Proceedings

The Second International Conference on

ARTIFICIAL INTELLIGENCE APPLICATIONS ON WALL STREET

TACTICAL AND STRATEGIC COMPUTING TECHNOLOGIES

EDITOR
Roy S. Freedman

April 19–22, 1993
New York City

Sponsored by

International Association of Knowledge Engineers
Division of Management, Polytechnic University
Wall Street & Technology
New York Chamber of Commerce

In Cooperation with

American Association for Artificial Intelligence (AAAI)
Association for Computing Machinery (ACM)
Society for the Management of AI Resources and Technology – Financial Services (SMART-F$)
AI Expert
European Coordinating Committee for Artificial Intelligence

Proceedings

The Second Annual International Conference on Artificial Intelligence Applications on Wall Street: Tactical and Strategic Computing Technologies

Dr. Roy S. Freedman, *Editor*

Proceedings

The Second Annual International Conference on Artificial Applications on Wall Street: Tactical & Strategic Technologies

Dr. Roy S. Freedman, *Editor*

April 19 - 22, 1993 • New York, New York

Sponsored by:

International Association of Knowledge Engineers (IAKE)
The Division of Management, Polytechnic University
Wall Street & Technology
New York Chamber of Commerce

In Cooperation with:

American Association of Artificial Intelligence (AAAI)
Association for Computing Machinery (ACM)
Society for the Management of AI Resources and Technology - Financial Services (SMART-F$)
AI Expert
European Coordinating Committee for Artificial Intelligence (ECCAI)

Published by the Software Engineering Press

ISBN 0-938801-07-4

Order from:
Software Engineering Press
973C Russell Avenue
Gaithersburg, MD 20879
Phone: (301) 948-5391

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

Logistics: Krebs Convention Management Services, San Francisco, CA

Chairman's Introduction

The *Second International Conference on Artificial Intelligence Applications on Wall Street* is organized to continue the momentum from the first conference in 1991. 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.

Throughout the world from ancient times onward, wherever there was trade between peoples, there were always new ways to raise money for commercial endeavors, and new technologies developed to support these activities. The earliest Babylonian clay tablets are database records of loans, receipts, promissory notes, leases, mortgages, taxes, and checks. Of course, none of these financial activities would make any sense without a standard notion for time; the earliest applications of technology were in the development of calendars. They were based on large numbers of observations and had to be accurate. Calendars fixed holidays as well as interest rates: interest rates were computed for lunar (monthly) and solar (annual) periods. Penalties for inaccuracies were severe.

Computational skills were developed to assist in all these tasks, especially for deriving mortgage and interest rate tables. In ancient Babylonia, interest rates were unregulated; simple interest rates fluctuated between 5% to 20% per annum. A thousand years later, computation was aided by devices such as the abacus— the portable laptop computer of the day. Its use was not restricted to the upper classes, but was carried all over the world by traders who introduced its use across cultures.

All of these developments were not without regulation. Formulas for asset allocation were carefully monitored, especially in regard to inheritance laws. The ancient story of the individual who, when asked to name his reward from the king, simply requested a chessboard with one grain of wheat on the first square, two on the second, four on the third, and so on, illustrates the fact that the power of compounding was known in antiquity. In Rome, one of the oldest interest rate regulations (450 B.C.) fixed the interest rate to one twelfth of the capital; one thousand years later, new regulations forbade compound interest in what remained of the Roman Empire. Nearly fifteen hundred years later, despite the advances in technology, the debate concerning financial regulation continues.

This conference addresses the modern technological aspects of trading, asset allocation, and regulation, and is truly organized across continents and time zones. I want to acknowledge the help of the sponsoring organizations and cooperating societies; our paper session and panel session chairs, our invited speakers, and of course, our Program Committee. Special thanks to Dr. Gia-Shuh Jang for his help in coordinating the speakers. Another special thanks to Paula Ungureanu and Julie Walker of IAKE, and Pat White of Systemsware Corporation, for their diligence and persistence in organizing this international event and quality Proceedings. At Inductive Solutions, I want to thank Marie Deluca for her help with the program, and Louise Parmegiani for her support through interesting and challenging times.

Roy S. Freedman, Ph.D.
Program Chairman — AI/WS-93
Inductive Solutions, Inc.
New York City, April, 1993

PROGRAM COMMITTEE

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

Chidanand Apte, *IBM*

Fevzi Belli, *Universitat Paderborn*

John DeSaix, *NASD*

Susan Garavaglia, *Dun & Bradstreet Information Systems*

Barry Glasgow, *Shearson Lehman Brothers*

Phillip Hayes, *Carnegie Group*

Ypke Hiemstra, *Free University of Amsterdam*

Gia-Shuh Jang, *National Taiwan University*

Warren Kaiser, *American Stock Exchange*

Ken Kleinberg, *New Science Associates*

Bradford Leach, *New York Mercantile Exchange*

Yuval Lirov, *Salomon Brothers*

Joseph Mathai, *INSTINET*

Ross Miller, *GE Corporate R&D*

Fumio Mizoguchi, *Science University of Tokyo*

Christian Nottola, *Banque de France*

Robert Phelps, *S.W.I.F.T.*

Dan Schutzer, *Citibank, N.A.*

Steven Slade, *New York University*

Kar Yan Tam, *Hong Kong University of Science and Technology*

Milton White, *DATANAMICS*

Takahira Yamaguchi, *Shizuoka University*

Lois Zarembo, *New York Stock Exchange*

International Association of Knowledge Engineers

Milton White, *Chairman of the Board of Trustees*
Julie Walker Lowe, *Executive Director*
Paula Ungureanu, *Conference Secretary*
Michael Paca, *Conference Treasurer*

TABLE OF CONTENTS

Paper Session: Understanding News

Chair: Chidanand Apte, IBM

Fact Extraction from Financial News

Kai Schirmer, Michael Kuehn
Intelligent Financial Systems GmbH,
Heinrich-Mann-Allee 105 b, 1561 Potsdam, Germany
CompuServe 100042,2376

Abstract

In this paper we describe the FINAS (FINancial News Analysis System) system, which is designed to analyze news texts from on-line news wires, to convert relevant information to a formal representation, and extract relevant facts in the prototypically chosen domain, the currency markets. The extracted facts are used for filtering and signaling specific online news information. In a second application fundamental and technical indicators are combined in a trading system environment. Word-oriented parsing is chosen as an adequate object-oriented methodology for the language analysis task. In word-oriented parsing each word has its own 'expert' who contains morphological, syntactic, semantic as well as pragmatic information about this word.

1 Introduction

The FINAS system is designed to analyze news texts from on-line news wires and extract relevant facts from the articles in the prototypically chosen domain, the currency markets. To execute the mentioned tasks FINAS has to provide

- a much more deeper analysis than automatic indexing systems carry out. Additional syntactic and semantic analysis is necessary to detect facts with high precision,
- a procedure which is adaptable to various domains,
- a procedure which can deal with the huge amount of texts to be processed online.

With conventional parsing formalize natural language by rules that describe general principles. Following Small and Rieger we belief that humans do not understand language "by orderly application of rules, but rather by the controlled interchange of semaphores and concepts among internally complex 'experts', one per linguistic unit" [SMA82]. 'Word-expert parsing' [SMA80, SMA87, EIM88] is an elaborated parsing strategy following this idea and is chosen as an adequate object-oriented methodology for the news analysis task.

Due to its very consistent knowledge model the method is easy to expand and to adapt to new domains, and is providing an efficient handling, maintaining and processing of language and domain knowledge. In contrast to sequential rule-based approaches the object-oriented implementation of the parser facilitates parallel processing and thus has an enormous influence on the system's performance.

As a first step in FINAS the incoming news are categorized to reduce the number of texts to be further processed. This is done by a neural net trained to categorize news according to news classifications. The classified news are then analyzed by the word-expert parser.

2 Word-oriented Parsing

In word-expert parsing each semantically or structurally interesting English morpheme (e.g. TAKE, -EN, WANT, -ING, DEEP, UP) has its own 'expert' who contains morphological, syntactic, semantic as well as pragmatic informations about this word.

A word expert starts processing, trying to determine the meaning role of its word in context, i.e. interacting with other word experts and with higher-order model processes to acquire the appropriate conceptual knowledge to make the correct inferences. Finally, all the word experts for a particular fragment of text come to mutual agreement on the meaning of the fragment, and the local distributed process terminates - local, in the sense that as long as there remains input text, the overall parsing process continues, while the disambiguation of individual lexical sequences making up the larger texts completes.

The individual word experts have a dual responsibility to coordinate to interconnected facets of the parsing process. Each active word expert must (a) determine its own meaning or function role in the larger text, and (b) provide conceptual and control information to other experts to enable them likewise to coordinate this complex task. Isolated word sense discrimination is not possible, since much of the dynamic knowledge required to complete that task must come from other experts. Thus each expert must both ask questions of

other experts and answer ones posed to it. The parser contains mechanisms for the passing of messages among word experts, in order to achieve basic interaction.

3 Semantic Modeling of Financial Domain Concepts

For the application domain, the currency trading, economic key indicators like

> Sales of Single-Family Homes, Productivity & Costs (Q3), Shipments, Inventories & Orders, Employment Situation, Wholesale trade, Housing Completions, Retail Sales, Auto Sales, Monetary Aggregates, Producer Price Index, Export & Import Price Indexes, Agricultural Prices.

will be described in detail. The semantic modelling of such concepts is done according to the object-oriented word-expert paradigm. Given the news text

> Germany's gross monetary reserves plunged 2.8 billion marks ($1.71 billion) to 109.5 billion marks in the week ended March 15, the Bundesbank said.

the following word expert will be activated:

WXP: Gross Monetary Reserves
Abbr.: GMR
Country: (Germany)
Institution: (Bundesbank)
Currency unit: (DM)
Time period: (weekly)
Qualifying date, act.: (March 15);
Amount, act.: (109.5); Unit: (billion)
Movement: (plunge)
Qualifying date, prev.: (March 8)
Amount, prev.: (112.3); Unit: (billion)
Foreign Liability
Net Monetary Reserves

The compound expert for Gross Monetary Reserves tries to fill his slots. For filling the slot Country the expert Country is called, which contains all the information on the concept and delivers the information back to the calling expert. In the same way the word experts for Germany, Bundesbank, plunge, mark, March etc. are called. By calling the WXPs Foreign Liability and Net Monetary Reserves the expert tries to complete the information about the monetary reserves topic.

Only information on semantic features is shown above - morphologic, syntactic and pragmatic information on the concept as well as control informations are not demonstrated in the example.

As a result of the processing all activated experts have fully or partially filled their slots and are connected compared to a semantic net. Thus news information is accessible and can be retrieved in many ways.

4 Applications Using News Understanding

An automated processing of news wire information in the sense of a linguistic text analysis and extraction of facts found in the articles would provide a tremendous advantage over reading and searching by hand especially in the following areas:

- Signaling the occurrence of specific events
- Filtering and routing specific information
- Combining fundamental information with Time Series Analysis.

Signaling

In the financial sector the availability of timely information is decisive: on product informations, market behavior, mergers and acquisitions etc. In an automatic trading system such informations can trigger buy and sell orders.

Filtering

News, incoming messages and other text documents can be filtered, sorted into meaningful categories according to detailed user profiles, and prioritized based on importance and urgery. With such a message router, e.g. installed company-wide, informations can be very specific and always be up to the minute.

Fundamental Neural Network Analysis

Fundamental Information will be used in this application for combining it with technical indicators in the framework of Time Series Analysis. In a first step a knowledge-based diagnosis component (KBD) evaluates investment conditions using fundamental information. The component combines uses extracted facts for an evaluation of general investment conditions and Using uses domain-specific knowledge, which depends directly on the targeted financial markets. The KBD component concludes about about prime factors and macroeconomic conditions and then provides a neural network with informations like

- where the economy is positioned in the business cycle,
- values of state variables,
- processes that take place in the economy.

Evaluations of a neural network (NN) based trading system can later be used for refinements of the domain specific rules (carried out by classifier systems).

Taken the classifications for prime economic factors a specialised neural net will integrate this information into a rating system, that will generalise (also non-linear) interactions between (historical) inputs, consisting of evaluated fundamental as well as technical data. The state of the art technical indicators are being implemented to give trading advices used as additional input for the learning algorithms.

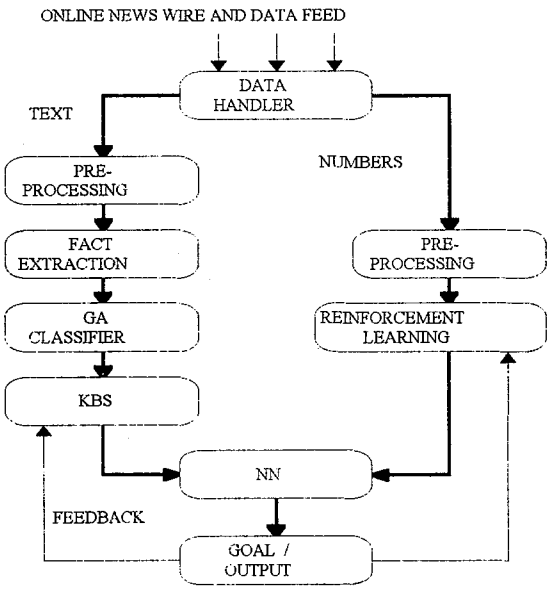


Fig. 1: Trading System Components

The technical data is retrieved from a preprocessor, that computes optimized indicators out of historical (stock) prices and measures their influence to price changes. The output of the neural computations are 6 trading signals: long, short, exit long, exit short, reverse long to short, reverse short to long. These outputs can be used by a KBD to manage different portfolios. Trading rules will be implemented for an effective money management adapted to the different needs of the market participants.

The key technologies used for this trading system are recurrent neural networks, reinforcement learning and genetic algorithms. In recurrent neural networks feedback is allowed from the output of a neuron to its own input and/or to the inputs of neurons within the same and/or other layers. In a next step, the recurrent NN is extended by reinforcement features, that focuses its training phase on handling 'real-world conditions' (and not on producing 'real-value predictions'!)

Genetic algorithms are used for optimizing tasks within the genetic reinforcement part as well as for finding suitable topologies of neural networks, parameter settings and calculating the best total error in respect of best results on test data.

The overall goal is to develop a trading system that performs well in trend markets, in non-trend markets as well as during turning points. To investigate the capabilities of the trading system, a wide range of time series data is being evaluated - including chaotic series and randomly generated series.

5 Summary and Outlook

We have outlined the word-oriented approch to text processing and news understanding. Word experts are the key element in the design of a system capable of continuously analysing and extracting concept and event information out of news text.

An initial baseline system is previously done in C++ and is covering reference, focus and disambiguation tasks. Work is going on with defining concepts and events of the applications domain. The final system will cover approximately 200 domain concepts.

References

[EIM88] Eimermacher M. (1988) Wort-orientiertes Parsen. Dissertation. TU Berlin

[SMA80] Small S.L. (1980) Word expert parsing: A theory of distributed word-based natural language understanding. Dissertation. Department of Computer Science, University of Maryland.

[SMA82] Small S.L., Rieger C. (1982) Parsing and comprehending with word experts (a theory and its realization. In: Lehnert W.G., Ringle M.H. (Hrsg.) Strategies for Natural Language Processing. Hillsdale: Erlbaum, p.89-147

[SMA87] Small S.L. (1987) A distributed word-based approech to parsing. In: Bolc L. (ed.) Natural Language Parsing Systems. Berlin et al.: Springer, p.161-202

Goal Based Reasoning for Securities Analysis

Raghav K. Madhavan
Information Systems Department
Stern School of Business
New York, New York 10012

Abstract

We describe Securities Analysis Program (SAP), a program that projects the impact of news events on securities market price movement. We base our program on principles consistent with those used by a securities analyst. Primarily designed as an advisory system[8], SAP can represent knowledge about the issues surrounding the recommendation of a security. Different levels of knowledge can be represented, ranging from the abstract evaluation level which encodes basic financial analysis, to the specifics of the individual company. Knowledge of the individual analyst, and the special interest groups that they are associated with can also be represented in SAP. SAP provides a recommendation on a given security from the perspective of a specific analyst, together with an explanation of how and why it arrived at the recommendation. SAP arrives at recommendations using goal-based (GBR) and case-based reasoning (CBR) mechanisms. We provide some implementation details of a prototype system and an example to illustrate our approach. Securities analysts as well as investors, including portfolio managers can benefit from such explanations. For example, these explanations may provide additional justifications in support of the analyst's position, or bring to light issues that an analyst may not have taken into consideration.

1 Introduction

Securities Analysis involves the assessment of the impact of new information such as news on the valuation of securities. Traditional approaches to such valuations are based on the translation of new information into quantitative impacts which can then be evaluated using mathematical models. Although this approach is rigorous, it ignores the impact of the goals of the analyst in interpreting this new information. We describe Securities Analysis Program (SAP), a program that projects the impact of news events on securities market price movement. We base our program on qualitative principles consistent with those used by securities analysts in their reasoning about recommendations on a security.

