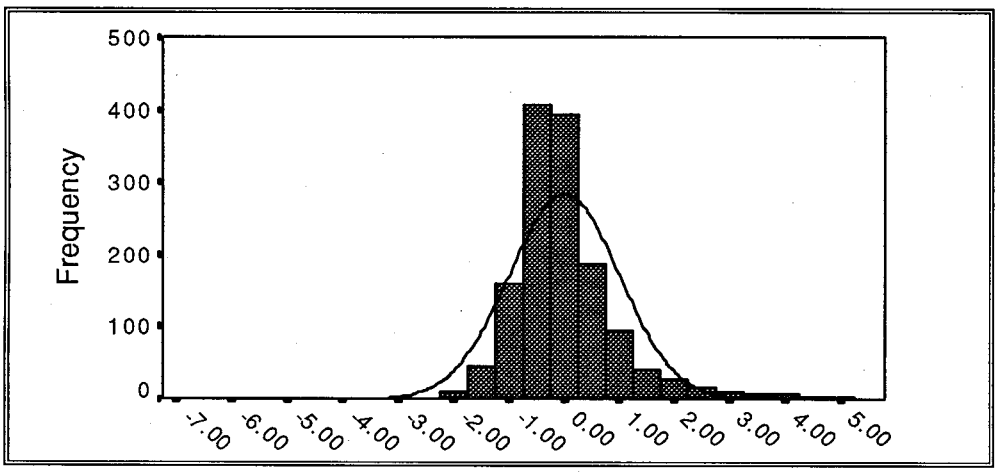


Figure 3. Histogram of Standardized Residuals from the Regression Model with Nonlinear Transforms

Copyright © 1995 Software Engineering Press

Subset A		Subset B		Subset A + B	
Regression Without Transforms		***Regression Without Transforms***		***Regression Without Transforms***	
Multiple R	0.4007	Multiple R	0.4478	Multiple R	0.4234
R Square	0.1606	R Square	0.2005	R Square	0.1793
ADJ. R Square	0.1534	ADJ. R Square	0.1937	ADJ. R Square	0.1758
Standard Error	0.1540	Standard Error	0.1559	Standard Error	0.1547
Observations	474	Observations	473	Observations	947
Regression With Nonlinear Transforms		***Regression With Nonlinear Transforms***		***Regression With Nonlinear Transforms***	
Multiple R	0.4337	Multiple R	0.4934	Multiple R	0.4626
R Square	0.1881	R Square	0.2434	R Square	0.2140
Adjusted R Square	0.1829	Adjusted R Square	0.2386	Adjusted R Square	0.2115
Standard Error	0.1243	Standard Error	0.1185	Standard Error	0.1213
Observations	474	Observations	473	Observations	947

Table 4. Summary of Regression Models over each of the Training Subsets

and the full data set models: the adjusted R^2 value for subset B is much higher than the adjusted R^2 value for the full data set model. Later, we will see that the underlying cause of this departure will have a detrimental effect on the predictive power of neural network models trained on this data.

5.0 Neural Network Models

Table 5 lists the features and training procedures of the neural network models. We found that the models generated by Brainmaker required more fine-tuning than the original code during training to generate acceptable results.

Brainmaker Models	
Architecture:	4 input nodes; 3 hidden nodes; 1 output node
Sigmoid slope*:	0.7 begin; 0.4 end
Learning rate:	0.5 begin; 0.1 end
Momentum:	0.5 begin; 0.1 end
Error threshold†:	0.125 (scaled)begin; 0 end
Stopping criteria:	1500 iterations
Procedure:	Reduce the parameters incrementally in a stepwise fashion when either of the following criteria are met: 1) 200 additional iterations have elapsed; 2) training MAE is not improving
Original Code Models	
Architecture:	4 input nodes; 3 hidden nodes; 1 output node
Sigmoid slope*:	0.3 node 1, 0.4 node 2, 0.5 node 3 begin; increment by 0.2 after 100 iterations
Learning rate:	0.2
Momentum:	0.4
Error threshold†:	0
Stopping criteria:	200 iterations
Procedure:	Increase sigmoid slope after 100 iterations as indicated above
* Sigmoid slope (β): $X_i = \tanh(u_i / \beta)$	
† Error threshold: Only backpropagates error, if absolute error > threshold	

Table 5. Architecture and Training Procedures for the Neural Network Models

Copyright © 1995 Software Engineering Press

This was accomplished by changing several parameters during training to avoid early convergence. In addition, Brainmaker performed better when the models were trained slowly. We accomplished this by slowly reducing the error threshold. In this respect, the original code was superior to Brainmaker, since we could conduct more experiments with the original code in the same amount of computer time.

6.0 Results

Table 6 presents the results of the model testing and validation for each experiment. In each case, the neural network models outperformed both simple linear regression and nonlinear regression. In addition, with the exception of the third experiment, the original code outperformed Brainmaker. Overall, models trained on subset B (experiment 2) did not have the predictive power of the others. As we observed in the discussion of the regression analysis of subset B, the data in this subset had significantly less unexplained variation than the full data set. Based on this observation, we believe that subset B is less representative of the full data set than subset A.

7.0 Conclusions

Average positive initial returns on IPOs is a real-world phenomenon which has puzzled financial experts for a long time. Although some progress has been achieved in understanding the factors which affect this phenomenon, serious effort has not been devoted to building models to predict the initial returns. Prediction of this sort, we believe, should be of importance to large institutions which regularly participate in the IPO market. It can be used, for example, as a guide when deciding the number of IPO shares to be purchased. This paper has taken a positive step in this direction. Using an original code and Brainmaker, we have built models which predict the initial returns on IPOs with a fair degree of accuracy.

As a result of our experiments, we have found that the neural network models consistently outperform the regression models over similar data sets. In addition, the neural network performance is satisfactory in an absolute sense: Our best model had a validated MAE of 0.0940. With this information, we could state that an IPO whose predicted initial return was over 9.40% would probably have a positive initial return. While this will not be true in every case, investors who use this information should be able to reduce their overall risk when using these models as a guide.

	Regression		***NL Regression***		***Brainmaker***		***Original Code***	
Exp.	*Testing Subset*	*MAE*	*Testing Subset*	*MAE*	*Testing Subset*	*MAE*	*Testing Subset*	*MAE*
1	B	0.0981	B	0.0940	B	0.0900	B	0.0889
	C	0.1030	C	0.1007	C	0.0968	C	0.0940
2	A	0.1014	A	0.0961	A	0.0973	A	0.0948
	C	0.1043	C	0.1010	C	0.1001	C	0.0987
3	C	0.1036	C	0.1008	C	0.0970	C	0.0981

Table 6. Comparison of the Predictive Power of the Tested Models

Copyright © 1995 Software Engineering Press

Acknowledgment

We would like to thank Kathleen Weiss Hanley for her input and assistance.

References

[1] Benveniste, Lawrence M. and Paul A. Spindt, 1989, "How investment bankers determine the offer price and allocation of new issues", *Journal of Financial Economics*. 24, 343-361.

[2] Hanley, Kathleen Weiss, 1993, "The underpricing of initial public offerings and the partial adjustment phenomenon", *Journal of Financial Economics*. 34, 231-250.

[3] Johnson, Richard A. and Dean W. Wichern, 1992, *Applied Multivariate Statistical Analysis*. New York: Prentice Hall.

Copyright © 1995 Software Engineering Press

Bank Failure and Categorization - A Neural Network Approach

Prof. Walter Miller
Quinnipiac College
Mt. Carmel Avenue
Hamden, CT 06518

Prof. David T.Cadden
Quinnipiac College
Mt. Carmel Avenue
Hamden, CT 06518

Prof. Vincent Driscoll
Quinnipiac College
Mt. Carmel Avenue
Hamden, CT 06518

INTRODUCTION

Contemporary bankruptcy research examines accounting data for matched-pairs, failed and non-failed, of firms. It employs statistical tests to detect the accounting data that bests discriminates between failed and non-failed firms. Recently, neural networks have been added to the statistical techniques in bankruptcy studies. Bank failure studies, generally, follow a similar research design; however, a bank's health can be categorized in more than two groups. Federal Depositors Insurance Corporation's examiners can place a bank in one of seven groups. This is done on the basis of hard accounting data and evaluator judgment. In addition, local economic conditions and other exogenous factors place a much more important role in bank-failure than corporate bankruptcy studies.

This research examines data on approximately 225 banks, including 46 failed banks, for the period 1987-1992. Two back propagation neural network models will be initially developed. The first model seeks to simply distinguish between failed and non-failed firms, and the second model seeks to match the 225 banks with their FDIC examiners' classification. The results of both models are compared with the results of quadratic discrimination models. Almost all bank-failure and corporate bankruptcy studies that employ neural networks have used back propagation. It is powerful and well understood; however, in bank studies we often wish to consider data which is evaluative, non-exact, that is, *fuzzy data*. This is particularly true in bank studies when dealing with exogenous data, such as multiple indicators of local economic conditions. A third neural network model is examined in this study - one based on Fuzzy ART architecture. This model will be used to classify banks into multiple groups.

MODELS OF BANK FAILURE

Although there was a large pre-World War II literature focusing on bank closings, few studies were able to distinguish operating from closed banks (Meyer and Pifer, 1970). Secrist (1938) argued that single measures - be they ratios or annual changes - could not discriminate between failed and non-failed banks; however, he believed discrimination could be achieved by means of multivariate statistical methods. In a very real sense, most contemporary bank failure studies can be seen as a subset of the classic statistical approaches to bankruptcy prediction. The premiere multivariate study of bankruptcy was Altman 1968 paper. It has become the benchmark against which most other bankruptcy studies are measured. Altman utilized the statistical technique of multiple discriminant analysis and found that bankruptcy could be explained quite completely by using a combination of five (selected from an original list of twenty-two) financial ratios. Linear Discriminant Analysis (LDA) is a statistical technique, developed by Fisher (1936), which is used to classify an observation into one of several a priori groupings dependent on the observation's individual characteristics. It is used primarily to classify and make predictions in problems where the dependent on the observation's individual characteristics. It is used primarily to classify and/or make predictions in problems where the dependent variable appears in qualitative form, e.g. male or female, bankrupt or non-bankrupt. After group classifications have been established, LDA attempts to derive a linear combination of these characteristics which "best" discriminates between the groups. Linear Discriminant Analysis requires certain assumptions about the data: (1) each group follows a multivariate normal distribution; (2) the variance-covariance matrices of the groups are equal; and (3) the prior probabilities are known.

Altman (1977) applied quadratic discriminant analysis to predicting performance in the Savings and Loan industry. This study differed from the classic corporate failure studies in two important points (1) it utilized *three* classification groups - banks with serious problems, temporary problems, and no problems; and (2) the use of trends of accounting ratios.

Pettway and Sinkey (1980) proposed using both accounting data and market information as a means of an early warning system for problem banks.

Copyright © 1995 Software Engineering Press

There have been several attempts to apply expert systems methodologies to the bankruptcy problem and the allied problem of creditor evaluation. Elmer and Borowski (1988) developed an expert system to evaluate the financial health of Savings and Loan (S & L) institutions and predict their failure. Their expert system took publicly available information and produced a single index to measure an institution's health. The rules were derived from the Federal Home Loan Bank Board (FHLBB) Examination Objectives and Procedures Manual and individuals' expertise. This system worked with five ratios drawn from CAMEL framework - CAMEL being an acronym for (C)apital, (A)ssets, (M)anagement, (E)arnings, and (L)iquidity. It, however, excluded the (M)anagement component since that was a subjective, quality measure. The S & L industry is seen as not being homogeneous; this system had the ability to identify thrifts with unusual characteristics and thus improve its own reliability.

The system's single index is a weighted average of scores for the four characteristics - (C)apital, (A)ssets,, (E)arnings, and (L)iquidity measures. The relative importance (the weights) for the four were derived from a poll of S & L presidents. Ten ratios are used to generate the scores for the four characteristics. The production rules treat these ratios either in the context of peer group comparison or with respect to fixed cutoff values. These rules provide sufficient flexibility to allow for changes in the industry.

The author tested this expert system's predictive capability against a logit analysis based on an Altman (1977) study of S & Ls and another study. Their test used 60 matched pairs of failed and non-failed S & Ls form 1986. On that data the two statistical approaches outperformed the expert system by a very slight margin. A second test was conducted. Here the models were used to predict failure 1-6, 7-12, and 13-18 months prior to failure for data for the first half of 1987. In the earliest time period prior to failure the expert system was as good a classifier as the Altman model and better than the second statistical model. As one moved further away from the failure date, the correct classification rates for the three models declined; however, the expert system's declined at a more modest rate. For the period 13-18 months prior to failure, the expert system correctly classified nearly 62% of the sample while Altman's model's value was approximately 48% and the second statistical model's value was 33%. The authors conclude that the expert system approach appears to be robust and that "correlational studies have difficulty adapting to new circumstances and are subject to error due to samples from which they are derived".

A study that compared the performance of a neural network model with a logit regression was Salchenberger, Cinar and Lash's (1993) study of S&L Thrift failures. As with the case of corporate failure, there have been numerous studies of thrift institutions. These studies have used linear discriminant analysis, quadratic discriminant analysis, logit, and probit. Salchenberger et. al. drew upon these prior studies to select an initial list of 29 financial variables. These were reduced down to five by means of stepwise regression. The training data consisted of 100 failed S and Ls, and 100 non-failed S and Ls. These were matched by both asset size and geographical region. They used, in effect, four holdout sample which consisted of matched pairs of thrifts. The first three sample consisted of failed and non-failed institutions 6, 12, and 18 months prior to failure. The total sample sizes were 116, 94, and 48, respectively. The fourth sample consisted of 75 failures matched with 329 non-failures. This fourth holdout sample was designed to more accurately represent the proportions of failed to non-failed institutions. They used a back propagation neural network with one hidden layer which had three nodes. In addition, a logit model was run on the initial training set. For both the logit and neural network models, two cutoff points (.5 and .2) were used. As previously mentioned, the lower cutoff point reduces the chance of a Type I error. For the training set and the 18 month holdout sample, the neural network statistically outperformed the logit model in forecasting failures. For the training set, he neural network model was also more robust when it came to lowering the cutoff point to .2; misclassifying fewer number of non-failures. For the fourth holdout sample, the neural network, again, was statistically superior in classifying failed institutions and non-failed institutions when the cutoff point is equal to .2. The authors conclude that the neural network model yield more useful results than the logit model, particularly when the data is reflective of the total population of thrift institutions.

Tam and Kiang (1992) have published two studies in which they applied neural networks to the study of commercial bank failure. The latter is perhaps the most comprehensive study in comparing neural network methodology to alternative approaches. In it they compare a neural network model's performance with a linear discriminant model, a logistic model, the ID3 algorithm, and the k Nearest Neighbor (kNN) approach. This last

Copyright © 1995 Software Engineering Press

approach is a non-parametric classification technique. It does not have any requirement for functional form nor does it assume normality in the distributions. The data was collected for the period 1985-87 and consisted of 59 failed and 59 non-failed banks. They were matched not only on the basis of assets but also on charter type and number of branches. Nineteen ratios, drawn from prior studies, were selected for use in this research. Although 15 of the 19 ratios were not normal in their distribution, they were used "as is" since attempts at transformation did not produce normal distributions. Tam and Kiang used two back propagation architectures - one with no hidden layer and another with one hidden layer which contained 10 nodes. They modified the learning function to consider both the differing probabilities for failure and non-failure and the differing costs of misclassification. The study considered two probabilities for failure and eight misclassification costs. The models were tested on data one and two years prior to failure.

For the training set, one year prior to failure the neural network with the hidden layer outperformed all other approaches; however, two years prior to failure discriminant analysis had the lowest total misclassification rate followed by the hidden layer neural network. Both neural networks had lower substitution risks (expected cost of misclassification) than the discriminant analysis across all combinations. The neural network with the hidden layer tended to outperform the two layer neural network.

The models were tested on a holdout sample of 44 paired failed and non-failed banks. The neural net with the hidden layer the best overall classifier one year prior to failure while the logit model scored best two years prior to failure with the hidden layer neural network coming in second. Since the results for the models for the training and holdout sets were inconsistent with regard to relative accuracy for the two time periods, Tam and Kiang used a jackknife method of estimation. Utilizing this method, the hidden layer neural network produced smaller total misclassification rates, for both time periods, than the other models. The neural network with no hidden layers tended to perform at a rate comparable to the discriminant function model.

These last studies clearly indicate the potential benefit to be derived from using neural networks in the study of bank failure.

DATA AND RESEARCH DESIGN

The first phase of this study will be data collection. Presently, the authors are planning a full study of all banks in the New England region; this paper will discuss the results obtained for a subset, namely, banks in the state of Massachusetts. Data was obtained from several sources - state published data bases; *Sheshunoffs*; *The Bank Quarterly: Ratings and* Analysis; interviews with bank executives and FDIC examiners, and a Freedom of Information request of the FDIC. Financial data was collected for each failed instituion for the three years prior to its failure. Failed banks were matched to non-failed banks, initially, on the basis of asset size and bank type - state, federal or national. In addition to controlling the selection of banks by size, we stratified by SMSA location. For each of the banks a set of financial ratios will be computed. Based upon prior studies, the research will compute twenty-six financial ratios . We also computed for the twenty-six ratios their annual rates of change. These variables were examined to determine the degree of correlation amongst themselves and T-tests and factor analysis were conducted to examine which variables were most significant, in terms of differentiating between failed and non-failed banks. This was done to reduce the data set.

In addition to these ratios, exogenous economic data for each SMSA location was gathered. This exogenous data and elements of the evaluations of the FDIC examiners can, at times, be imprecise. As an example, the designation of the local economy or the bank's management may be classified as *god, fair,* or *poor*. Such concepts need to be processed into its fuzzy logic equivalent. Fuzzy ART neural network, which will be reported in a subsequent paper, will allow for a quantification of such imprecise notions.

The authors employed four neural network back-propagation architectures designed to predict failed from non-failed banks. The architcture of the first model (NN1) consisted of ten nodes in one hidden layer; the second model (NN2) had twenty nodes in one hidden layer; the third model (NN3) had five nodes in the first hidden layer and five nodes in a second hidden layer; and the fourth model (NN4) had ten nodes in the first hidden layer and ten nodes in a second hidden layer. Based on the aforementioned statistical analysis, the data set was reduced to eleven financial ratios, six rates of change variables and the type of institution. The forty-six failed banks were segmented into two groups - thirty-one were used in a training sample and the

Copyright © 1995 Software Engineering Press

remaining fifteen were assigned to the test sample. Care was taken to assign failed banks to either training or testing sample based upon the year of failure. The training consisted of thirty one nonfailed banks while the testing sample consisted of 210 non-failed banks. This last sample represents nearly all the non-failed banks in the state. In addition to the neural network model, the data was tested using upon quadratic discriminant analysis (QDF). This classification scheme was used because of evidence of its superiority to linear discriminant analysis in bankruptcy studies (Mahmood and Lawrence, 1987). However, the authors plan to test the data for deviations from multivariate normality to which QFD analysis is sensitive. We also plan to Probit and Logit analyses to evaluate the data. The results of the QFD and back-propagation neural network are presented in Table 1. The authors plan to extend this analysis for the Massachusett's data by employing a Lachenbruch validation test design.

TABLE 1.

	Training Set				*Test Set*			
	Failed		*Non-Failed*		*Failed*		*Non-Failed*	
Model	*Actual*	*Predicted*	*Actual*	*Predicted*	*Actual*	*Predicted*	*Actual*	*Predicted*
NN1	31	29	31	30	15	10	210	196
NN2	31	30	31	31	15	12	210	201
NN3	31	28	31	30	15	12	210	187
NN4	31	31	31	31	15	15	210	206
QDF	31	31	30	30	15	10	210	195

The results indicate that the neural network with twenty nodes in two hidden layers outperforms all competing formulations.

Currently, the authors are examining how well neural networks can match the five category classification scheme used by FDIC examiners. This stage of the research will require a close linkage with such examiners in order to determine what specific variables they utilize.

Following that work, the next set of neural network models will incorporate fuzzy data that will consist of examinors evaluation of bank management and economic conditions.

A complete bibliography is available from the authors upon request.

Copyright © 1995 Software Engineering Press

Paper Session: Derivatives

Chair: Michel Benaroch, Syracuse University

A Genetic-based Approach to the Analysis of Derivative Securities

Sergio SCANDIZZO

Laboratorio di Urbanistica e Pianificazione Territoriale Università Federico II di Napoli
Via Toledo 402, 80134 Napoli, Italy
Phone:+39-81-5521011 Fax:+39-81-5513495 Email: scandizz@vm.cised.unina.it

Key Words: *derivative product, genetic algorithm, hedging, replicating portfolio, design of securities*

Abstract

In this paper the task of developing a genetic-based system for the building of optimal hedging strategies by means of derivative securities is addressed. The analytical potential of a genetic model as well as the representation issues connected with this approach are discussed. An attempt has been made to integrate the genetic maximization procedure with information available on the underlying price distribution, using a probabilistic fitness function. Finally some implementation details are considered. A genetic based model is likely to be both a flexible and an efficient tool of analysis, when problems are characterized by uncertain knowledge and complexly structured solution spaces.

1. Introduction

We call *derivative product* a security whose value depends on one or more other variables which are called the *bases* of the product (Ingersoll, [7]). During the last two decades, the development of such financial instruments has been enormous driven by the search for arbitrage opportunities as well as by the need for more efficient hedging strategies, arisen with the increased volatility of interest rates, exchange rates, and commodity prices.

As a consequence, derivatives are also object of a huge number of studies, addressing first of all the problem of valuing and pricing securities under the most various market circumstances. The task of building derivatives with desired characteristics is treated in the wider framework of financial innovation and design of securities. A key feature both in building and in valuing derivative products consists in the identification of a financial structure that mimics the characteristics of the desired product starting from the basis and the risk-free asset.

Both pricing and optimal hedging problems have been addressed in rigorous mathematical contexts. Modern derivative pricing theory relies on the seminal paper by Fisher Black and Myron Scholes [2]. The design of optimal hedging instruments under given exposure conditions has been studied by Allen and Gale [1] in a firms' value-maximizing context and by Duffie and Jackson [5], who studied the pareto-optimal allocation of resources connected with equilibrium conditions in futures markets.

At the same time, a considerable research effort has been undertaken to develop tools of analysis of economic systems, which were capable to treat dynamic, highly non linear, complex phenomena without relying on unrealistic, oversimplifying assumptions. Among these tools, some models borrowed from mathematics, physics, biology, and computer science are currently at the core of modern, non-traditional financial analysis. Chaos theory, fuzzy logic, artificial neural networks, and genetic algorithms are receiving increasing interests from students and are often synthetically referred to as the field of *soft computing*.

Being concerned mainly with non stable and far from equilibrium systems, soft computing tools are naturally well suited to address the topics of hedging strategies and risk management.

A similar approach was undertaken by Chorafas [3], who discussed the application of a genetic algorithm to the problem of managing off-balance sheet operations, representing combinations of financial instruments by means of *chromosomes* built up by a random-based process.

In this paper we will follow a building block approach (Smithson, [11]) to the construction of such "replicating portfolios", and we will discuss how to use the framework provided by so-called genetic algorithms (see Goldberg, [6]) to address the task of building derivative securities.

2. The Problem

The four fundamentals off-balance-sheet instruments issued to manage financial risk are: forward contract, future contract, swap contract and option contract. In fact, one of these instruments may be issued to hedge a firm's risk profile, more than one of these may be combined to suit with a

particular exposure or one or more may be issued together with a debt instrument to build up a so-called *hybrid security*. These four building blocks can thus be assembled to give customized solutions to hedging problems, that is to optimization problems (minimizing the expected cost of funds/maximizing the expected rate of return), given the probability distribution of the underlying price.

In turn, each of the four blocks is made up of the same fundamental components: the exercise price (the price at which the basic asset will be traded); the time to maturity; the payoff per period; the payoff at maturity; the cost of the contract. Managing these five variables we can specify each of our building blocks as well as a debt instrument and, thus, by assembling several packages of these five fundamental variables we can build virtually every kind of hybrid instrument.

Let P_t be the underlying price in period t, P* be the strike price, T the number of (elementary) periods to maturity, Π the payoff per period before maturity, Π^* the payoff at maturity and C the cost of the contract. Given the risk exposure of a firm to a particular price P, in the form of a functional relationship $\Pi_t = f(P^*, P_t)$, we wish to find, if possible, a string of the following form

P*	T	Π	Π^*	C	...	P*	T	Π	Π^*	C

which gives us the best available hedge without giving up all potential gains or, alternatively, gives the same payoff profile of an existing product at a lower cost (or a higher expected payoff at the same cost).

The second possibility is ruled out in absence of arbitrage opportunities, but even the first goal has no non-trivial solution if nothing can be said about the probability distribution of the underlying price.

3. The Genetic Approach

Genetic algorithms are computational devices designed to explore the solution space of an optimization problem by generating successive "populations" of solutions according to the laws of genetics and natural selection. By processing string structures through three fundamentals operators - Reproduction, Crossover, Mutation - and following probabilistic transition rules, genetic algorithms find the maximum of a function codified in the strings and representing the fitness, as a biologist would say, or the payoff, as an economist would say, of each string.

The applications of genetic algorithms are not confined to the area of optimization. The simulation of systems characterized by life-like behavior, often called artificial evolution, was addressed by Collins [4] and by Scandizzo [9], in the framework of evolutionary economic models.

It is interesting to note that the theoretical foundations of genetic algorithms rely on a "building block argument" which in some way recalls the building block approach discussed in the previous section. The main theorem about genetic algorithms is the *schema theorem* [6], which states that, in subsequent generations of a genetic algorithm, short, low order, above average schemata receive exponentially increasing trials. A schema is a string where, in one or more position, the original symbol was substituted with a wildcard mark (*) and represents all the strings that match it in all positions other than the wildcard. For example, the following schema

1***1**

represents all the strings with a *1* in the first and in the fifth positions, no matter what symbols are carried in the remaining positions.

A consequence of the schema theorem is the so-called building block hypothesis, that can be stated as follows [8]: "A genetic algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks."

Without taking the similarity too far, we can argue that genetic algorithms are computational devices particularly well suited to solve optimization problems whose solution spaces exhibits

Suppose the value V of a firm is exposed to changes in price P as in Figure 1.

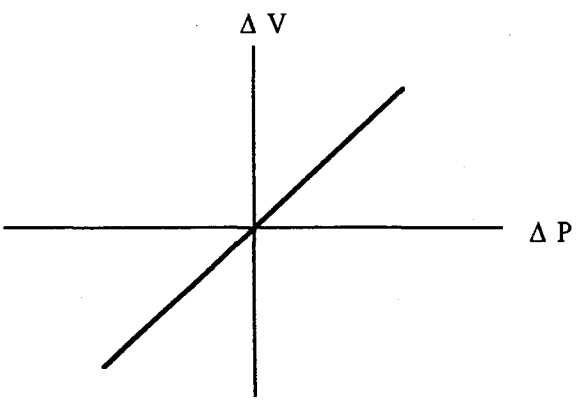


Figure 1

A perfect risk hedging is possible issuing a future or a forward on P, but this would eliminate all

possible gains from a rising of P. For example, if the probability distribution of P in the next period is such that there is a high probability of a substantial decrease in P, a small probability of a little increase, and virtually no possibility of significant increases, then we can combine a future contract with two option contracts, a call and a put, with exercise prices set such that selling the call yields more than what must be paid to buy the put[1].

Figure 2 shows the resulting risk profile.

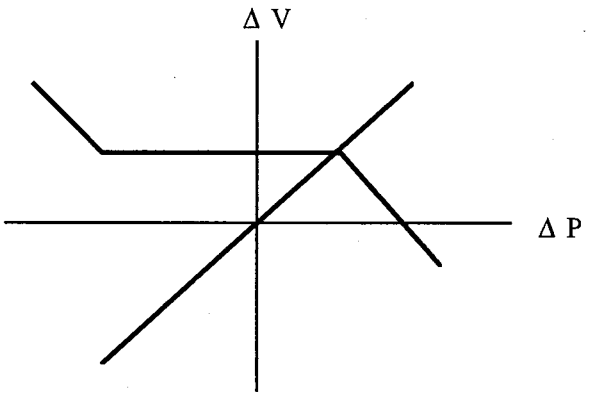


Figure 2

The solution may be represented in terms of our five basic variables as follows:

Forward
$P_t=P_t;P^*=P'^*;T=T';\Pi=0;\Pi^*=P_t-P'^*;C=C'$.

Sell a call
$P_t=P_t;P^*=P''^*;T=T';\Pi=0;$
$\Pi^*=-\max\{0;P''^*-P_t\};\ C=C''$.

Buy a put
$P_t=P_t;P^*=P'''^*;T=T';\Pi=0;$
$\Pi^*=\max\{P_t-P'''^*\}\ ;C=-C'''$.

where $C''>C'''$.

We have presented a relatively simple problem and a possible solution, given a synthetic description of our expectations about the underlying price, but neither we followed a rigorous maximization procedure nor we fully explored the solutions space, so we cannot be sure there isn't a different combination of our basic building blocks yielding a higher net payoff.

We suggest that a way to explore systematically this kind of solutions to hedging problems may be found by customizing genetic algorithms to process strings representing combinations of the basic financial instruments, and to consider a fitness function whose value depends on the payoff of the derivative product represented in the string.

An initial population may be built up by creating a set of basic instruments - each one defined by five values for variables P*, T, Π, Π*, C, and by randomly generating a large number of strings combining these basic blocks.

The resulting population will be characterized by an average fitness and by a certain level of genetic variance. The fitness function should be based on the expected net payoff of the solution represented by each string, on the inherent exposure as well as on information about future values of the underlying price. If a sufficient level of initial variance is available, the application of genetic operators - reproduction, crossover, and mutation - will lead the population to converge on a string with the maximum attainable value of the fitness function.

The strings on which the algorithm will converge will be the best performing solution to the problem represented by the price exposure together with the price distribution.

4. Implementation Notes

To describe our genetic algorithm we must specify the nature of populations processed, the selection mechanism, the operators used to generate new populations, the fitness function, how it is coded in strings, and how can it be calculated.

Population

The population consists of strings of the form

P*	T	Π	Π*	C	...	P*	T	Π	Π*	C

whose positions are interpreted as the variables discussed in blocks of five from left to right.

Selection mechanism

The classical roulette wheel selection mechanism (Goldberg [2]) chooses at random individuals for reproduction with a probability which is larger the larger the fitness of the individual. Because we may wish to explore more than one solution, other selection strategies may also be used that allows the algorithm to converge to more than a single genotype at the same time. It has been shown (Collins [1]) that, in spatially structured populations, selection models linking the probability of mating to distance among

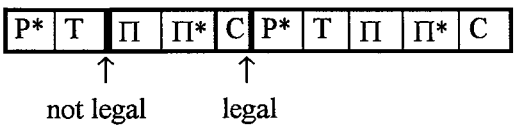
[1] For a wider discussion of this example see Smithson [5].

Copyright © 1995 Software Engineering Press

individuals (non-panmictic strategies) is likely to exhibit superior performance.

Operators and the fitness function

In the implementation of the familiar operators of reproduction, crossover, and mutation, we must take into account that non every combination of values in the basic blocks will result in a legal solution to our problem. Thus the operations of crossover and mutation are likely to be executed in a *positional* manner, that is cutting a string for crossover only between a block and another as the following string shows.



The evaluation of the fitness function, which drives the selection mechanism, depends on a variable, the underlying price P, whose values are unknown.

We will use a fitness function of the form

$$\sum_{j=1}^{n} \sum_{t=1}^{T} \frac{\Pi_{jt}}{(1+r)^{t}} - C_j$$

where *n* is the number of simple derivatives (building blocks) simultaneously represented in the string, *r* is an appropriate discount rate, and the $\Pi jt=f_j(P_j^*, P_t)$ are calculated generating at each step a random value for Pt, sampled from a distribution that synthesizes our knowledge about the future trend of the underlying price.

5. Conclusions

I have presented the general structure of a system conceived to treat risk management problems by building proper combinations of the basic derivative securities. The main characteristics of such a system are flexibility, the possibility of including available information on the underlying price, and the possibility of systematically exploring and comparing different complex solutions.

The general structure of the system proposed is summarized in Figure 3.

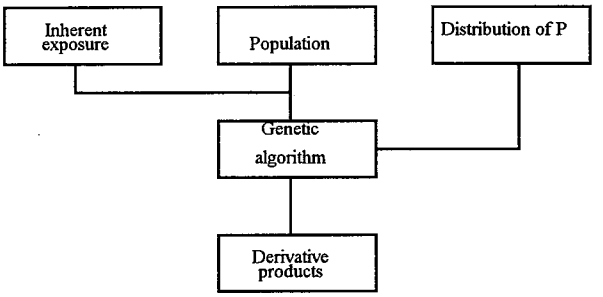


Figure 3

Such a system may be used both for optimization purposes and to perform simulation experiments under different conditions.

This paper is just a first attempt to explore the potential of genetic-based systems to handle risk management problems systematically, and a great number of possible applications is probably still to be discovered.

Two topics, however, are of immediate interest: to show whether the schema theorem still holds for a stochastic fitness function as the one suggested, and to test the system on different computing platform to assess its computational complexity as well as its effective power.

REFERENCES

[1] Allen F. and Gale D., "Arbitrage, Short Sales and Financial Innovation", Econometrica 59, 1041-68, 1991.

[2] Black F. and Scholes M., "The Pricing of Options and Corporate Liabilities",

[3] Chorafas D.N, "Chaos Theory in the Financial Market", Probus, Chicago, 1994.

[4] Collins R.J., "Studies in Artificial Evolution", Ph. D. Dissertation, University of California, Los Angeles, 1992.

[5] Duffie D and Jackson M.O., "Optimal Innovation of Future Contracts", Review of Financial Studies 2, 275-96, 1989.

[6] Goldberg D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, Reading, Mass., 1989.

[7] Ingersoll J., "Theory of Financial Decision Making", Rowman & Littlefield, Totowa, N.J., 1987.

[8] Michalewicz Z., "Genetic Algorithms + Data Structures = Evolution Programs", Springer Verlag, 1992.

[9] Scandizzo S., "Using Genetic Algorithms and Artificial Neural Networks in Evolutionary Economic Models", Proceedings of the 3rd International Conference on Fuzzy Logic, Neural

Copyright © 1995 Software Engineering Press

Nets and Soft Computing, Iizuka, Japan, August 1-7, 1994.

[10] Smith C.W.Jr. and Smithson C.W., "On the determinants of Corporate Hedging", Journal of Finance, vol.448 N.2, March 1993.

[11] Smithson C.W., "A LEGO Approach to Financial Engineering: An Introduction to Forwards, Futures, Options and Swaps", Midland Corporate Financial Journal 4, Winter 1987.

Copyright © 1995 Software Engineering Press

Forecasting Currency Futures Using Recurrent Neural Networks

Dr. Paolo Tenti
A & A Financial Management
Via Peri, 21
6900 Lugano - Switzerland
Tel.+41-91-229005 Fax+41-91-229077

Abstract

This paper proposes the use of recurrent neural networks in order to forecast currency futures. Recurrent networks, in which activity patterns pass through the network more than once before they generate an output pattern, can learn extremely complex temporal sequences. Selected recurrent architectures are compared in terms of prediction accuracy. A trading strategy is then devised and optimized. The profitability of the trading strategy, taking into account transaction costs, is shown for the different architectures. The methods described here, which have obtained promising results in real-time trading, are applicable to other markets.