The design goals for SAP are as follows. It should be an advisory system [8] that provides recommendations on a given security, together with an explanation of how and why it arrived at the recommendation. The benefits of such explanations include provision of additional justifications in support of the analyst's position, or bring to light issues that an analyst may not have taken into consideration.

SAP assumes the role of a hypothetical securities analyst and has a knowledge base consisting of past data on specific recommendations on securities, the goals it needs to achieve, its relationships with market groups, as well as the relationships amongst different groups of the market. Drawing from Case-based (CBR) and Goal-based reasoning (GBR) paradigms, SAP employs qualitative reasoning using past history of behavior and consequences of decisions to pursue rewarding alternatives and to avoid repeating mistakes.

For example, consider the process of recommending a specific software stock, Microsoft. The reason for recommending the stock can be because of the analyst believes that the future revenues for the company will be higher than present levels. This belief may be triggered by belief in a related outcome, for example high sales levels of the Windows NT product. On the other hand, an analyst might prefer software stocks in general, and hence recommend this specific software stock. Yet another reason for recommending the stock might be that a particular group, such as the company that the analyst works for, has a stake in the outcome of the decision. The relationship that the analyst has with this group can also shape the recommendation for the stock. The underlying goals, beliefs and relationships change over time, and with new information. SAP is proposed as a mechanism to maintain these goals, beliefs and relationships, and utilize them in arriving at recommendations for specific stocks.

1.1 Security Analyst Reasoning

A securities analyst follows several securities and makes recommendations on whether to buy, hold or sell the securities that she or he follows. Analysts evaluate securities using quantitative principles such as the Dividend Discount Model (DDM) [4, 3, 6] or the Yield to Maturity (YTM), and compare this value to the trading price of the securities. Analysts usually follow the business underlying a security very closely. They monitor the business' environment on a daily basis, looking at factors that affect dividends or coupon payments such as sales level, change in market share, costs of production, union negotiations, change in management, and overseas competition. The impact of such events is incorporated into projections of future earnings of a security, utilizing qualitative reasoning; the projected figures are then utilized in quantitative models that evaluate securities.

Seasoned securities analysts produce analyses faster and with fewer mistakes than novice securities analysts. There are three underlying reasons for these differences. First, a seasoned analyst has followed a company for a long time and hence, immediately recognizes and understands the relevance and significance of new events for the business. Second, a seasoned analyst has established relationships with other players in the market, and as a result recognizes the impact that the event will have on other market players. Third, a seasoned analyst has an implicit belief structure of how different groups in the market impact each other and the value of the security. The goals of SAP are to capture the complex decision structure of the seasoned analyst in an incremental fashion, starting from the level of a novice analyst and learning from its experience.

The rest of this paper is organized as follows. We outline below some of the traditional approaches to securities analysis, including some previous AI approaches. Then we contrast these with the approach taken by SAP, and discuss the advantages of using and combining case-based and goal-based reasoning. We then provide some details of our implementation efforts to date in modeling this domain. An illustrative example is then presented with discussion of the features of our implementation. Finally, we provide some conclusions and discuss future work in this area.

2 Traditional Approaches to Securities Analysis

There are several traditional approaches to securities analysis ranging from valuation based approaches to AI-based approaches such as rule-based expert systems and neural networks. For purposes of brevity, our discussion in the rest of this paper will focus on equity based securities, i.e., stocks. We outline below prevalent approaches in the industry for the analysis of stocks, including AI approaches.

Traditional approaches to evaluating a stock use standard decision theory [7] and deal with the present value of the expected future payoff from owning the stock, such as the DDM and P/E model and their variants. The principle behind the Dividend Discount Model (DDM) is that the expected future dividends of the stock are discounted using an appropriate discount rate for the stock, which is compounded yearly, and the sum of all the discounted dividends gives the value of the stock. Since projections of earnings are based on observable events and use of projected earnings in DDM is equivalent to using dividends under certain assumptions [5, 9], analysts most often focus on earnings projections as a central factor in the valuation of a stock.

These models rely on the processing of qualitative information in the form of observable events into quantitative information that can then be used within the framework of the model. Some observed information, such as the relationship between the analyst's investment services company and its client corporation, is not easily translated into a quantitative factor for such evaluation models. It is thus difficult to capture the role played by such relationships in determining the analyst's evaluation using quantitative models of decision making alone.

Rule-based systems capture the knowledge of human financial experts by extracting the rules that a group of experts is able to specify using formal if-then rules. Knowledge expressed in this form tends to be specialized to solve specific problems and is often termed as *shallow* knowledge. It is imperative that a program intended to handle a variety of situations be capable of *deep* reasoning, i.e. reasoning based on basic principles of the domain. Although expert systems can be generated with a mix of both of these kinds of knowledge, they are not able to learn from past experiences. Eliciting knowledge from experts in a form that can be represented as rules, and getting a group of experts to agree on rules are accompanying problems in building expert systems.

Neural networks (NN), in contrast, can be constructed without the involvement of experts and can learn from past data. However, the determination of what data should be used as input to the NN is based on expert knowledge. Problems with the NN approach include the interpretation of the final network configuration, the number of learning cycles required and ap-

plicability of a trained network across similar problems.

Quantitative models require projections of future dividends or earnings (the payoffs) and the uncertainty involved (premium required). A single number is computed as the value of the stock. All the models described above utilize forecasts of earnings and projections of risk associated with the stock to arrive at a stock value. These models demand the conversion of all available information into quantities that can be used in the model. In particular, impacts of news events have to be translated into numbers before they can be incorporated into the valuation process. Slade [11] and others [1, 2] have argued that it is not reasonable to expect tasks such as decision making to involve quantitative reasoning alone, and argue for incorporation of qualitative reasoning as a complement to the quantitative reasoning process in AI programs. We next present two AI approaches that provide both qualitative and quantitative reasoning.

3 An Approach Using CBR and GBR

Two psychologically motivated AI paradigms of reasoning are Case Based Reasoning [10] and Goal Based Reasoning [12]. CBR systems incorporate the familiar concept of being reminded of a similar episode from the past, the action that was taken in that instance, and the consequences of that action. This information is used in arriving at some conclusions for the current episode. For example, while arriving at a recommendation for a casino-resort stock, you are reminded of previous inaccurate projections of earnings by various casinos, and decide to discount heavily the forecasts made by the casino under consideration. GBR systems incorporate, in addition, the need to achieve certain goals, the relative importance of these goals, and the conflicts among goals. For example, the goals in recommending a stock would include getting high growth, low risk, high yield, increase market demand and so on. Getting high growth may not be as important as facing low risk, lowering the risk exposure might be in conflict with producing high yield. An implicit belief in the goal of increasing market demand for the stock might be that commissions on brokerage would increase or that the price of the stock might increase.

In addition to being psychologically appropriate [10, 12], the two paradigms also provide causal explanations of a recommended course of action. Such explanations often offer insights into the problem domain that are not available from other reasoning mechanisms. In addition, such causal mechanisms provide a method of changing a stance, and exploring the impact of such a change. Case-based reasoning systems use a case base of past history to reason about a new situation, incorporating the new episode and the knowledge from this episode into the case base when reasoning is complete. CBR systems provide a contrast to rule-based systems analogous to how an experienced analyst compares to a newly trained analyst. GBR systems incorporate knowledge of the agent's goals and the relationships the agent has with groups of others and the goals of these other groups, and balances these goals using several strategies in arriving at a solution. The impact of different actions on the attainment of goals is used in strategies of reasoning. Actions from previous cases are used in deriving implied relationships and the importance levels of these relationships. The agent's beliefs about consequences, as well as group beliefs, can be incorporated into the reasoning process. In addition, strategies can be used that incorporate the need to balance relationships of the agent with conflicting groups.

4 Implementing Goals and Relationships

Our implementation of a prototype for SAP is an extension of *VOTE* [13, 14] a GBR/CBR program for the domain of Congressional roll-call voting. We describe briefly the representation of goals and relationships in VOTE, and the mapping of this representation into SAP. Then we discuss the extensions of VOTE to SAP.

The architecture for VOTE, and ergo SAP, is an object-oriented representation of goals and relationships. The basic model elements consist of *issue*, which represents an abstract statement of a goal; *stance*, which represents a goal, by being for or against an issue; *group*, which represents special interest groups such as the National Rifle Association; *member* which represents individual decision makers, and *bill* which represents the Congressional bill under consideration. The particular interests of individuals or groups are captured through their preferences towards (pro) or against (con) specific issues. Such preferences form the premise of the goals to be achieved for the individual or the group. The consequences of being for or against an issue are yet other issues. A bill has certain consequences of voting for or against it.

A goal-based reasoning strategy is one that resolves the balance between the consequences for an individual of voting for or against a bill, based on their own preferences of goals (i.e issues) to be achieved or avoided and the preferences of those groups with which that individual has relationships. VOTE implements a database of ranked strategies that include "partisan decision",

which tries to achieve a vote by voting along party lines, and "minimize adverse effects", which looks at the importance of satisfying the preferences of those who have a majority opinion on the bill. Case-based reasoning is employed in evoking the prior voting record, and using this record to build preferences that might affect the vote of the individual in this instance.

VOTE has built-in knowledge for generating English explanations for decisions, and these explanations can be tailored to the particular strategy that was used to achieve a decision. For a more detailed explanation of the implementation of VOTE, the reader is referred to [14].

Mapping of this implementation into the domain of the securities analysis is as follows. Goals continue to be implemented as *stances* of *issues* and special interest groups as *groups*. Analysts are represented as *members*, and stocks to be recommended are represented as *bills*. Most strategies that have been implemented in VOTE are general in nature [14]. Some which are specifically for the Congressional domain, such as "unconstitutional" have been removed. Two new strategies, "best-for-the-company" and "not-of-value" have been introduced. One of these strategies, "best-for-the-company", will be explained in the following section. The English language generation has been tailored for the securities analysis domain. Making a recommendation on a stock is treated as voting on a bill.

We also introduce *beliefs* as a knowledge structure that are fundamentally different from goals. The difference arises from the fact that while goals are what one wants to achieve and thus form consequences, beliefs about issues can provide justifications for preferences without becoming consequences. Since beliefs are held about underlying issues much in the same manner as goals are held about issues, beliefs have been implemented as a variation of a stance. We are currently in the process implementing strategies which distinguish amongst beliefs and goals to arrive at a recommendation on a stock. Such strategies can be characterized as pursuing both goals that directly preferred, as well as those that are a consequence of beliefs. Under the present implementation of strategies in VOTE, "Deeper Analysis" works in a similar manner by pursuing the consequences of goals that preferred, if the first level goals themselves do not provide enough justifications to arrive at a recommendation. In a similar manner, we are looking at implementing a strategy that considers the impact of the beliefs of the analyst at the same time as it considers the goals of the analyst.

The overall algorithm for generating the recommendation is also adapted from [14], and proceeds as follows.

1. Extract preference stances from recommendation record and group relationships, if not done previously.
2. Match Stock recommendation consequence stances buy/sell with the analyst's composite preference stances.
3. Analyze relative importance of buy/sell stances with respect to norms, recommendation record, groups, and personal beliefs.
4. Apply decision strategies until one fits.
5. If no strategy fits, then expand the implications of the buy/sell stances to include the important consequences of the current decision stances. Go to step 2.
6. Update decision database, adding new recommendation, and print the result.
7. Produce an English summary of the decision.

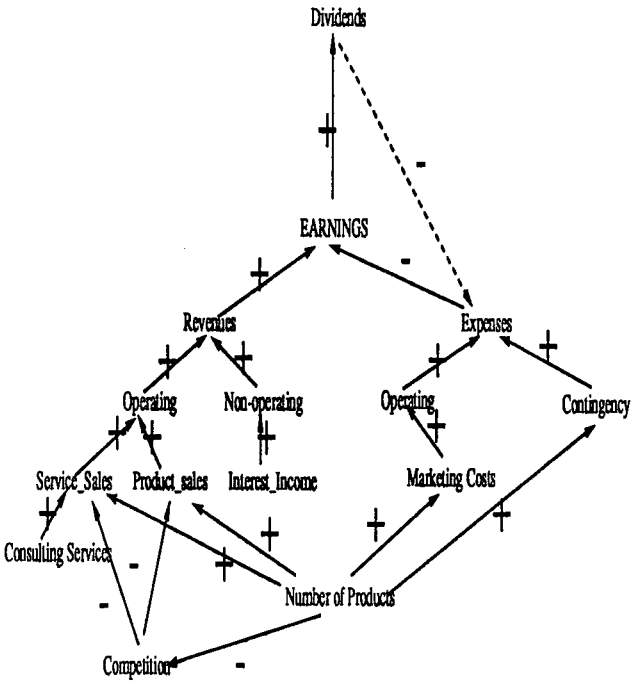


Figure 1: Issues in Business Cash Flow

5 Illustrative Example

We describe now examples of each of the elements that we have proposed in our paradigm for securities

analysis. We will first present the basic elements, and then present a sample recommendation session, together with a discussion of the performance of SAP.

We have implemented some basic business cash-flow issues in SAP, as represented in figure 1. This figure shows issues in text, and the nature of the relationships between issues using directed, labeled arrows. Fundamental to the cash-flow in a business is the sales of the services and products of the company, depicted as "Service_Sales" and "Product_Sales". These form the "Operating Revenue" of the company. Increases in the sales result in increases in revenue from operations, hence the + label on the arrow between these and "Operating Revenue". Similarly "Interest Income is related positively to "Non-operating Revenue". The relationship between "Expenses" and "Earnings" is negative, since increases in "Expenses" lead to decrease in "Earnings". Hence this relationship is labeled with a − sign. The other relationships that are depicted are, similarly, straightforward. In particular, the SAP implementation of these issues allows different levels of importance in such relationships. Based on such representation, SAP can reason about the impact of one of the underlying issues on a higher-level issue. A few sample descriptions of these issues from SAP are given below. **Note that all the descriptions from SAP provided in this paper are *generated* automatically by the program depending on the context, not canned.** Such descriptions will be shown in *italics.*

- Operating Revenue

  Most analysts are deeply committed to belief in the revenue from operations of the company. Increases in the revenue from operations of the company always leads to rises in inflow of funds into business, in addition to rises in cash flow of the business.

- Number of Products

  The general investing population believes in increases in the leadership position attained by offering an array of products. Increases in the diversity of the organization reinforces the revenue generated from selling the products that the company sells, the revenue generated from selling the services that the company sells, increases in the overhead costs associated with the company's existence, in addition to rises in the expense in marketing the goods/services that the company sells. Decreases in the diversity of the organization works firmly against the overhead costs associated with the company's existence, and escalation of extraordinary expenses arising out of abnormal situations.

- Product Sales

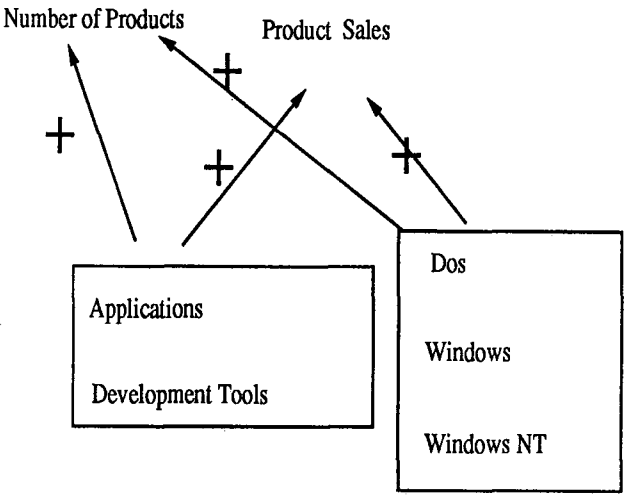


Figure 2: Issues related to Microsoft Security

  Most investors emphasize their deep-rooted belief in the revenue generated from selling the products that the company sells. Increases in the revenue generated from selling the products that the company sells essentially supports the revenue from operations of the company. It reinforces rises in the expense in producing the goods/services that the company sells.

- Consistency

  A normal analyst stresses his long-standing belief in being in-line with prior recommendations. Increases in being in-line with prior recommendations always leads to a good performance record.

In addition to the generic business issues, we have also modeled issues related to the Microsoft security, MSFT, as represented in figure 2. The figure depicts the impact of increases in sales of the products of the company on the previously described business cash-flow issues. Two of these issues are described below.

- Windows NT

  Average investors completely believe in rises in sales of Microsoft's NT Operating System. Increases in sales of Microsoft's NT Operating System is a fundamental cause of the revenue generated from selling the products that the company sells, the revenue generated from selling the services that the company sells, rises in sales of applications that are specifically developed to work with Windows, and rises in the leadership position attained by offering an array of products.

- Access

  The average investor has always believed in sales of Access, Microsoft's recent database debut. Increases in sales of Access, Microsoft's recent database debut strongly stifles rises in the competition for the company. It essentially supports the revenue generated from selling the products that the company sells. It reinforces rises in the leadership position attained by offering an array of products.

We model a few special interest groups related to financial analysis. Some of the groups that we have implemented are described below by SAP.

- Software Analysts

  The software industry analyst group has always believed in escalation of the value of computer industry stocks, being in-line with prior recommendations, keeping the market steady, rises in the value of software industry stocks, rises in the competition for the company, escalation of sales of Microsoft's GUI Operating Environment, and rises in the value of equity in the company. It is a skeptic of rises in extraordinary expenses arising out of abnormal situations, and moreover escalation of the risk associated with the company's future prospects. It is a defender of rises in the expense in running the operations of the company, rises in sales of personal computers, escalation of sales of Microsoft's NT Operating System, increases in distribution of earnings to stockholders, as well as rises in sales of Microsoft DOS. The financial community completely supports software analysts group.

- A Financial Services firm

  This financial services organization is a champion of sales of Microsoft's NT Operating System, sales of Access, Microsoft's recent database debut, and sales of personal computers. It is strongly against the competition for the company. The investment community has always believed in the strategic positioning of this firm and its management direction.

- Bulls

  Investors who are 'bullish' about equity-based securities have always felt strongly about rises in prospects of growth in equity value. They are skeptics of the risk associated with the company's future prospects. They are a defender of distribution of earnings to stockholders, and the legitimate concerns of the business community. The general investing population is eager to show its support for the calculated optimism of the bulls in rising levels of securities prices.

- Bears

  Investors who are pessimistic about the future prospects for equity based securities strongly distrust prospects of growth in equity value. They readily believe in big business, and extraordinary expenses arising out of abnormal situations. They disbelieve in distribution of earnings to stockholders. The general investor is strongly opposed to the paranoia of impending market decline of the bears.

We have represented two analysts in SAP. We present the description that SAP provides for each of these analysts, together with the representation of the list of stated goals ("credo"), and relationships to groups that each of them have.

- Larry Lynch

  Larry Lynch completely believes in the value of software industry stocks, being in-line with prior recommendations, and moreover software sales. He has strong concerns about extraordinary expenses arising out of abnormal situations, in addition to the risk associated with the company's future prospects. He strongly believes in the value of computer industry stocks, the diversity of the organization, the revenue from operations of the company, and the competition for the company.

```
Credo:
 (PRO B Member.879 Competition)
 (CON B Member.879 Contingency)
 (PRO B Member.879 Number_of_products)
 (PRO B Member.879 Operating_Revenue)
 (CON B Member.879 Uncertainty)
 (PRO A Member.879 Consistency)
 (PRO A Member.879 Software)
 (PRO B Member.879 Computer_Industry)
 (PRO A Member.879 Software_Industry)
```

- Peter Prudent

  Peter Prudent is a longtime skeptic of the diversity of the organization. He is unwavering in his belief in plans for future career opportunities, as well as increases in inflow of funds into business. He firmly disbelieves sales of Microsoft's NT Operating System. Prudent feels strongly in favor of the risk associated with the company's future prospects, as well as rises in cash flow of the business. Prudent is against the value of software industry stocks, in

addition to rises in software sales. He approves of extraordinary expenses arising out of abnormal situations. He is wavering about sales of Access, Microsoft's recent database debut, in addition to the competition for the company. Prudent is a champion of the strategic positioning of his firm and its management direction.

```
Credo:
  (CON D Member.878 Competition)
  (PRO C Member.878 Contingency)
  (CON A Member.878 Number_of_products)
  (PRO A Member.878 Revenues)
  (PRO B Member.878 Uncertainty)
  (PRO B Member.878 Cash_Flow)
  (CON C Member.878 Software)
  (PRO A Member.878 Career-Path)
  (CON C Member.878 Software_Industry)
  (CON D Member.878 ACCESS)
  (CON B Member.878 NT)

Relations:
  (PRO A Member.878 Company)
```

We have represented the following stock for recommendation by SAP.

- MSFT

  Increases in the Microsoft stock is a fundamental cause of escalation of the value of software industry stocks, escalation of software sales, increases in sales of Microsoft's NT Operating System, increases in sales of applications that are specifically developed to work with Windows, in addition to increases in the value of shares of the Microsoft Corporation. It results in sales of Access, Microsoft's recent database debut, in addition to increases in sales of Microsoft's GUI Operating Environment. It is bad for escalation of the risk associated with the company's future prospects, and rises in the cost of getting the product/service to the customer. Increases in the Microsoft stock is compatible with increases in sales of Microsoft DOS. Decreases in MSFT reinforces increases in the leadership position attained by offering an array of products, increases in the competition for the company, increases in the expense associated with suing or defending against a suit, and increases in the expense in marketing the goods/services that the company sells. It is compatible with escalation of extraordinary expenses arising out of abnormal situations.

A set of sample recommendations from SAP are presented below.

- Lynch's Recommendation

  Larry Lynch is backing stock MSFT, the Microsoft stock. He believes the adverse aspects of this stock are far outweighed by other issues. He is a champion of rises in the value of software industry stocks. Lynch is a champion of rises in software sales. Lynch is a skeptic of increases in the risk associated with the company's future prospects. Even so, he understands that he is a defender of rises in the leadership position attained by offering an array of products. Lynch readily believes in escalation of the competition for the company.

  The strategy used by SAP to arrive at this recommendation is to minimize the impact of adverse effects. The adverse impacts of making a recommendation to sell outweigh in number and or importance the adverse impacts of making the recommendation to buy the stock.

- Prudent's First Recommendation

  Peter Prudent is backing stock MSFT, the Microsoft stock. He believes this stock to be in the best interests of our company as a whole. He has always believed in rises in sales of Microsoft's NT Operating System. Prudent emphasizes his deep-rooted belief in sales of Access, Microsoft's recent database debut. Still, he understands that he endorses rises in extraordinary expenses arising out of abnormal situations.

  The interesting point in the above recommendation is that SAP used the strategy "best for the company". This strategy is evoked when the impact of making a recommendation all exactly match the interests of the employer. In this case the employer's interests override the interests of the analyst, since the analyst has a significant positive relationship with the employer. We are currently in the process of generalizing this strategy so that the interests of any group with whom the analyst has a significant relationship are considered before the individual interests of the analyst.

- Prudent's Second Recommendation

  If Prudent were to leave the company, his relationship with the COMPANY group would no longer exist. After such a revision is made, the description that SAP provides for him looks like:

  Peter Prudent adamantly disbelieves in increases in the costs in managing a host of product offerings. He completely believes in plans for future career opportunities, and moreover rises in inflow of funds into business. He is strongly against increases

in sales of Microsoft's NT Operating System. Prudent believes in rises in the risk associated with the company's future prospects, as well as rises in cash flow of the business. Prudent is against escalation of the value of software industry stocks, and rises in software sales. He approves of extraordinary expenses arising out of abnormal situations. He is against sales of Access, Microsoft's recent database debut, in addition to increases in the competition for the company.

The only difference between the current representation and the previous one is that Prudent's relationship with the "Company" has been removed. SAP now simulates his recommendation as follows.

Peter Prudent recommends selling stock MSFT, the Microsoft stock. He believes this stock not to be in the best interests of the investors. He is for rises in extraordinary expenses arising out of abnormal situations.

This demonstrates the ability of goal-based reasoning, and SAP in particular, to reason from the perspective of one's relationships and beliefs.

6 Concluding Remarks

We have introduced the notion of using goals, beliefs and relationships as a way of tracking and explaining security analyst reasoning using the paradigms of CBR and GBR. We have illustrated this approach by describing the prototype of SAP, together with an example. The benefits of this approach to represent the qualitative aspects of securities analysis include the availability of explanations for recommendations, and the flexibility to represent changes in the views of analysts based on new information. New information can be incorporated as a belief or as a goal which may then significantly alter the network of beliefs and goals for an analyst, or for a stock. Such impacts can be discerned by simulating the recommendation of different analysts on different stocks and interpreting changes in the recommendation, or changes in the nature of the recommendation.

We are continuing the process of implementing this approach by extending the VOTE program to handle beliefs, and by implementing strategies that utilize beliefs in the security analysis context. Future work in this area include:

- the implementation of CBR for extracting the consequences of individual issues based on historical consequences of similar issues
- persuasive reasoning: analyst and group level representation of perspectives of consequences of issues and use of these in strategies that deal with influencing other's perspectives as well as goals
- combining quantitative and qualitative reasoning
- improving the English generation for explanations
- investigating the prospects for "organizational learning" through the capture of analyst experience in terms of goals, beliefs, relationships and recommendations over time (cases), as well as the quantitative models used

In particular, research is needed in identifying the scheme for easy representation of news in terms of beliefs and goals.

Acknowledgment

We wish to thank Stephen Slade for generously offering his time and assistance in guiding us through this research. The assistance of Michael Bieber and Alfredo Diaz has also been valuable.

References

[1] J. de Kleer and J. S. Brown. A qualitative physics based on confluences. In *Qualitative Reasoning about Physical Systems*, pages 7–83. MIT Press, Cambridge, MA, 1985.

[2] K. Forbus. Qualitative process theory. In *Qualitative Reasoning about Physical Systems*, pages 85–168. MIT Press, Cambridge, MA, 1985.

[3] R.J. Fuller and C.C. Hsia. A simplified model for estimating stock prices of growth firms. *Financial Analysts Journal*, May-June 1984.

[4] R.J. Fuller and J. L. Farrell Jr. *Modern Investments and Securities Analysis*, pages 355–360. McGraw Hill, 1987.

[5] J.C.T. Mao. *Quantitative Analysis of Financial Decisions*, pages 464–476. Macmillan, Toronto, 1969.

[6] N. Molodovsky. Common stock valuation–principles, tables and applications. *Financial Analysts Journal*, March-April 1965.

[7] H. Raiffa. *Decision Analysis: Introductory Lectures on Choices under Uncertainty.* Addision-Wesley, 1968.

[8] R.C. Schank and S.B. Slade. Advisory systems. In W. Reitman, editor, *Artificial Intelligence Applications for Business*, chapter 14, pages 249–265. Ablex Publishing, Norwood, NJ, 1984.

[9] W.F. Sharpe. *Investments*, pages 369–371. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1981.

[10] S.B. Slade. Case-based reasoning: A research paradigm. *AI Magazine*, 12(1):42–55, Spring 1991.

[11] S.B. Slade. Case-based reasoning for financial decision making. In *Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street*, New York, NY, October 1991. IEEE Computer Society.

[12] S.B. Slade. Goal-based decision strategies. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 593–598, Chicago, IL, August 1991. Cognitive Science Society.

[13] S.B. Slade. *An Interpersonal Model of Goal-based Decision Making.* PhD thesis, Yale University, 1992. Technical Report 892.

[14] S.B. Slade. *Goal-based Decision Making: An Interpersonal Model.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

Extracting and Disseminating Information From Real-Time News

Edwin Addison, et. al., ConQuest Software, Inc.

Abstract

Real-time news is defined by two basic criteria. First, it is a feed (or consolidated feed) of the full text of articles, wire service stories or other documents which is not subject to any time embargo or delay. Second, it can be indexed, filtered or searched in real-time for routing to specific readers. The conventional approach to real-time news routing is to treat each profile as a query, which must be executed against the continuous inflow of data in order to extract relevant information. The processing time is directly proportional to the number of profiles.

An alternate approach, based on an advanced statistical retrieval system with processing enhancements from the field of natural language processing, reverses the conventional query/database relationship. This system treats user interest profiles, not as scores or even thousands of discrete queries, but as "documents" in a single information database. It treats the incoming news feed, not as a database, but as a single "query" executed against the aggregated profiles. Processing time is proportional to the logarithm of the database size, and can yield substantial efficiencies in performance and use of computer resources for systems supporting very large numbers of users.

Incorporating a semantic network derived from published dictionaries and thesauri, this new approach also automates the process of creating and maintaining interest profiles, and offers additional text management functionality in the form of intelligent hypertexting.

Extracting and Disseminating Information From Real-Time News

Edwin Addison, Judith Feder, Paul Nelson and Tom J. Schwartz,
ConQuest Software, Inc.
9700 Patuxent Woods Drive Columbia, MD 21046
410-290-7150/410-290-7155 (fax)

Introduction

Real-time electronic delivery of full-text news information has become an increasingly important complement to market pricing data for users in the financial community. Information vendors have also positioned real-time news as a means of reaching a broader range of corporate and institutional users, whose "need to know" extends to sales and marketing, competitive intelligence and other applications.

Real-time news emerged as an "end-user" information product in the late 1980s, with the announcement and/or introduction of such services as UPI's NewsManager/2000, Desktop Data's NewsEdge, and Dow Jones' DowVision (of which Desktop Data is one of several Alliance Partners). LINK Resources, an International Data Group company which follows the electronic information industry, estimated that such products accounted for $51 million of revenue in North America in 1992.

ConQuest Software, Inc. is the developer of a text search and retrieval system which can be used for both real-time and archival searching in a single enterprise. Real-time news is a feed (or consolidated feed) of the full text of articles, wire service stories or other documents which is not subject to any time embargo or delay. It must be able to be indexed, filtered or searched in real-time for delivery to specific readers. Real-time news products are increasingly integrated into such enterprise information systems as electronic mail, groupware or financial information delivery networks, using an audible or visual alert to indicate that relevant information has been received.

Conventional Real-Time Routing

The filters for incoming news information may be as basic as stock ticker symbols or other standardized alphanumeric identifiers, or as complex as user-defined interest profiles consisting of words and concepts a person needs to track. Note that these profiles are closely related--or in some cases bear a one-to-one correspondence-- to information agents.

Once these filters or profiles have been established, the conventional approach to real-time news is to treat each profile as a query, which must be executed against the continuous inflow of data in order to extract relevant information. Profiles must be maintained and updated to reflect changing user interests, changes in the database and even changes in the subject area and its terminology.

ConQuest's Approach

ConQuest's text information management capability represents a different approach, both to the fundamental process of real-time information filtering, and to the process of creating and maintaining the profiles on which that filtering is based.

Reversing the Query/Database Relationship

Simply stated, ConQuest accomplishes real-time information filtering through the converse of conventional approaches. The system treats user interest profiles, not as scores or even thousands of discrete queries, but as "documents" in a single information database. It treats the incoming news feed, not as a database, but as a single "query" executed against the aggregated profiles.

This approach greatly reduces the processing requirements for real-time information dissemination, while enabling ConQuest to support very large numbers of users. For example, a conventional real-time routing system with 1,000 users must execute 1,000 queries against an incoming feed. *The processing time is directly proportional to the number of profiles.*

Applying the converse approach, there is only 1 retrieval against the aggregate database of profiles. *Processing time is proportional to the logarithm of the database size*--that is, the logarithm of the number of profiles.

The difference between these approaches becomes substantial for systems supporting very large numbers of user profiles. For example, an organization with 10,000 active profiles using a conventional system will require processing time of 10,000 queries against a document. Reversing the query/database relationship would require 1 query against 10,000 documents, or approximately (log 10,000) time units, i.e., proportional to 4. If all else were equal, a conventional real-time system could only match ConQuest's time if it had 4 or fewer profiles.

Semantic Networks for Automatic Creation of Concept-based Interest Profiles

Creating and maintaining profiles with ConQuest also differs from conventional approaches. A chief distinguishing feature is the incorporation of a semantic network derived from full, published dictionaries and thesauri. The network consists of weighted links between words, encompassing multiple relationships of meaning, such as:

- synonym,
- antonym,
- part-of,
- kind-of,
- related term,
- contrasting term, etc.

The current network contains 250,000 concepts and 3 million word links Users do not need to construct their own networks of topics or concepts for interest profiles. Rather, they can type a plain English sentence or phrase describing their interest.

ConQuest indexes the words in these profiles. Using the semantic network, subsequent searches retrieve, not only the exact words, but related terms and concepts. The system also allows users to focus their interest profiles by selecting only specific meanings of their key terms from a range of dictionary definitions contained in the product.

Adding to or changing an interest profile can be accomplished by adding or entering a new phrase or sentence. The system indexes this as a discrete document, without having to update the entire profile database.

Query By Example

A Query By Example capability offers another method of creating an interest profile. Users can select a phrase or paragraph from an existing document, or an entire document itself, and instruct the system to treat that selection as an interest profile.

Intelligent Hypertext

Intelligent or automatic hypertext is another facet of this approach. Using the semantic network as a knowledge base, the search and retrieval process dynamically establishes links between related concepts within and between documents. This capability allows users, not simply to scroll, but to browse through documents from concept hit to concept hit.

Intelligent hypertext differs from conventional hypertext in several important ways:

1. It dramatically reduces the time and labor to construct links manually.
2. If offers greater flexibility than manual construction of links, which are "fixed" once they have been created. Users as well as publishers can create links, and the knowledge base of the semantic network can be used and re-used without restriction to establish links relevant to a particular user's needs.
3. It enables users to establish links to text added to an existing database

Technical Background

The technical foundation of this new approach to real-time news filtering is an advanced, statistically-based search system, with processing enhancements drawn from the field of Natural Language Processing (NLP). This approach also provides extended text information management capabilities for non-real-time applications. NLP has long held forth the promise of substantial improvements in text

retrieval capabilities, but has been hindered by two major problems:

1. NLP has until now typically required large, hand-crafted knowledge bases.
2. Traditional techniques are not robust in the face of text errors.