1. Introduction

For years opposing views existed between the trading and academic communities about the statistical properties of foreign exchange rates. Traders considered exchange rates to have persistent trends which permitted mechanical trading systems (systematic methods for repeatedly buying and selling based on past prices and technical indicators) to consistently generate profits with relatively low risk. Researchers, on the other hand, presented evidence supporting the *random walk hypothesis* in the behavior of exchange rates. When prices follow a random walk the only relevant information in the historical series of prices, for traders, is the most recent price. The presence of a random walk in a currency market is a sufficient, but not necessary, condition to the existence of a weak form of the *efficient market hypothesis*, i.e. that past movements in exchange rates could not be used to foretell future movements.

While there is no final word on the diatribe between practitioners and academics about the efficiency of currency markets, the prevalent view in economic literature that exchange rates follow a random walk has been dismissed by recent empirical work. There is now strong evidence that exchange rates returns are not independent of past changes. Before the advent of nonlinear dynamics, statistical tests for the random walk were usually conducted by verifying that there was no linear dependence, or that autocorrelation coefficients were not statistically different from zero. However, the lack of linear dependence did not rule out nonlinear dependence, the presence of which would negate the *random walk hypothesis*. Therefore, many tests were often inappropriate and some conclusions were questionable. Recent evidence has clearly shown that while there is little linear dependence, the null hypothesis of independence can be strongly rejected, demonstrating the existence of nonlinearities in exchange rates. [3,5,17]

The problem of predicting exchange rates, characterized by nonlinearities and high noise, seems to defy complex methods. Currency markets are to some extent still an enigma for economic theory. Sophisticated empirical econometric models using fundamental data to predict low-frequency (monthly or lower) exchange rates changes are characterized by parameter instability and poor forecast performance. Even recent nonlinear extensions of existing models do not provide any

Copyright © 1995 Software Engineering Press

improvements in the ability to forecast currency movements. [13]

With respect to the issue of the weak form of efficiency of the exchange rates markets it would seem very difficult to obtain positive results using only high-frequency (weekly, daily or even intra-day) past prices. Available evidence suggests that even nonlinear non parametric statistical methods have yet to show positive results in out-of-sample prediction.[1] Also, although the evidence of nonlinearities in exchange rates is compatible with the existence of low-dimensional chaos, only mixed results of chaotic behavior (and therefore short term predictability) have been obtained.[2]

Surprisingly, though, there are anomalies in the behavior of the foreign exchange markets that cannot be explained under the existing paradigm of market efficiency. New evidence has emerged, which reinforces previous tests, on the profitability and statistical significance of mechanical trading systems in currency markets, and negates the weak form of efficiency. [2,10,11] It seems that technical trading rules are able to pick up some of the hidden patterns in the inherently nonlinear price series. Mechanical trading systems appear to be the only approach that has demonstrated some validity. It should be used as a base on which to build on.

2. ANN as a Forecasting Tool

The potential advantages and limitations of an Artificial Neural Network (ANN), and in particular of a multilayer feedforward neural network, over other statistical methods or expert systems are well known. ANNs are universal function approximators, and being inherently nonlinear are notoriously good at detecting nonlinearities, but suffer from long training time and a very high number of alternatives as far as architectures and parameters go; they are also prone to overfitting data. Another common critique that is made about ANNs is that they are "black boxes", more difficult to decipher than traditional time series econometric models or expert systems. Critics go on to say that knowledge of the value of the weights and biases in the network gives, at best, only a rough idea of the functional relationships. Thus, even if ANNs are based on causally related data, the resulting model may not give a great amount of insight into the strength and nature of the relationships within it. This elusiveness of ANNs is the price to be paid in return for their being model-free estimators.

However, even when using econometric models it is an accepted fact that one cannot be sure about the direction of causal effects among different variables. As Fischer Black says [1]: "The trouble with econometric models is that, while they purport to tell us something about causal relations between variables, they almost invariably rely on correlations to imply causation. While correlations can tell us something about how variables are statistically related, they tell us little about how they are causally related." Furthermore: "Understanding a model helps only in that it may give us confidence that the coefficients will be stable through time." Thus, econometric models share the same ambiguity in respect to their interpretation as do ANNs.

Recurrent neural networks (RNNs), in which the input layer's activity patterns pass through the network more than once before generating a new output pattern, can learn extremely complex temporal patterns. Several researchers

[1] For example, Diebold et al. [6] used locally-weighted regression on weekly series of 10 OECD spot exchange rates, without being able to improve upon a simple random walk in out-of-sample prediction. The random walk had the lowest Mean Squared Error in at least 5 out of 10 cases for all the models they considered.

[2] De Grauwe et al. [5] applied some of the available techniques (rescaled range analysis and time delay method) to daily exchange rates and found only weak support for the occurrence of chaotic structure for the $/JY and BP/$. They found no evidence of chaotic behavior for the $/DM, in the period 1971-1990. In other tests, [6] found no evidence in favor of low-dimensional "regular" chaotic dynamics.

Copyright © 1995 Software Engineering Press

have confirmed the superiority of RNNs over feedforward networks when performing nonlinear time series prediction. [4,12] Recurrent architecture proves to be superior to the windowing technique of overlapping snapshots of data which is used with standard backpropagation. In fact, by introducing time-lagged model components, RNNs may respond to the same input pattern in a different way at different times, depending on the sequence of inputs. The appropriate response at a particular point in time could depend not only on the current input, but potentially on all previous inputs. The main disadvantage of RNNs is that they require substantially more connections, and more memory in simulation, than standard backpropagation networks. RNNs can yield good results because of the rough repetition of similar patterns present in exchange rate time series. These regular but subtle sequences can provide beneficial forecastability.

3. Empirical Design

The experiments, in order to be useful and applicable to real-time trading, must create conditions which are as close as possible to reality. Therefore, one must take into account, when using spot exchange rates, the interest rate differential among the currencies in question. Unfortunately, this crucial criteria is often overlooked. There will always be a difference between "paper" profits and real profits: the objective is to minimize it. By using forward rates or currency futures it is possible to overcome this problem because they already include a premium or a discount due to the differences in interest rates. Any trading system based solely on spot exchange rates is just an approximation, because it comes short of dealing with the problem of interest rate differentials.[3] The set of data used in the experiments consisted of IMM currency future (DM, SF, BP and JY) Opening and Closing prices from Jan-1990 to Dec-1994.

3.1.Adjusted Price Series Choosing the appropriate price series for currency futures is hardly a trivial matter and is the first step in building a trading system. Using individual contracts complicates the task of training and testing the system. The training and testing usually require a price data history that is much longer than the typical liquid trading period for an individual contract. Furthermore, the simultaneous use of individual contracts is difficult because it is necessary to combine a large number of individual results for each contract as well as dealing with possible divergences of trading signals when switching from the expiring contract to the next one. The commonly proposed solution is to create a single continuous price series by using the nearest futures prices, with a jump to the prices of the successive contract made at the beginning of the month or at a specified number of trading days before expiration. The fatal distortion of this system is that there could be significant price gaps created in the series at the roll-over dates, between the expiring and the subsequent contracts. The nearest futures series will create illusory price moves at the transition points, distorting both training and testing activities. In addition, the nearest futures series does not allow direct calculation of the profitability of a trading system. The solution adopted here is to use spread-adjusted continuous price series, by which, except for the most recent contract in the series, prices are adjusted by a constant that compensates for price differences which exist at roll-over dates. [16] This method alters the prices of the future contracts prior to the most recent one, but maintains identical price relationships, thereby avoiding the distortions mentioned above.

[3] It would be possible to use spot exchange rates by taking into account overnight interest rates on spot interbank deposits for returns calculations, but at the price of more approximations. In addition, the spread between bids and offers will tend to be greater for forward than for spot rates, increasing as the maturities grow longer.

Copyright © 1995 Software Engineering Press

The transition between contracts was performed seven days before expiration. Several sets of data were prepared: each one contained a training set of 424 consecutive trading days, a test set of 100 consecutive trading days (which begins the day after the training set ends), and a validation set of 100 consecutive trading days (which begins the day after the test set ends).

3.2.Generalization RNNs are predisposed, as are standard backpropagation networks, to overfit training data. Rather than learning the fundamental structure of the training set, which would enable them to generalize adequately, they learn insignificant details of individual cases. This problem is generated by two conflicting purposes of ANNs: they have to be as general as possible so that they learn a broad range of problems and yet they need to perform well in out-of-sample tests, on examples not previously seen. There are two approaches to the overfitting problem. The first one is to train the model on the training set and to evaluate the model's performance on the test set. The second approach is to use one of the many network pruning algorithms [19] to reduce the network size, thereby limiting the number of hidden neurodes and hence the number of parameters to be estimated. The solution I adopted is based on a parsimonious choice of the number of hidden neurodes as suggested by the generalization capability of the network on the test set. In this procedure I trained the network until convergence, observed the point at which the test set error began to rise, and then restored the network weights at the iteration cycle where the test set error was minimum. How well the network generalized was deduced by analyzing its performance on the validation set, and not on the test set as this was used to decide when to stop training, and therefore introduced a dangerous bias in the evaluation.

3.3.Prediction accuracy and profitability The ultimate goal of the experiments is to create a trading system, a set of interrelated rules to enter and exit the market, that produces profits. While accuracy is related to profitability, the trading system should not be evaluated using only standard statistical error measures (Mean Square Error and the like). As an example, a trading system might consistently miss a large number of small moves but correctly forecast a small number of large moves. Therefore, the researcher must take into account the out-of-sample profitability of the system, as well as its forecasting accuracy, when choosing the neural architecture, activation functions, data sets, and forecast horizon. To reiterate the concept, prediction accuracy is not the goal in itself, and it should not be used as the guiding selection criteria in the tests. While this simple concept is part of the wealth of knowledge of mechanical traders [16], it is rarely considered in tests undertaken by academics.

3.4. Outputs and Inputs Choosing the kind of outputs to be forecasted is an important decision. The most common options are:

- actual price values,
- first differences of prices,
- returns,
- binary signals, such as -1 short, 1 long.

As currency futures are non stationary, it is better to analyze price changes in terms of compound return: $r_{ct} = \log(f_t) - \log(f_{t-1})$. Experiments, conducted with compound returns, have shown that the forecasting horizon must remain very short to obtain good results. One of the problems in forecasting actual prices is that activation functions tend to emphasize the importance of intermediate output values, so that the range of predicted values is compressed with respect to target values. Solutions range from using special forms of normalization to linear activation functions.

Another critical point is to identify the appropriate set of inputs relevant for the RNN architecture and for the chosen output. In

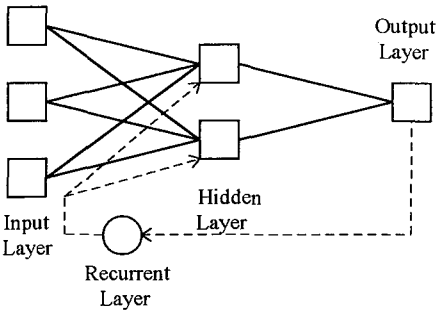
Copyright © 1995 Software Engineering Press

particular, the inputs should be adapted to the "needs" of RNNs: they should have a temporal structure and should not be too numerous.[4] An analysis of alternative sets of inputs based on transformations of the set of data, drawing from the vast base of technical indicators was performed. This selection was based on previous work performed on the optimal choice of parameters of different technical indicators and on their combined use in trading systems.[18] Inputs were normalized to zero mean and two standard deviations for all three data sets. The output was normalized at zero mean and three standard deviations.

4. Learning

Prediction using a RNN involves the construction of two separate components: one or more recurrent layers which provide the temporal context, usually referred to as short-term memory, and a predictor, usually the feedforward part of the network. The short term memory retains features of the input series relevant to the prediction task, and captures the network's prior activation history. The tests were performed with three variations of RNNs. They belong to the RNN family known as local feedback networks, where only local connections are activated. The rationale is that instead of learning with complex, fully connected recurrent architectures, redundant connections should be eliminated in order to significantly increase the network's generalization capability. The first architecture used is similar to that developed by Jordan [8], known as sequential network. The network has one hidden and one recurrent layer. The output layer is fed back into the hidden layer, by means of the recurrent layer, showing resulting outputs of previous patterns (Figure 1)[5]. The recurrent neurode allows the network's hidden neurodes to see their own previous output, so that their subsequent behavior can be shaped by previous responses. The recurrent layer is what gives the network its memory. Following the taxonomy proposed by Mozer [14], which distinguishes between the short term memory's content and form. The version I used was characterized by output-exponential memory.

Figure 1. *Recurrent Backpropagation with Output Layer Feedback Link*
(memory: output-exponential)



With respect to the form of the memory, the use of an exponential trace memory acts on the series of inputs x(1),..., x(t) creating a state representation $[\bar{x}_1(t), \bar{x}_2(t), \ldots, \bar{x}_t(t)]$, where each $\bar{x}_i(t)$ is related to the input sequence by the function e_i:

$$\bar{x}_i(t) = \sum_{\tau=1}^{t} e_i(t-\tau)x(\tau)$$

where $e_i(t) = (1-\mu_i)\mu_i^t$ with $0 < \mu_i < 1$.
An important property of exponential trace memories is that $\bar{x}_i(t)$ can be calculated incrementally:

$$\bar{x}_i(t) = (1-\mu_i)\mathrm{x}_i(\mathrm{t}) + \mu_i \bar{x}_i(t-1).$$

These memories can then be seen as exponentially weighted moving averages of past inputs. The exponential memory, used also for the other two versions, makes the strength

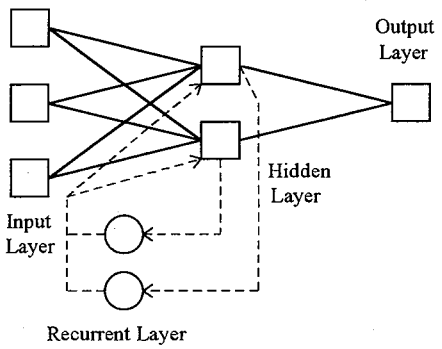
[4]This implies that comparisons among standard back propagation and RNN should not be based on the same set of inputs, but on comparing "best practice" with "best practice".
[5]Self-loops of recurrent neurodes are not shown in this and the following figures.

Copyright © 1995 Software Engineering Press

of more distant inputs decay exponentially. The rate of decay is governed by μ_i.

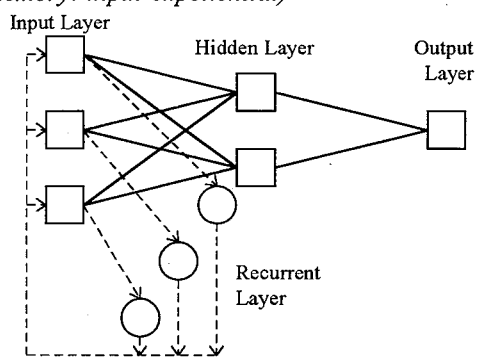
In the second version (Figure 2), similar to Frasconi et al. [9], the hidden layer is fed back into itself through an extra layer of recurrent neurodes. Both the input layer and recurrent layer feed forward to activate the hidden layer, which then feeds forward to activate the output layer. Therefore, the features detected in all previous patterns are fed back into the network with each new pattern. These recurrent neurodes remember the previous internal state.

Figure 2. *Recurrent Backpropagation with Hidden Layer Feedback Link*
(memory: transformed input-exponential)



In the third version (Figure 3), patterns are processed from the input layer through a recurrent layer of neurodes which holds the input layer's contents as they existed when previous patterns were trained, and then are fed back into the input layer.

Figure 3. *Recurrent Backpropagation with Input Layer Feedback Link*
(memory: input-exponential)



The memory's content is the dimension which differentiates the three versions of RRN. It refers to the fact that although it must hold information about the input sequence, it does not have to be a memory of the raw input series. In the three versions used here there were one-for-one linear connections between each recurrent neurode and, respectively, each output, hidden or input neurode.

Issues such as learning parameters, number of hidden neurodes and activation functions are also important in determining the chances of success of different configurations of RNNs. Several alternatives regarding the parameters were tested. The best results were provided, contrary to the tests performed in [15] with standard backpropagation, by the symmetric sigmoid logistic activation function:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

where f(x) has the same shape as the standard sigmoid function, except that its range is [-1,1] rather than [0,1]. The learning rate was initially set at 0.05 and decreased gradually to 0.0005 during the first 500 passes, while the momentum term was fixed at 0.1. The rate of decay μ_i was set at 0.6. Choosing the appropriate number of hidden neurodes was extremely important. The configuration that obtained the best results in terms of generalization had 38 inputs, 5 hidden neurodes and 1 output.

5. Trading Strategy

The importance of the trading strategy (entry orders, exit orders, number of contracts per trade, etc.) can hardly be underestimated. Research shows that identifying the appropriate trading strategy for each forecasting problem is vital to each system's trading performance. The forecast formulated by the three versions of RNNs is just the initial part of a trading strategy. The transformation from predictions into market actions is obtained by specifying a

Copyright © 1995 Software Engineering Press

set of rules to buy and sell currency futures. In particular, according to the uses of IMM, two types of orders were used:

- Market Opening Only Order, where the order is filled only during the opening range at the first available offer (sell order) or bid (buy order);
- Market on Close Order, where the order is filled at any time during the closing range.

Each trade had transaction costs consisting of commission and slippage, the difference between the theoretical execution price and the actual fill price. It was assumed to trade one contract at a time and to have transaction costs equal to $80 per trade ($25 for commission and $55 for slippage). Transactions costs are very important in short term trading systems because they can have a dramatic impact on performance. The network's objective was to forecast the compound return of the following day's Open and Close. Using prices up to the Open at time t (O_t) the forecast was made two steps ahead for the Open at time t+2 (O_{t+2}). Similarly, using prices up to C_{t+1} the forecast was made for C_{t+3}. For the sake of generating trading signals, it is possible to build a continuous price series $F_t,...,F_{t+n}$ by alternating O_t and C_t prices.

Trading Strategy 1 (TrSt1):

Entry rule at time t:

1. If $f(F_{t+2}) > x$ then Long F_{t+1}
2. If $f(F_{t+2}) < -x$ then Short F_{t+1}
3. If $-x < f(F_{t+2}) < x$ then Flat

Exit rule at time t+1:

1. If $f(F_{t+3}) > x$ then stay Long (if 1 at t), stop and reverse to Long (if 2), go Long (if 3)
2. If $f(F_{t+3}) < -x$ then stay Short (if 2 at t), stop and reverse to Short (if 1), go Short (if 3)
3. If $-x < f(F_{t+3}) < x$ then cover Short (if 2), cover Long (if 1), stay Flat (if 3)

where f( ) stands for the network's compound return forecast, r( ) is the compound return, and x is a numerical filter.

Trading Strategy 2 (TrSt2):

Entry rule at time t+1:

1. If $f(F_{t+2}) > [r(F_{t+1}) + x]$ then Long F_{t+1}
2. If $f(F_{t+2}) < [r(F_{t+1}) - x]$ then Short F_{t+1}
3. If $[r(F_{t+1})-x] < f(F_{t+2}) < [r(F_{t+1})+x]$ then Flat

Exit rule at time t+2:

1. If $f(F_{t+3}) > [r(F_{t+2}) + x]$ then stay Long (if 1 at t+1), stop and reverse to Long (if 2), go Long (if 3)
2. If $f(F_{t+3}) < [r(F_{t+2}) - x]$ then stay Short (if 2 at t+1), stop and reverse to Short (if 1), go Short (if 3)
3. If $[r(F_{t+2})-x] < f(F_{t+3}) < [r(F_{t+2})+x]$ then cover short (if 2 at t+1), cover long (if 1), stay flat (if 3)

TrSt1 is more realistic than TrSt2 because it forecasts and decides at time t to go Long, Short, or Flat. The purchase or sale of the future is then done at the next time step t+1. TrSt2, on the other hand, forecasts at time t, but waits until time t+1 to compare the subsequent market open or close with the forecast and then decides whether to buy, sell or do nothing. For this second strategy the slippage is likely to be significantly larger than for TrSt1, because the fill is not made at the Open but immediately after, and a decision must be reached within a few seconds of the Close.

Any sensible trading strategy should somehow restrict the number of trading signals because of the incidence of transaction costs. The filter x was used to provide a way to avoid as much as possible false signals. Its size was optimized using genetic algorithms based on the average profitability of the trading strategy across different RNN versions and periods.

6. Evaluation

The different versions of RNN were compared by focusing on the accuracy and reliability of the forecasts on training, test, and validation data. Differences in the architecture yield significantly different results. A standard error measure to evaluate the quality of predictions is the normalized mean squared error:

Copyright © 1995 Software Engineering Press

$$NMSE = \frac{\sum_{t\in\tau}(observation_t - prediction_t)^2}{\sum_{t\in\tau}(observation_t - mean_t)^2}$$

where t = 1,...,N enumerates the patterns in each data set (τ) used. The above is the ratio between the mean squared errors of both the prediction method and the method which forecasts by using the mean at every step. A value of NMSE = 1 thus corresponds to the value obtained by simply predicting the average. Yet, prediction accuracy statistics such as NMSE by themselves are of little use. The purpose is rather to build trading systems that would provide a consistent profitability on a risk-adjusted basis, with a high degree of confidence.

The results in terms of profitability of the trading strategy, net of trading commissions and slippage, are shown for the different versions. Margin requirements are usually satisfied by posting Treasury Bills. The interest income earned is not accounted in the following performances. Therefore, reported net profits are based on trading profits only, and represent the return earned in excess of the T-Bill rate. In itself, mere profitability is not enough to evaluate the relative value of a trading system. Profit has to be computed across several different periods, as it could be an expression of one isolated period of extraordinary performance. In addition, other measures of relative performance are needed, such as:

$$ROE = \left(1 + \frac{\text{Net Profit}}{\text{MaxValueFuture}}\right)^{(\text{days}/255)} - 1$$

Return on Equity measures the relative unlevered profitability, being the annualized ratio between the net profit and the maximum value of the DM future contract in the period analyzed.

$$ROC = \left(1 + \frac{\text{Net Profit}}{2(\text{MaxDr} + \text{InMar})}\right)^{(\text{days}/255)} - 1$$

Return on Capital expresses the dollar net profit relative to the funds required for trading by an individual trader who is subject to double margins. It is the annualized ratio of the net profit over twice the sum of the maximum drawdown on which the particular trading system would incur plus the initial margin required. It is measure of the efficiency in the use of capital of a trading system. Table 1, 2 and 3 show the net profit, the percentage of correct trading signals, ROE, ROC, and NMSE for each strategy. The figures shown are for the last of the 5 different time periods used (ending in December 1994) of the best network configurations. Each RRN version and trading strategy has its own filter, optimized using genetic algorithms.

Table 1. *Performance measures for RNN 1*

	Training	Test	Validation
	424	100	100
TrSt1 (0.13)	$37,488	$2,925	$2,750
TrSt2 (0.17)	$20,275	$1,950	$2,800
%TrSig 1	69.1%	69.4%	55.6%
%TrSig 2	45.2%	46.9%	45.7%
ROE TrSt1	24.9%	9.1%	8.6%
ROE TrSt2	13.9%	6.0%	8.7%
ROC TrSt1	375.7%	124.3%	241.1%
ROC TrSt2	203.2%	58.2%	234.3%
NMSE	0.8949	0.9622	0.9699

Combining the use of the validation set (in addition to the test set), with the use of different periods, adds greater reliability to the trading systems. Results here are reported only for the DM, though similar performances were obtained on the other currencies. Although it is probably unrealistic to expect any single system to work in all markets, a good system should demonstrate profitability at least in related markets, such as is the case of currency futures. In addition, if a trading strategy is

Copyright © 1995 Software Engineering Press

devised on currency futures very similar results can be expected by using forwards.

Table 2. *Performance measures for RNN 2*

	Training	Test	Validation
	424	100	100
TrSt1 (0.13)	$40,675	$3,925	$8,437
TrSt2 (0.17)	$35,713	$3,788	$8,262
%TrSig 1	64.4%	63.3%	63.5%
%TrSig 2	54.4%	52.1%	51.0%
ROE TrSt1	26.9%	12.4%	27.7%
ROE TrSt2	23.8%	11.9%	27.1%
ROC TrSt1	217.8%	256.9%	353.7%
ROC TrSt2	191.2%	247.9%	383.5%
NMSE	0.9519	0.9683	0.9795

Table 3. *Performance measures for RNN 3*

	Training	Test	Validation
	424	100	100
TrSt1 (0.05)	$28,413	$200	$1,250
TrSt2 (0.04)	$36,063	$1,625	-$625
%TrSig 1	48.9%	46.0%	48.50%
%TrSig 2	43.8%	44.1%	43.00%
ROE TrSt1	24.0%	0.6%	3.9%
ROE TrSt2	28.8%	5.0%	-1.9%
ROC TrSt1	185.1%	3.2%	8.7%
ROC TrSt2	232.4%	15.3%	-9.3%
NMSE	0.9654	0.9984	1.018

RNN2 provides the best overall profitability. It is the best of the three versions even if it is judged in terms of smaller decay of performance going from the training set to the test set, and then from the test set to the validation set. RNN1's results were not quite as good as those of RNN2, while RNN3 did not show a good generalization capability. However, these results should not be taken as the final verdict on the relative merits of the three versions of RNN. As far as trading strategies are concerned, TrSt2 had greater accuracy in forecasting large price movements than TrSt1, even though the percentage of correct trading signals was significantly smaller. This resulted in absolute and relative profitability that were slightly less than those obtained by TrSt1.

Comparisons with standard backpropagation methods have shown that RNN1 and RNN2 have better profitability and generalization capacities. However, as there are an almost infinite number of configurations and parameters in standard backpropagation, I cannot say that it would be impossible to find one which could yield better results than RNN1 and RNN2. Yet I can say for sure that in the many tests I performed between the two methods, RNN1 and 2 always had better results.

7. Conclusions

RNNs, often avoided because of fears of time consuming training sessions, are particularly useful for financial forecasting applications. The methods described here are equally applicable to other markets. Tests have been carried out on equity indices, bond futures, and commodities with encouraging results. Yet they are particularly well-suited to forecasting foreign exchange markets due to the network's adherence to nonlinearities as well as the subtle regularities found in these markets.

The above findings can be considered preliminary as I am in the process of expanding my research to the following areas:

- comparisons of ANNs with standard statistical techniques;
- comparisons of ANNs with mechanical trading systems;
- application of Modern Portfolio Theory framework to ANN financial forecasting;
- diversification of trading systems through the use of regime-switching models;
- development of criteria to be used in the evaluation phase.

The evaluation of test results is a very complex task because so many factors are involved. Certainly it cannot be based on isolated parameters but must incorporate situations

Copyright © 1995 Software Engineering Press

which are true to life. Real-time trading, the ultimate test, shows that ANNs are not a "passing fad" as critics would have us believe.

8. References

[1] Black, Fischer, "The Trouble with Econometric Models," *Financial Analyst Journal*, March 1982

[2] Bilson, John F.O., "Technical Currency Trading," The Chicago Corporation, internal document, 1990

[3] Brock, William A., D.A. Hsieh and B. LeBaron, *Nonlinear Dynamics, Chaos, and Instability*, MIT Press, 1991

[4] Connor, Jerome and L. Atlas, "Recurrent Neural Networks and Time Series Prediction," Proceedings IJCNN, 1991

[5] De Grauwe, Paul, H. Dewachter and M. Embrechts, *Exchange Rate Theory. Chaotic Models of Foreign Exchange Markets*, Blackwell, 1993

[6] Diebold, Francis X. and J.A. Nason, "Non Parametric Exchange Rate Prediction?," *Journal of International Economics*, 28, 1990

[7] Elman, Jeffrey L., "Finding Structure in Time," *Cognitive Science*, 14, 1990

[8] Jordan, M.I., "Serial Order: A Parallel Distibuted Processing Approach", University of California-San Diego, Institute of Cognitive Science, working paper, 1986

[9] Frasconi, Paolo, M. Gori and G. Soda, "Local Feedback Multilayered Networks," *Neural Computation*, 4, 1992

[10] LeBaron, Blake, "Practical Comparisons of Foreign Exchange Forecasts", *Neural Network World*, November 1993

[11] Levich, Richard M. and Thomas L.R., "The Significance of Technical Trading-Rule Profits in the Foreign Exchange Market: A Bootstrap Approach," New York University, Stern School of Business, working paper, 1991

[12] Logar Antonette M., E.M. Corwin and W.J.B. Oldham, "A Comparison of Recurrent Neural Network Learning Algorithms," Proceedings IJCNN, 1993

[13] Meese, Richard A. and A.K. Rose, "An Empirical Assessment of Non-Linearities in Models of Exchange Rate Determination," *The Review of Economic Studies*, 58, 1991

[14] Mozer, Michael C., "Neural Net Architectures for Temporal Sequence Processing," in Weigend, A.S. and N.A. Gershenfeld (eds.), *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley, 1993

[15] Refenes, A.N., M. Azema-Barac, L. Chen and S.A. Karoussos, "Currency Exchange Rate Prediction and Neural Network Design Strategies," *Neural Computing and Applications*, 1, 1993

[16] Schwager, Jack D., *A Complete Guide to the Futures Markets*, Wiley, 1984

[17] Taylor, Stephen, *Modelling Financial Time Series*, Wiley, 1986

[18] Tenti, Paolo, "Optimal Selection of Parameters in Technical Trading Systems," Boston University-Rome, working paper, 1991

[19] Weigend, Andreas S., B.A. Huberman and D.E. Rumelhart, "Predicting Sunspots and Exchange Rates with Connectionist Networks," in Casdagli, Martin and S. Eubank (eds.), *Nonlinear Modeling and Forecasting*, Addison-Wesley, 1992

Copyright © 1995 Software Engineering Press

Designing Financial Swaps with CLP($\Re$)

Evan Tick
Dept. of Computer Science
University of Oregon
Eugene OR 97403, USA
Tel: 503-346-4436 tick@cs.uoregon.edu

Key-words:
Financial swaps, constraints, logic programming.

Abstract
This paper describes how to design and evaluate custom financial swaps using CLP($\Re$), a constraint logic programming language over the real numbers. A prototype analysis tool, Paws, was implemented and its analysis of a large real-life example is given to illustrate the techniques.[1] The analyzer is useful to swap practitioners by allowing quicker and more flexible experimentation over the design space than is currently possible with spread sheets.

1 Introduction

Swaps are financial instruments that allow two parties to exchange interest payments in perhaps different currencies. A swap is a powerful building block from which exchange *networks* can be built, resulting in redistribution of economic surpluses and risks. Usually an intermediary designs and implements the swap network for a fee. A key criterion for a swap network to be viable is that no party must bear risk beyond its risk preference. For the intermediary, this often means *no risk*, i.e., all stochastic factors must "cancel out."

This paper describes how to design and evaluate financial swaps with a constraint logic programming language. We chose CLP($\Re$), over the real numbers [1, 5], for its robustness and availability. A prototype system was built, called Paws, which includes a sophisticated user interface for entering and displaying design solutions. Paws is of interest to swap practitioners by allowing quicker and more flexible experimentation over the design space than can be accomplished with current methods, e.g., spread sheets. For example, exploiting the ability of constraint languages to solve linear equations for any combination of unknown variables, a swap network can be partially constructed without binding all the input parameters. The system will then return the relationship among the unknowns, e.g., give the relation between two interest rates or long-term exchange rates to guarantee a profit within a certain range for a given entity. Using such a tool encourages flexible experimentation and optimization that is not possible with spread sheets, where the equations can be "solved" in only a rigid fashion.

This paper is organized as follows. A brief overview of swaps is given in Section 2. Section 3 reviews CLP($\Re$). Section 4 discusses how the swap analyzer is designed and implemented in CLP($\Re$). A large example, the Kodak swap, is explained in Section 5. The literature is reviewed in Section 6. Conclusions and future work are summarized in Section 7.

2 Review of Swaps

Hull [3], Macfarlane *et al.* [6], Shapiro [8], Smith and Smithson [9], and Wall [11], are just a few of the general expositions about swaps. The subtle assumptions involved in the zero-sum attributes

[1] Paws is available by anonymous ftp from ftp.cs.uoregon.edu:pub/tick/paws.tar.gz. Enquiries for obtaining CLP($\Re$) should be sent to joxan@watson.ibm.com.

Copyright © 1995 Software Engineering Press

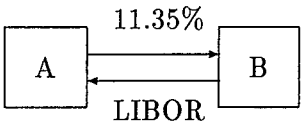


Figure 1: Interest Rate Swap (Building Block)

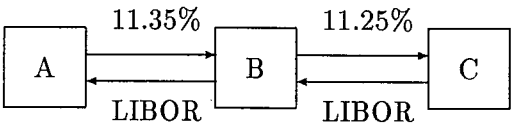


Figure 2: "Plain Vanilla" Interest Rate Swap

of the swaps are clarified by Turnbull [10]: our analyzer deals only with the simple model of no default risk and no hidden transactions costs (see Smith and Smithson for issues [9]). The knowledgeable reader may wish to skip to Section 3.

2.1 Interest Rate Swaps

Figure 1 illustrates the simplest interest rate swap wherein the two parties A and B have loans of the same principle amount, P. The type of loan we consider extends over some number of periods t_i for $1 \leq i \leq n$. Payments are made (according to the interest rate) each period t_i followed by a lump-sum payment of the entire principle at the last period t_n.

The swap consists of A making fixed interest payments of 11.35% to B in exchange for receiving floating LIBOR payments from B. For example, a scenario in which this makes sense is when A has a floating-rate loan pegged to LIBOR and B has a fixed-rate loan. We do not show these lenders in the swap network shown in Figure 1. For reasons of risk preference, A wants a fixed rate and B wants a floating rate, and so they swap interest payments.

A swap is effectively a simultaneous exchange of bonds. Using net present valuation, we can compute any of these bond values:

$$B = P - \sum_{i=1}^{n} \frac{s_i P}{(1+r)^{t_i}} - \frac{P}{(1+r)^{t_n}}$$

where s_i is the loan interest rate for period i and r is a fixed market rate. Here we assume that the bonds are risk-free and have the same principle P and length of term n. If we relax our restriction of a fixed market rate, we get:

$$B = P - \sum_{i=1}^{n} \frac{s_i P}{\prod_{k=1}^{i}(1+r_k)} - \frac{P}{\prod_{k=1}^{n}(1+r_k)}$$

where r_k is the market rate for period k (see Section 2.3). Although the latter formula is implemented in our swap analysis tool, for simplicity we explain swaps using the former equation. Thus for instance we see:

$$\begin{aligned} B_1 &= P - \sum_{i=1}^{n} \frac{0.1135(P)}{(1+r)^{t_i}} - \frac{P}{(1+r)^{t_n}} \\ B_2 &= P - \sum_{i=1}^{n} \frac{\text{LIBOR}_i(P)}{(1+r)^{t_i}} - \frac{P}{(1+r)^{t_n}} \\ \pi_A &= B_1 - B_2 \\ \pi_B &= B_2 - B_1 \end{aligned}$$

Another simplification is to remove the dependence on LIBOR_i for all periods i. Hull [3] briefly discusses how to do this. Effectively, B_2 is the same for any value of $n \geq 1$. Thus pick $n = 1$ to get the simplest relation, based only upon LIBOR_1. Assuming that all parties are risk-free banks, this can be justified by having the initial bondholder pass the bond through to another party after one period. Thus all subsequent cash flows cancel, leaving only the cash flows at the end of the initial period. Effectively the rate floats to ensure that this simplification holds!