The first problem has been addressed by incorporating machine-readable dictionaries and thesauri for all knowledge data. Dictionaries provide rich information on syntax, word variations and inflected forms. Thesauri provide semantic information and were compiled into structured networks. These resources have never before been combined and required significant engineering effort.

The problem of robust processing required work in two areas. First, NLP development was directed *toward* statistical approaches and *away* from rule-based approaches. Unlike rules, which are typically pass/fail, statistical approaches can handle unexpected or variable input without causing total system failure.

Second, an emphasis has been placed on software engineering. The guiding philosophy is that simple programs, well executed, will always out-perform complex tools poorly done.

System Architecture

The system architecture employs a dictionary with a semantic network to perform various NLP tasks for both indexing and search. Other modules in the system include: the library manager, responsible for system parameter, database configuration, resource allocation and physical partitioning of the indexes; and the dictionary editor, used to edit words, meanings, links and definitions.

The Dictionary/Semantic Network Relationship

The dictionary provides the meanings or words and their interrelationships which delinated nodes and links between nodes in the semantic network. The dictionary is also used for morphological analysis and idiom processing, while the semantic network is used to expand the query to include related terms. Because connections are made between word meanings in both the dictionary and the semantic network, processing is more accurate compared to a simple thesaurus.

Indexing Modules

The indexing modules are as follows:

- *Parse Document:* Looks for codes in the text database such as title, author, etc. These fields can be indexed, ignored or stored in a special database for fast access.

- *Tokenize:* Divides a string of characters into words, including special processing for dates, phone numbers, floating point numbers, hyphens, etc. This module further supports the construction of indexes for numeric and date ranges, such that users can include such a range in their queries or interest profiles.

- *Morphology:* This is a more accurate form of word reduction than stemming, which attempts to remove suffixes and perform spelling changes to reduce words to simpler forms found in the dictionary. For example: morphology stripes the "ies" from "babies," adds "y" and produces "baby," which is found in the dictionary. Morphology will not reduce proper nouns and produces more accurate reductions, for example for words that end in "e."

- *Find Idioms:* This module finds and indexes idioms such as "Dow Jones Industrial Average" as a single unit. Words within idioms can still be located individually, if desired.

- *Index:* The final step is to store the reduced words and collected idioms into the indexes. This is an inverted, positional word index, conceptually similar to the index at the back of a textbook.

Indexing speed has been measured to be approximately 40 megabytes per hour on a Sun IPC Sparc-Station (a 14 MIP computer) with 32 megabytes of RAM. The same speed has been achieved on a 486 IBM-PC, running at a clock rate of 33 MHz, with 16 megabytes of RAM.

Query

The query process encompasses word expansion and ranking. Generally speaking, the process attempts to refine and enhance the user's query, with the result matched against the indexes to look for documents with similar patterns.

Queries are not "understood" in the traditional sense of natural language processing. Rather, the attempt is made to understand the meaning of each individual word and its importance. That set of meanings and their related terms are then treated as a statistical set matched against document information stored in the indexes.

Query modules include:

- *Tokenize, Morphology, Find Idioms:* These are the same modules as for indexing.

- *Query Enhancement:* The user may enhance the query for additional improvement in precision and recall. The most important options are to choose meanings and weight query terms.

Choosing a meaning restricts the expansion of words only to related terms relevant to the chosen meaning. When running in automatic mode, the system expands all meanings of all of the query words.

Weighting query terms identifies the importance of the various words in the query. These weights are used by the search engine when ranking documents and computing relevance factors.

- *Remove Stop Words:* Determiners, conjunctions, auxiliary verbs and small adverbs are removed from the query, similar to the indexing process. This makes queries faster and reduces ambiguous noise in the query.

- *Expand Meanings:* Words in the query are expanded to include related terms. Users can control the level of expansion by choosing one of nine levels of word relationships in the semantic network.

- *Search and Rank:* The system uses an integrated search and rank algorithm which considers the relevance rankings of documents throughout the search process. Integrating search and ranking differs from most approaches, which typically retrieve all matching documents and then rank and sort as a separate step. The result is to produce the most relevant documents immediately.

Query speed is based on many factors, including database size, amount of expansion, size of dictionaries, etc. Many of these parameters can be modified to achieve the appropriate size trade-off between accuracy and speed. In an external test involving a relatively early release of the system, a query over 1 gigabyte with 15 terms took roughly 9 seconds to retrieve 500 documents on a SPARC-IPC with 32 megabytes of RAM.

In both external and internal testing, the system has also proven a top performer in retrieval accuracy using databases as large as 2.5 gigabytes.

Conclusion

ConQuest's semantic network approach provides text information management functionality which enhances real-time news information dissemination from the perspective both of vendors and users. ConQuest's inversion of the conventional query/database relationship enables real-time processing of a very large number of interest profiles with substantially less computing resources than other approaches. The semantic network automates the process of retrieving conceptually related information, while providing the ease of use of a plain English interface. Query By Example allows users to benefit from existing terms, phrases and documents, instructing the system to "find me other documents like this one." This functionality is equally applicable to non-real-time applications, including rigorous searches of archived text and document collections.

COPYRIGHT PROTECTION OF FINANCIAL, ECONOMIC, AND INDUSTRIAL DATA

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

Abstract

Financial, economic, and industrial data, which today is typically computer generated and stored in electronic databases, is fast becoming an increasingly valuable commodity, and play a central role in artificial intelligence applications. This paper analyzes a recent Supreme Court decision and recent decisions of the lower Federal courts that apply to copyright protection of data compilations.

Introduction

Financial, economic, and industrial data, which today is typically computer generated and stored in electronic databases, is fast becoming an increasingly valuable commodity. Such data compilations include stock, bond, and commodity prices, mortgage and interest rates, economic statistics, forecast data, customer lists, and much more. Purchasers of such information can access the data on-line using a PC, from disks and CD-ROMs, or even through satellite communication. Several prominent firms are in the business of exclusively marketing data, which include DRI/McGraw-Hill and Bloomberg Information Services. To be sure, these data compilations play a central role in artificial intelligence applications.

Because of the significance and value of such data compilations, it is only fair that anyone who undertakes the effort to compile and offer data should be entitled to proprietary rights to that information so that the investment in time and labor is protected from unauthorized access and copy. The professional software community has come to expect this kind of protection from the law of copyright, which among other things protects computer programs from unauthorized copying.

The Copyright Law

The copyright law protects *compilations* of facts and data bases, electronic or otherwise. Facts and data itself, however, have never been subject to copyright protection, since the law of copyright protects expression only; isolated facts, and similarly data, are considered to be in the public domain. Ideas and functionality (i.e., a method of operation) also fall outside the scope of copyright protection.

For a compilation to be protectable by copyright, three characteristics must be present: (1) the compilation must be a collection of data, facts, or pre-existing material; (2) there had to have been an act of selection, coordination, or arrangement involved in the creation of the compiled materials; (3) it must be an *original* work of authorship, i.e., the act of selection, coordination, or arrangement mentioned in (2) must have involved an element of originality. Selection generally refers to an act of judgement in choosing a subset of data from a larger universe from which the data comes. Arrangement means to order or group data into categories not deemed trivial. (Alphabetical or chronological arrangements are typically considered trivial.)

The Feist Case

In the 1991 decision Feist Publications Inc. v. Rural Telephone Services Company, the Supreme Court re-emphasized the originality requirement for copyright protection of data compilations., and ruled that data compilations that lack originality, either in selection or arrangement, are not copyrightable. More specifically, this decision held that an alphabetically arranged telephone directory (white pages) published by the Rural telephone company is not original enough to be protected under the laws of copyright.

The importance of this case goes beyond this reaffirmation, however. In Feist, the Supreme Court saw fit to abolish a long-standing doctrine, previously relied on by some courts, under which the required originality could be established by the implied time and effort that was required to produce the compilation. Although the Court indicated that the threshold of originality for copyright protection is low, i.e., novelty is not required, and the vast majority of compilations are sure to pass the test of originality, concerning the white pages at issue the court stated that the telephone company "simply takes the data provided by the subscribers and lists it alphabetically by surname. ... This is 'selection' of a sort, but it lacks the modicum of creativity to transform mere selection into copyrightable expression. Rural expended sufficient effort to make the white pages directory useful, but insufficient creativity to make it original."

The Feist decision is surely of concern to the financial community, since most data vendors want their data to be comprehensive, which implies a lack of creative discrimination in choosing certain data items while rejecting others. (Exercising judgement when selecting establishes originality.) On the other hand, the Feist decision tells us that the time and labor invested in compiling data is not relevant to copyright protection of exhaustive collections of data.

Copyright protection of a data compilation, such as a database, extends only to the original contribution of the compiler, i.e., to the particular selection or arrangement, but never to the data itself. Therefore, "the copyright in a factual compilation is thin. Notwithstanding a valid copyright, a subsequent compiler remains free to use the facts contained in another's publication to aid in preparing a competing work, so long as the competing work does not feature the same selection and arrangement."

After Feist, then, it is clear that a database containing a subset of data from a universe of data of a particular type is protectable if the subset is selected by an act of subjective judgement, though each individual item of data in the subset remains in the public domain.

Lower Court Cases

In Dow Jones and Co. v. Board of Trade, decided prior to Feist, the court found that Dow Jones' list of stocks, which is a carefully selected, small subset of all listed securities, is protectable because of "subjective judgment and creativity in choosing items to list." Similarly, in Key Publications v. ChinaTown Today Pub. Ent. the court found that the act of judgement in selecting entries for a directory of businesses that are of special interest to New York City's Chinese-American community was sufficient evidence of originality to establish copyrightability, since the businesses in the directory represented a selected subset of all New York's businesses.

There is one aspect of the Key Publications decision worth noting. Though time and effort involved in *creating* a database is irrelevant, time and effort involved in the *selecting* of data items may be used to establish that the selection process possessed the modicum of originality required. Here, the compiler collected business cards of businesses and professionals that worked with the Chinese-American community, which was noted in the decision. If the selection process is comparatively trivial, such as selecting a particular geographical location for a database of marketing data, or a time period for historical price quotations, then evidence that a selection process did indeed take place becomes crucial. (Though the white pages directory in Feist was limited to a particular year and geographical area, the telephone company was found not have exercised an act of judgement since it was required by law to provide a directory.)

Currently, no one knows precisely what the threshold is for the minimum exercise of judgement required to establish copyrightability. As mentioned, comparatively trivial choices are probably sufficient, while exhaustive listings are probably not. An uncertainty in this area is seen in the Bellsouth Advertising & Pub. v. Donnelley Inf. Pub. decision. Bellsouth published a yellow pages directory that divided listed businesses into several

categories, listing names and addresses. The directory also had a cut-off date or inclusion of businesses in the directory. Donnelley re-entered the data contained in the yellow pages into a computer and assigned codes to data records that corresponded to the business categories in the yellow pages. Donnelley used the stored data to print sales lead sheets that were organized differently than Bellsouth's directory. Nevertheless, the court found that Donnelley appropriated the copyrighted elements of the directory, i.e. "the coordination of informational components in a business listing and the selection of categories." After the court rendered its opinion, it vacated the opinion and ordered a rehearing of the case, the result of which has not yet been reported. We do not know why the court decided to rehear the case; perhaps the original opinion appears to contradict Feist, since apparently Donnelley adopted only the *facts* from the yellow pages.

A database consisting of an exhaustive collection of data is copyrightable if at least a portion of records includes a minimal creative contribution by the compiler. This is seen in Corsearch, Inc. v. Tomson & Tomson, which involved a database containing state trademarks of the fifty states and Puerto Rico that was found copyrightable. Though containing, essentially, publicly available trademark information, certain additions and modifications were made: the database contained an indication of whether a given trademark is a word, a design, or both; a classification of goods and services in certain records was modified or added; foreign words were translated into English; Greek letters were given Latin alphabet equivalents; etc. Thus if the items in an exhaustive database are enhanced by additional commentary, explanation, or cross-references, then the compilation ought to be protectable.

Putting aside selection for the moment and turning our attention to data arrangement, an exhaustive collection of facts is protectable if there is a certain originality in the arrangement of the items. Clearly, it is possible to write a program that extracts data from a diskette or CD ROM and *rearrange* them. Since the copyright extends only to the original arrangement, no infringement occurs. In practice, however, it is more probable that an unauthorized copy of the original diskette or CD ROM would be made, without investing the time and cost of writing a program for the sole purpose of circumventing the copyright. It is simply easier to make a copy than to develop software. In addition to malicious copying, unauthorized copying may occur because of negligence or a misunderstanding of a contractual agreement.

In general, a typical relational database can be viewed as a set of tables containing a particular arrangement of the data within it. The creator of the database exercises significant subjective judgement in establishing the database scheme, i.e., in partitioning the database into tables and defining each table's attributes. This exercise of judgement has been held copyrightable, which is seen in Budish v. Gordon. Very simplisticly tabulated data, in which practically no subjective choices in arrangement are made and which result in a simplistically structured database, may be found to be not original enough to be copyrightable. We see this in Victor Lalli Enterprises Inc. v. Big Red Apple, Inc., where the court found that a chart containing historical horse racing data that was organized as a grid, with columns headings indicating months and rows corresponding to days, was not protectable. The court said ruled that Feist dictates that more original expression is required than was evident.

As mentioned earlier, copyright protects expression, not ideas or functionality. In certain cases, an idea or a method is so inseparable from its expression that a court may deny protection of an expression because it would necessary imply a monopoly on functionality of some sort. This may occur with very simple databases, containing exhaustive data, in which practically no choices in arrangement of the data can be made. This concept is called the "merger doctrine," which is a merging of functional and expressive considerations. This kind of merging always works against copyrightability. In more sophisticated databases, the selection of data categories and arrangement is not forced by functional considerations, i.e., is not unique, and consequently reflects a creative act of expression by its designer.

Although, in theory, a database that is an electronic equivalent of the telephone white pages (i.e., a comprehensive list trivially arranged) is not protected, it is possible to "package" a database product so that it can be protected, but in a physical rather than a legal way. One way is to encode the data and then provide purchasers with software that is required to access it. A copy of such a database would be useless without copying the access program, which is itself protected by copyright, at least against direct copying. A diskette or CD

ROM may be protected against direct copying by placing several exhaustive data compilations on a single medium. For example, even if a given court finds that a data base of historical price data for a given commodity is not sufficiently original to be copyrightable, a CD ROM that contains a number of databases for several commodities may be protectable because an original *selection* was made.

If an exhaustive database is provided on-line, theoretically a subscriber can extract all the items and then resell it. Although an on-line database itself should be copyrightable since it is typically accessed using software provided by the vendor, an exhaustive collection of stored facts is not. In practice, however, the value of a typical on-line database is in its currency and up-to-dateness, since it is kept current on a daily basis, or even more frequently, and any data extracted quickly becomes outdated and loses its value. Moreover, an intelligent pricing structure, one that charges an appropriate amount for each and every query, will make it economically disadvantages to down-line load large portions of data for eventual resale.

Paper Session: Technology Transfer

Chair: Milton White, DATANAMICS

AI and Computer Supported Cooperative Work

L. J. Thomae

S.W.I.F.T.
P.O. Box 2005
Culpeper, Virginia 22701

Abstract

This paper is intended to show how AI practitioners need to examine Computer Supported Cooperative Work (CSCW), or groupware principles in order to build systems that will better serve the needs of the mainline, non-research users. This paper will discuss CSCW in AI terms and show how the technologies are complementary.

Introduction

The early history of expert systems and mainline Artificial Intelligence (AI) applications consisted of single user, stand-alone systems which tended to embody the know-how of an individual expert working alone. For example, a traditional expert system is one which could diagnose engine problems, for example General Electric's CATS-1 expert system [6]. While two individuals could run this expert system simultaneously (each running an individual diagnostic session of the expert system on a different computer), the expert system would not take into consideration the results of the input of the first user to aid the second user in reaching a successful diagnosis. Currently the emphasis of the expert system community in the business world is on embedded expert systems, but even these systems traditionally have single users, or computer programs, changing and using the data and knowledge of the system at any given point in time. In other words, expert systems tend not to be cooperative in nature.

While the advent of computers has definitely helped industry, they have sometimes detracted from the cooperative community spirit that is fostered in many organizations and the ability of personnel to share knowledge and information through the use of the computer. Information that the "bull pen" atmosphere of most work environments promote.