From this simple building block we can build more sophisticated networks. Figure 2 shows dual offsetting swaps through an intermediary B. Clearly A effectively transforms a floating to a fixed loan, and C transforms a fixed to a floating loan. B cancels its risk by passing the floating payments through from C to A. If we assume neither A nor C defaults, then B has no risk. In addition, B takes a profit of 0.1% for its service. Valuation gives the additional equations:

$$\begin{aligned} B_3 &= P - \sum_{i=1}^{n} \frac{0.1125(P)}{(1+r)^{t_i}} - \frac{P}{(1+r)^{t_n}} \\ \pi_A &= B_1 - B_2 \\ \pi_B &= B_3 - B_1 \\ \pi_C &= B_2 - B_3 \end{aligned}$$

Copyright © 1995 Software Engineering Press

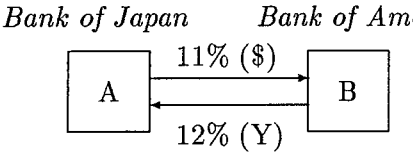


Figure 3: Currency Swap (Building Block)

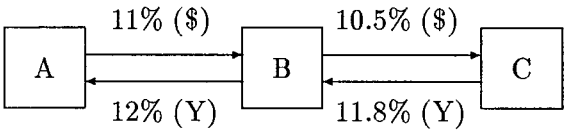


Figure 4: "Plain Deal" Interest Rate Swap

For this simple example, the profit to B can be computed more directly; however, in complex networks, the general formula is needed. To simplify things, we may elect to assume that the market rate is fixed over the length of the loan. In any case, it is critical for evaluating this formula in CLP($\Re$) that the market rate(s) be known *a priori*, otherwise nonlinear equations arise.

In addition to previous bond-like loans, amortized loans, wherein the principle is incrementally repaid, are easily modeled. Our system supports a library of various types of loans.

2.2 Currency Swaps

Figure 3 shows a simple currency swap building block. Here parties A and B lend each other principles in yen and dollars, respectively, of approximately the same value. They then pay each other interest based on those principles, until the end of the loan, when the principles are repaid. To alleviate foreign exchange risk at period t_n when the principles are repaid, a forward exchange rate, $F_{\$/Y}$ may be agreed upon in the swap agreement. For example, suppose the original principles are $P_\$$ and P_Y, where $P_\$ = S_{\$/Y} P_Y$ at t_0 given the spot exchange rate $S_{\$/Y}$. Then at t_n, parties A and B might replace principles $P'_\$$ and P_Y respectively, where $P'_\$ = F_{\$/Y} P_Y$.

The previous bond valuation formulae still hold, where the dollar bond value is B_1 and the yen bond value is B_2:

$$\begin{aligned} \pi_A &= B_1 - B_2 = S_{Y/\$} B_1 - B_2 \\ \pi_B &= B_2 - B_1 = S_{\$/Y} B_2 - B_1 \end{aligned}$$

In the above we compute the current value of the swap to parties A and B in today's yen and dollars respectively.

From this simple building block we can build more sophisticated networks. Figure 4 shows dual offsetting swaps through an intermediary B. Valuation of the swap follows from the previous discussion. A *circus swap*[2] is a combination of plain vanilla interest rate swap and plain deal currency swap [3], i.e., basically the swap of Figure 4 with either currency's loans on a floating rate. We will see an example of this in the larger example discussed in Section 5.

2.3 Discussion

As shown, the building blocks for composing swap networks use elementary cash flow mathematics, which facilitate their expression in CLP($\Re$). The key underlying stochastic variables — floating interest rates, market investment rates, and currency exchange rates — are however problematic. There are several ways of viewing this problem.

First, it is clear that the cash flows constituting any link in a swap network can be priced as accurately as can a bond or future [9]. Hull [3] reviews bond valuation methods such as Monte Carlo simulation and lattice evaluation. For example, using a lattice method, we evaluate all states of the world with respect to interest rates, e.g., using Ho and Lee [2] or advanced models, and value the bond over each scenario.

The problem of pricing all components of a complex swap network is more difficult. There appear to be two ways to structure the computation: evaluate the network "inside" the lattice (or simulation), or estimate the term structure first and apply it to the network. We chose the latter technique for three reasons: 1) *modularity*: we can utilize available, sophisticated interest rate prediction tools; 2) *speed*: we can run "what if" experiments quickly; and 3) *flexibility*: future extensions of our tool will hopefully exploit CLP($\Re$) to *solve* for traditionally stochastic variables (see Section 7).

[2] Picadilly or Ringling Brothers?

Copyright © 1995 Software Engineering Press

```
flat( Start, End, _, _, _, 0 ) :- Start >= End.
flat( Start, End, Principle, Rate, MR, Value ) :-
   Start < End,
   Value = Principle - Payments
   loan( End-Start, Principle, Rate/100,
         MR/100, Payments ).

loan( Time, In, Rate, MR, Value ) :-
   Time > 0, Time <= 1,
   Out = In / ( 1 + MR * Time ),
   Value = Out * ( 1 + Rate * Time ).

loan( Time, In, Rate, MR, Value ) :-
   Time > 1,
   Out = In / ( 1 + MR ),
   Value = Next_Value + ( Out * Rate ),
   loan( Time-1, Out, Rate, MR, Next_Value ).
```

Figure 5: Loan Valuation in CLP($\Re$)

3 Review of CLP($\Re$)

CLP($\Re$) is constraint logic programming language over the domain of real arithmetic. Programs appear in syntax to be Prolog programs, i.e., data and control structures are the same. The semantics of unification, however, are vastly different. We illustrate the language with a simplified version of a loan of the type previously discussed, shown in Figure 5. This procedure computes the net present value of fixed interest loans with fixed market rates only. Procedure `flat/6` has the following parameters: the `Start` and `End` periods of the loan, the `Principle`, the fixed loan `Rate`, `MR` (a fixed market rate), and `Value` (the net present value of the loan).

If the length of the loan is not positive, the loan value is zero (`flat/6` clause 1). Otherwise, the loan value is the principle minus the payments, computed by `loan/5`, starting at the next period. Procedure `loan/5` computes the payment value in what can be considered an iterative (or recursive) manner; however, the language lends itself to a more elegant *declarative* semantics. In effect, `loan/5` (and any procedure invocation in general) is true if the equations it engenders are consistent over the domain of the reals. Furthermore, these equations are not necessarily evaluated in any strict order: CLP($\Re$) has an internal equation solver that is transparent to the programmer.

The spawned equations form a recurrence. Each successive value is equal to the next value plus the discounted principle multiplied by the interest rate (`loan/5` clause 2). The final value (at the final period) also includes payback of the entire principle (`loan/5` clause 1). The final period can be fractional, requiring us to scale the loan and market rates by the remaining time.

Examples of queries to this program are instructive. The value of a three year $100 loan at 10% assuming a 5% market rate is: `flat(1,4,100,10,5,V)` returns `V = -13.6162`. Alternatively we can solve for loan rate: `flat(1,4,100,R,5,-14)` returns `R = 10.14`. However, we cannot solve for the market rate because the function is nonlinear in this variable. More strangely, we cannot solve for the time. For example, trying to solve for the ending period with the query `flat(1,E,100,10,5,-14)` returns:

```
E <= 2
1 < E
114 = _t13 * (0.1*E + 0.9)
100 = (0.05*E + 0.95) * _t13

*** (Maybe) Retry?
```

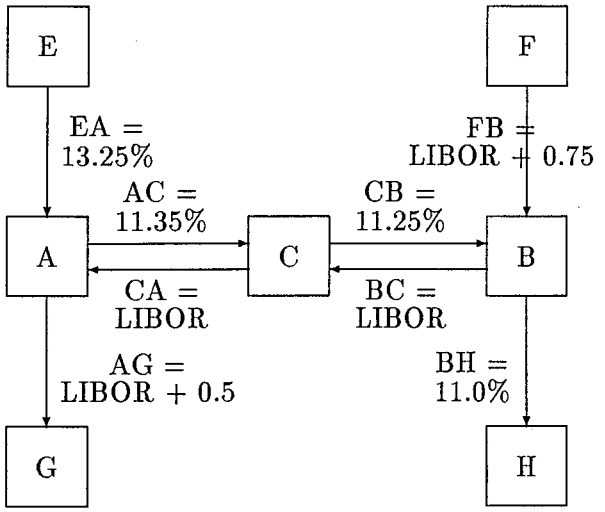
The "maybe" caveat in the result indicates that the non-linearity could not be removed and that the solution may be inconsistent. We can avoid CLP($\Re$) confusion by simplifying clause 1 of `loan/5` as:

```
loan( Time, In, Rate, MR, Value ) :-
   Time > 0, Time <= 1,
   Value * (1 + MR*Time) = In * (1 + Rate*Time).
```

There is an art to making such transformations! With this change, the system automatically solves: `flat(1,E,100,10,5,-14)` as `E = 4.55`.

It is important to note that to solve for time, the final fractional scaling of the loan rate is required. Without this, `loan/5` would not be able to ground the recurrence when solving for time (`E`), i.e., it loops forever. Although the analyzer we built uses loan procedures that are more sophisticated that this, the foundation is the same. Additional complexity arises from (optional) variable loan and market rates and amortization.

Copyright © 1995 Software Engineering Press




```
net1( Info, [ Pi_A, Pi_B, Pi_C ] ) :-
   Info = [ AC, CA, CB, BC, AG, BH, FB, EA ],
   Pi_A = CA - AC - AG + EA,
   Pi_B = FB - BH - BC + CB,
   Pi_C = AC + BC - CA - CB.
```

Figure 6: Plain Vanilla Interest Rate Swap in CLP($\Re$)

4 Swap Analysis in CLP($\Re$)

This section describes the design construction of the swap analyzer, as a series of increasingly sophisticated models. The analyzer is meant for fast prototyping of custom swaps rather than the development of generic products. Figure 6 shows an interest rate swap and its straightforward translation into CLP($\Re$) program, where the loan structures are identical except for the rates. Essentially each node in the network corresponds to an equation balancing the interest rates entering/exiting that node. This simple "rate" methodology for evaluating the swap is possible because the loan principles and terms are identical. This model also assumes a fixed market rate. A typical query to this program is:

```
?- net1( [ 11.35, LIBOR, 11.25, LIBOR, LIBOR+0.5,
           11.0, LIBOR+0.75, 13.25 ], Pi ).

   Pi = [1.4, 1, 0.1]
```

When the swap calls for differing principles or terms, then individual cash flows must be computing using the bond valuation formula. This model is considered in Figure 7, which shows the

```
net2([P,R_mkt,T], Info, Libor, [Pi_A,Pi_B,Pi_C]) :-
   Info = [ AC, CA, CB, BC, AG, BH, FB, EA ],
   Pi_A = AC_CF + AG_CF - CA_CF - EA_CF,
   Pi_B = BH_CF + BC_CF - FB_CF - CB_CF,
   Pi_C = CA_CF + CB_CF - AC_CF - BC_CF,
   loan(  P, EA,         R_mkt, T, EA_CF ),
   loan(  P, AC,         R_mkt, T, AC_CF ),
   loan(  P, CB,         R_mkt, T, CB_CF ),
   loan(  P, BH,         R_mkt, T, BH_CF ),
   floan( P, CA, Libor, R_mkt, T, CA_CF ),
   floan( P, AG, Libor, R_mkt, T, AG_CF ),
   floan( P, BC, Libor, R_mkt, T, BC_CF ),
   floan( P, FB, Libor, R_mkt, T, FB_CF ).
```

Figure 7: Cash-Flow Model in CLP($\Re$) for Previous Network

CLP($\Re$) implementation of a cash-flow model of the previous network. We invoke the `loan/5` and `floan/6` procedures for a fixed principle of \$100M and 5 period loan length. The net present values are computed from the cash flows rather than the interest rates as in Figure 6. Clearly we could give each loan independent principles and lengths if we desired. Interestingly, we need never define `Libor`: it will be instantiated as necessary and shared among the four floating loans. All unknown LIBOR terms will cancel from the solved equations! For example, typical queries to the program include:

```
?- net2( [100,10,5], [ 11.35, 0, 11.25, 0, 0.5,
                       11.0, 0.75, 13.25 ], _, Pi ).

   Pi = [5.3071, 3.79079, 0.379079]

?- net2( [100,10,5], [ X, 0, 11.25, 0, 0.5,
                       11.0, Y, 13.25 ], _, Pi ).

   Pi = [-4.19247*X + 53.454, 4.19247*Y + 1.04812,
                              4.19247*X - 47.1653]
```

These solutions are in dollars and are consistent with the previous (rate model) solution in terms of interest rates.

Figure 8 shows an extended implementation of the network with full input parameters allowing each node to have a different principle and loan length. For example, given this procedure, we can query:

```
?- net3( [ (90,6,11.35), (100,4,0), (100,5,11.25),
           (100,5,0), (100,5,0.5), (100,5,11.0),
           (100,5,0.75), (100,5,13.25) ],
         [L1,L2,L3,L4,L5,L6,L7,L8], Pi ).
```

Copyright © 1995 Software Engineering Press

```
net3( Info, Libor, [ Pi_A, Pi_B, Pi_C ] ) :-
   R_mkt = 10,
   Info = [ ( Pac, Tac, AC ),
            ( Pca, Tca, CA ),
            ( Pcb, Tcb, CB ),
            ( Pbc, Tbc, BC ),
            ( Pag, Tag, AG ),
            ( Pbh, Tbh, BH ),
            ( Pfb, Tfb, FB ),
            ( Pea, Tea, EA )],
   Pi_A = AC_CF + AG_CF - CA_CF - EA_CF,
   Pi_B = BH_CF + BC_CF - FB_CF - CB_CF,
   Pi_C = CA_CF + CB_CF - AC_CF - BC_CF,
   loan(  Pea, EA,        R_mkt, Tea, EA_CF ),
   loan(  Pac, AC,        R_mkt, Tac, AC_CF ),
   loan(  Pcb, CB,        R_mkt, Tcb, CB_CF ),
   loan(  Pbh, BH,        R_mkt, Tbh, BH_CF ),
   floan( Pca, CA, Libor, R_mkt, Tca, CA_CF ),
   floan( Pag, AG, Libor, R_mkt, Tag, AG_CF ),
   floan( Pbc, BC, Libor, R_mkt, Tbc, BC_CF ),
   floan( Pfb, FB, Libor, R_mkt, Tfb, FB_CF ).
```

Figure 8: Cash-Flow Model in CLP($\Re$) with Extended Parameters

```
Pi = [ -62.0921*L5 + 11.3422, 3.79079,
       62.0921*L5 - 5.65605 ]
```

which gives the profits when the fixed loan to A (from C) is extended to 6 periods on a reduced principle of 90 and the floating loan to C (from A) is reduced to 4 periods. Note that the profit to B has not changed. The change in the profits to A and C are a function of the parameter L5 which is the last period LIBOR rate. The previous four LIBOR rates still cancel by the swap. As the loan lengths become more disparate, we expect to see more LIBOR rate terms in the solutions.

This final model is similar to our prototype. Additional facilities include fixed amortized loans, principles in alternative currencies, and forward exchanges. The market rate is implemented in a manner similar to LIBOR. A fundamental difference is that in the analyzer we built, programs such as net3 are generated *automatically* from net specifications with are in turn generated from graphical input supplied by the user. Thus one crucial design philosophy we adopted was to shelter the user from CLP($\Re$). This impacted the flexibility with which the system can be used, as is discussed below.

Figure 9 shows the system overview of our current prototype analyzer. The user interface [7],

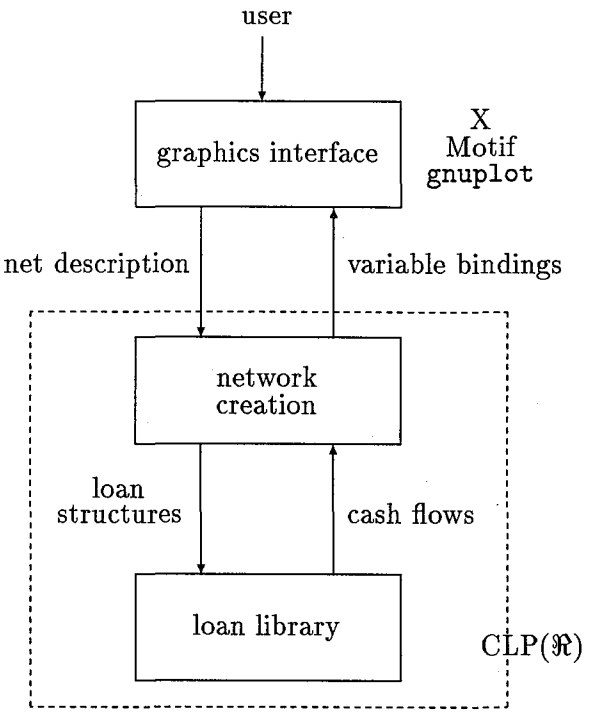


Figure 9: Overview of the Paws System

written in C, accepts graphical entry of the network and translates it into a net description accepted by the analyzer, written in CLP($\Re$). For a complete user's guide to the analysis tool, called Paws, see Scott [7].

The net description can have symbolic names for parameters, which if bound are returned as solutions. In addition, a profit is computed and returned for each node in the graph, which is the sum of its cash flows. Internal to the analyzer itself, the net description is used to invoke loan library routines that define various types of payments, such as simple or amortized. These invocations return the cash flow values needed to compute the net present value profits.

Examples of the user interface are illustrated in Scott [7]. The interface allows the user to graphically specify the swap network, entering parameters for each entity (node) and loan (edge). Either real values or symbolic names can be assigned to parameters. A sketch of the information is displayed on the illustrated graph, with detailed information available by explicit querying the interface (with a mouse). The user can also specify constraints in terms of both symbolic input pa-

Copyright © 1995 Software Engineering Press

rameters as well as profits. The use of this facility is illustrated in Section 5.

The interface also translates variable bindings into graphics in the limited cases when the binding is an equation in one or two independent variables. We generate a nonparameterized graph description for `gnuplot`. This is also illustrated in Section 5.

4.1 Symbolic Output

Whenever expressing symbolic solutions in CLP($\Re$), the issue of which symbolic variables in the formula are dependent and which are independent looms large. Flexibility in controlling the relative independence of variables is achieved with the `dump/3` predicate [1]. `dump/3` takes a list of variables as input, where the variables earlier in the list order are more independent than later variables. `dump/3` displays dependent variables in terms of independent variables specified by this list. If we purposely remove certain independent variables from the list, we can receive symbolic answers among the dependent variables.

Because of the great flexibility of output control, it becomes difficult for the analyzer to make autonomous decisions concerning symbolic output construction. Sometimes a user may wish to see a certain relationship among variables that would not abide by any default we could provide. Therefore, in the user interface we provide the ability for the user to specify the `dump/3` control list explicitly. A default is presented: `[L1,L2,..,M1,M2,..,U1,U2,..,Pi1,Pi2,..]` where `L1` is the LIBOR rate in period 1, `M1` is the market rate in period 1, `U1` is a user-defined variable, and `Pi1` is the profit of node #1, etc. Any of these may be absent if inappropriate to the problem at hand, e.g., the market rate may be a given constant.

By rearranging this list (usually by variable *type*), the user can produce any relationships needed. For example, `[U1,U2,...,Pi1,Pi2,...]` would show the profits in terms of the user-defined variables and not the LIBOR rates. Another example is: `[M,Pi1,Pi2]` might plot each of the two profits as a function of a fixed market rate, whereas `[Pi1,Pi2,M]` might plot the fixed market rate as a function of the two profits. By affecting the formulae produced by CLP($\Re$), this control list indirectly affects graphs produced by `gnuplot` because independent variables are plotted along the X and Y axes.

5 Kodak Example

The Kodak swap [8] illustrates the complexity of swaps in practice, involving two currencies, three banks, an intermediary (Meryll Lynch), and a firm (Kodak). Without going into the detail of the swap agreement, we illustrate the original terms of the swap in Figure 10. Each financial entity is given its own node in the graph, labeled by a node identifier. Edges are annotated with principle amounts (in millions). This figure does not show the implicit five year structure of all loans, nor does it explicitly specify the periods when the currency exchanges are made (when using our swap analyzer, such information must be entered). The swap analyzer can solve this version of the problem, where the result is

$$\begin{aligned} \pi_1 &= -\$9.49 \\ \pi_2 &= \$26.8 \\ \pi_4 &= -\pi_3 - \$17.2 \\ \pi_5 &= -\$1.09 \\ \pi_6 &= \$0.963 \end{aligned}$$

for U.S. dollar amounts in millions, assuming a spot exchange rate of \$1/0.75A. If we wanted the value of π_3 (or π_4) in detail, we would use an output control specification with LIBOR rates as the most independent variables, giving $\pi_3 = 0.34\texttt{L10} + 0.35\texttt{L9} + \cdots + 0.46\texttt{L1} - \$32.9$.

The previous network was slightly modified with with two unknown parameters, `Rate1` and `Rate2` substituted for 7.35% and 7.85%, respectively. A typical use of the system would be to view the profits as functions of these parameters. The internal solution produced are:

$$\begin{aligned} \pi_1 &= -\$9.49 \\ \pi_2 &= \$26.8 \\ \pi_4 &= -\pi_3 + 3.99 * \texttt{Rate2} - 32.9 \\ \pi_5 &= -(6.24 * \texttt{Rate1}) + 21.8 \\ \pi_6 &= 6.24 * \texttt{Rate1} - 3.99 * \texttt{Rate2} - 6.29 \end{aligned}$$

Copyright © 1995 Software Engineering Press

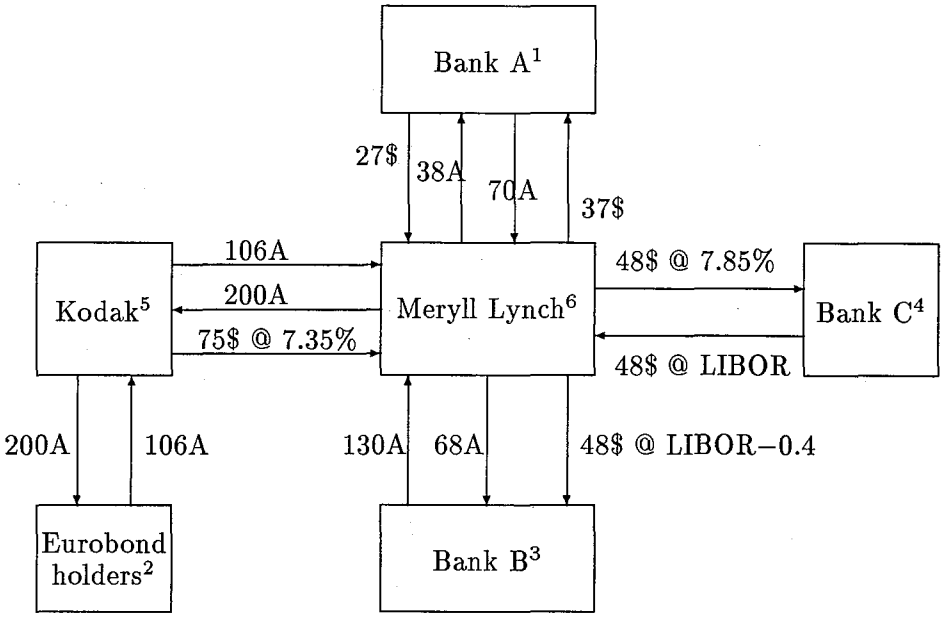


Figure 10: Original Kodak Swap (Principles in Millions)

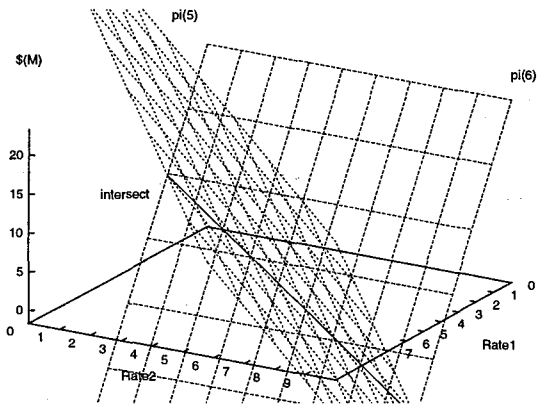


Figure 11: 3-D gnuplot of Profits π_5 and π_6

Any solution (left-hand side variable) that is a function of one or two independent (right-hand side) variables is displayed to the user via gnuplot. For example, profits π_5 and π_6 are shown in Figure 11. Making the plots is more difficult than it may look and further research is needed.[3]

A further user facility is the incorporation of constraints in these and other (profit) parameters. Suppose the user specifies that the profits of Meryll Lynch and Kodak are to be equal, by entering the constraint $\pi_5 = \pi_6$. The system can then simplify the solution:

$$\begin{aligned} \pi_1 &= -\$9.49 \\ \pi_2 &= \$26.8 \\ \pi_4 &= -\pi_3 + 3.99 * \texttt{Rate2} - 32.9 \\ \pi_5 &= \pi_6 = -2.0 * \texttt{Rate2} + 7.77 \\ \texttt{Rate1} &= 0.32 * \texttt{Rate2} + 2.25 \end{aligned}$$

which is displayed as in Figure 12. Note that π_4 is a function of π_3 because both are dependent on LIBOR rates (which Meryll Lynch passes through, so π_6 has no such dependency). The relationship between Rate1 and Rate2 to guarantee equal profits is shown above.

If we add the constraint $\pi_3 = \pi_4$, CLP($\Re$) gives us the solution:

$$\pi_1 = -\$9.49$$

[3]The actual plots are in color! The intersection of the planes is a parametric equation, and thus the planes must all be parametric for gnuplot to display them together. This was done in Figure 11 "by hand." Our current system produces only nonparametric equations, so we cannot compute and display plane intersections (yet). Finding a good vantage point and proper scaling of axes also remain unsolved problems. Currently we rely on gnuplot defaults.

Copyright © 1995 Software Engineering Press

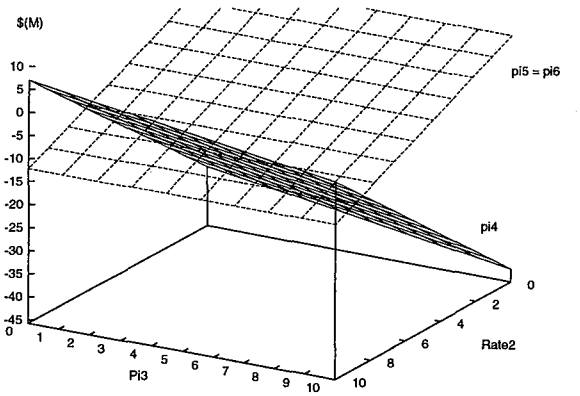


Figure 12: 2-D gnuplot of Profits π_5 and π_6

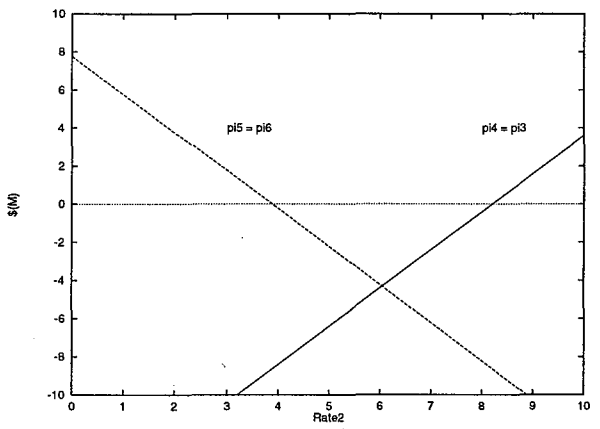


Figure 13: 2-D gnuplot of π_3, π_4, π_5 and π_6

$$\begin{aligned} \pi_2 &= \$26.8 \\ \pi_4 &= \pi_3 = 2.0 * \texttt{Rate2} - 16.4 \\ \pi_5 &= \pi_6 = -2.0 * \texttt{Rate2} + 7.77 \end{aligned}$$

plotted in Figure 13. Note that because π_3 and π_4 depend on floating rates in different ways (Bank B receives LIBOR whereas Bank C pays LIBOR), the only way to ensure that the profits are equal is to set `Rate2` as a function of LIBOR itself. The above equations disguise this as π_3 as a function of `Rate2`. However, if we modify the output control specification as: `[L1,...,L10,Rate1,Rate2]` we get a direct relationship: $\texttt{Rate2} = 0.17\texttt{L10} + \cdots + 0.23\texttt{L1} - \$8.24$.

From Figure 13 we see that a value for `Rate2` exists allowing the four profits to be equal. We could solve for this value directly by adding the constraint $\pi_3 = \pi_5$ to the system, getting:

$$\begin{aligned} \pi_1 &= -\$9.49 \\ \pi_2 &= \$26.8 \\ \pi_3 &= \pi_4 = \pi_5 = \pi_6 = -\$4.34 \\ \texttt{Rate1} &= 4.20 \\ \texttt{Rate2} &= 6.66 \end{aligned}$$

meaning that they all lose money. The subtle danger with this solution is that it implies a constraint on LIBOR that may be unrealistic. A mechanism for testing such an over-constrained floating rate is to set the output control specification to: `[L1,L2,...,L9,L10]`. If any constraints result then the system is over constrained. An empty output indicates a solution with no constraints and so everything is ok. A similar method can be used for testing market rates. We do not yet automatically perform these checks within our system, but it is straightforward to do so.

6 Related Work

A related work in the field of financial engineering is OTAS (Options Trading Analysis System) designed by C. Lassez *et al.* [4] at IBM Yorktown Heights. This system, also based on CLP($\Re$), evaluates the Black-Scholes solution to the partial differential equation describing an option's fair price. The arithmetic involved is more computationally intensive than that of swaps. Because the essential formula is non-linear in the volatility parameter, they explicitly linearize it.

Smith and Smithson [9] use time-line notation for the cash flows in a swap. This graphical technique does not gracefully extend to complex swap networks. Furthermore, they suggest pricing methods based on futures and bonds, again, which don't gracefully extend to complex networks (see Section 2.3).

7 Conclusions

A financial swap analysis tool, Paws, was described that can accept a high-level description of a swap network and produce functional relationships between unknown parameters, including the net present value profits of each entity

Copyright © 1995 Software Engineering Press

in the system. The engine of the tool was built in CLP($\Re$), exploiting its ability to perform symbolic arithmetic over the reals, and the user interface was built in C/Motif. The advantage of such a tool is the ability to quickly and flexibly design and evaluate custom swaps under incomplete information. Profits and parameters can be symbolically constrained to reduce the search space and symbolic solutions can be graphically displayed to help users gain intuition about parametric relationships. These attributes make the tool superior to current analysis methods, specifically those based on spread sheets.

There are two main directions in which we wish to extend our analyzer, both of which rely on separating market rate prediction from network valuation. First, we might avoid rate prediction in certain cases, by making simplying assumptions that allow linearization of the valuation equation with respect to market rate, thus enabling CLP($\Re$) to solve for it. More generally, we might open up the architecture of the analyzer. In such a "glass box" approach, the user will be permitted to write procedures describing custom payoffs (e.g., a "floating floor-ceiling swap" [9]) or constrain term structures to be simple functions of time. Again, the motivation is to offer flexibility and speed of prototyping, if accurate, estimates or simple models of stochastic variables are available.

Acknowledgements

This research was supported by an NSF Presidential Young Investigator award, with matching funds from Sequent Computer Systems Inc., and a grant from the Institute for New Generation Computer Technology (ICOT). Raul Clouse helped build the first prototype analyzer and David Scott built the user interface. I thank Bart Massey for many helpful discussions. Peter Stuckey and Roland Yap patiently explained the subtleties of CLP($\Re$) to me.

References

[1] N. C. Heintz, J. Jaffar, S. Michaylov, P. J. Stuckey, and R. H. C. Yap. The CLP(R) Programmer's Manual Version 1.2, 1992.

[2] T. S. Y. Ho and S.-B. Lee. Term Structure Movements and Pricing Interest Rate Contingent Claims. *The Journal of Finance*, 41:1011–1029, 1986.

[3] J. Hull. *Options, Futures and Other Derivative Securities.* Prentice Hall, 1989.

[4] T. Huynh and C. Lassez. An Expert Decision-Support System for Option-Based Investment Strategies. *Computers Mathematical Applications*, 20(9/10):1–14, 1990.

[5] J. Jaffar and J.-L. Lassez. Constraint Logic Programming. In *SIGPLAN Symposium on Principles of Programming Languages*, Munich, 1987. ACM Press.

[6] J. Macfarlane, D. R. Ross, and J. Showers. The Interest Rate Swap Market: Yield Mathematics, Terminology and Conventions. Salomon Brothers Inc., June 1985.

[7] D. H. Scott. A Tool for Designing Financial Swaps. Bachelor's thesis, The University of Oregon, December 1994.

[8] A. Shapiro. *Multinational Financial Management.* Allyn and Bacon, 4th edition, 1992.

[9] C. W. Smith Jr. and C. W. Smithson, editors. *The Handbook of Financial Engineering.* Harper Business, New York, 1990.

[10] S. M. Turnbull. Swaps: A Zero Sum Game? *Financial Management*, 16(1):15–21, 1987.

[11] L. D. Wall and J. J. Pringle. Interest Rate Swaps: A Review of the Issues. In C. W. Smith Jr. and C. W. Smithson, editors, *The Handbook of Financial Engineering*, pages 230–254. Harper Business, New York, 1990.

Copyright © 1995 Software Engineering Press

Fast Cost-Effective Computations of Derivatives

Roy S. Freedman
Inductive Solutions, Inc.
380 Rector Place
New York, NY 10280
Inductive_Solutions@MCIMail.com

Rinaldo DiGiorgio
Sun Microsystems, Inc.
1 New York Plaza — 35th Floor
New York, NY 10004
Rinaldo.Digiorgio@East.Sun.com

Abstract

The essential idea of this paper is that one should not separate the method of computing the expected present value of a derivative from its ultimate computing topology. In the following sections, we discuss the cost-benefit issues involved with implementing several methods for computing derivative statistics on alternate computing topologies. We show how the choice of topology impacts the computing time for a particular example of a time consuming derivative valuation. We conclude by showing how all these factors can be represented as a case-based expert system, which can be used to help an organization assess its computing alternatives.

1. Background: Algorithm Tradeoffs in Computing Derivatives

We are concerned with the computational problem of deriving the expected value and other statistics of a derivative security f at time T0. When the underlying security S and derivative security f are modeled as stochastic processes, the problem can be solved by reformulating it as a boundary-value problem: if it is known that the derivative pays out fT at time T, we just compute its value backwards from the risk adjusted random price movements of the underlying from t=T to t= T_0. The present value of f is just its expected discounted value in a risk-neutral world

$$\text{Expected Present Value} = E[e^{-r(T-T_0)} f_T] \qquad (1)$$

Here, r is the average instantaneous risk-free interest rate between t=T_0 and t=T. When the underlying S follows an Ito process, and if the derivative is a differentiable function of S and t, f=f(S,t), then by Ito's Lemma, f also follows an Ito process:

$$dS = \mu(t,S)\,dt + \sigma(t,S)\,dz \qquad (2S)$$

$$\begin{aligned} df &= (\partial f/\partial S)dS + [\partial f/\partial t + (1/2)\sigma^2(t,S)(\partial^2 f/\partial S^2)]\,dt \\ &= [\mu(t,S)\,(\partial f/\partial S) + \partial f/\partial t + (1/2)\sigma^2(t,S)(\partial^2 f/\partial S^2)]dt \\ &\quad + \sigma(t,S)(\partial f/\partial S)\,dz \end{aligned} \qquad (2f)$$

and f satisfies the Fokker-Plank forward diffusion equation:

$$\partial f/\partial t = (1/2)(\partial^2/\partial S^2)[\sigma^2(t,S)\,f] - (\partial/\partial S)[\mu(t,S)\,f]$$
$$\text{given initial condition } S(T_0) = S_0 \qquad (2FP)$$

Here S(t) is the probability distribution of the price of the underlying at time t, $\mu(t,S)$ and $\sigma(t,S)$ are the instantaneous drift and standard deviation rates, and dz is a Wiener Process that corresponds to Brownian motion. Note that if we know the probability distributions for S(t), and if we are given boundary conditions for f (which define the derivative), then we can solve (2FP) and derive the probability distribution for f, so that the expected present value of f can be computed from Equation (1).

The above equations are valid for all derivative securities with S as the underlying stochastic variable [4]. A vector form of Equation (2S) and (2f) is valid if S depends on other Ito processes (for example, if μ or σ are Ito processes). Here, the correlations of the underlying processes are additional factors in the dt term in Equation (2f).

Simplifications can be made: if the interest rate r is known to be constant, then it can be shown that the Ito process for $[S(\partial f/\partial S)-f]$ does not depend on dz — this "continuous" hedge is "riskless." Hence, in this case, f satisfies the Black-Scholes partial differential equation

$$\partial f/\partial t = rf - (1/2)S^2\sigma^2(t,S)(\partial^2 f/\partial S^2) - rS(\partial f/\partial S) \qquad (2BS)$$

Equation (2BS) can be solved if S(t) is known and the boundary conditions that define the derivative f are provided. For example, a boundary condition for a

Copyright © 1995 Software Engineering Press

European call option is

$$\text{At } t = T,\ f(T) = f_T = \max(S_T - X, 0) \qquad (2CO)$$

In practice, in all but the simplest cases, the price movements of S and f follow stochastic processes that involve substantial amounts of computation. There are three general methods that have different computational consequences for computing European-style derivatives (the holder has no decisions to make during its life) and American-style derivatives (the holder has decisions to make during its life):

Method 1. Analytic Approximation for Constant Parameters. If the derivative is a European-style derivative, and the Ito process in Equations (2S), (2f), and (2BS) has constant $\mu(t,S) = \mu$, constant $\sigma(t,S) = \sigma$, and constant interest rate, then computationally nice expressions exist for the derivative security — the famous formulas derived by Black and Scholes. Analytic expressions also exist for approximating the values of American-style derivatives. In Method 1, the time required to compute the expected value of f is proportional to a constant factor G — the time required to evaluate the formula. In general, G depends on the efficiency of computation of special functions (like the normal distribution).

Method 2. Recombining Lattice-Type Computations. If the Ito process in Equations (2S), (2f), and (2BS) has constant $\mu(t,S) = \mu$, constant $\sigma(t,S) = \sigma$, and constant interest rate, then the valuation of a European- or American-style derivative is usually computed by simulating the up-down price movements in a recombining binomial lattice. (The lattice is a discrete form of Equation (2S-2f), and is also related to a discrete form of (2FP) and (2BS)). In this method, the time required to compute the value of a derivative depends on the number of time units N, where $N = (T-T_0)/\Delta t$, and Δt is the smallest unit of time considered in the computation. In this method, a sequence of up movements followed by down movements are valued the same as the down movements followed by the up movements. At any given point in time $T_0 + i\Delta t$, the price of the underlying may increase or decrease by an amount u and d with probability p and (1-p) respectively. Hence, at time $T_0 + i\Delta t$, the price of the underlying may be any of a set of i+1 values:

$$S\,u^{j}\,d^{ij} \quad \text{where } i=0,..N;\ j = 0,..,i.$$

A recombining binomial lattice must compute and store a total of (N+1)(N+2)/2 prices for the underlying and derivative. For N=500, this requires approximately 10^5 computations, and represents much greater computational overhead than Method 1. This method may require several orders of magnitude of computation than Method 1.

Method 3. Non-Recombining Simulation. If f is a European-style derivative, and the Ito process in Equations (2S), (2f), and (2FP) has non-constant $\mu(t,S)$, non-constant $\sigma(t,S)$, and possibly non-constant interest rate, then Method 2 may not work because the up values and down values of a price movement may not combine: a sequence of up movements followed by down movements are not valued the same as the down movements followed by the up movements. Consequently, in evaluating the possible price of S, after N time increments there are 2^{N+1} possible prices (none are recombined as in Method 2; if recombining is allowed, there are only (N+1)(N+2)/2 prices). In Method 3, where recombining is not possible, all 2^{N+1} possible prices must be generated to get the "complete" distribution for the expected value in Equation (1). Pragmatically, this is impossible, since for n=500, this is approximately 10^{150} prices. The alternative here is to create a representative random "Monte Carlo sample" of f so that the expectation in Equation (1) can be computed directly from the random sample of prices, and not from the complete set of prices. In Method 3, the time required to compute the value of a derivative depends on the number of discrete time units N and the number of Monte Carlo samples M generated for f. Accuracy in the evaluation of f is a statistical problem relating to the standard error of the estimate of the sample mean. Since it is known that the standard error in computing an expectation is proportional to $M^{1/2}$, reduction of the error by a factor of 2 necessitates increasing M by a factor of 4. Consequently, different "variance reduction" techniques could be employed [2]. Note that in using Method 3, a model for S can depend on other Ito processes: for k processes, a complete set of N time samples would require $2^{k(N+1)}$ computations. Method 2 may require several orders of magnitude of computation more than Method 2.

Methods (1), (2), (3) can also be combined. For example, one can value an American-style derivative with stochastic average interest rate and stochastic average volatility by generating Monte Carlo samples for r and σ as input to a recombining binomial lattice for f.

Copyright © 1995 Software Engineering Press

Computational infrastructure is stretched when these three methods are used to value a portfolio of P derivatives. Consequently, the total amount of computation required for a portfolio is proportional to:

P*G, for Method 1
N*P, for Method 2
N*M*P, for Method 3

and, in general, the computation time for each method corresponds to

Method1 << Method 2 << Method 3.

2. Incorporating More Computing Power

There are tradeoffs in model accuracy and computing time in the three above Methods. These model tradeoffs are further compounded by the computational tradeoffs in alternative computing infrastructure. There are several ways of incorporating additional computing power to speed up the computation of derivatives, and the "obvious" answer of "getting a faster computer" may not be obvious, or may even be "obviously wrong." For example:

> "We have alot of programmers who write C applications. We have alot of Unix workstations, but most are efficiently used all day and all night. Our derivative evaluation application is based on Monte Carlo methods, and we need to improve the accuracy without sacrificing time."
>
> "We need to evaluate our very large portfolio in almost real time. We already have a supercomputer but we could use 2 more. Should we buy another million-dollar parallel processor? We have alot of idle workstations."
>
> "We run a lattice-type American-style valuation application each day on my entire inventory. We can do one evaluation each day. We keep getting more clients. Should I go back to a Black-Scholes formula? My application runs on a PC and I do not understand parallel computation. We have no programmers on staff."

The alternative computing topologies considered here are (listed in order of increasing cost):

1. Workstations
2. Faster Workstations
3. Networked ("Clustered") Systems, that could contain workstations, supercomputers, or both.
4. Supercomputers

Their general characteristics are summarized in Figure 1.

	Number of Processors	Speed in MFLOPS	Memory in MBytes	Cost $K
Workstation	1	1-25	32	<10
Faster Workstation	1-4	>25	64	>10
Cluster	>1	>2000	>128	40 -4000
Supercomputer	>1	2000	>128	1000-20000

Figure 1. Alternate Computing Topologies

The problem that we address in this paper is concerned with the cost effective computation of the expected value in Equation (1), with respect to the tradeoffs between Methods 1-3 and the above computing topologies. Note that these alternatives are not mutually exclusive, their boundaries are "fuzzy" and they may be combined.

The essential idea of this paper is that one should not separate the method of computing the expected value in Equation (1) from its ultimate computing topology. Different topologies may be more cost-effective than other topologies. This is a point also made in [1], even though their evaluation was basically concerned with showing the computing potential of the cluster topology, not its cost-benefit tradeoffs with respect to an organization's requirements.

In the following sections, we discuss the cost-benefit issues involved with implementing the above methods for computing Equation (1) on alternate computing topologies. We show how the choice of topology impacts the computing time for a particular example of a time consuming derivative valuation. We conclude by showing how all these factors can be represented as a case-based expert system, which can be used to help an organization assess its computing alternatives.

3. Risk Tradeoffs of Alternative Computing Topologies

The problem is: Given the algorithmic alternatives and parameters G, N, M, P as defined in Section 1, find a computing topology that minimizes the time and cost required for a valid computation. It is convenient to group

Copyright © 1995 Software Engineering Press

the costs into the categories of Opportunity Costs, Infrastructure Costs, and Algorithmic Costs. The first two costs are general and may be applied to any kind of alternative topology problem; Algorithmic Costs are specific to derivative computations. From another perspective, these costs can be used to describe potentially new benefits of changing to an alternative topology: if the business benefit does not outweigh the other costs, then there may be no cost-effective reason to change.

Opportunity Costs. These costs reflect the risks associated with the nature of the routine function of the business. Assessed here are the costs of a late answer, cost of a wrong answer, cost of no answer, and cost of infrastructure breakdown. For example, a fixed income group may require real time evaluation of their entire derivative position 30 minutes before the monthly speech of the Federal Reserve Chairman. If this cannot be done, then there is an opportunity cost.

Infrastructure Cost. These costs reflect the risks associated with maintaining the existing computing infrastructure as well as the additional risks of modifying the infrastructure to a new topology. Assessed here are Client-Server Costs (costs of additional workstations and servers, together with software); Network Costs (costs of network hardware and software); Infrastructure Modification Costs, Runtime Costs; and System Administration Cost.

The cost and benefit tradeoffs can indicate whether "getting a faster computer" presents a good alternative: the network performance impact is almost as great as the computing processing. For example, purchasing a supercomputer may result in slower performance if the network the supercomputer is on is slow or is saturated with traffic. Figure 2 further illustrates the impact of network performance on computation. This table shows, for example, that during the time that one computer is sending another computer 1 Megabyte of data, the other computer could have done over 100 million floating point divides. This latency only gets worse for memory-intensive computation. The derivative evaluation problem is more compute-intensive than memory intensive. On the other hand, some implementations of Method 2 may send large lattices around a network : for N=500, this would amount to about 1 Megabyte.

Ethernet (Network)	To send 1MB requires (sec)...	No. of float divides on 200MHz chip...
Regular Ethernet	0.56	112,000,000
Fast Ethernet	0.056	11,200,000
OC-3	0.051612903	10,322,581
OC-24	0.006451613	1,290,323
SP2	0.033333333	6,666,667
Tightly Coupled (Parallel Processor Backplane)		
100MB/sec bp	0.01	2,000,000
320 MB/sec	0.003125	625,000
640 MB/sec	0.0015625	312,500
1200 MB/sec	0.00078125	156,250

Figure 2. Network Speed vs. Computation

Algorithmic Costs. These costs reflect the risks associated with maintaining the existing computing algorithm as well as the additional risks of modifying and porting the algorithms so they work on the new topology. Assessed here are the costs of optimizing an algorithm. While many compilers offer one such level of optimization, two other levels of analysis should also be performed. On a macro level, there is a cost-benefit analysis involved in determining the best combination of Methods 1-3. This is essentially the job of the model builder. From a micro perspective, there is a degree of algorithm optimization that is orthogonal to that produced by compiler optimizations. One such optimization is concerned with building a parallel version of the algorithm. The idea here is to implement the algorithm in such a way so that n-processors can solve the problem in (1/nth) the time as one processor. Moreover, new processors actually require "supercomputer style" optimizations (such as loop unrolling, blocking, and memory access patterns) to keep data paths efficient.

At this point the tradeoffs between a weakly-coupled parallelism versus a fine-grained parallelism should be addressed.

Method 3 is a problem that can be solved with weakly-coupled parallelism: for example, Monte Carlo samples can be generated on two different processors, f can be evaluated, and the discounted expected value computed on a third processor. The first two processors are totally independent of each other (assuming they both do not generate the same set of "random" samples). Consequently, one can optimally expect a 2:1 speed-up (minus the communication overhead discussed above). Weakly-coupled applications require relatively little effort

Copyright © 1995 Software Engineering Press

in creating a parallel speed-up.

Method 2 is a problem that can be solved with fine-grained parallelism It can be shown that each computation along the diagonal of the lattice can be done in parallel. Consequently, an algorithm can be configured (or "vectorized" by a skilled programmer) that, at time k, computes the values of S and f in the k+1 nodes on k+1 processors (see Figure 3).

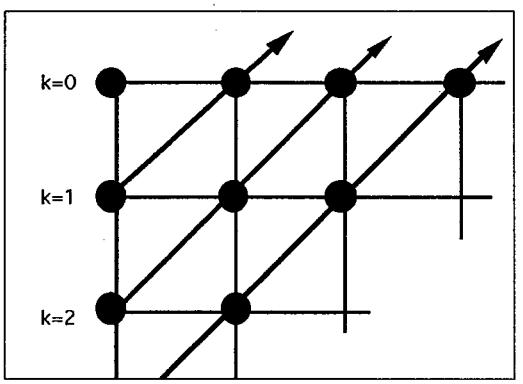


Figure 3. Fine-Grained Parallelism of Recombining Lattice Method

Consequently, if N processors are available, instead of performing (N+1)(N+2)/2 sequential computations, a fine-grained parallel implementation requires only (N+1) sequential computations. Fine-grained parallelization usually requires more effort in modifying the algorithm than weakly-coupled parallelization.

Both weakly-coupled and fine-grained parallelization techniques require a topology to support different parallelization operators. Fine-grained topologies often rely on semaphores, condition variables, and shared memory areas. Some of the operators for the weakly-coupled topologies include:

Broadcast. One processor node sends the same message to other nodes. The simplest broadcast operation is to start running all programs on all nodes.

Scatter. One processor node sends a different message to each node. An example: in Method 3, we can use a scatter operation start running all programs with a different seed to the random number generator.

Gather. Every processor node sends a message to a single member. An example: we gather the Monte Carlo sampled values f for averaging at processor node 0.

Barrier Synchronization. All processor node must reach the same point before any can proceed. An example: in the fine-grained parallelism of Method 2, computation must synchronize for each diagonal to be completed.

4. Evaluating Tradeoffs: An Example

The following problem, using the most compute-intensive aspects of Method 2 and Method 3, was used as a benchmark in evaluating topology tradeoffs. S follows an Ito process with constant μ and σ, and f is an American-style derivative. We use a recombining lattice to find the expected value of f. Next, we vary the average instantaneous interest rate r in by taking 1000 Monte Carlo samples. Thus the value of f is the sample average of 1000 lattice evaluations. The algorithm was implemented to support the weakly-coupled parallelism of Method 3.

We compare the impact of several implementations in Figure 4.

	Time for 1 Sample (sec)	Time for 1000 Samples (sec)	No. of Processors
Workstation	5	5000	1
Faster Workstation	1	1000	1
Cluster (PVM)	3	crashed	4
Cluster (PVM/custom)	3	4-103	1000
Cluster-(PVM/SMP)	0.225	225	4
Supercomputer	0.03	30	40

Figure 4. Benchmark Performance of 6 Computing Topologies

The clusters were implemented under Parallel Virtual Machine, a package that permits the utilization of a heterogeneous network of parallel and serial computers as a single computational resource [7].

The three cluster implementations of the benchmark problem. In the first cluster inplementation, the benchmark problem crashed the system. There were too many Monte Carlo requests for the network task scheduler to handle the barrier operations. In the second cluster inplementation, the problem was reconfigured to allocate one Monte Carlo sample to each processor. The time rquired to perform 1000 samples then depended on the latency of the network: it is variable because the network is a shared resource. In the third cluster inplementation, the network was a dedicated high-speed backplane (see Figure 2). In this "Symmetric Multi-Processing" implementation, only four processors were

Copyright © 1995 Software Engineering Press

allowed to be active at one time.

These results show that the underlying network topology, is the crucial factor in designing cluster computing solutions. Similar results on cluster computing performance are discussed in [8].

5. An Expert System for Assessing Computing Alternatives

We have collected several cases that can be used to assess the transition between alternative computing technologies for the optimal computation of Methods 1-3. There are 16 basic cases, corresponding to the pairwise transitions between each of the 4 topologies, and the null transition — the alternative of keeping the computing topology the same. Our cases were derived by examining similar transition problems for other compute-intensive applications. Our case profiles include the attributes discussed in Section 2, concerned with opportunity cost, infrastructure cost, and algorithmic cost. As in other case-based reasoning systems, our cases contain typical examples and counter-examples (exceptions). We summarize the conclusions of the typical cases:

Case 1. Workstation to Workstation.
Alternatives provide marginal gain in performance. Alternatives are too expensive. No skills to perform algorithm modification. Algorithm is difficult to parallelize.

Case 2. Workstation to Faster Workstation.
No algorithm modification required. Limited Budget.

Case 3. Workstation to Supercomputer .
Algorithm exploits utilization of vectors and vector operations. Budget for the supercomputer is available. Workstations all busy. Bad network infrastructure. Require consistent performance. Skills available to modify algorithm and optimize in FORTRAN. Low modification costs.

Case 4. Workstation to Cluster .
Have many workstations and budget is available to buy more workstations. Other departments will allow limited use of their workstations. Problem cannot be solved with supercomputers.

Case 5. Faster Workstation to Workstation.
Lose of Budget.

Case 6. Faster Workstation to Faster Workstation. Alternatives provide marginal gain in performance. Alternatives too expensive. No skills to do the rehosting. Algorithm is difficult to parallelize.

Case 7. Faster Workstation to Super-computer. Generally same as Case 3.

Case 8. Faster Workstation to Cluster.
Generally same as Case 4.

Case 9. Supercomputer to Workstation.
Lose of Budget. Performance not good enough to continue justification of Supercomputer. Algorithm is too memory-intensive and too large for the Supercomputer. Staff unable to program in FORTRAN to get maximum Supercomputer performance.

Case 10. Super Computer to Faster Workstation. Fast Workstations can provide 50% of Supercomputer performance at 10% of the price.

Case 11. Super Computer to Cluster.
Lose of Budget. Performance not good enough to continue justification of Supercomputer. There are many workstations available. Algorithm is too memory-intensive and too large for Supercomputer. Staff unable to program in FORTRAN to get maximum Supercomputer performance.

Case 12. Super Computer to Super Computer.
Algorithm performance is satisfactory. New algorithm developed for supercomputer will not work on anything else: cost to reimplement is high. New model upgrade costs are low.

Case 13. Cluster to Workstation.
Solution is having a negative impact on business, primarily due to the saturation of the network. Performance at desktop is being hurt. Everyone is getting a workstation to exploit the computing capability.

Case 14. Cluster to Faster Workstation.
Same as Case 13. Can afford more power per desktop.

Case 15. Cluster to Cluster.
Future model of computing topology. Algorithm performance is satisfactory. New model upgrade costs are low. New faster network topologies becoming available.

Copyright © 1995 Software Engineering Press

Case 16. Cluster to Supercomputer.
Solution is having a negative impact on business, primarily due to the saturation of the network. Performance at desktop is being hurt. Everyone is getting a workstation to exploit the computing capability. Algorithm exploits utilization of vectors and vector operations. Budget available. Workstations all busy. Bad network infrastructure. Require consistent performance. Skills available to modify algorithm and optimize in FORTRAN. Low modification costs.

In operation, a problem profile representing attributes relating to the opportunity costs, infrastructure costs, and algorithmic costs are entered in case fields. The expert system then compares each case to the problem profile, and then ranks all cases by similarity.

It seems that as workstation costs decline, the cluster topology becomes more cost effective. However, as seen in the above cases, this alternative is not without problems. A better statement is that as workstation costs and networks improve, the cluster topology will become more cost effective. An important trend that can further improve cost-effective computation is the development of intelligent resource (process and processor) allocation and network load schedulers built into all operating systems ([3], [5],[6]).

6. References

[1] Cagan, L., Carriero, N., and Zenios, S., "A Computer Network Approach to Pricing Mortgage-Backed Securities," **Financial Analysts Journal**, March-April 1993.

[2] Clewlow, L., and Carverhill, A., "Quicker on the Curves." **Risk**, 7(5), May 1994.

[3] Huang, C., and McKinley, P., "Communication Issues in Parallel Computing Across ATM Networks," **IEEE Parallel & Distributed Technology**, Winter 1994.

[4] Ingersoll, J., **Theory of Financial Decision Making**, Rowman & Littlefield, 1987.

[5] Kaplan, J., and Nelson, M., "A Comparison of Queuing, Cluster, and Distributed Computing Systems," NASA Technical Report TM-109025 (Revision 1), June 1994.

[6] Lirov, Y., et al, "Intelligent Infrastructure for the Distributed Front Office," in **Artificial Intelligence in the Capital Markets**, ed. by R.S. Freedman, R. Klein. & J. Lederman, Probus, 1995.

[7] Beguelin, A., et al, **A Users' Guide to PVM Parallel Virtual Machine**, Oak Ridge National Laboratory, U.S. Department of Energy Contract, DE-AC-05-84OR21400.

[8] Anderson, T., et al, "A Case for NOW (Networks of Workstations)," Report for Advanced Research Projects Agency, Contract N00600-93C-2481.

Copyright © 1995 Software Engineering Press

Paper Session: Trading Floor Support

Chair: Yuval Lirov, Lehman Brothers

Intelligent Help for Wall Street

Dimitri Rotov
BFR Systems
31 Clyde Rd.
Somerset, NJ 08873

An extended abstract

1. Introduction

On-line Help facilities provide users with a fast alternative to searching through manuals or calling support lines. This relative quickness, this immediacy, this promise of a quick-fix to an interrupted process, is probably what sells Help systems to budget-conscious software development teams. I like to amuse such teams by telling members that any Help system I design will eliminate the need for telephone support or hardcopy manuals. This is droll only because in our everyday experiences with Help we see that manuals retain the advantage of depth of information and telephone support systems retain the advantage of customizing responses for the user's precise circumstances. And yet, there are no technology limitations enforcing this state of affairs. The better on-line Help is, the fewer the technical support staffing costs and manual development and production costs.

There is a lot of consumer software on the market today that doesn't need Help, or that has Help as a nice-to-have extra. This is software with relatively few features, or simple features, or an obviousness or familiarity that makes Help redundant. Financial software, on the other hand, tends to be feature-rich. Financial calculations need to be subject to "proof" and "examination" (no black boxes, please); scenario manipulations can transform the functionality of a window; outside "feeds" or even products fold into each other; some functions lead to subordinate activities that must be completed before certain tools can be used; etc. Such richness makes Help essential, not peripheral, to financial software. If the software is used under intense time/transaction pressure, with huge amounts of money at stake, the need for first-rate Help is intensified.

2. Roadblocks to "good" Help

Good Help is not easy to find. There are three general conditions limiting Help and keeping it on the margins of its own potential.

(1) Documentation development methods. The generally accepted procedures for document development inhibit the creation of complex Help. There are two pervasive scenarios. Under the first, a project sets up a separate documentation budget, then hires or assigns staff to write manuals and/or Help; writer access to developers' time decreases with the approach of the deliverable deadline. A variant scenario involves developers or programmers writing their own manuals and/or Help at the tail end of a project, using time available (instead of time needed) while improvising a look, feel, consistency, and depth for this documentation. An ambitious Help development methodology would make Help a thoroughly conceived, parallel development effort rather than an ad hoc, adjunct activity. And it would customize Help features rather than rely on existing off-the-shelf packages and standards (see below).

(2) Emerging Help writing tools. These reinforce existing project methodologies

Copyright © 1995 Software Engineering Press

(above) by facilitating Help creation as an adjunct effort. For instance, some Help development software allows conversion of book files (as if an on-line manual equals an on-line Help facility!); some allows codeless Help window creation by any technical writer; some packages, marketed directly to programmers, promise speed, ease and comfort of use -- in other words, a quick end to a dirty little job. The whole range of Help development tools appeal to the developer's ease of use rather than the user's satisfaction: they are "writerly" rather than "readerly" and fall into that bane of good service, the "Easier for Them" syndrome [2], "them" being the providers rather than users.

One sees this most sharply in the limited number of Help features designable with off-the-shelf software and in the tendency to follow the standard of Help seen in Microsoft products. MS Windows Help features and functions are at least one generation removed from state-of-the-art but remain a standard because people encounter them constantly. I am sorry to say they are inadequate for the needs of deep, complex financial software. Sorry, because so many of you must design software to operate on a Windows platform and you do not have a choice in the matter.

(3) Help is passive (dumb). To provide some of the value we get from live, technical support lines, Help needs artificial intelligence.

3. Functions of today's passive Help systems

While financial software gets smarter, Help systems remain simple and passive.

Help systems currently can offer access to manuals (that is, function as on-line viewers); they can offer general (non-specific) information; they can offer lists, tables or other arranged data; and they can offer Help in the context of an activity or process. In the Microsoft paradigm, Help consists of topics accessed through menu selections, or searched for after data entry, or viewed via pop-up displays invoked by selecting "live" window objects. It also includes, if we broaden the meaning of Help, tutorials, demonstrations and "Wizards," which are "interactive assistants" that step users through some part of a process. [2] All of these are somewhat useful, but the (potentially) most useful Help information is about the user's current context. This context-sensitive Help has three aspects: what is it?; how does it work?; how does it relate to my task or objective? [3]. Since this context-based Help is harder to create than the lists, tables, search facilities, etc., it tends, in my buying experience, to get the shortest shrift in commercially available software.

I once had the pleasure of working on a system which offered much context sensitivity. First, the developers ensured every object in the window was "live" and "wired." I then ensured that every field, every label, the title bar of every window, every box line that grouped functions, every spot of real estate in a window, had an ID and a unique bit of Help text. To get Help, a user moved the mouse pointer over any object and clicked the right mouse button. Up would come a window that described the object, told how to use it, and offered jumping-off points to more information. These were not static application windows, either. Some had hundreds of objects, some had multiple "parents" and "children," some had alternate data import sources, and because they involved complex financial scenario analytics, many allowed the user to rearrange column and row headings, to select among alternate calculations or denominations, to activate or decommission functions and then to take the output to another window just as plastic and complex for further work. As massive and powerful as that Help system was, it could only provide limited context-type Help. For instance, on

Copyright © 1995 Software Engineering Press

the window level, it could tell you the purpose of the window; how to use it; what its objects do; where the output goes; what happens if you violate the sequence. Or, on the field level, it could tell you what the field represents; if the field is in a spreadsheet, what its position signifies; what input is allowed and what not; how the data will be used; what step should follow data entry; whether data is optional; etc. This is vastly more context Help than one gets from most programs, and yet, if we honestly consider the needs of the user, it is still not enough.

4. Intelligent Help

Context-sensitive Help can be dramatically helped by AI. Since I am not an AI developer or programmer, my notion of the some of the children of an AI/Help marriage may seem fantastic. I offer them as reasonable from the perspective of a Help developer concerned with the usability of extremely complex software. Consider these possibilities for Help (and consider it *help* in the broadest possible sense):

Process analysis is something that already exists, on a simple level, in some strategic game software. The user elects to be tutored while doing and is rewarded with a stream of advice about the implications of inputs. In other words, there is an expert system effect. This seems easy and reasonable to require in applications that require large amounts of precise numerical data entered in exact order. Some of the content of this kind of Help can include information about input deviations, data completeness, sequence issues, fulfillment of preconditions, data format requirements, and possible next steps. It can prompt next steps, flag active rows and columns, flag active input fields, determine (by activity measurement) whether a task has been interrupted and provide summaries describing work done so far.

Process mapping would provide a thumbnail sketch of where, in a complex process, the user is. For instance, one hypertext software authoring product called Storyscape shows readers (users) diagramatically which piece of text, in a web of textual cross references, they are viewing. The diagram shows where the user came from and what next steps are available. The more variables involved, the more intelligence would be required. This information need not be diagramatic.

Usability analysis would provide quality-of-data information. Developers sometimes provide this in the form of an error message about an outcome: for example, a total must be an integer, or a number cannot be outside a range. A more sophisticated analysis would make a general statement about the usability of an output (what this number will be good for and what not). An even more sophisticated analysis would make general statements about the quality of a number: "This is the highest price allowed by law;" "The size of this spread fails to meet the transaction requirement minimum," etc.

Efficiency measures are unlikely to be popular and are not necessarily intelligent. I note that for every document created in Microsoft Word, there is a statistical summary that includes time spent working on it. This can be useful.

Query interventions (smart queries) recall the HAL computer in 2001: A Space Odyssey. At the high end of sophistication, they would combine process and usability analysis to stop error conditions early on, e.g., "You have not finished entering data and your total already exceeds that allowed." Or they might flag mode issues: "You've entered a prepayment speed appropriate to the PSA model, not the SMM model. Should I change your model selection?" They should not be limited to error detection, however, or they become no more than warning messages. They can initiate Help queries by noticing slow input, frequency of

Copyright © 1995 Software Engineering Press

errors, "erase and redo" activity and other signs of difficulty.

Dynamic formula displays have a number of possible uses. There is tremendous utility in seeing the formula at work that represents the calculations underlying a GUI display. Financial workers who have to justify the numbers they generate have the least incentive to change software, unless they can look "under the hood" of an interface and satisfy themselves. Formula frames might inset themselves into any window displaying a GUI calculator; or they might run in separate windows (like Help text windows); or they might "crawl" across the bottom of the screen like tickertape. Formula displays might populate as data is entered; they might display the subcalculations in a way to allow step-by-step cross-checking; they might lead to Help text with descriptions of formula elements; they might even allow modification of the formulas themselves by the user. After the recent news about the Pentium, checking and verification systems might sell well.

Error Help would use expert system approaches to analyze and recommend corrections to software error conditions. Perhaps time, memory, and storage are issues. And yet it is not unusual to see up to 50 percent of an avionics system dedicated to duplication, self-testing, and error handling.

Some advanced Help features would not need AI:

State information is always useful, and the more dynamic it is, the better. This is sometimes seen as a strip of text at the bottom of a window. Unfortunately, it's often used merely to tell what the software is doing while the user is locked out of the system. State information is always available to X-Windows users, who can open an underlying UNIX system window that will act as a real-time log.(All the X-Windows users I know habitually do this, which testifies to the appeal of this feature.) Some MS-Windows products satisfy this need by providing system information -- Central Point Software's Crash Guard, for example.

Label modification should allow users to relabel objects and for the relabeling to not only stick, but to migrate into the Help and error systems. This accomodates Wall Street's habit of calling the same things by several names.

File naming conventions must somehow transcend the limitations of DOS for those working in DOS or Windows. Perhaps an alias system can be devised that will save analysts from having to use numbers or cryptic abbreviations for all the scenario files they constantly must generate and retrieve.

Stick-on commentary, in the form of virtual post-it notes, is already available as a feature for document viewers and groupware. It needs to be more widely available.

5. Conclusions

The commitment to user Help should increase in proportion to the complexity of a software product. The commitment to helping the user -- in a general sense -- may require junking the old developer-documentor division of labor in favor of a codevelopment partnership. It definitely requires ridding ourselves of bad project habits and the straitjacket of bad paradigms and off-the-shelf Help development software.

6. References

[1] Landow, George P., *Hypertext Theory*, Baltimore and London, Johns Hopkins, 1994

[2] See any work by Paul Fussell for a description of Easier For Them systems.

[3] Fowler, Susan L. and Stanwick, Victor R., *The GUI Style Guide*, Cambridge, MA: Academic Press, 1995

Copyright © 1995 Software Engineering Press

CALYPSO Goes to Wall Street: A Case Study *

Arash Baratloo [†] New York University
Partha Dasgupta [‡] Arizona State University
Zvi M. Kedem [§] New York University
Dmitri Krakovsky [¶] New York University& Lehman Brothers

Abstract

Many computationally intensive problems are parallel in nature. This means that at least theoretically, *parallel solutions can be developed for these problems. A wide range of problems from the fields of scientific computation, databases and financial analysis fall into this category. So why is parallelism so rarely used? It is not the case that parallel processing platforms are economically infeasible: the economic advantages of adapting a network of workstations as a parallel platform are well established. It is the extra cost of* developing *parallel programs that has made this infeasible.*

CALYPSO *is a software system for writing parallel programs, and software support for distributed execution in a network of workstations. Recognizing that the amount of money saved by utilizing free CPU cycles in a network must out-weigh the extra developmental cost, it provides a simple interface for expressing parallelism. It shields the programmer from the nuances of remote execution, data partitioning and synchronization, load balancing, and the dynamic behavior of multiple machines scattered all over the network.*

In this document we briefly introduce CALYPSO *and describe a case study. We start from a sequential program for the calculation of Option-Adjusted-Spread of the corporate bond index, and analyze (a) the effort required to parallelize the program, (b) the performance gained, and (c) the behavior of the system in a network where workstations can arbitrarily slowdown or crash at any time. The measured overhead of a* CALYPSO *program running on six workstations in presence of slowdowns and failures ranged from 7.9% to 16.3%. We finally conclude that parallel applications can be a cost effective solution to coarse-grain computationally intensive problems that exist in many financial applications.*

*This research was partially supported by National Science Foundation under grant numbers CCR-94-11590, and CCR-95-05519.

[†]Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., New York, NY 10012-1185, (212) 998-3350, baratloo@cs.nyu.edu.

[‡]Department of Computer Science, Arizona State University, Tempe, AZ 85287-5406, (602) 965-5583, partha@cs.asu.edu.

[§]Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., New York, NY 10012-1185, (212) 998-3101, kedem@cs.nyu.edu.

[¶]Lehman Brothers Inc., 3 World Financial Center, New York, NY 10285-1100, (212) 526-6731, dkrakovs@cs.nyu.edu.

1 Introduction

In the recent past, networks of low cost workstations and personal computers have become the norm in many academic and corporate institutions, and their number is growing rapidly. Even from their early days, researchers saw a tremendous hidden potential.