Recently, there has been a growing emphasis on a concept called "groupware" or Computer Supported Cooperative Work (CSCW). Although the field of groupware is still relatively young (the first CSCW conference was held in Austin, Texas in 1986), there have been a few attempts at designing "intelligent" groupware systems. Examples of intelligent groupware systems include: Information Lens for E-mail systems at MIT [5], Callisto for project management at Digital [7], and ArgNoter for group decision making at Xerox Palo Alto Research Center [8]. These systems have all tried to emphasize presenting intelligent functionality to cooperating end users; functionality beyond the normal "user friendly" interface.

Information Sharing

Groupware's main concern is with information sharing. Information sharing "has to do with disseminating information so that it reaches those people to whom it is valuable without interfering with those to whom it is not" [5]. The emphasis in groupware is normally on utility, or functionality, and usability, or interface. Groupware systems are set apart from other software systems such as distributed databases, word processors, spreadsheets, and even most expert systems. This is because in order for a system to be considered "groupware" the system needs to have a sense of different individuals within an organization and of their individual roles. The needs and dynamics of those individuals as a group must be taken into consideration.

Interestingly, CSCW is a technology that is being pushed by the needs of the business community. To date there are few commercial CSCW systems in the market place and, as seen at the last CSCW conference (CSCW '92, Toronto), there is a growing need in the workplace to allow individuals to work together and still benefit from advancing technology. Yet there are very few products that will satisfy those needs.

This concept is the inverse of many other technical conferences and areas. Ordinarily, research technology is more mature than the business community, and technical vendors spend their time trying to convince organizations of how a specific technology will satisfy a corporation's needs. With CSCW the business world is crying for help and the technology is lagging behind.

Working Definitions

Definitions for evolving technologies are hard to explicitly define. Definitions change over time and with each new exploration and system developed. CSCW and AI are terms that express different concepts to different people and different technologists. Below are the definitions used for the sake of common understanding while reading this paper.

CSCW

Few researchers or technologists have tried to explicitly define computer supported cooperative work, and those who do often base their definition on the ability to share tools, materials, and data. The idea of designing computer technologies to support people working together dates back to Douglas Engelbart's Augment system in the early 1960's [1], but has only recently acquired the status of a new direction in computer research and development. The term computer supported cooperative work was coined by Irene Greif in 1984 to delineate a new field of research focused on the role of the computer in group work [9]. In the development of CSCW over the last several years, the term CSCW has opened as many questions as it has answered.

Although researchers disagree about the definition of computer supported cooperative work, the current definitions tend to focus on the technology being used. Under some definitions, CSCW applications encompass hypertext, computer conferencing tools, electronic calendars, computerized video-conferencing, intelligent electronic mailing systems, and any other system that can be said to enhance communal work among people. This definition-by-technology resembles the way AI is frequently defined: neural nets, production systems, natural language, etc. CSCW is typically envisioned as a conjunction of numerous technologies and users, with an overall view that emphasizes convivial work relations. CSCW is almost more of a social movement within a technical sphere as opposed to a specific technology or philosophy.

Certain terms are used almost interchangeably with CSCW: "groupware," "shared minds," "seamless systems," "collaborative systems," and "intellectual teamwork." These terminologies all show the social concept of CSCW and the concept of social interaction. This differs from some visions of AI which often have the view of the individual--a sole "intelligent" computer or set of computers which can pass the Turing Test--not the vision of a group of computers conversing with each other or of a group of individuals conversing through a computer.

In 1984 when the term CSCW was first used, it was to summon researchers and developers to a series of ACM sponsored conferences that examined how individuals worked in groups and how technology can be used to enhance group interaction [3]. In 1986 the ACM began sponsoring biannual CSCW conferences. Now the term "groupware" has become a desirable product buzzword, which is often used to suggest the concept of technology breaking away from its confinement of individual desktops or individual users.

Most researchers in the field consent, though, that the two dimensions, or variables, that are critical to CSCW are time and space. That is, determining whether the work is to be accomplished synchronously or asynchronously and whether the collaborators are to be co-located or not co-located. These dimensions affect the degree to which the interfaces of the computer

systems must be shared or temporally constrained. See Figure 1.

AI

Artificial intelligence, while an older and more mature technology, is in some ways as difficult as CSCW to find two technologists who will agree upon a specific definition. Patrick Winston defined AI in his book Artificial Intelligence in 1984 as ". . . the study of ideas that enable computers to be intelligent" [12]. While this definition of AI is simple and clean, the problem, for AI technologists has always been with defining what is exactly intelligence and how one goes about defining intelligence. Does one need to be able to communicate, reason, and apply knowledge to be intelligent? Or perhaps should one also be able to cooperate and interact in one's environment to be considered intelligent?

CSCW and AI: How They Tie Together

Computerized cooperation or collaboration is difficult to obtain and requires an understanding of the way groups and organizations function. In the same way AI is difficult, for AI requires an understanding of the way experts and people function. As AI researchers have dealt with the problems of natural language and understanding over the years, CSCW researchers are now having to analyze similar world knowledge to allow individuals to communicate effectively in the complex area of their workplaces. Ideally, the research from both fields will bring about new ideas, and knowledge from each field will enhance the other.

Many of the original, so-called successful, groupware projects and research (for instance: Information Lens, Object Lens, and Callisto) make some use of artificial intelligence techniques. For as Irene Greif has stated, "It is interesting to note that these systems all make some use of artificial intelligence techniques. While CSCW systems are not necessarily dependent on artificial intelligence breakthroughs for successful implementation, the coordination-technology pairs of these domain-specific CSCW systems make heavy use of well-understood artificial intelligence technology. These systems are rule-based, and represent information about the tasks by means of data structures tuned to supporting reasoning about such things as user roles and routing of information" [2].

Patrick Winston has listed the goals of artificial intelligence as: one, "to make computers more useful" and, two, "to understand the principles that make intelligence possible" [12]. Therefore, accepting Winston's definition of AI, and Winston's goals as the goals that constitute the goals of the AI field, it is difficult to distinguish the goals of AI from the goals of CSCW.

Hurdles

Groupware has its proponents and opponents, as has artificial intelligence. Even many of

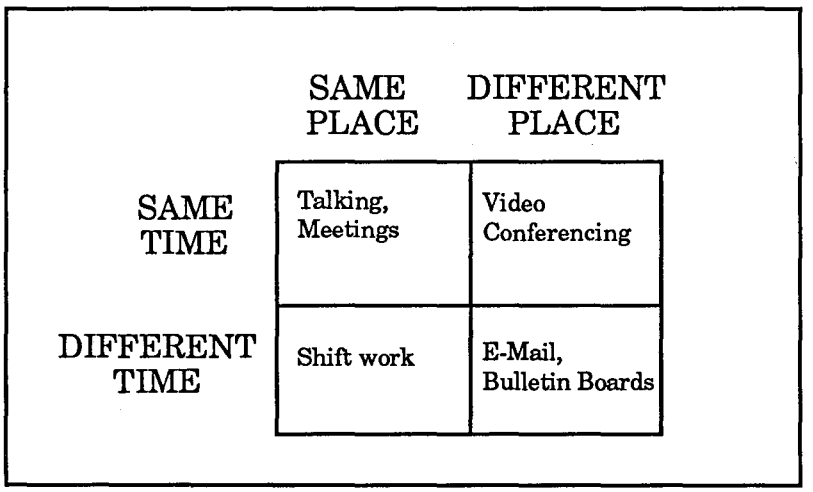


Figure 1. Temporal/Spatial CSCW Constraints

groupware's proponents feel as if there have been few, if any, successful groupware applications. E-mail has been considered by some to be perhaps the only marginally successful and accepted groupware system commonly out in the workplace; and E-mail can only be considered to be groupware in the broadest sense.

One of the biggest hurdles of groupware is the hurdle of the critical mass. For as Patrick Winograd has stated, "Work is not carried out by a homogeneous collections of individuals. Every work setting contains groups with collective interests, which can be affected by the introduction of computer systems. The redesign of work is a negotiation among the groups already doing and supervising the work, and the results will be shaped by the interests of these groups and the compromises among them" [11].

Therefore, in order to ensure the success of any computer system, one needs to obtain the cooperation of the users of the system. For unless the users were willing to use a system, and not revert back to previous work practices, a system will be rendered ineffective. This is even more critical for CSCW applications. For the most part, groupware systems, even more than other computer systems, must be used by all individuals, or they become a burden for the few users who try and use them. (Think of electronic schedulers, if only half the individuals are diligent about logging meetings, vacations, etc. the system becomes useless for the individuals who do try and use the system to schedule a meeting for all must attend.)

Conclusions

Since the advent of the term CSCW in 1984 there has been considerable controversy, hype, and talk about the groupware, but very few actual applications. This evolution of a "new" technology has had a similar background to how AI emerged and how AI, itself, is still being defined. It is the premise of this paper that the two technologies are complementary and require each other in order to fully succeed as deployed systems that are used daily in the business workplace.

As the needs for CSCW become more and more apparent in today's corporations, more effort, resources, and budget will become available for CSCW research. It is apparent from CSCW conferences and CSCW literature that AI and CSCW will have no choice but to aid each other in order for successful CSCW products to be developed. Many good AI systems have floundered when they could not be seamlessly embedded within companies applications, and even embeddable systems had the same problem of all other computer systems, that of being individually driven.

The view that work is fundamentally social is not a view that has traditionally been taken by AI systems or expert systems. The view that most activity, even in decision making, is cooperative is a new one for AI technologists, but one that must be addressed to handle the problems that are facing many businesses.

But, in the end, as our world becomes smaller and corporations become more distributed and computerized one of the most obvious effects of computer systems is the replacements of face-to-face verbal interactions, with computer-mediated exchanges, which will use a combination of AI and CSCW technology. Sharing information yet at the same time reducing information overload by disseminating the information so that it reaches those people to whom it is valuable without interfering with those to whom it is not.

Acknowledgments

Douglas Dankel II, my major professor at the University of Florida and Robert Phelps, my manager at S.W.I.F.T.

Note: The majority of this paper was taken from thesis work done for the University of Florida.

References

[1] Engelbart, Douglas C., Ed. (1991) *The Augmentation Papers: A Collection since 1960.* Fremont, CA: The Bootstrap Institute.

[2] Greif, Irene Ed. (1988) *Computer-Supported Cooperative Work: A Book of Readings.* San Mateo, CA: Morgan Kaufmann Publishers, Inc.

[3] Grudin, Jonathan. (1991) "CSCW Introduction." *Communications of the ACM.* 34(12): 30-34.

[4] Malone, Thomas W., Grant, Kenneth R., Lai, Kum-Yew, Rao, Ramana, and Rosenblitt, David. (1988) "Semistructured Messages are Surprisingly Useful for Computer-Supported Coordination." *Computer-Supported Cooperative Work: A Book of Readings.* Edited by Irene Greif. San Mateo, CA: Morgan Kaufmann Publishers, Inc.: 311-331.

[5] Malone, Thomas W., Grant, Kenneth R., Turbak, Franklyn A., Brobst, Stephen A., and Cohen, Michael D. (1987) "Intelligent Information-Sharing Systems." *Communications of the ACM.* 30(5): 390-402.

[6] Rauch-Hindin, Wendy, B. (1988) *A Guide to Commercial Artificial Intelligence.* Englewood Cliffs, NJ: Prentice Hall.

[7] Sathi, Arvind, Morton, Thomas E., and Roth, Steven F. (1986) "Callisto: An Intelligent Project Management System." *AI Magazine.* 7(5): 34-52.

[8] Stefik, Mark, Foster, Gregg, Bobrow, Daniel G., Kahn, Kenneth, Lanning, Stan, and Suchman, Lucy. (1987) "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings." *Communications of the ACM.* 30(1):32-47.

[9] Suchman, Lucy. (1988) "From the Program Chair." *Proceedings CSCW'88 (Portland, Oregon).* New York, NY: ACM Press: V-VI.

[10] Thomae, L. J., (April 1992). *TM: AI and CSCW.* Master's Thesis, University of Florida.

[11] Winograd, Terry. (1988) "A Language/Action Perspective on the Design of Cooperative Work." *Computer-Supported Cooperative Work: A Book of Readings.* edited by Irene Greif. San Mateo, CA: Morgan Kaufmann Publishers, Inc.: 623-653.

[12] Winston, Patrick Henry. (1984) *Artificial Intelligence.* Reading, PA: Addison-Wesley.

Technology Transfer: An Expert System for Adoption of Innovation Decisions

James Bowen
CompEngServ Ltd.
Suite 300, 19 Fairmont Ave.
Ottawa, Ontario K1Y 1X4

Uma Kumar
School of Business
Carleton University
Ottawa Ontario, K1S 5B6

Abstract

In the present environment of increasing rate of technological change, successful adoption of technological innovations is a prime competitive advantage. Thus, understanding the possibilities of innovation diffusion is of concern for both the innovator and adopter. While many researchers have developed frameworks of adoption or diffusion, the issue of testing these frameworks still needs to be addressed. This paper addresses that issue by first combining several propositions or considerations/issues which various authors claim are critical into an adoption of innovation knowledge-base. This knowledge-base can then be used for determining the possibility of innovation adoption in an organization. Second, this knowledge-base was implemented as an expert system and tested through case studies. The results indicate that the system can successfully predict adoption of innovation. In addition, such an approach could have tremendous use in validating adoption of innovation frameworks and assisting in the actual evolution of innovations.

1. Introduction

In general, there are two kinds of innovations. Discontinuous innovations use new products, services or systems to alter existing patterns of production or consumption or create new patterns [1]. In other words a product or service or process which has the potential to "create or revolutionize markets and demand" [2]. Adoption of continuous innovations is perhaps easier than discontinuous innovations in that it does not cause such a disruption and that methods such as technology monitoring/forecasting attempt to foresee such events [3].

For the purposes of this paper we will declare an innovation as being adopted when the organization decides to adopt e.g. at the end of Rogers' [4] Decision stage and before his Implementation stage. Each organization when deciding to adopt an innovation must deal with several considerations, most of which involve the uncertainty and cost of switching to the innovation [5]. As a result many authors have studied adoption from different perspectives e.g. consumer research ([4], economics [6] and organizational behaviour [7]. A great deal of research has focused on developing frameworks for diffusion or adoption of innovations. Some authors [5] [8] have brought together such work into a combined framework for further research. Arnould [9] has tested a framework with cases in different culture settings for consumer adoption. Where it was found that culture influences affect diffusion of innovations. Upon developing their combined framework Robertson and Gatignon [5] believe "a new stream of research" using "experimental designs" is necessary to test their framework. While they recognize the problem of external validity, they suggest moving from the survey methodologies and developing an approach "in which respondents make adoption decision facing different

competitive environment as described in scenarios" [10] [11]. The purpose of this paper is address the research issue of testing such a framework for industrial adopters in a manner suggested by Robertson and Gatignon [5]. This will be accomplished by distilling from previous authors knowledge about adoption of innovation, expressing this knowledge as an adoption model in the form of rules, constructing an expert system version of the model and testing it using case studies. The following section presents the adoption of innovation by organizations knowledge-base. The subsequent section discusses expert systems and is followed by a section discussing the computerized implementation of the knowledge-base into a rule-base expert system. The next section discusses the results of test cases examined by the expert system. The final section presents conclusions and further research areas.

2. The Distilled Adoption Of Innovation Knowledge-Base

To begin to understand whether an organization will adopt an innovation we divide the considerations/issues into those related to the external environment and those related to the internal environment. In the external environment, such issues as the suppliers of innovations actions, and the competitor's practices are the focus. While, the internal environment focuses on the organization itself, for example, its stability and current practices.

The decisions for adoption of innovations are partly influenced by the compatibility between the innovation's characteristics and those of the potential adopting unit [5]. As well as, the competitive forces by other organizations in the same industry. The literature review produced the following considerations/issues.

Supplier: Since innovations contain an element of technological uncertainty e.g. will this innovation work for us, the reputation of supplier of a innovation is important [5]. If a supplier has the potential adopters' confidence and an established reputation then the faster and higher the adoption rate of technology [5]. Thus, two factors must be considered: the technological uncertainty of the innovation, and, the strength of relationship and confidence attributed to the supplier.

Competition: The higher the level of competition in an adoptor industry the higher and quicker the adoption of innovation takes place [5]. This can also be expressed as the greater the importance of innovation to achieving or sustaining competitive advantage the greater the adoption rate [12]. The competitiveness of the industry can be measured as three factors: number of competitors, the concentration ratios and mobility barriers or barriers to competition [13]. However, if competition becomes too high, industries will not have the financial resources for constant innovation. Before the maximum level is reached innovation needs could switch from product to process innovations [14].

The Innovation's Development Status: The innovation itself may come to the attention of the adopter organization at different stages of development. Two such stages could be before or after the innovation, in the form of technology, has settled on a dominate design [14]. If a dominate design or standardization technology has been developed this allows the organization to take advantage of the experience curve. A organization must ensure that it is the dominate design and will not be quickly succeeded by another design [15].

Vertical Coordination: between Supplier and Customer: A high degree of vertical dependence should help diffuse the innovation faster since it leads to better information flow [7].