1. Networks of workstations are a common commodity: they already exists, have been paid for, and are operational in many institutions.
2. Their aggregate computational power rivals many supercomputers. Furthermore, their cost/performance ratio makes them an attractive alternative to relatively expensive hardware.
3. As many studies have shown [5,6], on average workstations are utilized 15% of the time. Thus, the hidden unutilized computing power that has already been paid for is phenomenal.

Given the previous reasons, then why is it that parallel programs that utilize networks of workstations have not proliferated? A major reason is that the cost to *harness* this power is too high. That is, although networks of workstations are a good value in terms of raw computing power (meaning hardware), the cost to harness this power (meaning software development) still remains high and unattractive.

Although there have been many years of work in providing software toolkits for parallel and distributed programming, it is generally believed that writing a parallel program is still a hard task. This complexity arises because of many reasons:

- The standard programming languages are sequential. Many parallel programming environments require the programmer to learn a

Copyright © 1995 Software Engineering Press

new programming language and a very different computational and execution paradigm. This can be an expensive and a time consuming transition for many corporate organizations.

- For multiple programs (executing at different sites on a network) to work toward a common goal, they must exchange data. Different techniques for data-sharing have large consequences on (a) the performance, (b) the ease of programming and debugging, and (c) the maintenance and portability of the program. It is generally accepted that message passing systems provide the best performance at the expense of programmability: they frequently require extensive changes to a sequential program, and they are also hard to debug. In spite of such drawbacks, PVM [7], for instance, is one of the most popular parallel programming systems for distributed hardware. This demonstrates the acute need for providing such facilities.

- Load balancing is a critical issue in the performance of any parallel application. This becomes even more acute when we consider running a program on multiple workstations with different speeds. In general, a substantial amount of the time and effort of development goes into proper load balancing issues — none of which is required for sequential programs.

- When writing a sequential program, tolerating hardware (the machine itself) or software (the operating system) failures is usually not a consideration. However, in order to allow long-running programs to execute on multiple workstations, some of which we have no control over (e.g. they are located in another office), dealing with failures becomes an important issue. And again, this is an added complexity.

Recognizing that for a parallel programming system to gain acceptance in the "real world", (i.e. outside the research community) we address ease of programmability as well as performance in the building of our prototype.

CALYPSO *is a software system that provides a parallel processing platform on a network of workstations or personal computers.* Several similar systems already exist. However, unlike those systems, CALYPSO provides an exceptionally easy programming interface by shielding the programmer from the nuances mentioned above:

- it extends the C++ programming language by *only four* keywords to express parallelism,

- provides a virtual shared address space to free the programmer from explicit data movement,

- transparently balances the work among available workstations, allowing faster machines to do more work,

- is resilient to machine failures and slowdowns.

By providing these features our goal is to allow high utilization of network of workstations, while allowing the effort to develop parallel programs to remain comparable with their sequential counterparts.

The commercial and the administrative realities prohibit running "private" operating systems for the vast majority of potential users. Thus, CALYPSO *requires no modification to the kernel of the host operating system.* The current prototype runs under SunOS, but the system has been designed and implemented to be portable, and we expect ports to run on most Unix-based operating systems as well as Windows NT.

We feel that many computationally intensive, coarse-grain problems can utilize CALYPSO to harness the computational power hidden in many networks. In the next section we give a brief overview of CALYPSO. In Section 3 we consider a specific problem: Option-Adjusted-Spread of a corporate bond index is a typical example of a coarse-grain computationally intensive problem. In section 5 we report experimental results by parallelizing and measuring the performance of such application. We conclude by summarizing our findings.

2 Overview of CALYPSO

Simplicity is fundamental to CALYPSO. This work has its roots in several years of theoretical and systems research. For the complete history see [4]. Here we briefly describe the CALYPSO Source Language (CSL) used to express parallelism, then proceed to discuss its features which are transparent to the user, and in many instances to the programmer. CSL extends the standard C++ with only four keywords. Once a program has been written, a translator reads in the extended C++ and outputs standard C++. This output is then compiled with a standard compiler, and linked with the CALYPSO library to produce an executable program. That is all: the fact that this program is running on multiple workstations with dynamic behavior is hidden from the user. All the user experiences is a speedup whenever there are idle workstations anywhere in the network. For a complete description of CALYPSO see [4,2,1].

2.1 Language

CSL extends the standard C++ with the following keywords: `shared`, `parbegin`, `parend`, and `routine`.

CALYPSO views the virtual address space of each process as partitioned into two disjoint areas: shared and private. Shared data is used to refer to the section of the memory that is accessed by the concurrent threads of the parallel program. The keyword `shared` is used to declare this region, as shown at the top of Figure 1. In that example, variables `i`, `j`, `k`, and `str` are declared as shared variables. In general, any variable can be declared to be shared.

In CSL, parallelism is expressed by one or more `routine` statements within a `parbegin ...parend`

Copyright © 1995 Software Engineering Press

```
shared {
   int   i, j, k;
   char  str[100];
};

parbegin
   routine[5](int num, int id) { ... standard sequential code ... }
   routine[x+2](int num, int id) { ... standard sequential code ... }
parend;
```

Figure 1: Expressing parallelism in CALYPSO.

structure. The body of a **routine** statement can be any valid C++ code, accessing (reading or writing) either (a) shared variables, (b) any locally declared variables, or (c)the two parameters passed in as arguments to the thread. Consider Figure 1 again. In that example two routines are defined within the parallel step. Five instances of the first routine are spawned in parallel—notice that the number of instances, five, corresponds to the number 5 within the brackets following the keyword **routine**. At runtime each instance of the concurrent thread will get the number of instances (5) and its own instance number (from 0 to 4), as input parameters.

A key point here is to realize that the number of concurrent threads do not dependent on the number of workstations involved in the computation: in the case that there is only one machine working on the problem, the concurrent threads are executed one at a time, sequentially; and when multiple workstations are involved in the computation, CALYPSO distributes the load among all workers, giving faster machines more of the load — transparent load balancing. For example, if 100 computations can be done concurrently, then the program can be written for 100 concurrent computations, and not as a function of available workstations.

2.2 Transparent Features

CALYPSO runs parallel programs on a set of workstations connected by a standard network and running a standard operating system. A CALYPSO computation utilizes a central *manager* process and a dynamically changing set of *worker* processes.

For such a system to be effective, the development of parallel programs must be comparable with their sequential counterparts. A simple programming interface is important for this, but alone, it is not sufficient. There are many programming difficulties that arise in distributed environments that are not present on a single machine. These include partitioning of the workload, data sharing, load balancing and fault-tolerance to name a few. In this section we briefly cover other aspects of CALYPSO that are hidden from the programmer for a good reason: they are considered nuances.

- **Separation of Logical Parallelism from Physical Parallelism:** Programs are logical entities. Thus, the parallelism expressed by a programmer is independent of the parallelism provided by the execution environment, which is tied to the availability of workstations. This mapping between the program parallelism and the execution parallelism is transparent in CALYPSO.

- **Fault Tolerance:** CALYPSO executions are resilient to failures. Worker processes can fail, and possibly recover, at any point without affecting the correctness of the computation. Unlike other fault-tolerant systems, there is no significant additional cost associated with this feature—in the absence of failures, the performance of CALYPSO is comparable to a non-fault-tolerant system. The impact of fault tolerance on performance is discussed in Section 5.

- **Dynamic Load Balancing:** CALYPSO automatically distributes the work load depending on the dynamics of the participating machines. The result is that faster workers do more work than slower workers. Not only there is no cost associated with this feature, but it actually *speeds up the computation*, as fast workers are never blocked waiting for slower workers to finish their work assignments—they bypass the slower ones.

- **High Performance:** While providing the features listed above, our initial experiments indicate that the overhead is surprisingly small for coarse-grained computations. Later, we will provide detailed performance examples using a financial application problem. A sequential C++ program executes in 1730 seconds. With 6 machines, a CSL program for the same problem takes 331.5 seconds. This yields a speedup of 5.2. That is, the *total* runtime overhead of CALYPSO which includes network communication, load balancing, fault tolerance, and other housekeeping tasks is only 14.9%. (This follows as, $(6 \cdot 331.5)/1730 - 1 = 0.14.9$. We elaborate on our performance metrics in Section 5.) In our experiments, under varying conditions of failures, slowdowns and changing number of

Copyright © 1995 Software Engineering Press

available machines, the total overhead varied between 7.9% and 16.3%.

3 Application to Wall Street Problems

Experimenting with CALYPSO we have come to the realization that a number of the problems that require a significant amount of the computational power lend themselves easily to the CALYPSO paradigm and can be solved efficiently and with the minimum programming effort in the CALYPSO framework. The objective of this work was to illustrate how CALYPSO can be utilized for the problems in the realm of finance that are currently considered to be too computationally involved.

One of these problems is the real-time or at least adequately fast computation of the characteristics of the market-weighted averages of the collection of securities grouped by a certain criteria—indices. Given the fact that indices can contain the large number of securities (e.g., there were 3,158 securities in the Lehman Brothers Corporate Bond Index as of March 29, 1995) it is impossible to calculate the statistics of the index sufficiently fast on the currently existing hardware. Particularly complicated and computationally involved are the computations related to the calculation of the Option-Adjusted-Spread (OAS) and the embedded option value of a bond.[1] To illustrate how the performance of these computations can be significantly improved we have implemented the OAS index statistics calculations. We also demonstrate that a remarkable performance improvement can be achieved without incurring much of the additional hardware or programming overhead.

To calculate the OAS we utilized a binomial tree option pricing model as described by Black, Derman, and Toy [3]. Despite the fact that some of the simplifications have been made, we believe that the computational complexity of the problem has not been reduced significantly, especially from the viewpoint of the model's application to the index calculations. The experiments serve as a showcase of the magnitude of the improvement that CALYPSO can provide for the financial problems with substantial computational complexity.

3.1 Overview of Corporate Bond Index Statistics

The following is the data flow of the index statistics calculation.

1. The snapshot of the observed market data is input and analyzed. This data consists of the yields of the risk-free benchmark securities, US Treasury bonds with different maturities, which we call a yield curve. Yield curve, along with the number for the percent volatility of the short rate, serves as an input for the option-pricing model.
2. The option pricing model calculates the value of the OAS for each bond in the index.
3. And finally, the market-weighted average of the resulting OAS and option values are computed.

The pseudocode for the index statistics program is illustrated in Figure 2.

`GetIndexBonds()` routine inputs the information about all of the bonds belonging to the index. The following inputs are collected:

- Coupon
- Observed price of the bond
- Time to maturity
- Coupon schedule
- Type of embedded option
- Redemption schedule

`GetTermStructure()` inputs the array of long rates, that are the yields of the US treasury securities, as well as the volatility of the short rate.

`CalculateShortRateTree()` routine calculates the short rate tree that corresponds to the term structure. The tree has one-month steps and a 30-year horizon.

parbegin ...parend block calculates the OAS for all of the bonds belonging to index in parallel. It spawns as many threads as there are securities in the index and runs Price-To-OAS analysis on them. The `PriceToOAS()` routine, that is called for every bond, takes as input the description of an index bond and the short-rate tree calculated by the `CalculateShortRateTree()` procedure. As the output it produces: option-adjusted-spread, which is a measure of the incremental return provided by a non-benchmark bond as compared to a risk-free treasury benchmark bond and option price, which is the price of the embedded option provisions.

`CalculateMarketStatistics()` calculates market-weighted average of the OASes and option values. It does so according to the following formula:

$$\text{AverageOAS} = \frac{\sum_{i=1}^{n} T_i \cdot (P_i + A_i) \cdot \text{OAS}_i}{\sum_{i=1}^{n} T_i \cdot (P_i + A_i)}.$$

where

- T_i is the total debt outstanding for the i-th bond in the index
- P_i is the price of the i-th bond in the index
- A_i is the accrued interest for the i-th bond in the index
- OAS_i is the option-adjusted-spread of i-th bond.

[1] In the rest of the paper when referring to computing OAS we also refer to the embedded option calculations.

Copyright © 1995 Software Engineering Press

```
IndexBonds = GetIndexBonds();          /* get the bonds in the index                          */

TermStructure = GetTermStructure(); /* input the treasury yield curve and volatility      */

                                       /* calculate short rate tree                          */
ShortRateTree = CalculateShortRateTree(TermStructure);

parbegin                               /* run price-to-OAS for all securities in the index */
     routine[number of bonds in index] {
          Results[num] = PriceToOAS(ShortRateTree, IndexBonds);
     }
parend;
                                       /* calculate the statistics of the index OAS          */
CalculateMarketStatistics(Results, IndexBonds);
```

Figure 2: Pseudo-code for the index statistics program.

3.2 OAS Calculation Model

The option/OAS calculation model implemented in this experiment is a version of the Black-Derman-Toy option pricing model, as described in [8], with minor modifications. This paper assumes the basic familiarity of the reader with the model; here we shall only briefly outline the algorithm of the computations performed.

We implement the model in an imaginary world where we adhere to the simplifying assumptions outlined in [8].

The algorithm for the OAS and option-value calculations is the following:

1. Given the risk-free term structure calculate the short-rate binomial tree T. The length of the tree should be sufficient to accommodate for the length of a longest bond being evaluated, currently we produce a 30-year tree with monthly steps.

2. Initialize OAS to 0.0, and ϵ to 0.0001.

3. Calculate the price $P1$ of a security with embedded option using the short-rate tree T.

4. Compare the $P1$ with the observed (market) price of the security P and if $|P1 - P| \leq \epsilon$ go to step 7.

5. Shift the rates in the short-rate interest tree T by a spread (OAS).

6. Go to step 3.

7. Calculate the price of the BULLET (without option) bond $P2$ with the same characteristics as B, and using the value for OAS as it was calculated in steps 3-6.

8. The difference between the $P1$ and $P2$ is the price of the option $P3$.

3.3 Further Parallelism

Due to the nature of the index calculations, which are characterized by a large number of relatively coarse-grain computations, there was no need to to exploit anything but the highest level of parallelization—one thread was allocated to do all of the calculations for one security. Considering, however, the fact that the same price-to-OAS module could be used in the individual security computations outside of the index context we briefly outline the steps of the algorithm that could have been run in parallel:

- One security's price-to-OAS could have been run on a few workers simultaneously with different OAS assumptions. This would assure faster convergence of the calculated price to the observed price value. This method would parallelize on the iterations.

- Additional parallelization could be achieved by discounting different parts of the tree by the different threads. This would be a parallelization within the same step of the iteration.

4 Application to Finance

The described corporate bond index calculation problem is a real problem dealt with by any major financial firm that calculates the averages of the large collection of securities. The real-life bond OAS calculation problem would become even more time consuming if the following changes were to be made:

- A finer grid tree (ideally with daily nodes) were generated

- Multiple factors to describe the shifts in the yield curve were considered

One could imagine numerous other applications that would significantly benefit from the utilization

Copyright © 1995 Software Engineering Press

of CALYPSO. Given the fact that CALYPSO is mostly geared towards coarse-grain computations, most of these applications would probably be the ones, where there are independent highly-intensive subproblems that may use the same logically shared input data and which outputs are to be combined after all these subproblems have been completed. There are various computationally involved problems that lend themselves easily to such a computational model. To name just a few we would suggest the following:

- Mortgage-Backed-Securities (MBS) OAS calculations
- Collateralized Mortgage Obligation (SMO) OAS calculations
- Total return/return attribution horizon analysis under different yield curve scenarios

5 Performance Experiments

Measuring actual performance of CALYPSO is an ongoing activity. In this section we present performance results obtained by running a parallel program to compute OAS of the corporate bond index. In every execution we bore the cost of fault tolerance and any required load balancing.

Results are shown for the following cases, all for the same CSL program:

1. The program ran on identically behaving free workstations. Here we measured speedups when there were no failures and no slowdowns.
2. The program ran on six free workstations for the first 100 seconds at which time it utilized 100% of the CPU cycles. After which a subset of machines slow down by 50% for the duration of 200 seconds, and then returned to their original state. Here we measured the speedups when transient slowdowns occured, which are common in networks supporting multiple users. It also measured the effectiveness of the load-balancing schemes implemented in CALYPSO.
3. In this experiment we model transient machine availability. The program initially started on a subset of workstations. After 100 seconds, other workstations joined-in the computation, but for only a short period of time: 200 seconds after joining the computation they were crashed. This experiment demonstrates ability to tolerate failures, as well as a measure of effective usage of transient idle workstations.

All times reported are "wall clock" or elapsed times, *not* CPU or virtual times. As the initialization and output of the results are an artifice, the times were measured from the start of the OAS computation to its completion.

In each experiment, we use up to 6 machines from the 3 profiles described below. Whenever a machine is "available" to us, we are charged for its use, regardless of whether we are in fact able to benefit from its work or not. Hence, our results are based on quite conservative assumptions.

For this experiment we define 3 machine *profiles*: Machine A, Machine B and Machine C. Each machine profile a-priori determines its behavior. Here are the descriptions of each profile, see also Figure 3.

Machine A is available to us 100% for the duration of the computation. It is a regular machine, that does not fail or slow down during the execution.

Machine B is available to us 100% for the first 100 seconds, then 50% for the next 200 seconds, and then 100% for the rest of the computation. This models a transient network or machine slowdown.

Machine C is available to us 0% for the first 100 seconds, then 100% for the next 200 seconds, and then 0% for the rest of the computation. This models a transient machine availability, as well as faults.

5.1 Theoretical and Actual Speedup

We now describe our cost model. Each machine profile P, is defined by the function $\mathsf{availability}_P$. Availability is a function of time, and depicts the fraction of the CPU resource available to us from that machine. Thus, the availability of 1 denotes the machine is free and completely available, and the availability of 0 denotes that the machine is unavailable. Then, if the computation lasted for time T, the *work* that the machine gave us is $\int_{t=0}^{T} \mathsf{availability}_P \, dt$. This is the area of the shaded region for the time interval $[0, T]$ in the graphs of Figure 3.

In general we will have several machines in the computation, say n machines with profiles, $P_1, \ldots, P_n$, respectively. Then if a sequential program takes time S to complete and a parallel computation lasts for time T, then the theoretical speedup that we can achieve is given by:

$$\frac{\sum_{i=1}^{n} \int_{t=0}^{T} \mathsf{availability}_{P_i} \, dt}{S}$$

Since machine availability is an external function, we are accounting a "charge" whenever a machine is available–whether we use it effectively or not. Also given this charging method, it is obvious that we account for the all overhead that the system incurs, which includes the time taken to move data between different machines, and any runtime overhead.

5.2 Performance Results

To obtain a fair comparison, we always compare the speedup of a CALYPSO program to what could have been achieved theoretically if every machine cycle were utilized effectively. The sequential program executed

Copyright © 1995 Software Engineering Press

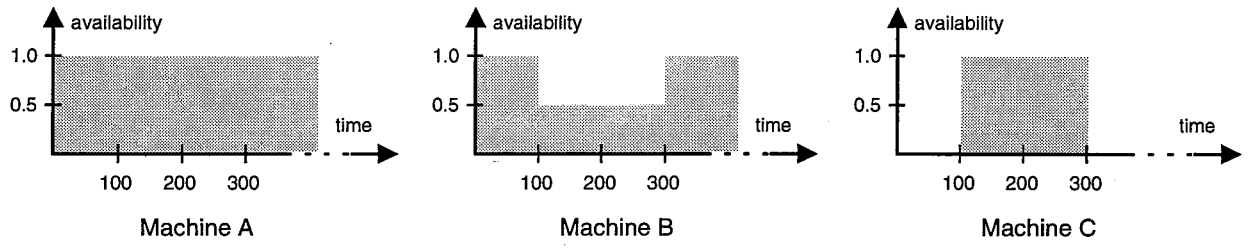


Figure 3: Profiles of available machines.

in 1730 seconds, and we use this number to calculate the theoretical speedup.

We conducted 3 families of experiments, utilizing up to 6 workstations in each experiment, and we describe them in turn. For each experiment the results are graphed showing the total time, the actual speedup and the theoretically best possible speedup. *It should be reiterated again that the same* CALYPSO *program was tested in all experiments, without any (compile- or runtime) alteration.*

The labels on the x-axis of the graphs denote the machines that were used in that experiment. For example, 2A means two machines with profile A, and 4A+2B means four machines with profile A and two machines with profile B were used. The left y-axis indicates execution time, and the right y-axis denotes speedup.

In the first experiment we examine how the performance scales with the number of workstations. We ran the same CALYPSO program on 1 to 6 workstations with profile A, which devote all their resources to the computation. Refer to Figure 4. The execution times for 1, 2, 3, 4, 5 and 6 workstations were 1768, 922, 624.5, 476.5, 392.5, and 331.5 seconds respectively.

This family of experiments show that when there are no failures or slowdowns CALYPSO bears little overhead, even though it is "prepared" to handle such adverse cases.

In the second family of experiments, we measure the speedup when transient slowdowns occur. Here we used 6 different combinations of profiles A and B. See Figure 5. The CPU utilization remains constant independent of the number of machines that slowdown during a computation.

This demonstrates the ability for our runtime system not to "charge" any extra penalty for dynamic behavior. Dynamic behavior is very common in a multi-user network of workstations. In many systems the effort of load balancing is left to the programmer, where as in CALYPSO it is provided transparently, and it has shown to be efficient.

In the final family of experiments we measure the utilization of idle workstations, even if they are idle momentary. In addition, by using different numbers of workstations with profile C we investigate the effect of failure during a computation. See Figure 6. Here the efficiency ranges from 7.7% to 14.2%.

These experiments demonstrate the ability of CALYPSO to utilize resources as they are available, even if availability changes over time.

6 Conclusions

We are working on further enhancements and extensions to the system. We summarize and list some of the important properties of the system, especially as it is related to what has been discovered by us while experimenting with the corporate bond index statistics calculations.

6.1 Performance

The measured overhead is low. This is of a special significance, as CALYPSO is able to execute programs in situations where other systems are inefficient or simply fail to execute. These are not the isolated situations either. We are targeting network of workstations with fluctuating processor loads, network traffic, and even crash-failed processes; if anything this is a conservative view of the real world.

The performance improvement is significant. Bond index OAS calculations achieve near linear speedup, which means that changing the number of workers from m to n, where $n \geq m$, increases the speed of computations by the factor of n/m (assuming that all of the workers have the same computational power).

6.2 Programmability

The programming interface is simple and easy to learn. We converted a sequential program that consisted of nearly 700 lines of C++ code into a parallel program capable of running on a network of workstations by simply modifying 26 lines of code.

- It took us approximately two hours to transform a regular sequential code into a CALYPSO program, which illustrates the ease of programming in this framework. The ease of programming is

Copyright © 1995 Software Engineering Press

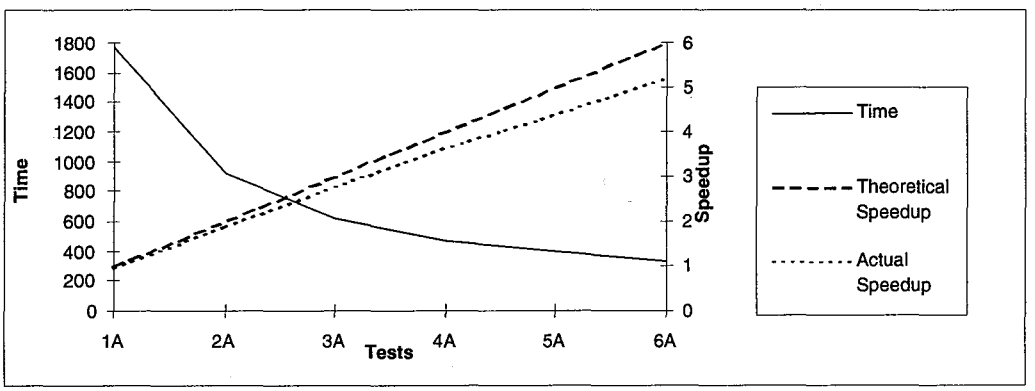


Figure 4: Parallel computation on machines with profile A.

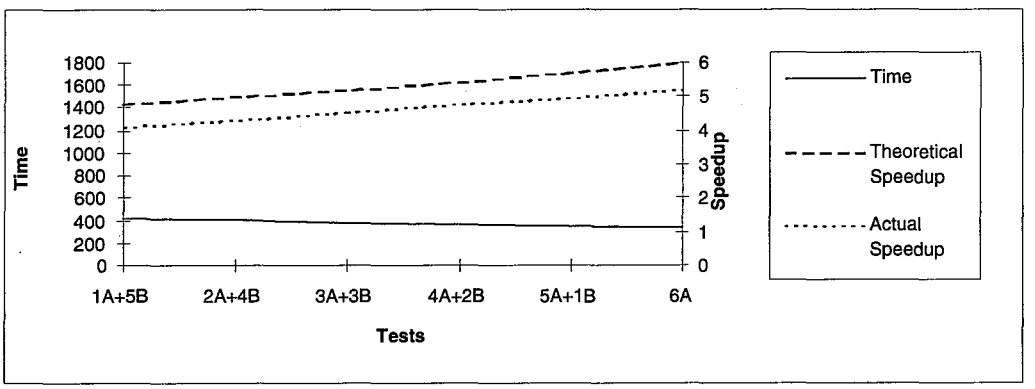


Figure 5: Parallel computation on machines with A & B profiles.

enhanced by the separation of logical and physical parallelism, shared memory model, transparent data movement, and the simple style of expressing the parallelism.

- Another factor is that an extension of the standard C/C++ is utilized. Therefore, programmers do not need to burden themselves by studying a new language or a new environment to improve the performance of their software. What may be even more important, is that a significant number of existing programs are in C/C++ which can be easily converted to take advantage of a substantial boost in performance.

- Shared memory-based programming model and "free" fault tolerance are of great importance. There are various other approaches that try to achieve the same objectives by utilizing Also, heavy-weight processes, file-based interprocess communication and some variation of a scripting languages to glue them together. However, we argue that (a) CALYPSO is much more natural for a programmer to operate in terms memory data structures; with the scripting/file-based communication, a significant amount of memory-to-file and file-to-memory data structure conversion has to be performed unless a language provides built-in persistence mechanism, (b) there is no need to mix different languages (programming language and a scripting language), and finally (c) a programmer using the CALYPSO framework does not concern himself or herself with issues of fault-tolerance and load-balancing.

6.3 Applicability

- Hardware required to solve complex problems becomes much more affordable. Due to the remarkable increase in speed many problems that would otherwise require expensive hardware to

Copyright © 1995 Software Engineering Press

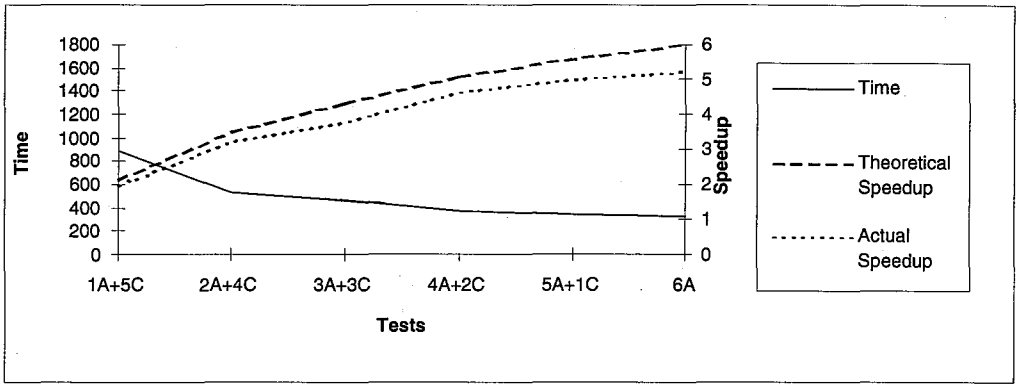


Figure 6: Parallel computation on machines with A & C profiles.

be solved in an adequate amount of time can be solved by a network of inexpensive workstations. Also, since the networking naturally fits the CALYPSO model, even under-powered, low-cost personal computers can contribute their share to computations.

- The configurability of the system is very high. This may mean that users can increase the speed of the computations at the runtime, by adding additional workers. Therefore, users can exercise some control over the speed of the computations even after the computations started.
- Users are given the transparent access to the distributed resources located on the network, which insulates them from dealing with various issues of the network programming.
- The system smoothly and efficiently adapts to fluctuating abilities of workers.

7 Acknowledgments

The authors acknowledge the following for their contributions to the CALYPSO project: Churngwei Chu for recognizing the importance of dynamic termination conditions; Deepak Goyal for augmenting CALYPSO with daemon processes; Mehmet Karaul for developing a graphical monitoring tool; Fabian Monrose for implementing dynamic-memory allocation; Naftali Schwartz for his persistent work on an intelligent program translator; David Stark for advice on key implementation issues; and Jun Xu for implementing a portable GUI.

References

[1] Arash Baratloo, Partha Dasgupta, Mehmet Karaul, Zvi M. Kedem, and Fabian Monrose. CALYPSO *0.9 Manual.* New York University, August 1994.

[2] Arash Baratloo, Partha Dasgupta, and Zvi M. Kedem. CALYPSO: A novel software system for fault-tolerant parallel processing on distributed platforms. Manuscript, February 1995.

[3] Fischer Black, Emmanuel Derman, and William Toy. One-factor model of interest rates and its application to treasury bond options. *Financial Analysts Journal*, pages 33–39, January–February 1990.

[4] P. Dasgupta, Z. M. Kedem, and M. O. Rabin. Parallel processing on networks of workstations: A fault-tolerant, high performance approach. *Proceedings of the 15th Intl. Conf on Distributed Computing Systems, to appear*, June 1995.

[5] David Gelernter and David Kaminsky. Supercomputing out of recycled garbage: Preliminary experience with Piranha. *Sixth ACM International Conference on Supercomputing*, July 1991.

[6] D.A. Nichols. Using idle workstations in a shared computing environment. *ACM Operating Systems Review*, 21(5), 1987.

[7] V.S. Sunderam. PVM: A framework for parallel distributed computing. *Concurrency: Practice and Experience*, 2(4):315–339, 1990.

[8] Tom Windas. *An Introduction to Option-Adjusted-Spread Analysis.* A Bloomberg Magazine Publication, 1990.

Copyright © 1995 Software Engineering Press

Notification and Contact Management for Distributed Systems Support

Boris Grinfeld, Yuval Lirov, Andy Sherman, and Frank Wadelton
Fixed Income Research Infrastructure
Lehman Brothers, Inc.
New York, NY 10285

Abstract

In a large and complex distributed computing environment, the failure of a single resource can impair multiple systems and affect many users dependent upon them. The combined automation of fault detection and fault notification increases the efficacy of the support function while simultaneously reducing its cost. A critical issue is the determination of whom to notify, based on the nature, location, and severity of the failure. BING, an intelligent and reliable contact and notification system, manages rosters of contacts and their relationships to various support entities, and provides email and pager notification, as well as acknowledgment and escalation facilities.

1 Introduction

The increasing complexity of distributed mission critical applications requires a cost-effective approach to systems management. In a distributed environment, the failure of a single shared resource may affect multiple applications and their users. Wherever possible, automated probes monitor the health of systems and the status of processes running on them.

Various support groups, managers, developers, and users require timely failure notification dependingupon the nature of the failure, its severity, and duration.

Traditional notification is simultaneously *too slow* and *too fast* as it reaches the support person last and the end user first. An outage is first noticed by the application end-users, who notify the application developers, who in turn notify the systems or database administrators. We would argue, however, that such an arrangement delays unnecessarily the troubleshooting process. The deployment of three automated systems, namely: *Gryphon*, *Proteus*, and *Bing*, has helped reverse the traditional notification chain. Gryphon monitors for outages (and, as the knowledge base grows, predicts them); Proteus [Desmond, 1994] tracks problem resolution process; and Bing notifies the appropriate parties (see Figure 1). The new arrangement not only expedites problem resolution, but also extends support scope: by the time users are notified, problem resolution has begun and there is a likely schedule for service restoration.

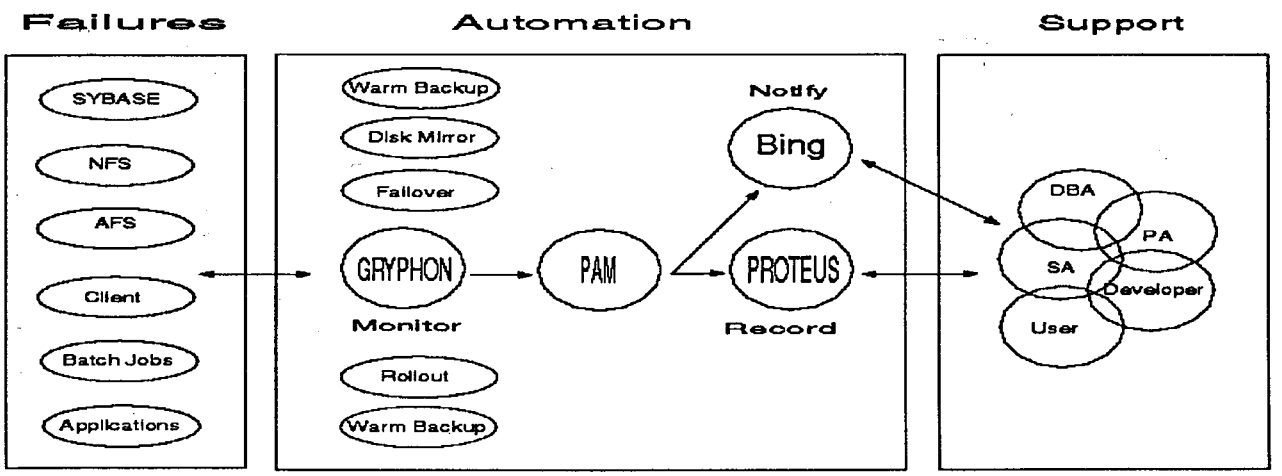


Figure 1: Support Information Flow

Copyright © 1995 Software Engineering Press

The main notification challenge is to correctly notify the responsible people every time. By increasing the set of notified parties, we can guarantee access to the responsible people every time, but we also increase the likelihood of nuisance alerts.

Conversely, by decreasing the set of notified parties, we minimize the likelihood of a mistaken notification, but also reduce the chance to access all needed people. For *precise* management of contact requirements in every crisis scenario, we need to strike a balance between these two conflicting approaches.

Quick and precise management of vast amounts of data is notoriously hard because of the complexity of distributed systems [Lirov *et al*, 1993, 1995]. Worse, the peripatetic nature of support personnel in a distributed computing environment exacerbates the situation. At any time administrators may be away from their desks and fixing something in another part of the building, logged in or not. A "system" now spans thousands of hosts, and hundreds of servers, applications, and support people. Additionally, a lot of support is delivered by telecommuting from home. As a result, all of our support personnel carry alphanumeric pagers, and our notification system is built around them. Finding the right person in a crisis can be a logistical nightmare.

Earlier attempts to address this problem include the system decribed in [Rice 1995]. It consists of two subsystems: monitoring and notification. The monitoring scheme is a proprietary finite state machine, while the notification uses a simple, non-queued method of dialing a static list of numeric pagers. Our approach extends the notification methodology in both the time and space dimensions. First, because of the need to provide coverage at all hours, our system provides for a time dependent notification list. Second, because of the size of our network and the extent of our monitoring, our approach manages the determination of the service group and the person within the group to be notified for particular classes of alarms.