Lead Users: Lead users are those "who have needs which are not now prevalent among users of a given product but which can be predicted to become general". If an organization can be identified as a lead user then it should more quickly adopt an innovation. As well, this could spur other organizations to adopt [16].

R&D: If suppliers spent a deal of money on R&D then more technological choices should become available. More technological choices should find a wider range of adopters with suitable needs and thus increase innovation adoption [5].

Marketing: Many marketing studies have shown that the greater the resources committed to marketing of the innovation by the supplier firms the greater the adoption/diffusion rate [5]. The factors of marketing which have been identified as most relevant are: advertising, personal selling, promotional support and distribution support.

Type of Industry: The type of industry is also important in that a highly homogeneous industry has basically the same information concerning innovations and thus the competitive advantage of (and need for) the innovation is reduced [5].

Uncertainty of Demand: The greater the uncertainty of demand of the adopter's product and its target markets needs, the greater the adoption rate [5]. The following statement has several factors which must be determined before applying it: cost is a barrier to entry; the adopters target market is a heterogeneous market; marketing potential will be realized from price reductions; and, there currently is changing consumer needs.

Communication between Potential Adopters: The greater the frequency of announcements or other information flows between potential adopters in the samc industry the greater the adoption rate [5]. The one factor serving as an condition to that statement involves the clarity of the communication e.g. is it truthful and unambiguous [10].

Type of Workers: Professionals are defined as those who identify with a profession as well as a firm. Thus, they receive information from both their professional sources and the organization they work for, and, can understand an innovation potential in relation to their profession [17]. If the organization has a great number of professionals then innovations from outside the industry could be more readily discovered and received [5]. However, they may be harder to propose [4].

Global View: In order to access information about innovations, organizations need to have a wide range of sources. Thus, the greater the range of sources used by the adopting organization, the more innovations will become known and the higher the rate at which they will be adopted [5].

One of the areas which slow adoption of innovation is the resistance to change existing internally to an organization. This can be divided as several considerations/issues as follows:

The innovation most have an advantage relative to the concerns of the target organization [18].

If the organization engages in technology monitoring then it is more likely to find and adopt innovations [3] or if constant innovation is necessary to survive [12].

A higher potential for adoption is possible if the innovation fits the current practices of the organization [18]. For example, will the innovation cause a major disruption of current organizational business practices [12]. Other departments which will be effected should be in agreement or informed [12].

The enduser should perceive the innovation to be simple to use [18].
The innovation should be explainable in terms the user is familiar with [18].
The innovation should be designed so it can be introduced as a prototype [18].
Acceptance is speeded if the user can withdraw from the trial with small losses [18].
The bigger the relative cost the less likely the innovation will be introduced [18]. In contrast, the bigger the relative advantage over current operations the higher adoption rate [4]. This can be expressed as economic profits, initial costs, savings in time and effort, as well as, the resource constraints, will impact the decision [19] [4].
If the organization has identified this innovation as a good one but has not found a problem for it to solve it is more likely to be adopted [4]. As well, has the organization a particular problem which they are currently spending resources trying to invent a solution [20].
What is the public cost of failure? The higher the public embarrassment of failure of the innovation the less likely it will be adopted [18].
If the organization is in a situation of great turbulence or crisis the innovation should be perceived as different [18].
The greater the organizational bureaucracy the less likelihood of innovation-induced change [18]. In other words, the greater the centralization the less innovativeness [4].
If the organization's compensation or reward structure punishes creativity, etc. the less likely of champions for innovations [18].

3. Expert Systems

Artificial Intelligence (AI) atttempts to model decision problems requiring symbolic reasoning, for example, deciding personnel policy issues. The field of Artificial Intelligence covers many technologies such as machine learning, robotics, natural language processing, vision, speech recognition, and expert systems [21]. Each Artificial Intelligence technology typically is explained in terms of reasoning and representation approaches. Representation involves the form in which knowledge is encoded in the computer while reasoning is the approaches to utilizing that knowledge. This paper focuses on two Artificial Intelligence technologies: Expert systems and machine learning.
Expert systems are interactive programs suitable for semi-structured problem domains, where knowledge is incomplete and/or uncertain and/or inconsistent and/or dynamic. Relationships between concepts are symbolic rather than numerical [22] [23]. An expert system helps to bring together diverse knowledge (experience, heuristics, academic knowledge, etc.) from many possible sources into a coherent useable form. Knowledge can be represented in an expert system in many forms such as semantic nets or frames; the most common representation is rules.
A rule-based expert system is an attempt to derive decision recommendations [24] based upon the application of rules. A rule-based approach is most suited for knowledge intensive decision problems [25]. Rules are typically in the form of:

> IF condition is true
> THEN draw conclusion or
> perform action.

A typical expert system is divided into the following four interacting parts. The first part is the knowledge-base containing the problem's facts or other static information and heuristics possibly in the form of rules. The second part is the working memory containing relevant data for the current problem being solved. The third part is the reasoning method or inference engine controlling the organization of the problem data and the route to be followed through the knowledge-base. The fourth part of an expert system is an interface module which interacts with the user or databases.

4. ADINES: Adoption Of Innovation Expert System

Once a knowledge-base, which predicts adoption of innovation has been developed, the next step is to implement it into a testable form. This section describes the expert system which was developed from the above mentioned knowledge-base.

The approach used to develop this expert system is to combine the distilled knowledge available from theoretical, empirical and practical experience into an expert system. This approach has been successfully used in devising an expert system for advertising design [26]. The adoption of innovation considerations/issues has been expressed above as a knowledge-base in the form of rules. The weight/significance/importance between the rules will be developed on the basis of experience.

In order to determine whether an innovation will be adopted a scoring scheme will be used [27]. This scoring scheme represents the relationship between the rules. The scoring scheme uses integer values ranging from -10 to +10 points for each issues's score. In the current version of ADINES, this scheme will increase the likelihood of the conclusion being true by 10 points with each piece of positive evidence (or each of the adoption of innovation knowledge-base's consideration/issues being stated as true) examined by the prototype. While negative evidence will decrease the likelihood of the innovation being true by 10 points. There will be no change in the score if the question or points is not applicable or if the user is unable to answer e.g. unknown. In later versions of ADINES the integer score value assigned individual rules can be changed if further study determines a different importance or weight of each issue. Given the number of issues to consider the maximum total score is 250 points. Based upon practical experience, a threshold of 100 points will be assumed to divide the potential for adoption between those likely and those that are unlikely.

By reducing the complexity of the adoption of innovation issues into simple queries, the user can be queried for simple responses (such as yes, no, not applicable). The use of simple queries allows the prototype to use the expert's knowledge to evaluate the responses. This removes the uncertainty the user may have introduced into the system and has worked well in previous applications. A previous application developed for electromagnetic interference resulted in a robust system which could function even with a great number of wrong responses by the user [28]. This method can be modified to allow the user to judge the significance of each question, if desirable, in later versions.

The following examples illustrate a sample of the rules which have been implemented:

ASK the user - **Does the organization's compensation or reward structure reward creativity?**
IF the organization's compensation or reward structure rewards creativity THEN SCORE IS SCORE + 10.
IF NOT THEN SCORE IS SCORE - 10 AND APPLICABLE IS YES

ASK the user - **Will the organization suffer a larger public relations embarrassment if the innovation adoption project fails?**
IF the organization will suffer a large public relations embarrassment THEN SCORE IS SCORE - 10 AND APPLICABLE IS YES.
IF NOT THEN SCORE IS SCORE + 10

A response of 'not applicable or unknown' does not change the score, since many factors could invoke such a response, such as incomplete information.
The expert system shell chosen to implement the system was Level 5 Object. An expert system shell was chosen for development instead using a programming language to take a cost advantage of the debugging, screen generation, pre-prepared inference engine and other features available in such shells.

5. Testing Of ADINES

ADINES will be tested using four test cases: The Grumman Corporation by Harvard Business School (1979); Transprovincial Engineering Limited by The University of Western Ontario (1985); Adjeleian And Associates by The University of Western Ontario; Hudson Bay Mining and Smelting Co. by the University of Western Ontario.
The Grumman Corporation case is set in early 1975 and involves a corporate decision as to which technology the company should pursue in its R&D efforts. The corporation is trying to determine which innovation area to pursue research in. The company is considering investing in projects designed to produce solar products/services for consumer sales. They must chose between solar heating, wind, photovoltaics, solar space satellites, ocean thermal electric, biomass, agricultural and industrial process heating, and, hydro power. For the purpose of this case, we will only look at one: photovoltaics. The choice is arbitrary, since ADINES could be applied to any of the innovation areas.
The Transprovincial Engineering case is set in early 1982 and involves a corporate decision to purchase a CADD system. The management must analyze this relatively new technology and decide whether to invest in it. They are motivated by rumours that customers will start discriminating against them if they do not offer such abilities as CADD and that their competition will soon offer such abilities. However, the technology is not mature, expensive and will create problems internally in the organization.
The Adjeleian Assocates case set in late 1983 is similar to the transprovincial case, in that the company is looking at purchasing a CADD system.
The Hudson Bay Mining case is set in early 1969 and involves chosing between two mining processes. One a conventional approach and one based on proof-of-concepts. The choice of the conventional approach could mean lost market share due to less efficent operation, the new approach could mean lost of market due to poor

technology or its implementation. However, the new approach could bring lower costs.

Appendix A reproduces a sample of the dialogue between the user and the ADINES for one case. ADINES predicted that Grumman should not pursue photovoltaics and that at this time CADD was not appropriate for Transprovincial. These outcomes agree with the actual outcomes of each case. However, it is interesting to note that some of the 'no' responses could change as the technology of company's business environment changes. For example, the score for Grumman to pursue photovoltaics was near the threshold of 100 points and a few different answers could change the outcome. As an example, in the Adjeleian and Associates case ADINES recommended not to adopt the innovation. However, if the vendors of the CADD systems had made a trial or prototype versions available or some how reduced the cost of failure ADINES would have increased the score by 40 points (although the new score would still be below the threshold and ADINES still would not recommend its purchase). In fact, this is what happened when subsequently one of the vendors reduced the price and the Adjeleian and Associates decided to purchase. However, at a later date the company admitted the system was not successfully adopted. Interestly enough at a later date it was used as a graphical media proposal demonstration device and helped land a major contract. Finally, in the Hudson's Bay case ADINES recommended not to adopt the new mining approach.

6. Conclusion And Further Research

In conclusion, the system worked and successfully implemented an "experimental design" "in which respondents make adoption decision facing different competitive environment as described in scenarios" [10] [11].

Obviously, for future research more test cases could be used, the scoring system can be refined and more expertise could be entered. Another interesting area is to re-examine the weight assigned per rule and perhaps empirically derive appropriate weights for each adopter (or their industry) and each kind innovation. However, in its present form many interesting results were obtained.

One interesting result from testing with the cases was the ability to focus an innovation supplier's efforts. Upon running the system, an innovation supplier can see which points are working against their innovation and their importance in adoption for customers. The suppliers could then try to change their innovation for the better. This is consistent with Kline and Rosenberg's view that innovations start crude and evolve within the original framework. Perhaps this is the real benefit out of developing such systems because they could assist in helping evolve innovation themselves.

One other result is worth noting, technology transfer can be defined as that the process of incorporating research, empirical and practical results into a form for others e.g. industrial users, to benefit from. Thus, ADINES is an example of using expert systems for technology transfer [29] [30]. In this case, an inter-organizational transfer from universities to industry. While, the present knowledge about technology transfer can be generally expressed as rules, are there superior methods of representation? It has been said that rules are used by beginning experts and reasoning by analogy by experienced experts. Perhaps, as a technology transfer vehicle expert systems could use case-based reasoning and neural nets [21].

APPENDIX A: Sample Dialogue between ADINES and User for the Grumman Case

Expert System: "This expert system incorporates research literature of empirical results and practical experience to suggest if a organization will adopt a new innovation.

The following questions should be answered with knowledge about the adopting organization, the supplier of the innovation, the industry of both supplier and adopter, and, the innovation itself.

Note: That a response of 'unknown' may be entered if the query is not applicable or if information to answer the query is not available."

Expert System: "Does the innovation have a great deal of technological uncertainty?"
User: High

Expert System: "Does the organization have a good relationship with its innovation supplier?"
User: Good

Expert System: "Does the organization have confidence in the innovation supplier?"
User: High

7. References

[1] Robertson, Thomas, *Innovative Behaviour and Communication.*, New York, N. Y.: Holt, 1971.

[2] Shanklin, William, and Ryans, John "Organizing for High Technology Marketing", *Harvard Business Review*, November, 1984, 659 - 679.

[3] Balachandra, B., "Technological Forecasting: Who Does IT and How useful Is It", *Technological Forecasting and Social Change*, Vol 16, 1980, 75 - 86.

[4] Rogers, Everett, *Diffusion of Innovations*. New York, N. Y.: The Free Press.

[5] Robertson, Thomas, and Gatignon, Hubert, "Competitive Effects on Technology Diffusion", *Journal of Marketing*, Vol. 50, July, 1986, 1 - 12.

[6] Stoneman, P., "Intra-Firm Diffusion, Bayesian Learning and Profitability", *The Economic Journal*, Vol. 91, June, 1981, 375 - 388.

[7] Kimberly, John, and Evanisko, Michael, "Organizational Innovation: The influence of individual, Organizational and Contextual Factors on Hospital Adoption of Technological and Administrative Innovations", *Academy of Management Journal*, Vol. 24, December, 1981, 689 - 713.

[8] Mahajan, Vijay, Muller, Eitan, Bass, Frank, "New Product Diffusion Models In Marketing: A Review and Directions for Research", *Journal of Marketing*, Vol. 54, Jan., 1990, 1 - 26.

[9] Arnoulds, Eric, "Toward a Broadened Theory of Preference Formation and Diffusion of Innovations: Cases from Zinder Province, Niger Republic", *Journal of Consumer Research*, Vol. 16, Sept., 1989, 239 - 267.

[10] Heil, Oliver, "Signaling in Competitive Marketing Environments", Working Paper, The Warton School, *University of Pennsylvania*, 1985.

[11] Robertson, Thomas, and Wind, Yoram, "Organizational Psychographics and Innovativeness", *Academy of Management Journal*, Vol. 26, June, 1980, 332 - 338.

[12] Day, Ralph, and Herbig, Paul, "How the Diffusion of Industrial Innovations is Different from New Retail Products", *Industrial Marketing Management*, Vol. 19, 1990, 261 - 266.

[13] Porter, Michael, *Competitive Strategy*, New York, N. Y.: The Free Press, 1980.

[14] Abernathy, William, and Utterback, James, "Patterns of Industrial Innovations", *Technology Review*, Vol. 80, June, 1978, 41 - 47.

[15] Farrell, Joseph, and Saloner, Garth, "Standardization, Compatibility and Innovations", *Rand Journal of Economics*, Vol. 16, Spring, 1985, 70 - 83.

[16] Kennedy, Anita, "Development, Adoption and Diffusion of New Industrial Products", *European Journal of Marketing*, Vol. 17, 1983, 31 - 87.

[17] Leonard-Barton, Dorothy, "Experts as Negative Opinion Leaders in the Diffusion of Technological Innovation, *Journal of Consumer Research*, Vol.11, March, 1985, 914 - 926.

[18] Bright, James, "Strategies to Speed the Adoption of Innovation", *IEEE*, 1991, 660 - 664.

[19] Malecki, Edward, *Innovation Diffusion Among Firms: A Dissertation*, Ohio State University, 1987.

[20] Foxall, Gordon, "User Initiated Product Innovations", *Industrial Marketing Management*, Vol. 18, 1989, 95 - 104.

[21] Carbonell, J., *Machine Learning: Paradigms and Methods*. Cambridge, Mass.: The MIT Press, 1990.

[22] Harmon, P., and King, D., *Expert Systems*. New York, N. Y.: John Wiley and Sons, 1985.

[23] Waterman, D.A., *A Guide to Expert Systems*. Reading, M. A.: Addison-Wesley, 1985.

[24] Leigh, W.E., and Doherty, M.E., *Decision Support and Expert Systems*. South-Western Publishing Co. 1986.

[25] Laudon, K. C., and Laudon, J. P., *Management Information Systems*. New York, N. Y.: Macmillian Publishing Company, 1991.

[26] Burke, Raymond, Rangswamy, Arvind, Wind, Jerry, and Eliashberg, Jehoshua, "A Knowledge-Based System for Advertising Design", *Marketing Science*, Vol. 9, No. 3, 1990, 212 - 229.

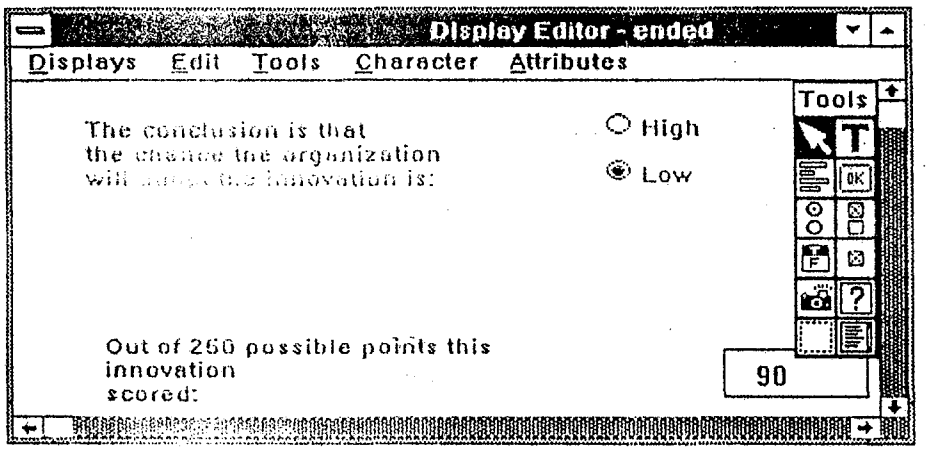
[27] Bowen, James, "An Expert System for Police Economic Crime Investigators", *Expert Systems With Applications*, Vol. 7, No. 1, 1994, Forthcoming.

[28] Bowen, James E., "EMIR: Expert System for Electromagnetic Interference Resolution", *Second International Conference on Industrial and Engineering Applications of Artificial Intelligence & Expert Systems*, ACM, 1989, 75 - 80.

[29] Rao, H. "Using Expert Systems for Technology Transfer", *Information Systems Management*, 1991.

[30] Rao, H. "Technology Transfer: Making Expert Systems Commerically Successful", *IEEE Expert*, Vol. 7, No. 2, 1992, 5 - 10.

Figure 1: Grumman Corporation



Paper Session: AI and Asset Allocation and Trading Signals

Chair: Barry Glasgow, Shearson Lehman Brothers

POET, an expert system weighted by time criticality, for Portfolio Enhancement

Bonnie Schnitta
South Fork Technological Consultants
29 Gann Road, East Hampton, New York 11937

Abstract

Wall Street profits often require not only a successful investment strategy, but also the input from rapid and accurate interpretation of real-time updates to information that is obtained from more than one source. Artificial intelligence (AI) systems have been designed to facilitate decision making under such adverse conditions. While these system automate much of the decision making, often too much information is provided to the user, or the information is not provided in a timely manner.

This paper presents a unique expert system for POrtfolio EnhancemenT, POET. POET can be used as a stand alone system, or it can be used in conjunction with an expert system like a neural network algorithm, as found in various standard approaches(1). What is unique about POET is that it mathematically alters the input, or AI results, to be inclusive of time criticality and dynamic prioritization.

Introduction

Successful investment strategy alone does not a profit make. Wall street profits are made when not only the indicators of potential profit or minimized loss for the investment strategy are noticed, but rapid interpretation of all the updates to the indicators from various sources that are required inputs for a chosen investment strategy are made in time to maximize the profit within the portfolio. Success under these adverse constraints of Wall Street requires the investor to continuously and accurately assess his or her position, taking into account necessary avoidance criteria and the fact that there could potentially be critical information that has been concealed or is presently not available. At the same time the investor must access the relative positions of the surrounding market and make rapid interpretation of real-time updates derived from more than one source. In summary when a decision must be made that could result in a great profit, there is more information than can be properly evaluated.

Also, when a certain event or events happen in the market place, such as a price differential or volume increase of stocks above or below the norm, a shift can be made in the priority of which stocks need action, depending on several parameters. One simple example of such a parameter would be the time limit before a decision is no longer as profitable. Another example is that two events can imply the potential for several profitable actions, but the action upon one stock would result in a greater profit or prevent a larger loss. That is, in order for decisions to be optimum they must be prioritized in a dynamic domain. (2,3)

Originally Wall Street addressed these problems by automating various investment strategies, using a diversity of expert systems. The expert system has been shown to have a good rate of return for its application (4,5). Unfortunately, systems like these have been noted to be less than effective, since by time the information provided by the system has been examined much of the investment timing has been lost. Or worse yet, when sell signals have been generated by the expert system, the time critical information went unnoticed, since the system had generated many sell signals that were not critical. Thus these artificial

intelligent systems have on occasion been perceived as not working properly.

These examples indicate at least two crucial improvements that can be made to enhance the portfolio, whether or not an artificial intelligent system is in use. First, a system should be designed so that not only are the rules which govern the actions on the investment strategies important, but also the priority by which each action is executed. That is, the time domain "meaning" or semantics of the information that is being processed by the investment strategy is of great important. Conceptually speaking this is the mathematical core of semantic data compression (2). Also, not only should only time critical information be presented, but all information should be presented in a format that summarizes the information into the most important information first. Both of these facts imply time criticality. In order to address the time criticality issue, an algorithm was developed which had as its foundation the mathematics that has been used in semantic data compression.

POET was designed to also mathematically hand shake with most existing artificial intelligent systems. POET's mathematical approach embraces the fact that time criticality is an important factor in portfolio decision making. What is unique about POET is that it mathematically addresses the fact that semantic data compression, time criticality and dynamic prioritization is an important factor in portfolio decision making and can be independent of any investment strategy used.

This paper presents the fundamental mathematics of the data compression and prioritization that comprise POET. An implementation of POET that was developed for evaluation is then presented.

Mathematical Design of POET

The algorithm design of POET not only had to include how the data was read into the POET processor, but had to consider the retrieval of the data for processing and output to some type of user interface. Various retrieval structures entail commitments that make it difficult to incorporate concepts that appear after the original structure is assembled (6). Thus the design was based on an open system that could be easily appended at future times for alternate applications. It was also important that the search not grow rapidly with the size of the files. Therefore, the processing included first the data compression and then the prioritization.

POET first compresses the original amount of data, D_0, into a user defined preferred quantity, D_C, that consists of a reduced data quantity, p. In order to understand the compression consider that all inputs to the investment strategy are defined by a function of m variables. All functions also have a designated standard variance, $\partial(0)_i$ $(i=1,p)$. This in turn allows the system on a real-time basis to review all functions for changes in its function value at an epoch in time n, $\partial(n)$ which have deviated from $\partial(n-1)$ greater than $\partial(0)$. We now define $D_C(i) = f(\beta)$

where:

$i = 1,p$

β = m parameters of the investment strategy specific to the i^{th} set

If $f(\beta) \leq \partial_i$, then $r=0$ and $P_r = 0$.

Each of the m variables of the function are also associated with a specific time response coefficient. The each member of the compressed set $\{D_C(1), \ldots, D_C(p)\}$ is then given a priority, $P_r(i)$ according to there resultant ranking weights ,r (i) where $(r=-w,1,\ldots,w)$, based on time criticality, t_c

The only essential design criteria that remains to be specified is how often does each data point or function cell send information to the ranking mechanism. Consider the one-dimensional situation in which each decision is a function only of r and t. Denote by n(r,t) the rank of the cell. In the absence of any neighboring influences we assume the cells can be in a quiescent state or send autonomously at a uniform rate which we normalize to 1. If the cells'

ranking is perturbed we assume it evolves according to

$$dn/dt = f(n)$$

where f(n) has zeros at n=0 and n=1. Let us now expand this to include neighboring variation and incorporate the effect on the ranking of a strategy such as $D_C(i)$, due to ranking of neighboring cell, $D_C(i+1)$. The variation is incorporated in a weighting function w. We must integrate the effect of all neighboring cells on the sending rate and we model this by a convolution integral involving an influence kernel. In POET's model we set a cell to have a short range activation effect and a long range inhibitory effect. This is typical of the cell behavior in pattern formation concepts based on short and long range inhibition (7).

Implementation

The mathematics that includes the weights of time criticality are independent of an investment strategy, whether or not it is implemented on an expert system. It was decided that the implementation and evaluation of POET was best performed in conjunction with an expert system whose mathematical foundation is common in the investment community. This was done to eliminate the debate whether or not its success as a stand alone could not be equaled by an expert system alone. An expert system that is used in the investment community is the neural network. The neural network that was implemented for the evaluation used a standard delta rule learning neural network (8).

An expert system, when it is processing stocks for portfolio optimization, produces a list of stocks with either buy or sell signals. This list depends on whatever parameters govern the algorithm. The problem that can occur is that should there be a significant move up or down in the market and the indices are one of the parameters of the expert system, the resultant number of stocks that are identified, or flagged, for action is quite large. Time being of the essence it is possible that important flags could be neglected, until it is too late for the action on the flag to be profit giving or protecting. This is due to the fact that all the information provided is more than the user can process in a reasonable amount of time, or the important actions are mixed in with the less important or profitable actions.

POET changes an expert system to reflect a time criticality factor. Like a neural network the basic mathematical model used for recall is based on nodes interconnected by different kinds of associative links. Unlike the neural net, each node for POET is a descriptor of an event and its associated action. The associative links contain rules or functions governed by a dynamic time criticality function that effects not only whether or not the information advances to the next node, but also orders the final information that is presented to the user.

What is interesting to note is that this approach is easily upgraded to include "trajectory" positions. That is, the term that is added to the expert system is such that one could expand the POET equation to project what the future deviation would be. This could speed up the prioritization, as well as allow the risk weight to be inclusive of a future deviation that might rank it substantially higher or lower.

Performance evaluation

The completion of the POET algorithm has been recent and its documented advantages only occurred for a small test time. None the less the system ran the neural net alone versus the neural net with POET from 1/93 to 4/1/93, using 600 stocks as input. It was assumed that the investor could only process 30 information blocks a day. Using the 30 top information blocks of the neural network vs. the 30 information blocks of POET, POET increased the performance of the neural net by 2%. If the processing limit was reduced, the performance was increased.

Summary and Conclusions

What POET provides is an improvement in not the data that is being presented to the user, but the improvement in the ability of the investor to make expeditious and insightful judgments. Many investors prefer not to use an expert system for what they do, or have an expert system and have found the system to be less effective than originally hoped.

The effectiveness of the expert system is not only algorithm dependent, but also user dependent. Once the expert system has been installed the issue becomes whether or not the system is being used and in a timely manner. Thus in a time critical environment the amount of information needs to be reduced and prioritized. It is important to note that the mathematical foundation of POET is designed to not only be used as a stand alone, but also append to most expert systems.

The goal of the POET structure was the elimination of the numerous unnoticed situations that can build up. Thus POET can be thought of as a mathematical structure to create "situational" awareness.

REFERENCES:

1. Roy Freedman, et al, Expert Systems in Spreadsheets: Modeling the Wall Street User Domain, The First Int. Conf. on Artificial Intelligence Applications on Wall St., October 1991, pp. 296-301.

2. Bonnie Schnitta and Roy Freedman, Increasing the Usefulness of AWACS Downlink Transmissions by Semantic Compression, NATO AGARD conference Proceedings No. 414, Information Management and Decision Making in Advanced Airborne Weapons Systems, ref. 18.

3. S. R. Davis, From Technology Push to Market Pull: Expert Systems in the Front Office, The First Int. Conf. on Artificial Intelligence Applications on Wall St., October 1991, pp. 220-225.

4. G. Jang, et al., An Intelligent Trend Prediction and Reversal Recognition System Using Dual-module Neural Networks, The First Int. Conf. on Artificial Intelligence Applications on Wall St., October 1991, pp. 42-51.

5. D. T. Cadden, Neural Networks and the Mathematics of Chaos - An investigation of These Methodologies as accurate predictors of Corporate, The First Int. Conf. on Artificial Intelligence Applications on Wall St., October 1991, pp. 52-57.

6. Marvin L. Minsky, Semantic Information Processing, Marvin L. Minsky (ed.), MIT Press.

7. G.F. Oster and JD Murray, Pattern formation models and developmental constraints, J.P. Trinkaus Anniversary Volume 1989.

8. Y. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Publishing Company, Inc., New York.

STAR: A Rule-Based System for Asset Allocation

Stephen W. Brockbank
Moran Asset Management
41 West Putnam Avenue
Greenwich, CT 06830-5300

Everett J. Rutan, III, CFA
Dolcyna Research
54 White Cedar Drive
Madison, CT 06443-1652

Abstract

STAR--Strategic and Tactical Asset Reallocation--is a rule-based system for asset allocation among stocks, bonds and cash. In STAR, conventional investment wisdom has been specified in exact relationships that have been rigorously tested. The objective is to maximize geometric mean return while limiting turnover and sensitivity to short-term market moves. The result is an active discipline that approximates a long-term 60/40 asset mix, but significantly outperforms its static counterpart in both risk and return by avoiding weak markets and exploiting strong ones.

STAR distinguishes between two types of rules: valuation rules that determine asset preferences and limit factors, that determine asset reallocations. Rule candidates were developed by questioning investment professionals and examining the correlation between market indicators and returns. The original trial-and-error testing and validation process has been replaced by genetic search optimization. This permits us to create customized models for other asset classes, markets and investment goals.

1. Introduction

Most investment authorities consider the asset allocation decision--the distribution of funds among major categories--to be the most important determinant of portfolio returns. Brinson *et al* [3, 4] find that over 90% of the variation in portfolio returns is due to differences in investment policy, far outweighing security selection.

Many studies [7, 9, 10] contend that attempting to dynamically adjust asset allocation in order to enhance returns is a risky if not futile endeavor. Because of the distribution of returns, failure to be invested during just a few months of high returns, or success in being invested in just a few months of low returns leads to long-term, below-market results. You don't have to miss by much to miss by a lot. When combined with studies showing the failure of active policy to add much value [3, 4] "market timing" has gained a reputation as a too-clever-by-half attempt to outwit the laws of nature.

STAR is a conservative, non-traditional asset allocation model which recognizes these facts of life and turns them to its own advantage. STAR recognizes the risk/return hierarchy of the Capital Asset Pricing Model (CAPM), favoring those assets with the highest return to risk ratio. There is a preference to be totally invested in stocks. STAR's total return is driven by avoiding down markets rather than seeking up markets. [12]

The next section describes the STAR model is described in detail, with attention to its rules, structure, estimation and use. The paper then examines different aspects of STAR's pro forma and actual performance. The lessons learned with STAR lead to two areas of current research: the use of genetic search optimization to select and refine the decision rules; and the generalization of the model structure to other markets and asset classes.

STAR was developed while we knew little of the world of artificial intelligence, knowledge engineering and rule-based systems. This paper re-casts much of what was done in a more favorable light than it may deserve. STAR arose as a natural solution to a real problem, not as a conscious attempt to apply a new technique.

2. The STAR Model

2.1 Rules

STAR is a rule-based system. It operates by processing a series of "IF-THEN" statements in a fixed order. Actions implied by some rules enable or disable others. All the rules can be expressed in plain English and are recognized by most investment professionals as expressions of conventional investment wisdom. The complete system is simple enough to execute by hand, yet complex enough to provide superior investment performance over a 23 year period.

STAR has two types of rules: valuation rules and limit factors. The interaction of these different types of rules is one factor in the model's superior performance. Conceptually, **valuation rules** indicate a preference for one class of assets over another, driving the movement of funds among stocks, bonds and cash. **Limit factors** alter the timing of the desired movement of funds because the markets may not always recognize value.

STAR has two valuation rules. The first determines the preference for equities versus fixed income. The second

determines the preference for bonds versus cash. Each rule is a comparison of asset yields, with precise cut-off values.

Each asset class (stocks, bonds and cash) has a set of limit factors. If the valuation rules indicate a call for funds to be moved into a particular asset, then the limit factors for that asset class are considered. They may increase, decrease, delay or cancel the movement of funds; the condition and action of each rule is precisely specified.

The model has a number of other parameters and structural features in addition to the valuation rules and limit factors. The normal amount of funds which can be moved into each asset class is set to maintain reasonable turnover and reduce "whip-saw" (the switching of funds into and immediately out of an asset class at the next decision point). The model cannot "short" an asset, or leverage its investment by borrowing. Accounting consistency rules keep allocations in each asset between 0 and 100 percent, and total investment at 100%. The rules, and, implicitly, the asset classes, are considered in a fixed order--first stocks, then bonds, finally cash--reflecting the risk/return hierarchy of the CAPM. Structuring the agenda in this way biases the allocation of funds in favor of the assets which are treated first.

Consider the allocation of funds to bonds. This decision requires first that the valuation rule for bonds be true:

Bond yield is significantly greater than the T-Bill rate.

The actual allocation amount and timing depends on the bond limit factors:

1. ***If T-Bill rates are up, delay allocation.***
2. ***If the bond prices are significantly up and T-Bill rates are significantly down, move the normal amount of funds and any delayed amounts.***
3. ***If the bond prices are significantly down, delay allocation***
4. ***If T-Bill rates are down significantly, increase the amount of funds allocated to bonds.***

In each rule, "up," "down," "significantly," "normal amount," "delay" and "increase" have precise meanings determined by a process described below. STAR does not make use of "fuzzy logic" in determining which rules are used.

Rules for bonds are considered after the valuation rules and limit factors governing the allocation of funds to stocks. The effect of the stock rules may be to prevent the bond rules from being considered or from having any effect. If the bond valuation rule is false, then cash (Treasury bills) are considered, and the limit factors affecting that decision are tested.