Finally, we use alphanumeric paging to provide as much information as possible in the initial notification, and deliver the pages through a queued mechanism that requires active acknowledment and manages an escalation chain for unacknowledged pages.

Notification knowledge representation maps our understanding of production dependencies between networks, servers, applications, and users. Given outage symptoms, it enables automated deduction of associated support personnel and impaired applications. We approach knowledge acquisition, the traditional bottleneck of artificially intelligent systems, in two phases: manual and automated. We acquire manually the production dependency information for all entities except the hosts. We acquire automatically the host information and their relations to the applications and servers. This paper presents *Bing*, a system that manages both the informational and operational aspects of notification. First, it manages rosters of contacts and their relationships to various supported entities such as hosts, databases, and applications. Second, it facilitates notification by pager and email with acknowledgment and escalation features. The paper proceeds in three major sections. First, a case study will be presented which highlights the need for such a system and some of the complexities in the requirements. Second, a theoretical discussion of the roster management and notification problem is presented. Finally, we discuss the architecture of the Bing system implemented at Lehman Brothers.

2 Systems Administration 24 hours 7 days a week

In the traditional support model, each subnetwork or administrative domain has its own primary and backup system administrators who receive all support calls for that domain regardless of the hour. This leads to a fragmentation of support and difficulties in dealing with vacations, sick days, and multiple concurrent problems. A support model that requires an expert for every problem does not deliver effectively on a bad day, and certainly cannot scale up efficiently.

Copyright © 1995 Software Engineering Press

2.1 A New Paradigm - Team-based Systems Administration

Using a single software template, and only three possible disk configurations makes all of our Sun desktops and servers identical. As hosts and procedures become increasingly uniform, any member of the support team can handle a wide variety of the support requests that come from anywhere in our span of control, and are encouraged to do so. We estimate that over 80% of the routine support requests can be handled by any administrator on the team. If the vast majority of support problems can be handled by any administrator, then there is no need to wake up *every* administrator for night problems every night. Instead, we instituted a rotation, so that *one* administrator was on call each night.

All of our support staff can log into our systems from home, and the administrator on call is issued a portable telephone to answer pages while not at home.

2.2 Roster Management and Notification Requirements for Team-Based Support

With the introduction of an off-hour support rotation, the problem of whom to notify of an event on a given host became time dependent. We have become increasingly dependent upon automated monitoring of our systems, and it is essential that the monitors be able to initiate the proper notification without human intervention. Therefore, we required an additional layer of intelligence above our normal email and alphanumeric paging commands.

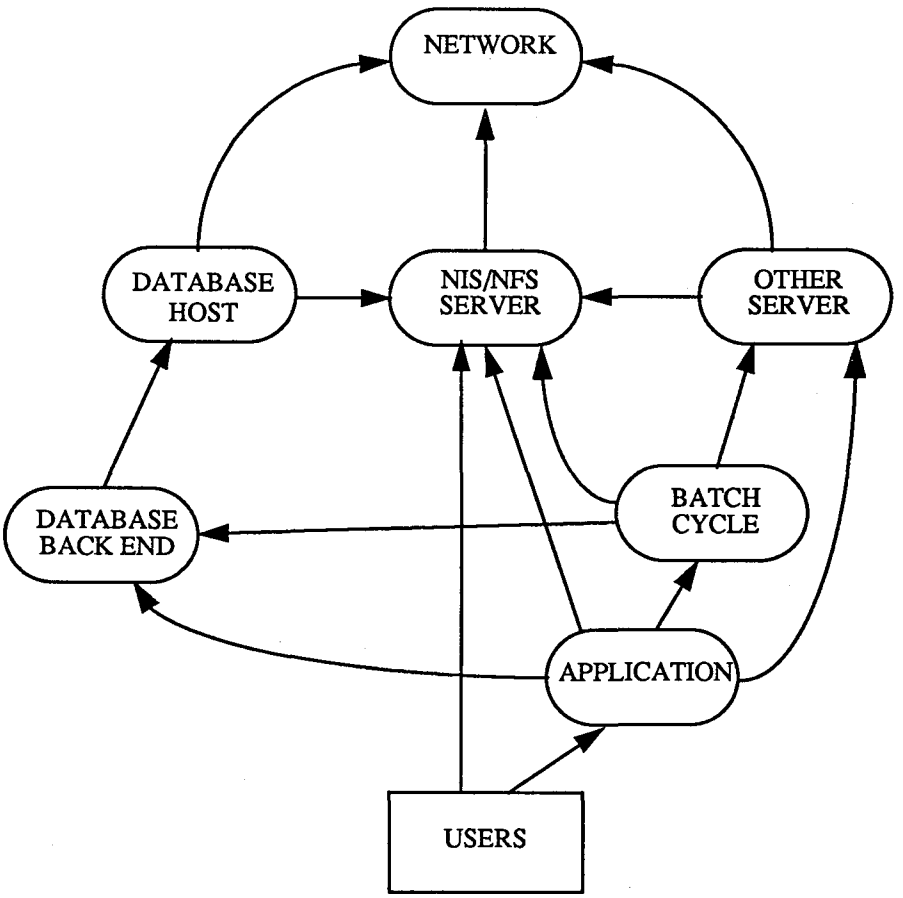


Figure 2: Directed Graph of Support Dependencies

Copyright © 1995 Software Engineering Press

This layer has 5 requirements: daytime notification shall go to the primary SA for the network or host; night time notification shall go to the SA on call; unacknowledged notifications shall be dispatched up an escalation chain, including the backup SA and the SA manager; a "panic button" facility shall notify the entire SA group by day and the off-hour escalation chain by night; the daytime notification shall skip the primary SA the day after that SA was on call.

3 Notification Knowledge Representation

Notification knowledge base operates three basic concepts: groups, entities, and domains. A "group" is a set of login-ids representing an administrative organization or a meaningful collection of users (e.g. using a particular application); sa, dba, and prod are administrative groups. An "entity" is a unit of responsibility (a host, subnet, dbserver, batchlonline application e.t.c.). Groups are responsible for one or more "domains" (i.e. sets of entities for which there are separate duty schedules). The Bing Database contains tables for: information about each contact; primary,secondary admins, domain; work shifts, rotation, etc.; log of pages and when/if acknowledged; on-call admins by domain; entity to entity dependency relations; and company observed holidays. Figure 2 uses a directed graph to illustrate the relationships and dependencies between various entities.

For each responsibility domain there is a dayshift, nightshift, and weekend/holiday shift.

The starting times of the three shifts are kept in the database. Administrators are assigned to entities. These assignments are what the system uses during the dayshift. If there is no assignment, the system looks for a day schedule. For scheduled duty a rotation is kept in the database which is used to generate an initial schedule for the month. At the end of each month the administrative group's manager runs a utility which creates the schedule and outputs a flat file $HOME/domain.Month containing a line for each day of the month with primary and secondary on-call assignments. A utility command allows single entry updates to the database schedule directly.

But, if a significant number of changes are anticipated to the generated schedule, the user can vi the flat file and update from it.

Each of the supported entities (the rounded boxes in figure 2) has its own primary support group. The network domain and the various types of hosts are the responsibility of the Systems Administrators. The database backends are the responsibility of the Database Administrators, the batch cycles are the responsibility of the Production Administrators, and the applications are the responsibilities of their developers. The notification chain for a condition affecting one of the entities on this graph is derived from the dependencies on that entity and the severity of the condition. For example, a hardware failure on a database host will certainly require notification of the SA responsible for the host and the DBA responsible for the database backend (at the time of the error). Depending upon the severity of the error and the length and timing of any downtime, the batch production group, application support group, and the user community may need to be notified.

In general, a notification system provides five basic services: maintain rosters and shift schedules for multiple support and development groups; maintain dependencies (or registration of interest) on supported entities by multiple groups; allow selective notification of a subset of all groups interested in a given entity; distinguish between action notifications (requiring acknowledgment) and information notifications (acknowledgment not required); and log all notifications and acknowledgments.

4 Implementation

Bing, a knowledge-based roster management and notification system is a set of utilities which are wrappers around email and paging commands, or provide for database query and update. Because the system requires high availability, the knowledge base is contained in a Sybase relational database, which is duplicated on a hot standby system and kept in synchronization using periodic transaction log dumps.

Copyright © 1995 Software Engineering Press

In the remainder of this section we first describe the set of knowledge management and notification functions that are currently implemented and then describe work in progress to automate parts of the knowledge acquisition process.

4.1 Knowledge Management

The database contains rosters for the various support groups, including contact information and shift schedule information. For each support group an entity encodes that group's primary responsibility. Characteristics of the entity include identifying information, such as hostname, network address or dataserver name, as well as the primary and secondary daytime contacts. Groups may express interest in an entity that is the primary responsibility of another group, for example, the DBA group is interested in notifications relating to the host for a database backend.

We distinguish between maintenance and notification commands. Maintenance commands allow the support manager to adjust rosters, schedules, and entity contacts, as well as allowing any user to list contact information without actually initiating a notification. The notification commands allow action or information notifications of either all interested groups or some subset. The notification of each group is controlled by that group's contact list and shift schedule. For action notifications, an acknowledgment must be issued by one of the contacts or the notification will escalate through the secondary contact or if necessary the group manager.

The system logs all notifications and acknowledgments. Utilities allow the support manager to scan the log, which provides a convenient summary of what happened, for example, during a particular night shift.

4.2 Automated Knowledge Acquisition

We are currently exploring three avenues for acquiring additional knowledge through automated means. In every case, we are looking for ways to identify the users who are associated with a given entity.

We have focussed on user identification because that population is continually changing, and is difficult to track manually.

An application usage accounting system that tracks usage of various in-house applications serves as a highly accurate knowledge base of who needs to be notified for faults affecting a particular application. Specifically, it contains information ranging from all users who have ever used application X to the set of users who currently have session of application Y running on their workstations.

Another way to associate users with applications is to look at their database backends, which are dedicated to each application. We can log connections to the database by user or host. As with the usage database, this gives us a picture of who is actively using the application at any given time, and, in particular, who will be affected by faults relating to the database backend.

Another important problem is that of defining the community of interest for problems with NFS servers. While everyone with a given server in their automount maps is potentially an interested party, we might really only be concerned with people who actually access it. Using network sniffing software, we can look at what users are generating NFS traffic with a given server, and from where.

4.3 Examples

The system uses two basic commands for notification and acknowledgment: bing, and back. For instance,

```
bing    DEVILS      #server is downunder
bing    sunbond     #has the market crashed
                    or is it just my
application?
back1917            #acknowledged msg from
                    Russia # Revolution
back user frankw boris#acknowledged all
messages
                    from Frank or Boris
```

Copyright © 1995 Software Engineering Press

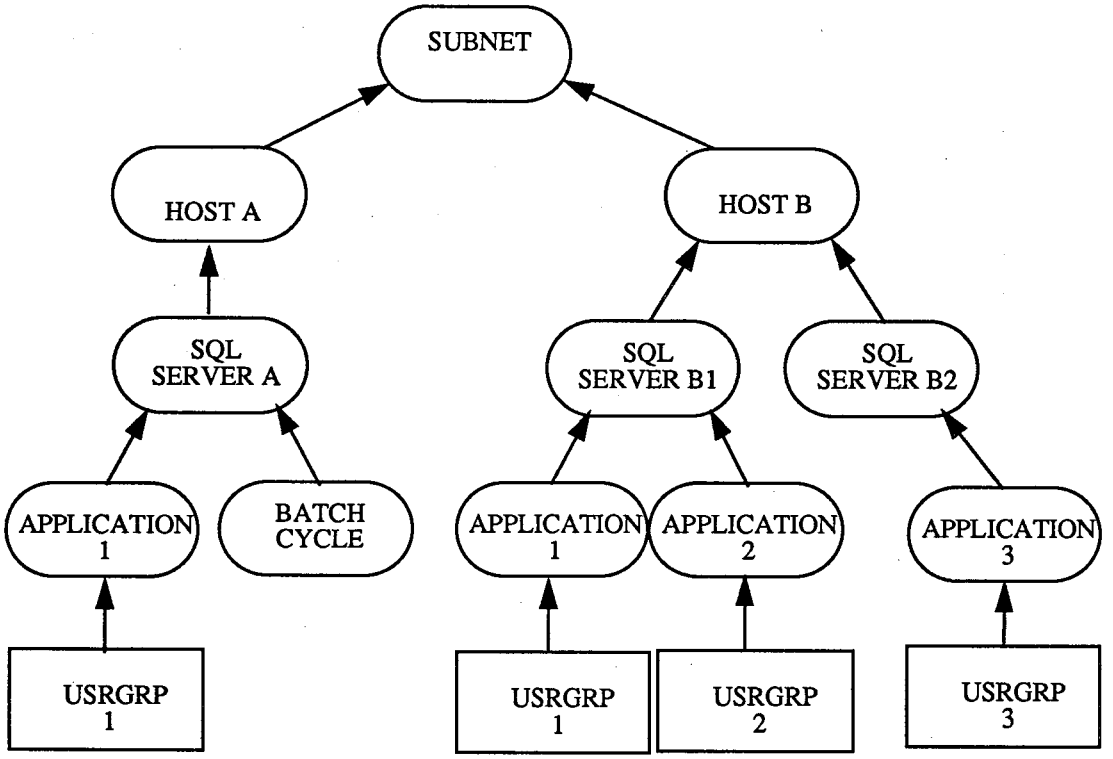


Figure 3: Tree of Entity Dependencies

back admin wizkid #acknowledged all msgs for whizkid

The bing command waits for acknowledgment from an administrator. Calls are acknowledged by running a command with the log number of the message. The log# is obtained by a listing the outstanding messages. A retry time is specified by domain within the database.

The system first tries to contact the primary administrator(s) for the appropriate shift and responsibility entity. After the retry time has expired, it tries the primary again. Then it goes to the secondary administrator(s) on duty. It tries him also a second time and then escalates to the group manager. The system gives up when the manager has been beeped twice and there is still no acknowledgment. Email is sent to an administrator when he is beeped.

Three command line options generalize the functionality of the basic commands in three ways: scope, action, and time.

Specifically, the scope option includes four possibilities: 1 for primary group's administrators, 2 for secondary groups' administrators, 3 for application developers, and 4 for user groups. These values are not exclusive and multiple codes may follow -s.

The action option allows to specify a subset of possible actions (mail, tell, zephyr, etc.). For instance, bing RISKPROD_A -s1 -a13 -s234 -a2 asteroid falling on datacenter, means "notify primary administrators via email and beeper, and secondary administrators, developers, and application users via Zephyr only." The action option enables the Bing user to specify a subset of the full scope of an entity which can also be represented as a subtree of the entity dependency tree rooted at the impaired entity.

Time dependence may be optionally specified. The database contains the days and hours of the week in which various user groups require applications up. The time dependence is represented as a bitmap.

The value of the bitmap for each node is determined by OR-ing together the bitmaps of all the children of the node. A -t option on the command line indicates that the time dependence bitmap should be utilized to filter the dependence relation represented by the entity tree.

Although generally administrators are assigned to entities by day and to a duty schedule by night, the system allows day schedules and night assignments. For some entities an override may be specified for the entity which demands selection of the assigned admin even though for the given domain a duty schedule might normally be consulted.

Suppose that usrgrp1 requires application1 between 9:30AM and 4:30PM and usrgrps2 and 3 require applications2 and 3 respectively between 7:30AM and 5:30PM. host b crashes at 5:00PM. Then, of course, the primary administrators of host b (sa's) would be called regardless of time. Usrgrp1 does not need its application at 5PM but usrgrps 2 and 3 do. So those groups and the administrators of the entities above them constitute the set of potential notification targets in this case. The scope option in the Bing command would indicate which of these to include.

5 Summary

Reliable notification of primary support personnel and interested parties is an essential tool in the management of large distributed computing networks. Team-based support paradigms add complexity to the notification problem because the notification chain is time dependent. In addition, because the work of various support groups is interrelated, events for an entity supported by one group may require notification of the contact list of another.

A set of tools has been developed to manage rosters, entities and relationships, and to perform notification with acknowledgment and escalation. These have been deployed as part of a large distributed systems monitoring environment. Our experience is that this has taken the guesswork out of many notifications scenarios, and has resulted in fewer missed pages and increased reliability and accountability.

Acknowlegment

The authors thank Jeff Borror for driving innovation in support of the business.

References

J. Desmond, "First Application Automates Tracking of Thousands of Support Requests per Month," Application Development Trends, pp. 68-69, December 1994.

Y. Lirov, O. Aloni, B. Grinfeld and A. McMichael, "Embedded Artificial Intelligence for Trading Floor Support", Second International Conference on Artificial Intelligence Applications on Wall Street, 123:130, April 19-22, 1993, New York City.

Y. Lirov, A. Goldberg, and A. Tzvieli, "Intelligent Infrastructure for the Distributed Front Office," Artificial Intelligence in Capital Markets, Chicago: Probus, 1995.

E. H. Rice, "Have Your System Page You Automatically," Unix Review, March 1995.

Copyright © 1995 Software Engineering Press

Intelligent Batch Testing of Distributed Interactive Applications

Aaron Goldberg and Yuval Lirov
Fixed Income Research Infrastructure
Lehman Brothers, Inc.
New York, NY 10285

Abstract

The increasing frequency of new software releases conflicts with the need for stable and reliable systems. A Polymorphic Application Specific Test Encoding Language (PASTEL) captures application level primitives and exploits artificial intelligence to facilitate and expedite release testing. PASTEL descriptions are used to set up tests which replicate full production conditions, running applications across hundreds of workstations and testing difficult to model shared system resources before software reaches the trading floor.

1 Introduction

Computerized trading and sales systems provide investment banks with a critical competitive edge in today's global markets. However, the competitive nature of these dynamic markets sets up a fundamental conflict for applications developers. On the one hand, systems must be absolutely reliable to insure there are no problems during the business day. On the other, systems must continuously change to incorporate new financial instruments, currencies, and analytic techniques. To balance change and reliability, applications developers require advanced release testing technologies that detect problems before software reaches the trading floor. However, full release testing is extremely difficult because of the inherent complexity of distributed systems. For example, the aggregate behavior of the distributed system depends on the individual actions of all its users.

Our goal is to create a system load (Figure 1) that mimics the actions of potentially hundreds of independent human users executing a complex

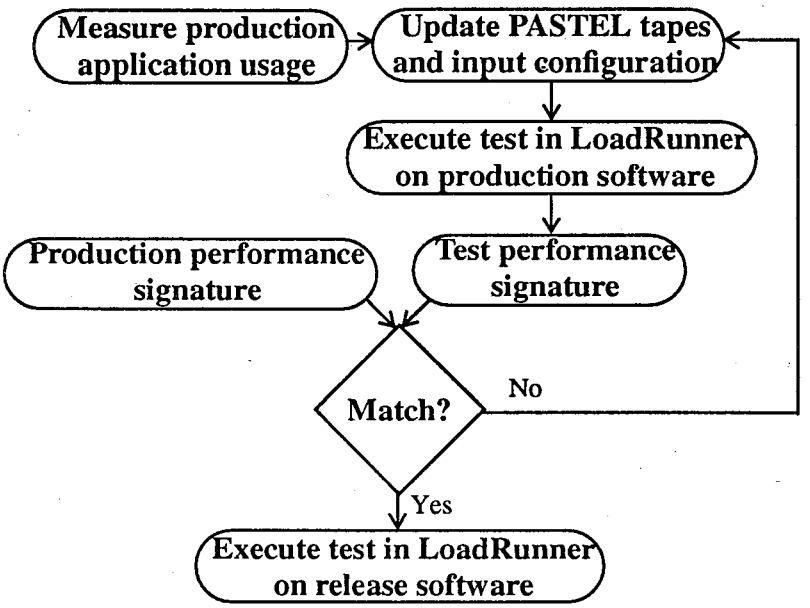


Figure 1: Test Calibration Process

Copyright © 1995 Software Engineering Press

trading system. Ultimately, we use our knowledge representation to map application usage scenarios into computational load patterns, providing a mechanism for validating system performance. We currently have an operational prototype of a batch load testing system that is regularly used in the release cycle to reproduce the workload of fifty distinct users executing different application paths across fifty workstations.

The basic problem is to represent the system under test. We distinguish two kinds of knowledge: local application usage patterns and global resource performance loads The local patterns consist of sequences of elementary application primitives such as trade entry and trade edit. Our PASTEL testing paradigm supports both classes of knowledge. Usage patterns are represented as tapes that contain lists of application primitives. Global resource loads are represented as graphs of resource utilization (performance signatures).

Knowledge acquisition consists of enumerating application primitives, characterizing application usage patterns, and collecting shared resource performance signatures. While primitive enumeration can be performed by human developers and business analysts, the amount of data required both to characterize application usage and to capture a performance signature is too vast for humans to collect manually. Both tasks are automated as described later in the paper.

Our prototype combines four concepts. First, we exploit commercially available Graphical User Interface (GUI) testing software to develop a driver that allows us to execute interactive applications in batch mode. Second, we create a set of application specific test languages to encode our tests. Third, we describe techniques to characterize normal application behavior. And finally, we show how to validate the test, checking that it accurately replicates the real world workload.

2 Remote Batch Test of Interactive Applications

Our implementation relies on the LoadRunner GUI tester from Mercury Interactive. The test designer invokes LoadRunner in learning mode and executes the operator test sequence. In learning mode, LoadRunner acts as a virtual window display server, making note both of user inputs such as mouse movements, button clicks, and keystrokes and of high level application responses like a new window being brought up. LoadRunner automatically stores the sequence of inputs and outputs in a file using a batch scripting language with primitives like `click(MiddleButton)` and `wait_window("File Menu")`. Thus, with the help of the LoadRunner GUI tester, the test designer transforms a high level interactive application test into a batch script. Invoking LoadRunner in batch mode, the learned script can

Table 1: Developing PASTEL for Trading System

Application Primitive	PASTEL Construct
Edit a trade	`EditTrade <Offset>`
Enter a trade	`RegularTrade <Bond Offset> <Price> <Quantity>`
View current yield curve	`YieldCurveRefresh`

Copyright © 1995 Software Engineering Press

be played back, running the application and entering user input at the appropriate points. Further, LoadRunner is designed for remote batch test and it allows the user to designate the set of remote hosts where the test will be run.

3 PASTEL

The key to empowering developers to create their own application tests is developing an encoding specific to the application. We refer to this knowledge representation approach as PASTEL, or Polymorphic Application Specific Test Encoding. The first portion of this section introduces PASTEL via an example. The second portion gives the flavor of the actual implementation.

The Taxable Fixed Income Risk group needed to develop a batch, performance test of their trade entry and risk analysis system. Development began with a 30 minute meeting to discuss the basic primitives of their system. Attendees at the meeting included both the application developers and the Business Analysts (BA's)who support the application on the trading floor. The BA's described their perception of how traders use the system. The application developers helped to collapse the set of primitives by noting user input sequences that were functionally equivalent in the code. This meeting was followed up with a half hour session with one of the lead developers to formalize the basic primitives into an application specific test encoding language. A subset of the primitives selected in the initial meeting and final language developed in conjunction with the developer are shown in Table 1. Because, the developers and business analysts are intimately familiar with their application domain, the knowledge acquisition process is straightforward and inexpensive.

The PASTEL interpreter is written in the Test Scripting Language (TSL) provided with the LoadRunner software. Figure 1 provides example TSL implementations of a primitive and an extract from the main interpreter loop. In addition to the primitives, the prototype interpreter

```
function refresh_yield_curve() {
  # bring up yield curve
  select_display_menu(); move_locator_rel(0, 45, 1); click("Left");
  if (!wait_window (60, "", "Yield Curve", -1,-1,-1,-1)) {
     report_msg("Yield curve window did not come up; Exiting");
     return;
  }
  # press refresh button
 move_locator_abs(675,610,1); click("Left");
  wait(10);
  # dismiss yield curve
  move_locator_abs(1041,610,1); click("Left");
}

while ( getline hostline < tapefile ) {
    fields = split(hostline, hostarr, " ");
    if (hostarr[1] = "EditTrade") {
       edit_trade(hostarr[2]);
    } else if (hostarr[1] = "RegularTrade") {
            regular_trade(hostarr[2], hostarr[3], hostarr[4]);
    } else if (hostarr[1] = "YieldCurveRefresh") {
            refresh_yield_curve();
    } else if .....
```

Figure 2: PASTEL Implementation (primitive and interpreter loop)

Copyright © 1995 Software Engineering Press

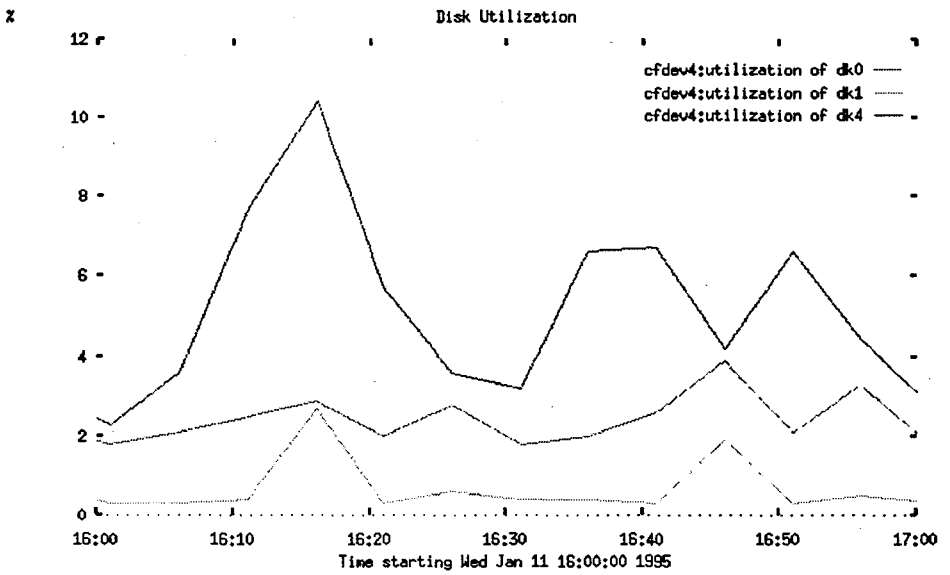


Figure 3: Disk Performance Signature

provides limited control flow and synchronization constructs. PASTEL programs are structured as sets of files called *tapes*, with control flow changes accomplished by executing `RunTape <next-tape-file>` to transfer control to the top of the named *tape*. The prototype interpreter does not yet support a stack so PASTEL does not provide call/return semantics. Synchronization is implemented assuming a shared file system, with a `touch/synch` pair supporting a blocking rendezvous where `touch` creates a file and `synch` waits for the removal of the file. There is also a `delay(seconds)` construct to insert pauses

4 Characterizing Application Behavior

PASTEL provides a platform for executing a sequence of application primitives over a set of hosts. It does not, however, address the problem of characterizing application usage: "Who executes what sequences of primitives at what interval?" The problem is particularly challenging because simple statistical models are inadequate. When an employment number comes out or the Fed announces a rate change, many traders simultaneously access the system To capture these effects, the developers instrument the existing application code to log each time application primitives are invoked by each user. Then, PASTEL tapes are constructed to recreate the actions of specific users at specific times of specific days. The actual load test is driven by a configuration file that has one ordered triple for each test user containing: (hostname, username, initial-tape). Typically, the last two lines of the initial-tape will be

```
synch start_test
RunTape nextape.USERNAME
```

The file `start_test` is deleted to allow the test to proceed, and each user begins executing a designated PASTEL script.

5 Calibrating the Test with Performance Signatures

The technology described in the previous section allows us to execute a full performance test, but how do we calibrate the test, to insure that it does in fact represent the behavior of the true production system? We address this problem with

Copyright © 1995 Software Engineering Press

shared resource *performance signatures*. Our applications all depend on shared resources. In the trading application discussed earlier, the key shared resource is the Sybase dataserver. To capture its performance signature, we collect CPU and Disk Utilization versus time using the standard UNIX `iostat` command as well as information on the number of active client connections. Figure 2 illustrates a typical disk performance signature. Figure 3 shows the calibration process. There are two inputs to the process: the performance signature from the instrumented production dataserver and the test load (PASTEL tapes) that are intended to reproduce this signature. We run the PASTEL tapes under LoadRunner and monitor the dataserver to collect a performance signature. Finally, we compare the production and test signatures, iterating if the match is inadequate. Currently, this is a manual trial and error process, though we expect to apply pattern filtering techniques to compute quantitative measures of closeness of match between the production and test signatures [Leclerc 1994]. Once the match is sufficiently close, the PASTEL workload is applied to the new version of the application software

6 Experience and Future Work

We have presented a new framework for application release testing that reduces the risk of introducing bugs into production software. The PASTEL system represents a significant step forward because it allows the developer to reproduce full distributed production workloads. In a real world trading application, we developed 15 PASTEL primitives with less than three hours of developer interaction, encoded these primitives in perhaps eight more hours, and had the developers running 50 workstation tests on their own. PASTEL is a system where proper knowledge representation empowers developers to test applications code under controlled production conditions.

In practice, PASTEL has isolated two classes of problem and allowed us to quantify performance improvements across releases. First, while developing the primitives, we tried numerous arbitrary interleavings of primitives, encountering one that simply did not work because of a bug in the code. Second, under a full production load, we drove the Sybase dataserver to deadlock in a pre-release version of the application. Finally, by running the workload in a loop for a twenty minute period before each release, we have been able to develop the "Peak trades/minute" metric which allows us to compare performance across releases.

While the PASTEL prototype is in production use, several areas would benefit from improvement. One limitation of the LoadRunner software is that it runs all software as a single UNIX user ID. For our current test, this is not a significant problem, because the application has its own user identification system, disjoint from the UNIX system. However, it does present obstacles to accurately testing certain other Lehman products.

A second limitation is the complexity of implementing a full set of primitives. For example, the 15 primitives available in the system test described in this paper ignore many features of the application. We overcame this shortcoming with human agents: two or three developers exercise bring up the application while the batch test is running to exercise the "exotica".

Acknowledgment

We would like to thank Jeff Borror for an environment that fosters innovative solutions to business problems.

References

F. Leclerc and R. Plamondon, "Automatic Signature Verification: the State of the Art-1989-1993," International Journal of Pattern Recognition and Artificial Intelligence (June 1994) vol.8, no.3, p. 643-60

Y. Lirov, A. Goldberg, and A. Tzvieli, "Intelligent Infrastructure for the Distributed Front Office," Artificial Intelligence in Capital Markets, Chicago: Probus, 1994.

TSL Script Language Reference Manual, Mercury Interactive Corporation, 1993.

Copyright © 1995 Software Engineering Press

INDEX BY AUTHOR

Copyright © 1995 Software Engineering Press

Addendum

Avoiding overfitting by locally matching the noise level of the data

Andreas S. Weigend
Department of Computer Science
and Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430
andreas@cs.colorado.edu*

Morgan Mangeas
Electricité de France, Direction des Etudes et Recherches
1, av. du général de Gaulle, 92141 Clamart, France, and
Department of Computer Science
University of Colorado, Boulder, CO 80309-0430
mangeas@cs.colorado.edu

Abstract When trying to forecast the future behavior of a real-world system, two of the key problems are nonstationarity of the process (e.g., regime switching) and overfitting of the model (particularly serious for noisy processes). This articles shows how *gated experts* can point to solutions to these problems. The architecture, also called *society of experts* and *mixture of experts* consists of a (nonlinear) gating network and several (nonlinear) competing experts. Each expert learns a conditional mean (as usual), but each expert also has its own adaptive width. The gating network learns to assign a probability to each expert that depends on the input.

This article first discusses the assumptions underlying this architecture and derives the weight update rules. It then evaluates the performance of gated experts in comparison to that of single networks, as well as to networks with two outputs, one predicting the mean, the other one the local error bar. This article also investigates the ability of gated experts to discover and characterize underlying the regimes. The results are:

- the gating network discovers the different regimes that underlie the process: the outputs of the gating network segment the data correctly into the different regions
- the widths associated with each expert characterize the sub-processes: i.e., the variances give the expected squared error for each regime
- there is significantly less overfitting compared to single nets, for two reasons: only subsets of the potential inputs are given to the experts and gating network (less of a "curse of dimensionality"), and the experts learn to match their variances to the (local) noise levels, thus only learning as much as the data support.

This article focuses on the architecture and the overfitting problem. Applications to a computer-generated toy problem and the laser data from Santa Fe Competition are given in [Mangeas and Weigend, 1995], and the application to the real-world problem of predicting the electricity demand of France are given in [Mangeas et al., 1995].

1 Introduction

Conventional time series models are global models. They can be linear, assuming that the next is superposition of preceding [Yule, 1927, Chatfield, 1989], or they can be nonlinear, typified as neural networks with hidden units [Lapedes and Farber, 1987, Weigend et al., 1990]. Global models are well suited to problems where the underlying dynamics is stationary.

However, man real-world time series are not stationary, but rather switch between different regimes. For example, the regimes of electricity demand depend on the seasons, and regimes of financial forecasts depend on the economy (e.g., recession or growth) [Granger, 1994, Hamilton, 1994]. Although—in principle—a single global model can emulate any function, including regime switching, in practice it might be very hard to learn. A typical problem in trying to learn regimes with different noise levels by a single network is that the network starts to extract features in some regime that do not generalize well (local overfitting) before it has learned all it could have in another regime (local underfitting).

1.1 Gated experts

We here present a class of models for time series prediction that we call *gated experts.* They were introduced into the connectionist community as *mixture of experts* [Jacobs et al., 1991]; [Rumelhart et al., 1995] use the term *society of experts.* The basic idea behind gated experts is simple: rather than using a global model, we try to learn from the data several local models (*experts*) simultaneously with the splitting of the input space.

*http://www.cs.colorado.edu/~andreas/Home.html

We use the term *gated experts* for nonlinear gated nonlinear experts: the input space can be split nonlinearly by using the hidden units of the gating network, and the sub-processes can be nonlinear through the hidden units of the expert networks. In contrast to related work (e.g., [Hamilton, 1994, Jordan and Jacobs, 1994]) we allow the noise-level parameter associated with each individual expert to adapt separately to the data. Different regimes can thus be approximated with different precision. This turns out to be a new approach to the problem of overfitting, of matching model-complexity to data-complexity.

Apart from excellent predictive performance and robustness against overfitting, gated experts lend themselves to a rigorous statistical interpretation that allows to segment the series and identify the underlying regimes. This approach is more fundamental than previous connectionist methods for segmentation, based on the errors [Elman, 1990], and on the activations of the hidden units [Doutriaux and Zipser, 1990]. In order to achieve reliable convergence and numerical stability, we had to combining the EM algorithm (explained in Section 2.3) with a second-order method for the nonlinear optimization. Given our experience so far, we expect this class of models to scale up well to larger problems.

1.2 Organization of the article

Section 2 gives a mathematical and probabilistic perspective on the architecture, the cost function and the search. Section 3 analyzes why the gated experts help avoid overfitting and compares gated experts with a method to determine *local error bars* introduced in [Weigend and Nix, 1994]. This is done on the task of predicting the electricity demand of France.

1.3 Related Work

The idea of splitting an input space into subspaces is not new. In the time series community, one of the first examples is the *threshold autoregressive* (TAR) model [Tong and Lim, 1980]. In contrast to gated experts, the splits there are very simple and ad hoc; there is no underlying probabilistic interpretation.[1] More closely related to gated experts are the mixture models of the econometrics community [Hamilton, 1990, Hamilton, 1994]. Expressed in connectionist language, the mixture models used there do not have any hidden units: both the gate and all the experts are linear. To our knowledge, neither the double-nonlinear gated experts used here nor the flexible individual noise levels for the different regimes have been used in economics or econometrics [Granger and Teräsvirta, 1993, Hamilton, 1994].[2] The rigorous probabilistic interpretation of the linear gated experts fully generalized to the gated experts discussed here.

An important inspiration for our work has been the introduction of mixture models into the connectionist community by Jacobs, Jordan, Nowlan and Hinton (1991),[3] and the convergence proof [Jordan and Xu, 1995]. [Jordan and Jacobs, 1994] developed a related architecture of a *hierarchical* mixture of *linear* experts (with fixed widths). [Waterhouse and Robinson, 1995] applied this architecture to time series prediction of the sunspots [Weigend et al., 1990, Nowlan and Hinton, 1992] and for nonlinear regression on an example of noise heterogeneity [Weigend and Nix, 1994]. Further related work is [XU, 1994] who applies this architecture to two linear AR(2) processes, and [Müller et al., 1994] who use "hard competition" for a similar task.

Before turning to the theory, we would like to point out that gated experts do not just simply average different "experts:" in contrast to an *additive model* where the weights of the individual predictors are fixed, the outputs of the gating network vary dynamically with the input. This allows the experts to specialize and learn the areas of their responsibility, whereas simple averaging (e.g., as in [Perrone, 1994]) requires all sub-models to be equally responsible over the entire space.

---

[1] TAR models still are quite popular in economics and econometrics. Typically, a cut in one of the input variables is introduced, and two hyperplanes are fitted, each of them to the points in each corresponding subspace. The constraint of continuity across the cut is introduced by hand (whereas it emerges naturally for gated experts). Successful applications of TAR models are typically on problems with relatively few data points ($\mathcal{O}(100)$) and splits now splits are often made in an exogenous variable, such as the volatility [Engle, 1982, Bollerslev, 1986, Bollerslev et al., 1990]. A more flexible model of multivariate adaptive regression splines (MARS) [Friedman, 1991] has recently been applied to forecasting of financial data [Lewis et al., 1994].

[2] The problem of estimating local noise levels is known in the statistics literature as *noise heterogeneity* [Seber and Wild, 1989]. Statistics, however, tends to assume a specific, often rather stringent model for the noise as a function of the input.

[3] Steve Nowlan suggested the application of Gaussian mixture models to time series analysis to us in 1991; the present article summarizes the work done since and includes some of the results presented at an invited talk at *IEEE Workshop on Neural Networks for Signal Processing* and at *Neural Networks in the Capital Markets (NNCM)* in 1994.

2 Theory of Gated Experts

This section describes the ingredients needed to specify a connectionist model: the architecture (network topology and activations functions), the cost function (in a maximum likelihood framework related to an error model), and the search algorithm that minimizes the cost function.

2.1 Architecture

Fig. 1 shows the architecture of the gated experts model. The entire model consists of K experts and one gating network. The task of each expert is to approximate a function over a region of the input space. The task of the gating network is to assign to each input vector one expert. Both the experts and the gating network have access to the inputs. The teacher signal that is directly available is the target (i.e., the next value in time series prediction)—the splitting of the input space is not known. To solve the problem, we need to blend supervised and unsupervised learning: the supervised component learns to predict the next value, and the unsupervised component discovers the (hidden) regimes.

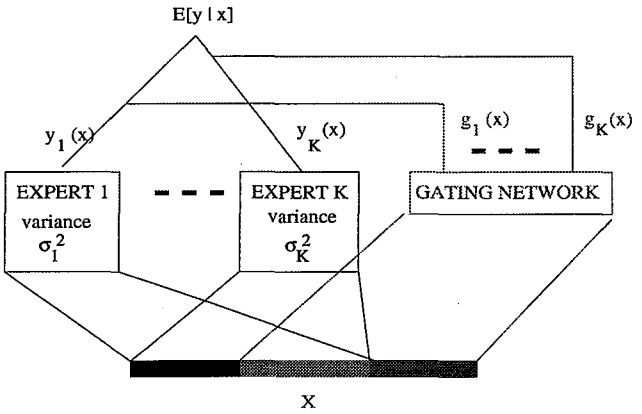


Figure 1: The architecture of gated experts. The inputs $\mathbf{x}$ are at the bottom of the figure. The boxes indicate nonlinear neural networks. The gating outputs $g_i(\mathbf{x})$ weight the expert outputs $y_j(\mathbf{x})$; the expectation value of the output is $\sum_{j=1}^{K} g_j(\mathbf{x}) y_j(\mathbf{x})$.

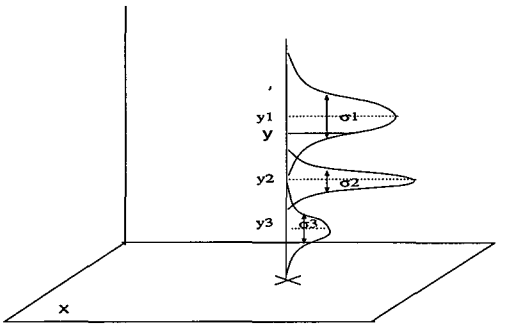


Figure 2: Probability density function given by a mixture of Gaussians (Eq. 6). The outputs of the experts, $y_j(\mathbf{x})$, give the centers of the Gaussians and vary with the input. The widths, indictated by σ_j are indpendent of the input. The three Gaussians densities sketched are multiplied with different gates g_j.

In more detail, expert j learns a function $y_j(\mathbf{x})$, implemented as standard neural network with tanh hidden units and a linear output unit. The output y_j can be interpreted as a parameter of a conditional target distribution. For example, if it parametrizes a Gaussian, y_j corresponds to its mean.[4] The other parameter of a Gaussian, its width σ_j, is a property of the expert; it does not depend on the specific input vector, but it adapts during learning to the noise level in the regime it is an expert for (see Eq. 19).

The gating network has one output for each expert. The goal of output j is to estimate the probability that a given input was generated by expert j. We use *normalized exponentials* (also called "softmax"-units) for the outputs of the gating network to incorporate into the architecture the constraints that the outputs should be positive and sum to unity. The first level of the gating network is an ordinary single network with hidden units. The hidden unit activations ξ are then combined with a weight vector $\mathbf{w}_j$ for each $j = 1, 2, \cdots, K$ into an intermediate activation

$$s_j = \mathbf{w}_j \cdot \xi + c_j \tag{1}$$

where the dot product implies the sum over the hidden units and c_j is a constant ("bias") term. The s_j are then exponentiated and normalized to sum to unity, giving for the weighting of jth expert

$$g_j = \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}} \quad . \tag{2}$$

[4]If there is only a single expert and we assume a Gaussian error model with constant noise level (variance), then this is equivalent to minimizing the squared error between the output and the target value (as can be seen by taking the negative logarithm of the Gaussian) [Rumelhart et al., 1995]. If we allow the width of the Gaussian to become a function of the inputs (e.g., by adding a second output unit to the network to predict the local error bar), we obtain a model for estimating the local noise level [Weigend and Nix, 1994, Nix and Weigend, 1995].

The gating network can be viewed as generating K mutually completing probabilities as a function of the input $\mathbf{x}$. The built-in constraint of outputs of the gating network summing to unity implements a competition between the experts.

Having described the topology and activation functions, we now need to specify the cost function. The next section uses a maximum likelihood framework to derives a cost function.

2.2 Cost function

We begin by defining the variables we use:

- $\mathbf{x}$ is the input vector
- d is the target (or "desired output value")
- y_j is the output of expert j (corresponds to the mean of the Gaussian)[5]
- σ_j is the width of the Gaussian represented by expert j
- $P(Y = y \mid \mathbf{x}, j)$ is the probability density associated with the jth expert for the stochastic variable Y to take the value y
- $g_j(\mathbf{x})$ is the probability the pattern is generated by the jth expert, given the input $\mathbf{x}$
- $h_j(\mathbf{x}, d)$ is the posterior probability that the pattern was generated by the jth expert, given input $\mathbf{x}$ and target d
- j denotes the event that the pattern is generated by the jth expert, $(1 \leq j \leq K)$.

We now make an important assumption: only one expert is responsible for a pattern, i.e., we assume that the events of choosing the experts are mutually exclusive, allowing us to write for the probability of observing the data point d given the input and the model:[6]

$$P(Y = y \mid \mathbf{x}) = \sum_{j=1}^{K} P(y, j \mid \mathbf{x}) = \sum_{j=1}^{K} P(j \mid \mathbf{x}) P(y \mid \mathbf{x}, j) = \sum_{j=1}^{K} g_j(\mathbf{x}) P(y \mid \mathbf{x}, j) \tag{3}$$

where the sum extends over the experts.

Eq. 3 is written in terms of probability distributions. In order to give a single number as "the prediction," we take the expectation value of the probability density.[7] It is given by the linear combination of the expectation values of the individual experts, $y_j = E\left[y \mid \mathbf{x}, j\right]$, weighted by the g_j's:

$$\widehat{y} = \sum_{j=1}^{K} g_j(\mathbf{x})\, y_j(\mathbf{x}) \quad . \tag{4}$$

Note that this model is not included in the usual class of feed-forward networks: the expectation value is a *product* of the outputs of single networks. However, as usual, the $\widehat{y}$ is a deterministic function of the input, and the noise is included in the assumption of an error model [Rumelhart et al., 1995].

We now want to evaluate goodness of the model by how well the data are predicted by the model. To be specific, we now assume each experts to describe a Gaussian. The probability of generating a value y by expert j is then proportional to

$$P(y \mid \mathbf{x}, \theta_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(y - y_j(\mathbf{x}, \theta_j)\right)^2}{2\,\sigma_j^2}\right) \quad . \tag{5}$$

The parameters θ_j and the variance σ_j^2 characterize expert j. P is the probability density of observing a $Y = y$ where Y is a stochastic variable.

[5] In this article the output is a scalar; the generalization to a vector is straightforward.

[6] If k events $(A_i)_{i \in \{1,2,\ldots,k\}}$ are mutually exclusive, then $P(A_i \wedge A_j) = 0$ if $i \neq j$, and $P(A_1, A_2, \ldots, A_K) = \sum_{i=1}^{K} P(A_i)$.

[7] The expectation value is only a good statistic if the distributions is more or less unimodal. If the assumption that each pattern was generated by a single expert is correct, the g's should during learning become binary. In that case only one Gaussian remains active for every data point, and the goal of a point-prediction is well justified. It is a good idea to check the distribution of the g values; if it remains at intermediate levels, there might be a mis-specification of the model (e.g., nothing is gained by splitting the data into different regimes) or the task (e.g., predicting a single value is not appropriate. There are approaches to predicting arbitrary probability densities; the mixture of Gaussians prior to taking the expectation value can be used [Bishop, 1994]; an alternative is the nonparametric "fractional binning" technique [Weigend and Srivastava, 1995].

The probability density of the mixture in response to an input pattern $\mathbf{x}$ is given by the weighted sum of the individual Gaussians (see Fig. 2)

$$\sum_{j=1}^{K} g_j(\mathbf{x}, \theta_g)\, P(y \mid \mathbf{x}, \theta_j) \quad . \tag{6}$$

Assuming statistical independence (the superscript t enumerates the patterns, their total number is N) allows us to obtain of the full likelihood by taking the product of the likelihoods of the individual patterns:

$$\mathcal{L} = \prod_{t=1}^{N} P(Y = d^{(t)} \mid \mathbf{x}^{(t)}) = \prod_{t=1}^{N} \sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g)\, P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_j) \tag{7}$$

$$= \prod_{t=1}^{N} \sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g) \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\,\sigma_j^2}\right) \tag{8}$$

The cost function $\mathcal{C}$ is given by the negative of the logarithm of the likelihood function:

$$\mathcal{C} = -\ln \mathcal{L} = \sum_{t=1}^{N} -\ln \left[\sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g) \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\,\sigma_j^2}\right)\right] \tag{9}$$

Having described the global probability model, we now proceed to the estimation of the parameters $\theta_g, \theta_1, \theta_2, \cdots, \theta_K, \sigma_1, \sigma_2, \cdots, \sigma_K$. The usual way of minimizing the cost function $\mathcal{C}$ with respect to the parameters through gradient descent did not work out here: it turned out to be too hard for simple backpropagation of this cost function to learn at the same time both the individual maps of the experts as well as the splits of the input space through the gating network. Note that the sum inside the logarithm makes the cost function significantly more complicated than in the case of a single network.

Following [Hamilton, 1994, Jordan and Xu, 1995], we now use the *Expectation-Maximization* algorithm to solve the optimization problem. This algorithm is based on the assumption that some binary variables are missing. In our case, the information that is missing is which expert it was that generated a given pattern.

2.3 Search: Expectation Maximization

The cost function Eq. 9 is quite difficult to minimize with backpropagation (gradient descent). However, we can reformulate the problem such that it allows us to apply the *Expectation-Maximization* algorithm (EM). To map the problem onto EM, we first need to identify some missing (or "hidden") variables. We choose as the missing variables the probabilities that a given pattern (t) was generated by expert j; $j = 1, \cdots, K$. Second, to get rid of the awkward sum inside the logarithm, we assume (consistent with Eq.3) that only a single expert generated the pattern; this is implemented by an "indicator variable." We thus choose the missing data to be a set of indicator random variables $\mathcal{Y}_{mis} = \{I_j^{(t)}, j = 1, ..., K, t = 1, ..., N\}$ with

$$I_j^{(t)} = \begin{cases} 1 & \text{if pattern } (t) \text{ is generated from the } j\text{th model} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

This allows us to rewrite the likelihood, replacing that *sum* over experts by a *product* over experts—this is ok since the indicator variable filters out all but the true term.

Now, the problem is that we do not know the values of I_j. This is where the two-step EM algorithm comes in. In the first step (E), we compute the expectation values for I_j (assuming that all the network parameters are known). And in the second step (M), we update our the parameters of the model (assuming that the I_j's are known: we just take those from the E-step).

More formally, we assume that the distribution of the "complete-data" $(\mathcal{Y}, \mathcal{Y}_{mis})$ is given by the following likelihood function:

$$P(\mathcal{Y}, \mathcal{Y}_{mis} \mid \Theta) = \prod_{t=1}^{N} \prod_{j=1}^{K} \left[g_j(\mathbf{x}^{(t)}, \theta_g)\, P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_j)\right]^{I_j^{(t)}} \tag{11}$$

where $\Theta = (\theta_g, \theta_1, \theta_2, \cdots, \theta_K, \sigma_1, \sigma_2, \cdots, \sigma_K)$. So far we have chosen two distribution, one for $\mathcal{Y}_{mis}$ (Eq. 10), and one for $(\mathcal{Y}, \mathcal{Y}_{mis})$ (Eq. 11). Note that when we integrate out $\mathcal{Y}_{mis}$, we obtain the probability of $\mathcal{Y}$ given Θ (Eq. 8) as the marginal distribution, $P(\mathcal{Y} \mid \Theta) = \int P(\mathcal{Y}, \mathcal{Y}_{mis} \mid \Theta)\, d\mathcal{Y}_{mis}$.

Unfortunately, we cannot directly use this new likelihood (Eq. 11) because we do not know the missing variables. So, the idea of the EM algorithm is to replace the missing variables $I^{(t)}$ by their *expectation values* $h_j^{(t)}$ (The superscipt i denotes the iteration number; we iterate back and forth between the E-step and the M-step). These expectation values are computed in the **E-step**:

$$h_j^{(t)} = E\left[I_j^{(t)} \mid \mathcal{Y}, \Theta^{(i)}\right] = P(j \mid \mathbf{x}^{(t)}, d^{(t)}) \tag{12}$$

$$= \frac{P(j, d^{(t)} \mid \mathbf{x}^{(t)})}{P(d^{(t)} \mid \mathbf{x}^{(t)})} = \frac{P(j \mid \mathbf{x}^{(t)})\, P(d^{(t)} \mid \mathbf{x}^{(t)}, j)}{P(d^{(t)} \mid \mathbf{x}^{(t)})} \tag{13}$$

$$= \frac{g_j(\mathbf{x}^{(t)}, \theta_g^{(i)})\, P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_j^{(i)})}{\sum_{k=1}^{K} g_k(\mathbf{x}^{(t)}, \theta_g^{(i)})\, P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_k^{(i)})} \tag{14}$$

Assuming Gaussian distributions for the experts, we can express h entirely through the easily available quantities g, d, and y_j (and the parameters σ and θ):

$$h_j^{(t)} = \frac{g_j(\mathbf{x}^{(t)})\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j^{(t)}\right)^2}{2\sigma_j^2}\right)}{\sum_{k=1}^{K} g_k(\mathbf{x}^{(t)})\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_k^{(t)}\right)^2}{2\sigma_k^2}\right)} \tag{15}$$

Taking the negative logarithm of Eq. 11, and replacing the I's by the h's (i.e., their expectation values) allows us to arrive at the EM cost function:

$$C_{\text{EM}} = -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)} \ln\left[g_j(\mathbf{x}^{(t)}, \theta_g)\, P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_j)\right] \tag{16}$$

$$= -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)} \ln\left[g_j(\mathbf{x}^{(t)}, \theta_g)\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\sigma_j^2}\right)\right] \tag{17}$$

$$= -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)} \left[\ln\left(g_j(\mathbf{x}^{(t)}, \theta_g\right) - \frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\sigma_j^2} - \frac{1}{2}\ln 2\pi\sigma_j^2\right] \tag{18}$$

The **M-step** uses this cost function and adjusts the parameters of the network in order to minimize it.

Specifically, the updates for the variances can be computed directly:

$$\sigma_j^2 := \frac{\sum_{t=1}^{N} h_j^{(t)} \left(d^{(t)} - y_j(\mathbf{x}^{(t)})\right)^2}{\sum_{t=1}^{N} h_j^{(t)}} \quad . \tag{19}$$

The variance of the jth expert is set to a weighted sum of squared errors; the weight is given by the posterior probability that expert j generated that pattern. The denominator normalizes the weightings for that expert.

Since we use nonlinear hidden units, the weights of the networks cannot be computed but are found through gradient techniques. The weight change is proportional to the difference between the desired value d and the expert output y_j,

$$\frac{\partial C_{\text{EM}}^{(t)}}{\partial y_j} = -h_j^{(t)} \frac{1}{\sigma_j^2}\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j^i)\right) \quad . \tag{20}$$

This leaning rule adjusts the parameters such that the expert output y_j moves towards the desired value d. However, note the two factors in front of the usual difference between desired value and prediction:

- The first factor, $h_j^{(t)}$, modulates the weight change proportional to the importance of that expert for the pattern.
- The second factor, $1/\sigma_j^2$, modulates the learning according to the general noise level in the regime of expert j. If the average squared error (Eq. 19) in the regime is large, the influence is scaled down. If the regime is believe to only have little noise, small differences in $(d - y_j)$ are exaggerated by dividing through a small number. This can be interpreted as a form of "weighted regression," increasing the effective learning rate in low-noise regions and reducing it in high-noise regions. As a result, the network emphasizes obtaining small errors on those patterns where it can (low σ^2); it discounts learning patterns for which the expected error is going to be large anyway (large σ^2).

We have found it useful to introduce a lower bound for σ^2; its exact value depends on the specific problem. For the laser data of the Santa Fe competition, for example, we set it to the experimental resolution given by the analog-to-digital converter. This hard limit corresponds to the assumption of a prior distribution for the variance that is flat above the cut-off and zero below the cut-off. Choosing an appropriate prior is an important part of modeling, particularly for short and noisy data sets.

The weight changes of the gating network are proportional to the gradient of the cost function with respect to the intermediate variable s_j (prior to exponentiation and normalization in the "softmax" part, see Eq. 1):

$$\frac{\partial \mathcal{C}_{\mathrm{EM}}^{(t)}}{\partial s_j} = -\left(h_j^{(t)} - g_j(\mathbf{x}^{(t)}, \theta_g^i)\right) \tag{21}$$

This parameters are adjusted such that $P(j \mid \mathbf{x}) = g_j(\mathbf{x}^{(t)}, \theta_g^i)$ gets pulled toward $P(j \mid \mathbf{x}, d) = h_j^{(t)}$ Note the difference between the g's and the h's. h_j is the *posteriori probability* of using the jth expert—its computation uses both input and output information. g_j is only a function of the input; it tries its best to approximate h_j without knowing the target value. In learning the g's move toward the h's; a scatter plot of g_j vs h_j where each pattern gives an entry is a good diagnostic.

In all of our experiments, the gating network and the expert networks are nonlinear, and we use a second-order method to update the parameters in the M-step(the Broyden-Fletcher-Goldfarb-Shanno algorithm, or BFGS, see [Press et al., 1992]). This batch method computes a descent direction as function of the first and second derivatives, and chooses the best step in this direction in order to minimize the cost function.

2.4 Comparison to other cost functions

We close this section by interpreting the cost function that we minimize (for clarity suppressing the implicit dependencies on the parameters, and by comparing it related cost functions. Dropping also the sum over patterns t, i.e., writing it as per-pattern cost function, we started out with a mixture of Gaussians,

$$\mathcal{C}_{\text{Gated Exp}} - \ln\left[\sum_{j=1}^{K} g_j(\mathbf{x}) \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d - y_j(\mathbf{x})\right)^2}{2\,\sigma_j^2}\right)\right] \tag{22}$$

(as the exact cost function), and in the EM implementation after introducing h_j as posterior probability that a pattern was generated by expert j

$$\mathcal{C}_{\mathrm{EM}} = \sum_j -h_j \ln g_j + \frac{1}{2}\sum_j h_j \left[\frac{\left(d - y_j\right)^2}{\sigma_j^2} + \ln\sigma_j^2 + \ln 2\pi\right] \tag{23}$$

The first term here can be viewed as an entropy term. Since g_j gets pulled to h_j (Eq. 21), the term can be approximated by $\sum_j g_j \ln 1/g_j$. This entropy is a measure of "disorder" of the experts: it is cheapest if there is most order, i.e., if only one expert is fully responsible for the pattern. The cost increases if more than one expert gets gated in and reaches is maximum if all $g_j = 1/K$, i.e., if an average over all experts is taken.

The expression in squared brackets, weighted for each experts by its relevance, is identical to the cost function derived in [Weigend and Nix, 1994] for the case of predicting "local error bars" (i.e., of a network with two output units, one for the conditional mean, the other one for the conditional variance),

$$\mathcal{C}_{\mathrm{LEB}} = \frac{1}{2}\left[\frac{\left(d - y(\mathbf{x})\right)^2}{\sigma^2(\mathbf{x})} + \ln\sigma^2(\mathbf{x}) + \ln 2\pi\right] \quad , \tag{24}$$

where LEB stands for "local error bars". This architecture is more complicated in that the variance $\sigma^2(\mathbf{x})$ is an explicit function of the input, and it is more simple in that there is no gating network. Eq. eq:CLEB, and the square bracket in Eq.24 share an important feature: there is a trade-off between the two terms containing σ^2. The squared-error term could be made small by a large value of σ^2, but the cost increases logarithmically with σ^2. The minimum w.r.t. σ^2 just corresponds to setting σ^2 to the expected squared error, see Eq. 19.

A standard least mean square minimization of

$$C_{\text{LMS}} = \frac{1}{2}\left[\frac{\left(d - y_j(\mathbf{x})\right)^2}{\sigma^2} + \ln\sigma^2 + \ln 2\pi\right] \tag{25}$$

assumes that σ^2 is a constant. If we are only interested in finding the minimum, dropping all constants from 25is equivalent to minimizing the squared error, $\left(d - y_j(\mathbf{x})\right)^2$.

3 Avoiding Overfitting by Estimating the Noise Level: Weighted Regression

To investigate the learning dynamics, we compare three architectures: gated experts (Fig. 3), learning the variances (Fig. 4), and a single neural network (Fig. 5) on the problem opf predicting the energy demand of France. Details are given in [Mangeas et al., 1995]. In all cases, we plot as a function of training time (on the same scales) the *normalized mean squared error*

$$E_{\text{NMS}} = \frac{\sum_{t\in\mathcal{T}}(\text{observation}_k - \text{prediction})^2}{\sum_{t\in\mathcal{T}}(\text{observation}_t - \text{mean}_{\mathcal{T}})^2} \tag{26}$$

E_{NMS} compares the performance of the model on set $\mathcal{T}$ to simply predicting the mean on that set.

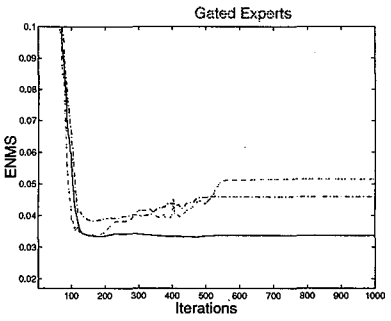


Figure 3: The normalized mean square error E_{NMS} (Eq. 26) as function of training iterations for gated experts. The solid line is the in-sample error (training), the two broken lines are out-of-sample errors.

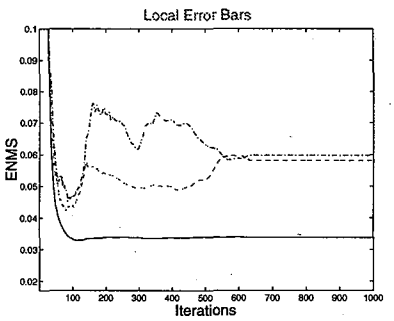


Figure 4: E_{NMS} learning curve for the model that learns both to predict the next value and prediction and the local error bar (variance).

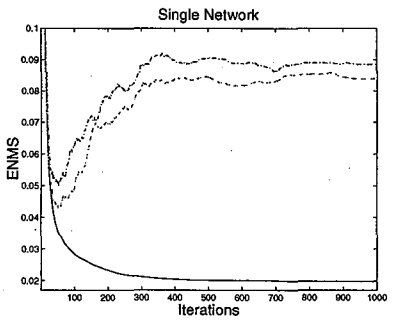


Figure 5: E_{NMS} learning curves for a single network trained by backpropagating the squared errors. In this case the cost function is identical to the plotted E_{NMS} performance.

Note that the in-sample error (solid line) is significantly lower for the single network than for the other two architectures. However, the out-of-sample performances (broken lines) never reach as good minima as the other two methods, Furthermore, the single network overfits significantly worse, i.e., determining the exact stopping point becomes very important.

Analyzing the costs as functions of training time (not shown here) shows significant overfitting on the respective *costs* in all three examples. However, Figs. 3–5 show that the *performance* differs significantly. Whereas the gated experts show very stable learning and do not overfit much (Fig. 3), the local error bar network is somewhat worse (Fig. 4), and the single network (Fig. 5) a lot worse.

It is thus important to distinguish between the full cost (which might include penalty terms, robust errors, etc.) and the performance term we are ultimately interested in (which we take here to be squared error, but it could be anything from percent correct, to the profit a neural network trading strategy makes). In our experience with flexible neural networks and noisy data, the cost function almost always overfits [Weigend, 1994]. The key is to choose the cost function that it learns features that generalize well such that its overfitting has little effect onto the true performance we are interested

in. Distinguishing between the cost function and the performance part is an important degree of freedom in modeling, particularly for short data sets and noisy problems [Weigend et al., 1990].

4 Application to financial data

We have applied gated experts on the financial problem of foreign exchange trading. The main architecture have been outlined here; the size of the data set (both the number of inputs and the record length) was comparable to the problem reported here (and similar to [Weigend et al., 1995]). The key feature we found for the gated experts was that two of the experts become active on about 1/10th of the trading days (active being defined as $g > 0.8$). On those days, their out-of-sample accuracy is larger than 70%. This architecture manages to find regimes where the variance is lower than average, and allows for successful modeling of the dynamics in those lower-noise regimes.

Acknowledgments

Applying a Gaussian mixture models to time series analysis was first suggested in 1991 to Andreas Weigend by Steve Nowlan. This material is based upon work supported by the National Science Foundation under Grant No. RIA ECS-9309786 to Andreas Weigend. Morgan Mangeas is grateful for discussions with Mike Jordan during the summer school of the Electricité de France (EDF) in 1994 and acknowledges support by EDF while visiting the Computer Science Department of the University of Colorado at Boulder.

References

[Bishop, 1994] Bishop, C. M. (1994). Mixture density networks. Technical report, Aston University.

[Bollerslev, 1986] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 21:307–328.

[Bollerslev et al., 1990] Bollerslev, T., Chou, R. Y., Jayaraman, N., and Kroner, K. F. (1990). ARCH modeling in finance: A review of the theory and empirical evidence. *Journal of Econometrics*, 52(1):5–60.

[Chatfield, 1989] Chatfield, C. (1989). *The Analysis of Time Series*. Chapman and Hall, London.

[Doutriaux and Zipser, 1990] Doutriaux, A. and Zipser, D. (1990). Unsupervised discovery of speech segments using recurrent networks. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 303–309, San Fransisco, CA. Morgan Kaufmann.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

[Engle, 1982] Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50:987–1007.

[Friedman, 1991] Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, 19:1–142.

[Granger, 1994] Granger, C. W. J. (1994). Forecasting in economics. In Weigend, A. S. and Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 529–538, Reading, MA. Addison-Wesley.

[Granger and Teräsvirta, 1993] Granger, C. W. J. and Teräsvirta, T. (1993). *Modelling Nonlinear Economic Relationships*. Oxford University Press, Oxford, UK.

[Hamilton, 1990] Hamilton, J. D. (1990). Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45:39–79.

[Hamilton, 1994] Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press, Princeton.

[Jacobs et al., 1991] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

[Jordan and Jacobs, 1994] Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

[Jordan and Xu, 1995] Jordan, M. I. and Xu, L. (1995). Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, (in press).

[Lapedes and Farber, 1987] Lapedes, A. and Farber, R. (1987). Nonlinear signal processing using neural networks. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

[Lewis et al., 1994] Lewis, P. A. W., Ray, B. K., and Stevens, J. G. (1994). Modeling time series using multivariate adaptive regression splines (MARS). In Weigend, A. S. and Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 296–318, Reading, MA. Addison-Wesley.

[Mangeas et al., 1995] Mangeas, M., Muller, C., and Weigend, A. S. (1995). Forecasting electricity demand using a mixture of nonlinear experts. In *World Congress on Neural Networks (WCNN'95)*.

[Mangeas and Weigend, 1995] Mangeas, M. and Weigend, A. S. (1995). First experiments using a mixture of nonlinear experts for time series analysis. In *World Congress on Neural Networks (WCNN'95)*.

[Müller et al., 1994] Müller, K.-R., Kohlmorgen, J., and Pawelzik, K. (1994). Segmentation and identification of switching dynamics with competing neural networks. In *Proceedings of International Conference on Neural Information Processing (ICONIP'94)*, pages 213–218.

[Nix and Weigend, 1995] Nix, D. A. and Weigend, A. S. (1995). Local error bars for nonlinear regression and time series prediction. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7 (NIPS*94)*. MIT Press, Cambridge, MA.

[Nowlan and Hinton, 1992] Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493.

[Perrone, 1994] Perrone, M. P. (1994). General averaging results for complex optimization. In Mozer, M. C., Smolensky, P., Touretzky, D. S., Elman, J. L., and Weigend, A. S., editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 364–371, Hillsdale, NJ. Lawrence Erlbaum Associates.

[Press et al., 1992] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge.

[Rumelhart et al., 1995] Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). Backpropagation: The basic theory. In Chauvin, Y. and Rumelhart, D. E., editors, *Backpropagation: Theory, Architectures, and Applications*, pages 1–??, Hillsdale, NJ. Lawrence Erlbaum Associates.

[Seber and Wild, 1989] Seber, G. A. F. and Wild, C. J. (1989). *Nonlinear Regression*. Wiley, New York.

[Tong and Lim, 1980] Tong, H. and Lim, K. S. (1980). Threshold autoregression, limit cycles and cyclical data. *J. Roy. Stat. Soc. B*, 42:245–292.

[Waterhouse and Robinson, 1995] Waterhouse, S. R. and Robinson, A. J. (1995). Non-linear prediction of acousitc vectors using hierarchical mixture of epxerts. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7 (NIPS*94)*. MIT Press, Cambridge, MA.

[Weigend, 1994] Weigend, A. S. (1994). On overfitting and the effective number of hidden units. In Mozer, M. C., Smolensky, P., Touretzky, D. S., Elman, J. L., and Weigend, A. S., editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 335–342, Hillsdale, NJ. Lawrence Erlbaum Associates.

[Weigend et al., 1990] Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1:193–209.

[Weigend and Nix, 1994] Weigend, A. S. and Nix, D. A. (1994). Predictions with confidence intervals (local error bars). In *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, pages 1207–1212, Seoul, Korea.

[Weigend and Srivastava, 1995] Weigend, A. S. and Srivastava, A. N. (1995). Predicting probability distributions: A connectionist approach. *International Journal of Neural Systems*, 6.

[Weigend et al., 1995] Weigend, A. S., Zimmermann, H., and Neuneier, R. (1995). The observer-observation dilemma in neuro-forecasting: Reliable models from unreliable data through CLEARNING. In *AI Applications on Wall Street.*

[XU, 1994] XU, L. (1994). Signal segmentation by finite mixture model and EM algorithm. In *Proceedings of the 1994 International Symposium on Artificial Neural Networks (ISANN'94)*, pages 453–458, Tainan, Taiwan.

[Yule, 1927] Yule, G. (1927). On a method of investigating periodicity in disturbed series with special reference to wolfer's sunspot numbers. *Phil. Trans. Roy. Soc. London*, A 226:267–298.

The Observer-Observation Dilemma in Neuro-Forecasting: Reliable Models From Unreliable Data Through CLEARNING

Andreas S. Weigend
Department of Computer Science
and Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430
andreas@cs.colorado.edu*

Hans Georg Zimmermann
Ralph Neuneier
Siemens AG, ZFE T SN 4
Otto Hahn Ring 6
D–81739 München, Germany
georg.zimmermann@zfe.siemens.de

This paper introduces the idea of *clearning,* of simultaneously *cleaning* data and *learning* the underlying structure. The cleaning step can be viewed as top-down processing (the model modifies the data), and the learning step can be viewed as bottom-up processing (where the data modifies the model). After discussing the statistical foundation of the proposed method from a maximum likelihood perspective, we apply clearning to a notoriously hard problem where benchmark performances are very well known: the prediction of foreign exchange rates. On the difficult 1993-1994 test period, clearning in conjunction with pruning yields an annualized return between 35 and 40% (out-of-sample), significantly better than an otherwise identical network trained without cleaning. The network was started with 69 inputs and 15 hidden units and ended up with only 39 non-zero weights between inputs and hidden units. The resulting ultra-sparse final architectures obtained with clearning and pruning are immune against overfitting, even on very noisy problems since the cleaned data allow for a simpler model. Apart from the very competitive performance, clearning gives insight into the data: we show how to estimate the overall signal-to-noise ratio of each input variable, and we show that error estimates for each pattern can be used to detect and remove outliers, and to replace missing or corrupted data by cleaned values. Clearning can be used in any nonlinear regression or classification problem.

1 Introduction

Traditionally, observed data are assumed to be "the truth," and model building reduces to data fitting. In contrast, in human reasoning, people constantly use their internal model of the world to (re-)evaluate and possibly discard observations. This can be called the *observer-observation dilemma:* Neglecting the data entirely reduces to dreaming or hallucinating, and neglecting the model entirely without building models and hypotheses prevents us from forming a consistent view of the world, recognizing outliers, etc. Winograd and Flores (1986) nicely describe the use of implicit knowledge and top-down processing in human perception and cognition, and how it could be applied in the computer sciences.

Most mathematical modeling assumes noise-free inputs: the model is built from the data in bottom-up fashion. Our basic assumption is that the data is noisy and the data set size is limited, conditions surely fulfilled when modeling financial data on the time scale or daily data. The method we propose here also includes top-down processing: the emerging model is allowed to modify data if the cost of moving the data is smaller than the gain in cost associated with the output error.

This paper introduces a formalism for building numerical models that uses the top-down information from the model in order to correct outliers and characterize and understand the inputs better. Conceptually, the method consists of two steps:

1. **learning:** use the data to modify the model (structure);

*http://www.cs.colorado.edu/~andreas/Home.html

2. **cleaning:** use the structure to modify the data (observations).

We use the term **clearning** to describe the simultaneous application of both steps in model building. Note the trade-off between the belief in the data and the belief in the model, the *observer-observation dilemma.*

The idea is broadly applicable; as motivation, we give an example of classification, as it occurs in financial decision making. Let us consider the case of two classes (e.g., input patterns of the first class belong to a trending market, input patterns of the second class represent a side market). On the one hand, if these clusters are well separated, it is easy to find a decision boundary. On the other hand, if the clusters overlap, a flexible model will be able to find a complicated boundary that will not generalize well to new patterns. Cleaning the data corresponds to moving patterns on the training set closer to their centers, reducing the overlap on the training set, and allows a simpler decision structure to match the cleaned data.

Although the proposed method is useful for decision making (classification) and portfolio applications, this paper focuses on its application to forecasting or time series prediction, essentially a regression problem. So far, all connectionist models for regression assume noise-free inputs and try to find a regression surface that approximates the targets (desired values) associated with the inputs as well as possible. The standard approach breaks the potential symmetry between inputs and outputs. Particularly in univariate time series prediction where the inputs are just lagged values of the output, this assumption is clearly inconsistent.

When trying to work with financial data, there are several sources of errors. One source is the entering of the data (such as wrong numbers, sometimes as blatant outliers), or unreliable timing in quotes (no longer tradable). Even if the available data are entered correctly, they might be poor indicators for the underlying economic processes (e.g., industrial production and unemployment). Another source stems from the uncertainty in measuring quantities such as the GNP or the tax revenue, quantities that are difficult to assess and typically revised. Finally, there always are external influences that are not captured by the inputs into the model. They show up as noise.

Having very noisy data and very flexible methods (such as neural networks) is potentially a dangerous combination: if the model is too flexible, it will not only model the signal but also the noise, and yield poor out-of-sample performance. In particular, outliers absorb resources: In case of an outlier, the network moves a hidden unit to the outlier in order to reduce the error. As a consequence, there are not enough resources left to approximate the true structure in lower-noise regions. The many attempts of applying neural networks to financial problems have made one fact clear, that the problem of controlling the flexibility of the model is central.

In order to obtain good generalization (out-of-sample performance) on problems with finite, noisy data sets, such flexible models require regularization. We briefly describe some of the methods that are useful on financial data. One of them, pruning, will play a crucial role in combination with clearning (discussed in the Section 2).

- **Stop early.** Backpropagation is an iterative procedure: the complexity of the model gradually increases with training time [Weigend, 1994]. Starting training with small weights and stopping early introduces a preference for linear models since the weights do not have enough time to grow large enough to express significant nonlinearities. This can be a serious problem when trying to find nonlinear structure in noisy data. In any case, we always monitor an error on a cross-validation set; when it starts going up as function of training time, we begin to bring in some of the other techniques.
- **Penalize network complexity.** Adding a complexity term to the cost function that effectively counts the number of significantly sized weights is known as *weight-elimination* [Weigend et al., 1990]. This method treats the weights as independent; it has first been applied to financial data in [Weigend et al., 1991].
- **Prune weights.** To evaluate the reliability of the information coded in a weight, we use the size of the weight relative to the standard deviation of its fluctuations. (The fluctuations occur in response to the training inputs on a pattern-by-pattern basis.) We start with an oversized network, rank all weights in terms of a test statistic (Eq.9), and remove those of low significance in order to obtain a sparse network topology [Finnoff et al., 1993]. The pruning step is performed in conjunction with early stopping. This pruning limits the ability of the network to memorize the training data without introducing a bias towards linear models—this is an important distinction to the use of early stopping alone [Weigend and LeBaron, 1994].

- **Neuro-fuzzy methods.** Constraints, extracted from humans in the form of logical rules, can be used to constrain the network architecture and learning. In terms of overfitting problems, this corresponds to building a model that is resistant against outliers. A recent example is the insertion of priors that describe the derivative of the outputs with respect the inputs [Neuneier and Zimmermann, 1995].
- **Hints.** [Abu-Mostafa, 1995] gives the example of the symmetry-hint for predicting foreign exchange data. This hint corresponds to viewing exchange rate returns first from one country, then from the other country. The hint suggests that the dynamics should be the same. During training, the cost functions switches back and forth between gradient descent in the performance cost function (i.e., learning to predict the returns), and learning the hint (i.e., learning to minimize the difference in response to a pattern and to the flipped version of the pattern).
- **Pseudo-data: add noise to inputs that reflect our belief in their accuracy.** The idea is best explained through an example. Let us assume that we believe that the response of the network to a certain pattern should be the same also when the pattern is slightly compressed or stretched in time. Backpropagation is an iterative procedure; at each presentation of a training pattern, we allow for some stretching or compressing of the pattern by repeating or dropping an observation of the time series with a certain probability. The network thus learns to also recognize stretched and compressed versions of the training patterns. It will subsequently generalize better on the test set if this belief about the data is indeed correct. The difference between hints and pseudo-data is that any input vector can be used to descend on the hint, whereas pseudo-data tend to stay close to the actual data since the added noise has them explore the vicinity of the data points [Weigend, 1995].

Our approach here is quite different.[1] Rather than adding different random noise to the inputs at each training iteration in order to prevent the network to overlearn outliers, we prevent the network from overfitting by continually moving the inputs to a more likely point by using information from the observed output and the model. There are two assumptions involved. First, that there is a clean or "true" input value. This gives us the hope of building a better model, i.e., a model with less stochasticity. Second, that cleaning moves the inputs closer to that true value; we will show in the next section how this is achieved with gradient descent in a backpropagation framework. Note that during training, we both use the data to adjust the model and the model to adjust the data: cleaning is only possible through a model.

2 Clearning = Cleaning and Learning

We here derive the regression case; the method can also be applied to the classification case (using sigmoid outputs for predicting the probability of increase of the price of an asset), and to the portfolio case (using normalized exponentials). Suppressing pattern indices, the total per-pattern cost is given by the sum of two terms,

$$E = \frac{1}{2}\eta \left(y - y^d\right)^2 + \frac{1}{2}\kappa \left(x - x^d\right)^2 . \tag{1}$$

The first term, $E^y = \frac{1}{2}\eta \left(y - y^d\right)^2$, is the usual squared error term between network output

$$y = y(x, w) \tag{2}$$

and the desired value of the output y^d. (The symbol w denotes the vector of model parameters.) The second term, $E^x = \frac{1}{2}\kappa \left(x - x^d\right)^2$, is the squared deviation between the cleaned input x and the data input x^d.

There are two sets of **update rules**, the update rules for the weights, and the update rules for the input values. The gradient descent update rule for the weights is identical to standard backpropagation

$$w_{i+1} = w_i - \frac{\partial E}{\partial w} = w_i - \eta \left(y - y^d\right) \frac{\partial y}{\partial w} \quad . \tag{3}$$

The update rule for the cleaned input x_i at iteration i is given by

$$x_{i+1} = x_i - \frac{\partial E}{\partial x} \quad . \tag{4}$$

[1] This approach can be related to the method of *total least squares* [Huffel and Vanderwalle, 1991] (in the context of linear models), to *error in variables* [Seber and Wild, 1989] and to methods dealing with *missing data* [Buntine and Weigend, 1991, Tresp et al., 1994].

Rewriting x_i as a sum of the original data point x^d and a correction term (Δ_i, also at iteration i),

$$x_i = x^d + \Delta_i \quad , \tag{5}$$

the update rule can be expressed most easily as

$$\Delta_{i+1} = (1 - \kappa)\Delta_i - \eta \left(y - y^d\right) \frac{\partial y}{\partial x} \quad . \tag{6}$$

The elements of the update rule for the input correction term are:

- Exponential decay of Δ.
  Without new impulses, Δ shrinks back to zero, proportional to $1 - \kappa$. ($0 \leq \kappa \leq 1$).
- Proportionality to the output error $\left(y - y^d\right)$.
  This is the same proportionality as in "normal" error backpropagation: the larger the deviation, the larger its effect on the update (in this case on the cleaning). Note that η enters here since it describes the scale of the output error.
- Proportionality to the sensitivity of the output with respect to the input, $\partial y / \partial x$.
  This quantity describes the slope (gradient) of the surface at the present operating point x (the cleaned value–the network does not see x^d any more). (This quantity is already computed in error backpropagation.)

Note that there are two step-sizes involved, the learning rate η, and the cleaning rate κ. From the perspective of classical mechanics, they can be interpreted as spring constants (E = $\frac{1}{2}k\Delta^2$); from the perspective of statistics as the inverse of a noise variance.

A **mechanical interpretation** of the cost function and the (relative) learning and cleaning rates is given in Fig. 1. The standard case (without cleaning) can be viewed in the following way. The data points are put in the (input $\times$ output) space. The regression output (network response) can be viewed as a surface above the input space. The data points are vertically attached to the surface with springs; these springs store the (internal) energy. The model complexity lies in the trade-off between the stiffness of the regression surface and the stiffness η of the springs. In the one extreme, an infinitely flexible model would just go through all of the data points. On the other extreme, infinitely weak springs would not modify the model from its prior value (e.g., from a hyperplane).

The new addition are the springs in the input space, between each input data point x^d and its cleaned value x. The energy stored in that spring is $\frac{1}{2}\kappa\Delta^2$. κ is the spring constant and $\Delta = x^d - x$ the amount the spring is stretched. Minimizing the total cost function (Eq. 1) corresponds to minimizing the total energy stored in the input springs and the output springs (averaged over all patterns). The ratio between η and κ describes the trade off between the stiffnesses (or importances) between the output errors and input errors.

A **statistical interpretation** of the cost function can be given in a maximum likelihood framework [Rumelhart et al., 1995]. We assume that each pattern was generated by a "true" input (estimated by x) and a "true" output (estimated by y). We then assume that it is corrupted by Gaussian noise (additive to all the inputs and all the outputs, independent in each component). This statistical interpretation, in conjunction with the model that we have obtained, allows us to characterize the inputs by their noise levels. The total variances of the inputs (i.e., sum of the noise and the signal) are easily computed (and, in connectionist modeling, used to scale the inputs). The present method allows us to estimate the two parts (the noise level and the signal level) separately[2] by using the model that we have learned already. We record the squared deviations $\Delta^2(t)$ of each input as functions of the pattern index t (the time when each prediction is made). This allows us to gain insights that cannot be obtained otherwise into the process:

- **Detect outliers.** Plotting the squared errors of all the inputs and the outputs for each pattern as a function of time (like a spectrogram) allows us to extract three signatures: Individual spikes point to a typo-like error. Horizontal bars points to an outlier in the input that has been generated by a smoothing operation (e.g., a moving

[2]This is in principle an ill-defined problem: there are many ways of writing one number as the sum of two numbers (Jerry Friedman, public communication, NIPS'90). However, since the final model is quite respectable asjudged from the out-of-sample performance, the estimates of the noise levels are also reasonable.

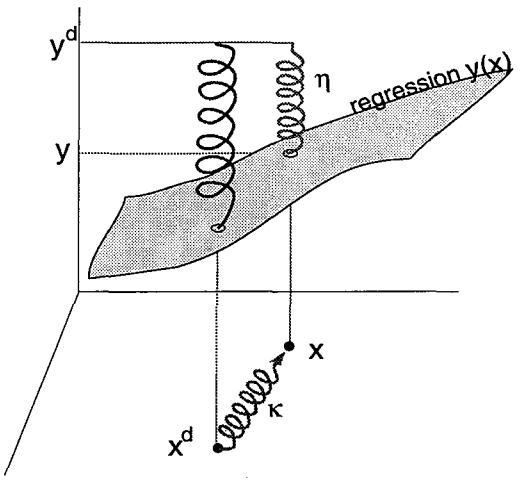


Figure 1: Mechanical analogy of the trade-off between learning and cleaning. In contrast to error-free input modeling, we here attach a spring to the data point x^d and allow that spring to be stretched to x for a price: the energy stored in that spring is $1/2\kappa \left(x^d - x\right)^2$, where κ is the spring constant. The ground state is reached when the sum of this energy and the energy stored in the spring in the output (next to η in the figure) is minimal.

average training signal). Horizontal bars point to outliers in the output: several inputs try to compensate for the output outlier; the output itself also has a large error.

- **Characterize input variables by their stochasticity.** Taking the mean of the squared errors across time (i.e., computing the mean squared errors of each of the inputs) allows us to characterize the signal-to-noise-ratio of each input feature. This is not possible without using a model.
- **Error covariance matrix.** Computing the covariance matrix of the errors allows us to investigate the validity of the assumption of statistical independence of the noise of the inputs. If there are significant non-zero off-diagonal contribution, the modeling can be improved by transforming the data by the inverse of the noise covariance matrix.
- **ARSCH Models (AutoRegressive Special Conditional Heteroskedasticity).** Estimating the noise levels enables us to generalize ARCH and GARCH models [Engle, 1982, Bollerslev, 1986, Bollerslev et al., 1990]: since we allow for nonlinearities at every level, we call these models where we input the noise levels averaged over an exponentially decaying window in time ARSCH models (AutoRegressive *Special* Conditional Heteroskedasticity).

The remainder of this section discusses how models that were built on cleaned data are to be used for prediction. We first discuss the case of point predictions where the goal is to predict the expectation value. We then give an algorithm that exploits the noise structure in the inputs to obtain a probability distribution over the output values.

Predicting the expectation value (point predictions). Once we have built the model, point predictions are obtained by a simple feed-forward pass through the network. In principle, we use cleaned data whenever available: e.g., any lagged variable in the input should be replaced by its cleaned value.[3] In practice, however, when the majority of variables are not simply past values of the outputs but more complicated indicators, we simply use the raw data to obtain the predictions. The results reported in this papers were obtained with the raw inputs in the final feed forward step.

Predicting the probability density of the next step. We have emphasized the importance of knowing the accuracy of the prediction from a number of sources.[4] The clearning algorithm puts us in the fortunate position of being able to

[3]This also applies to variables derived from cleaned inputs; e.g., a moving average should be replace by its cleaned value. This process can be iterated.

[4]Examples are (1) estimating uncertainties due to the splitting of the data [Weigend and LeBaron, 1994]–crucial when

estimate the error in the outputs due to the uncertainty in the inputs. We start by computing the matrix of empirical input errors $\Delta(t)$; this matrix consists of one vector (across inputs) for each time step (or pattern). In order to forecast the probability density of the next value, we use today's input vector (as in the case of point predictions), randomly pick one of the empirical noise vectors, and add it to today's input, and record the resulting output. We then draw (with replacement) a second vector from the set of empirical noise vectors, add it to today's (original) input, and generate and record the corresponding output. We repeat this procedure typically with several hundred to a thousand resamplings. We then present the prediction for tomorrow as a histogram of these predictions.

This section has discussed the clearning strategy, and indicated how predictions are to be obtained. To not clutter the presentation, we have omitted the details of the pruning that is done in parallel with the clearning. The next section discusses our pruning algorithm.

3 Pruning

In Section 1, we discussed the standard methods of regularization in neural networks. In Section 2, we discussed how to use the evolving model to modify the input data. We now combine the two concepts; using cleaned data instead of the noisy data allows us arrive at even smaller networks since there are fewer irregularities in the data that would absorb resources that are not supported by the underlying dynamics: simpler data allow for a simpler model. In our experience, we were able to obtain truly ultra-sparse models, unprecedented in what has been done before. We typically end up with fewer weights than inputs! The final networks are immune against overfitting; having found the nonlinear structure in the data, we train the resulting ultra-sparse networks to minimize the error on the test set as well as they still can.

In more detail, we clearn until we observe overfitting, measured by an increase of $\sum(y - y^d)^2$ on the validation set. Our goal is to thin out the connections between input and hidden units; we want to remove the weights that respond most to the noise. Consider a specific weight in the network; w denotes its present value in training. The key idea is the following: we present one epoch (iteration) of input patterns $t = 1, \ldots, N$. We then compare the size of the weight (at the end of the epoch) to its fluctuations in response to the inputs during that epoch.

Let ξ_t denote the weight change in response to pattern t. (In gradient descent: $\xi_t \propto -\partial E_t / \partial w$). The mean and standard deviation of the weight change over the epoch are given by

$$\text{mean}(\xi) = \frac{1}{N} \sum_{t=1}^{N} \xi_t \qquad \text{mean weight change (over epoch)} \quad , \tag{7}$$

and

$$\text{std}(\xi) = \sqrt{\text{var}(\xi)} = \sqrt{\frac{1}{N} \sum_{t=1}^{N} \left[\xi_t - \text{mean}(\xi)\right]^2} \qquad \text{rms weight change (fluctuations)} \quad . \tag{8}$$

This allows us to formulate the following test statistic:

$$\textbf{test value} \; : \; \frac{|w + \text{mean}\,(\xi)\,|}{\text{std}\,(\xi)} = \frac{|\text{weight at end of epoch}|}{\text{fluctuations during epoch}} \quad . \tag{9}$$

If this test value is large, we keep the weight since it is well determined (the fluctuations are small compared to size of weight). If the test value is small, we prune the weight since it is not well determined (the size of weight is small compared to its fluctuations). We are primarily interested in pruning connections from the inputs to the hidden units.[5] All the weights of the input-to-hidden layer are ranked according to their test value.

cross-validation sets are set aside to determine meta-parameters, (2) estimating confidence intervals for unimodal distributions [Weigend and Nix, 1994]–useful for problems that consider Sharpe ratios, (3) obtaining essentially model free arbitrary distributions with the method of fractional binning [Weigend and Srivastava, 1995]–important for multi-modal processes, e.g., when we expect a big move that could go either up or down, and (4) finding trading days where we can trust our model to a higher than average degree, using the method of gated experts [Weigend and Mangeas, 1995]–important for very noisy processes.

[5] We are particularly interested in removing inputs completely; an alternative method for variable subset selection based on the information theoretic measure of mutual information is described in [Bonnlander and Weigend, 1994].

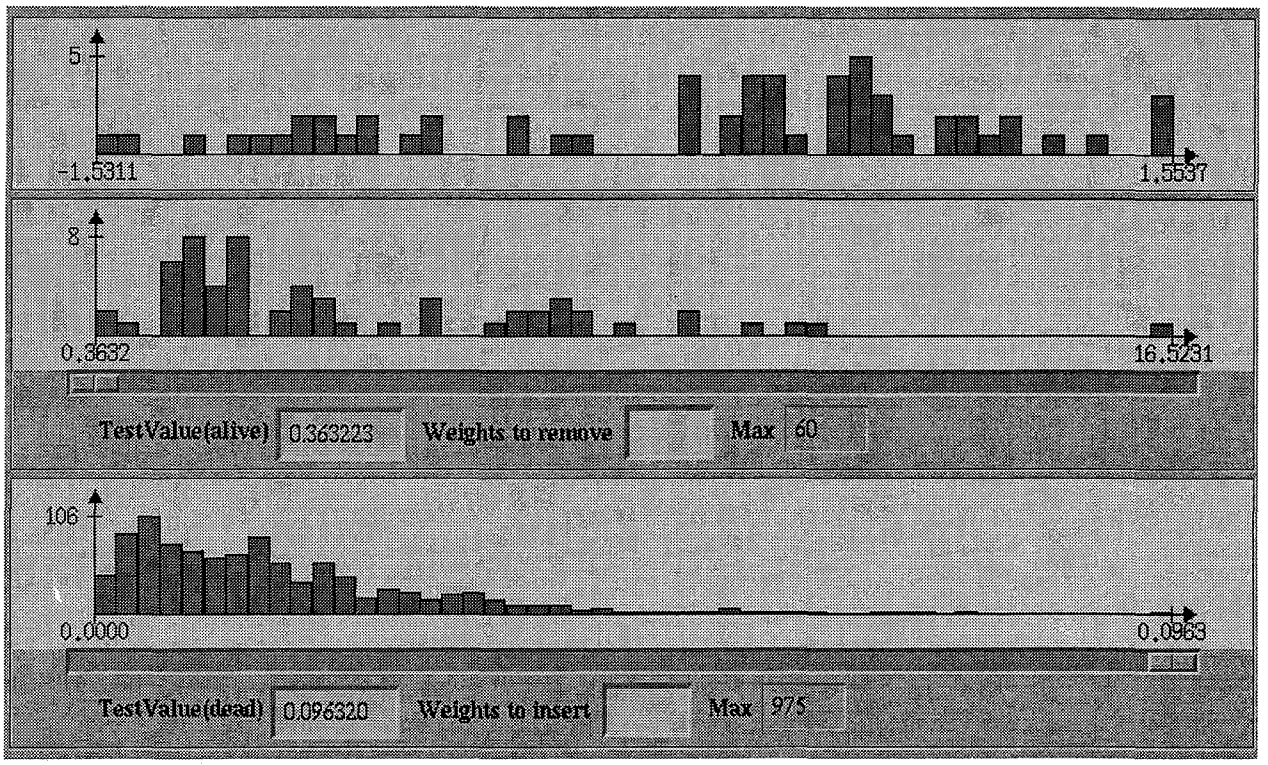


Figure 2: Distribution of test values (Eq. 9) for the weights for the final network trained on exchange rate prediction with pruning only. (Screendump from the SENN simlulator.) The top two rows are for the weights that are active; the bottow row for the weights that are pruned away.

One further decision has to be made: we have the choice of evaluating the test value using the raw or the cleaned inputs. We here compute this statistic on the raw (uncleaned) data for two reasons:

1. The fluctuations of the weights mirror the noise present in the data. In this sense it can be viewed as noise-filter.

2. In prediction mode, the very recent inputs are only available in uncleaned form. Using the raw inputs in pruning is thus closer to the final task.

Furthermore, the fluctuations of the data are mirrored in the fluctuations of the weights. Removing the part of the network that is in resonance with the external noise can be viewed as a nonlinear noise filter. Fig. 2 shows the typical histograms of the test values (Eq. 9), both for the active weights and the pruned weights.

After one pruning epoch, we return to clearning. At overall early stages of the entire procedure, we reinitialize the now smaller network with a new set of random weights ($\mathcal{O}(10^{-4})$) and also reset the cleaning correction vector Δ to zero. This restart of the smaller architecture corresponds to a search in a reduced sub-space. At the later stages of the entire procedure, we omit the re-initialization part. Re-initialization presents, just like early stopping, a strong bias towards linear models that we want to avoid at the final model. Only sufficiently long learning allows the network to extract the nonlinearities present in the data.

When does the entire process stop? The test statistic is evaluated on all weights, not only the survivors. If the test value of weights that were pruned away becomes large again, this is an indication for overpruning. We then revive these overpruned weights by resurrecting them, and finally train to the local minimum. Given the ultra-sparse architectures we have reached, there is no overfitting left. Fig. 3 shows the remaining network of the example discussed in the next section.

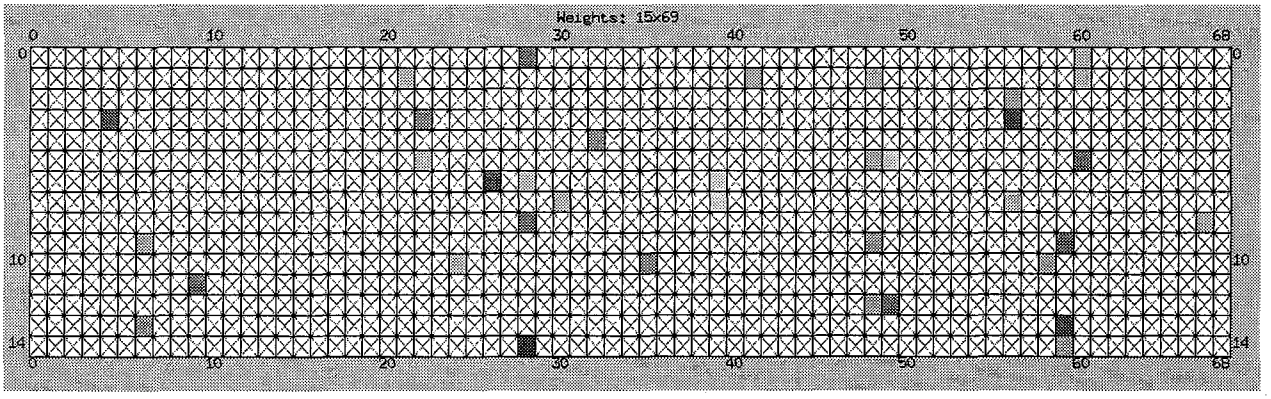


Figure 3: The weight diagram of the weights between inputs and hidden units for the exchange rate example. The larger the weight, the darker the field. (In gray-scale rendering, we lose the information about the sign of the weight.) A gray "X" means the weight has been pruned away. Note that more than half of the potential inputs are completely disconnected. In this sense the clearning and pruning procedure acts also as feature selection.

4 Example: Exchange rate predictions

We demonstrate the proposed method on the problem of predicting daily foreign exchange rates between the US Dollar and the German Mark. The entire time period ranges from January 15, 1985 through January 27, 1994. We first set the test data aside as out-of-sample data: we pick the last 216 days for the test period, covering the prediction for January 4, 1993 through January 27, 1994. From the remaining block, we put aside every fourth day as our cross-validation set to estimate the generalization performance.

The architecture is a simple feed-forward network with 15 tanh hidden units.[6] There are 69 inputs. 12 inputs reflect *chart information* derived from the series itself (relative strength index, skewness, point and figure chart indicators, ...). 57 inputs reflect *fundamental information* beyond the series itself (indicators depending on exchange rates between different countries, interest rates, stock indices, currency futures, ...). These inputs contain information from six countries (France, Germany, Japan, Switzerland, UK, and USA). The network has 3 outputs. The first output predicts the *return* (we use a normalized version of the return; we divide it by the standard deviation computed over the last 10 trading days. In our experience, solely training the network on the one-day forecast makes it hard to capture long-term dynamics of the market. We thus add two further tasks, provide the network with information about the next turning point, defined as the next maximum or minimum of the daily series. The two additional outputs are the *number of days to next turning point,* and the *return between today and the next turning point* (divided by the standard deviation of the data).

When computing the return on investment on the test set, we take the position size as given by the sign of the first output (return output). The profit and loss curves shown in Fig. 4 include a transaction costs of 0.001. We were able to reduce the network with pruning alone to 60 weights between the inputs and the hidden units. Using clearning in conjunction with pruning, we manage to arrive at the ultra-sparse architecture of only 39 weights. The annualized return on investment is significantly above 30% in the difficult 1993/94 period.

Acknowledgments

We thank Art Owen for suggesting to bootstrap the input errors to obtain the distribution of the forecast, and Barak Pearlmutter for sharing his intuitions about the cost function. The simulations were carried out with SENN (Simulation Environment for Neural Networks) at the University of Colorado at Boulder. Andreas Weigend acknowledges support by the National Science Foundation (Grant No. RIA ECS-9309786).

[6] We usually obtain faster convergence and more stable results by using a layer of normalized sigmoids as hidden units; since this paper focuses on clearning, we use the standard tanh architecture for the comparison.

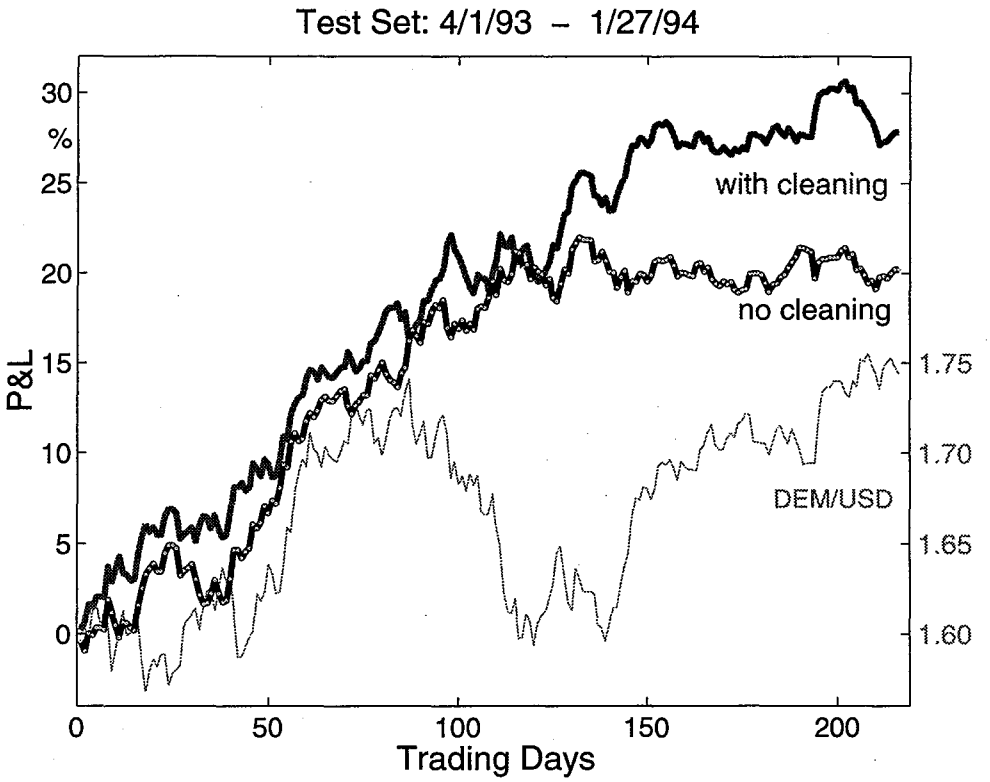


Figure 4: We compare clearning to learning without clearning (both with pruning) on the held-out test set, ranging from April 1, 1993 to January 27, 1994. The top curve is with cleaning. The curve below without cleaning. The scale on the left corresponds to these profit and loss curves. We also give the exchange rate during this time period (bottom curve); its corresponding scale is indicated on the right.

References

[Abu-Mostafa, 1995] Abu-Mostafa, Y. (1995). Hints. *Neural Computation*, 7:(in press).

[Bollerslev, 1986] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 21:307–328.

[Bollerslev et al., 1990] Bollerslev, T., Chou, R. Y., Jayaraman, N., and Kroner, K. F. (1990). ARCH modeling in finance: A review of the theory and empirical evidence. *Journal of Econometrics*, 52(1):5–60.

[Bonnlander and Weigend, 1994] Bonnlander, B. V. and Weigend, A. S. (1994). Selecting input variables using mutual information and nonparamteric density estimation. In *Proceedings of the 1994 International Symposium on Artificial Neural Networks (ISANN'94)*, pages 42–50, Tainan, Taiwan.

[Buntine and Weigend, 1991] Buntine, W. L. and Weigend, A. S. (1991). Bayesian back-propagation. *Complex Systems*, 5:603–643.

[Engle, 1982] Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50:987–1007.

[Finnoff et al., 1993] Finnoff, W., Hergert, F., and Zimmermann, H. G. (1993). Improving generalization performance by nonconvergent model selection methods. *Neural Networks*, 6:771–783.

[Huffel and Vanderwalle, 1991] Huffel, S. V. and Vanderwalle, J. (1991). The total least squares problem: Computational aspects and analysis. In *Frontiers in Applied Mathematics*, volume 9. SIAM.

[Neuneier and Zimmermann, 1995] Neuneier, R. and Zimmermann, H. (1995). An efficient method of neurofuzzy in forecasting. In *Proceedings of ICANN 95 (in press)*.

[Rumelhart et al., 1995] Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). Backpropagation: The basic theory. In Chauvin, Y. and Rumelhart, D. E., editors, *Backpropagation: Theory, Architectures, and Applications*, pages 1–??, Hillsdale, NJ. Lawrence Erlbaum Associates.

[Seber and Wild, 1989] Seber, G. A. F. and Wild, C. J. (1989). *Nonlinear Regression*. Wiley, New York.

[Tresp et al., 1994] Tresp, V., Ahmad, S., and Neuneier, R. (1994). Training neural networks with deficient data. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6 (NIPS*93)*, pages 128–135, San Francisco, CA. Morgan Kaufmann.

[Weigend, 1994] Weigend, A. S. (1994). On overfitting and the effective number of hidden units. In Mozer, M. C., Smolensky, P., Touretzky, D. S., Elman, J. L., and Weigend, A. S., editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 335–342, Hillsdale, NJ. Lawrence Erlbaum Associates.

[Weigend, 1995] Weigend, A. S. (1995). Time series analysis and prediction. In Smolensky, P., Mozer, M. C., and Rumelhart, D. E., editors, *Mathematical Perspectives on Neural Networks*. Erlbaum Associates, Hillsdale, NJ.

[Weigend et al., 1990] Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1:193–209.

[Weigend and LeBaron, 1994] Weigend, A. S. and LeBaron, B. (1994). Evaluating neural network predictors by bootstrapping. In *Proceedings of International Conference on Neural Information Processing (ICONIP'94)*, pages 1207–1212. Technical Report CU-CS-725-94, Computer Science Department, University of Colorado at Boulder, also submitted to IEEE TNN.

[Weigend and Mangeas, 1995] Weigend, A. S. and Mangeas, M. (1995). Experts for prediction: discovering regimes and avoiding overfitting. Technical Report CU-CS-764-95, University of Colorado at Boulder, ftp://ftp.cs.colorado.edu/pub/Time-Series/MyPapers/experts.ps.Z.

[Weigend and Nix, 1994] Weigend, A. S. and Nix, D. A. (1994). Predictions with confidence intervals (local error bars). In *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, pages 1207–1212, Seoul, Korea.

[Weigend et al., 1991] Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3 (NIPS*90)*, pages 875–882. Morgan Kaufmann.

[Weigend and Srivastava, 1995] Weigend, A. S. and Srivastava, A. N. (1995). Predicting probability distributions: A connectionist approach. *International Journal of Neural Systems*, 6.

[Winograd and Flores, 1986] Winograd, T. and Flores, F. (1986). *Understanding Computers and Cognition*. Ablex Publishing, Norwood, New Jersey.

Pace University
School of Computer Science and Information Systems
One Pace Plaza,
New York, NY 10308

$65.00

ISBN 0-938801-09-0

Software Engineering Press
an imprint of the Systemsware Corporation