2.2 STAR Development Process

The four steps in developing a model like STAR are: 1) generating rule hypotheses, 2) precise rule specification, 3) parameter estimation and 4) model testing.

Rules in STAR came from two sources. The first, and preferred, source was **expert opinion.** Successful fund managers were asked about their asset allocation decisions and the rules of thumb which guided them. This was particularly useful in developing the limit factors, as managers seem particularly attuned to short-term timing.

The second source was **statistical correlation.** The long-term relationships between variables were explored to develop possible valuation rules. Most candidates were from economic and financial theories which prescribe how to determine relative asset values. Fundamental analysis is traditionally correct in the long-run, and disastrous in the short.

Each rule hypothesis had to be stated precisely with respect to the data series used, how the values should be transformed, the specific lead or lag used to judge changes, the exact values which trigger the rule, and the resulting actions. An investment manager may know when "interest rates have fallen significantly," but a STAR rule has to know whether to use T-Bill, intermediate Treasury or corporate bond rates, whether it requires a 50, 100 or 150 basis point change, and whether to measure it from last month, last quarter or last peak, and how to determine when the last peak occurred. Thus every potential rule generates a whole family of rules depending on how these details are specified.

A different model can be created by altering the set of rules used, their order of evaluation, the trigger values in each rule, the percent of assets to be moved and how the transfers should be adjusted. In building STAR, hundreds of combinations were tested in a cycle of specifying an alternative, calculating its performance, examining the result, and looking for improvement. This process is equivalent to the estimation phase of a traditional statistical model.

With enough rules, it would be possible to build a model which perfectly navigates the past and has no predictive power at all, "over-fitting" the model. Statistical theory provides guidance for traditional econometric estimation: how to decide which parameters are identifiable, how to compute confidence intervals, how much data is needed and so on. It is not clear how this can be applied to estimating non-traditional models such as STAR. The number of parameters can be totaled and compared to the number of data observations, but does a parameter in a rule "use up" as many degrees of freedom as a coefficient in a regression equation? If the "parameter" is a choice of actions, or permits or disallows the action of other rules,

then the matter is even more uncertain.

Neither is it clear that least-squares, maximum likelihood or other error-minimization criteria can be adapted to judge the results. The form of the "ideal" trajectory is unknown and perhaps unknowable. The primary estimation objective for STAR was to maximize compound return. However, a model which managed to be 100% invested in the asset with the highest return in every period, even if it had a reasonable number of rules and parameters, would be unacceptable for two reasons. First, no one would believe it had any forecasting power. Second, the turnover of this set of allocations is too high for most institutional investors. Transaction costs alone do not properly measure discomfort with high-turnover investment strategies. As perfection is assumed to be unattainable, high turnover looks like desperation whenever a manager is running behind performance benchmarks. There is no clear standard against which to measure error.

STAR parameters were established to maximize compound total return with reasonable turnover. The model has only two valuation rules and eleven limit factors. While weekly and bi-weekly decision making were tested, monthly asset reallocation provided the best results for the data series used and the investment goals. Most of the work was done in 1985 and 1986. The model was estimated using data from 1970-1979: different rule sets, data series, transformations and lags were tested and satisfactory parameters determined. The final version of the model was validated using the data from 1980 through 1984 which had been held out during the estimation process. Regular testing since that time has yet to result in a decision to change the model.

The STAR model looks at relatively few data series to calculate asset reallocations:

- S&P 500 Earnings Per Share, as reported for the past twelve months
- S&P 500 Price
- Dow Jones 20 Bond Index Yield
- Dow Jones 20 Bond Index Price
- 90 Day U.S. Treasury Bill Auction Rate

2.3 Behavioral Implications

The interaction of multiple rules makes STAR **non-linear** in its behavior. A steady increase in interest rates or prices does not result in a smooth shift in funds from one asset class to another. Reallocations may require that more than one rule be used. Hence two situations which both trigger the same valuation rule may result in different actions because different limit factors come into play. Alternately, some limit factors may not be considered at all depending on which valuation rules hold true.

Many asset allocation models tend to move either too soon or too late, because basic **valuation changes do not coincide with market movements.** STAR valuation rules signal an intention to reallocate, but the limit factors permit the actual flow of funds to depend on more complex circumstances. By keeping the number of rules and parameters small STAR manages to find enough behavioral regularity in the markets to provide good out of sample and forecast results.

3. STAR Performance

3.1 STAR Performance Objectives

The primary goal of STAR is to produce **higher returns** than any constant mixture of the asset classes that it uses while achieving a **lower standard deviation** of return than the most risky of those asset classes. This result is expected **over a market cycle,** not necessarily in any given month, quarter or annual period.

However, there are a number of secondary performance goals which are important for the acceptance of the model's reallocation schedule as an investment product. Over time, the allocation levels must show a reasonable distribution among the three assets. The turnover generated by the reallocations must be reasonable. Because three assets are involved rather than two, it would be reassuring to see that, on average, each class helps improve overall returns.

STAR allocations and returns are calculated starting from an arbitrary 50% equities, 50% cash from January 1970 through December 1992. The results for 1970-1984 are *pro forma*, while those for 1985-1992 were calculated as events occurred.

Definitions for the performance results are:

STAR Return to allocations generated by STAR, using the S&P 500, Bonds, and Cash asset classes defined below, without investment management fees.

S&P 500 Return to the S&P 500 with dividends reinvested.

60/40 Static investment allocation of 60% equities and 40% bonds, using the S&P 500 and Bonds asset classes here defined. This is the "standard" long-term asset allocation in investment lore. [1]

Bonds Return to the Dow Jones 20 Bond Index.

Cash Return to 90-Day US Treasury Bills.

Performance is calculated monthly as total return, then compounded and annualized for longer periods. Detailed performance data is available on request from the authors.

3.2 Investment Returns and Risk

STAR attains its performance objective using stocks, bonds and cash primarily by being in bonds and cash when the stock market is weak. In particular, STAR does well by missing periods of substantial market decline.

Figure 1 shows that STAR's allocation to stocks tends to fall, often to nothing, during those periods when returns to equities were negative. However, not every stock market decline is avoided; rather, allocations are smoothly rebalanced to capture medium to long term trends.

An examination of annual returns would show:

- STAR returns exceeded the return to the S&P 500 in 9 out of 23 years, with an average excess return of 14% in those years. STAR returns fell short of the S&P 500 in 6 out of 23 years, with an average shortfall of 5%.
- STAR returns exceeded the static 60/40 asset mix in 19 of 23 years with an average excess return of 7%. STAR returns fell short in the other four years, with an average shortfall of 4.0%.

Figure 1. Growth Of $100

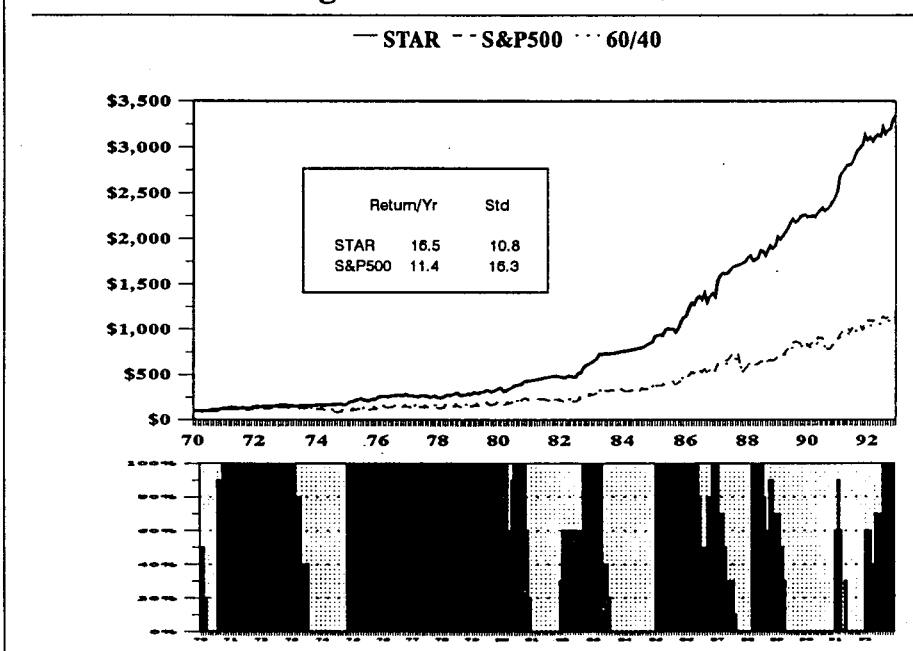


Table 1 -- Return and Risk by Asset Class and Period

	STAR	S&P 500	60/40	Bonds	Cash
1970-1992 Average Return	16.0	12.2	11.2	9.7	7.5
Standard Deviation	10.9	16.0	10.8	6.9	0.8
Sharpe Measure	0.78	0.29	0.34	0.33	
1970-1984 Average Return	15.1	9.6	9.3	8.9	7.8
Standard Deviation	11.6	15.7	10.9	7.6	0.9
Sharpe Measure	0.63	0.11	0.14	0.14	
1985-1992 Average Return	17.6	17.0	14.7	11.4	6.9
Standard Deviation	9.4	16.5	10.8	5.2	0.5
Sharpe Measure	1.14	0.61	0.73	0.86	

Sharpe Measure = (Average Portfolio Return - Average Risk Free Return) / Standard Deviation of the Portfolio
Cash (3 Month Treasury Bills) is the risk free asset

Risk/return statistics in Table 1 show STAR comes close to meeting its second objective. While risk (standard deviation) is not quite reduced to the levels of bonds and cash, it is approximately that of the "60/40" stock/bond asset mix. With returns significantly above those of stocks, STAR is the clear leader over all periods on a risk adjusted basis. In some periods, like the late 1980's, risk reduction may have been more valuable than the enhanced returns relative to equities.

3.3 Allocation Levels and Turnover

Over long periods, STAR approximates conventional asset allocation wisdom: the average levels are close to the "60/40" equity/fixed income split often cited in investment literature. This result is neither intentional, nor is it an argument for or against the STAR method.

Table 2
Average Asset Allocation in Percent

	Cash	Bonds	Stocks
All Years	20	18	62
1970-1984	24	9	67
1985-1992	14	35	51

Monthly data show that STAR allocations do not remain constant over time, reaching, or nearly reaching 100% allocation in each asset class on different occasions. STAR recently (August 1992) attained 100% investment in equities for the seventh time in its 23 year history. This suggests a three or four year market cycle. The cycles have been shorter during the more exciting markets of the late 1980's than in earlier periods, though this difference may be the result of forecast versus in-sample performance.

Total trading activity is measured by the percent of assets sold. As STAR is totally invested at all times, any sale results in an immediate and equivalent purchase. Annual turnover is the sum of all sales in a twelve-month period. Because stocks are more costly to trade, and are usually the highest returning asset, stock turnover--the sum of buys and sells--is broken out separately.

Table 3
Average Annual Turnover

	All Assets	Stocks Only
All Years	117%	77%
1970-1984	86%	55%
1985-1992	175%	118%

The average turnover levels in Table 3 are satisfactory for an asset allocation strategy implemented with index funds. There are periods of months (and in the 1970's especially, even years) during which no trading occurs. As noted above, STAR went 100% invested in stocks on August 1, 1992, and remains that way as of March 31, 1993.

3.4 Allocation Decisions and Returns

The relationship between allocation and return over different horizons shows the responsiveness of an asset allocation model. High correlation at short time periods indicates the model reacts quickly. Correlation at longer periods indicates a model tuned to longer trends.

Table 4
Excess Return to Stocks
by Allocation and Horizon

Average Allocation to Stocks	1 Month Periods	3 Month Periods	12 Month Periods
0-10%	3.29	-4.59	-11.64
10-20%	-2.70	4.83	3.18
20-30%	2.33	3.69	-5.77
30-40%	0.81	1.80	-4.72
40-50%	-0.52	2.67	0.72
50-60%	0.91	-0.46	9.82
60-70%	1.53	4.23	15.17
70-80%	-1.82	5.11	16.13
80-90%	3.08	4.12	22.83
90-100%	1.06	3.34	8.65

Each entry shows the average return to stocks minus the average return to cash during all 1, 3 and 12 month periods where the average STAR allocation to stocks fell in the range specified in the first column. The 3 and 12 month measures include all overlapping periods.

Over one and three month horizons, STAR allocations to the stock market are not particularly aligned with returns. As Table 4 shows, only at the 12 month horizon does STAR clearly tend to be more heavily invested in stocks when returns to stocks are high.

STAR favors the CAPM risk/return hierarchy by considering the equity allocation decision first. STAR adds value to a portfolio largely by investing in stocks except when the market is weak. Another way of looking at performance is to treat the model as making two separate asset allocation decisions, one between stocks and fixed income, and another between long-term and short-term

fixed income instruments. Ideally, both decision should add value.

The equity allocation decision can be examined in the strongest possible terms by ignoring the returns to fixed income. In Figure 2, the ***S&P 500 unit value*** line is the result of a buy and hold strategy beginning on January 1, 1970 and continuing to the present. The ***S&P 500 as allocated*** line is the result of investing the STAR-determined share of assets in the S&P 500 and holding the rest uninvested each month over the same period. Note that the ***S&P 500 as allocated*** line is flat when STAR has withdrawn all funds from equities, indicating no return during these periods.

Stocks dominate STAR's investment allocations, but the use of bonds also enhances STAR returns. By removing equities from the portfolio, the remaining STAR investment in bonds and cash improve upon being invested in cash alone.

In Figure 3, the ***Bonds and Cash***, and ***Cash Only*** portfolios yield no return when STAR was invested entirely in equities. When fixed income investments were made, the ***Cash Only*** portfolio was 100% invested in T-Bills. During those same periods, the ***Bonds and Cash*** portfolio was invested in the Dow Jones 20 Bond Index and T-Bills in the same proportions as the STAR allocation to those assets, treated as 100% of the portfolio. For example, if STAR recommended 40% in Bonds and 40% in Cash, the ***Cash Only*** portfolio was 100% invested in cash, and the ***Bonds and Cash*** portfolio was invested 50% in Bonds and 50% in cash.

Figure 2. STAR Allocated S&P vs S&P

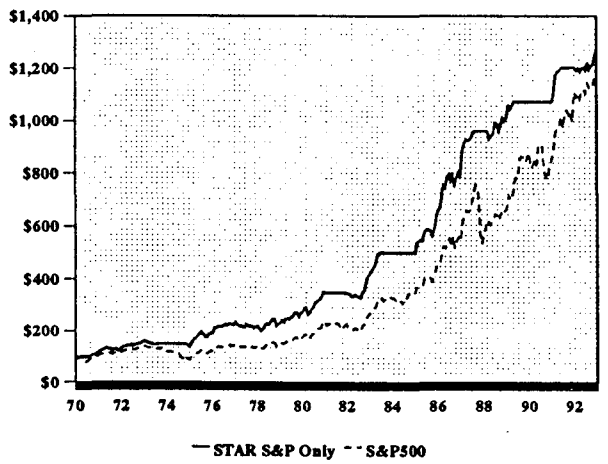
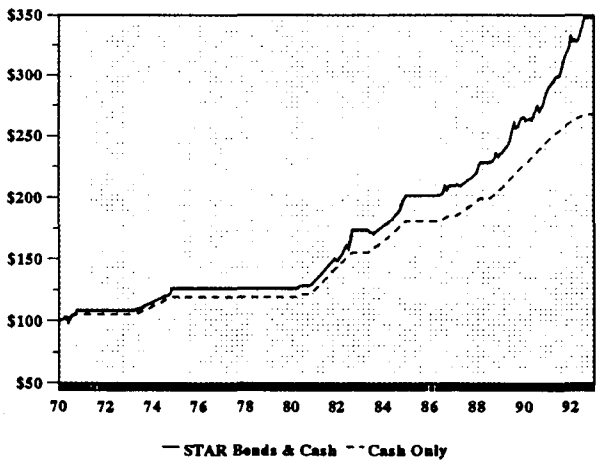


Figure 3. Bonds & Cash vs Cash



4. A General Asset Allocation Model

4.1 STAR as a Paradigm

STAR was developed as a practical response to a real investment management problem, not as an attempt to apply particular techniques to finance. Details of rule structure and precedence were often mixed with or resulted from coding decisions made for efficiency or convenience. General in some ways, the structure was very limited in others. This made it difficult to extend the model.

There are many obvious avenues to explore. First, STAR uses only a few data series to make decisions. Monetary aggregates, real economic activity and exchange rates are all believed to have an effect on markets. Second, technical analysts have long used a very rich set of lags, moving averages and other data transformations, especially of price data. Third, within the U.S. there are asset sub-classes which have at times outperformed the broader averages. Small capitalization stocks [8] and intermediate term bonds are notable market "anomalies." Allocation signals for these assets must be derived independently. Fourth, there are any number of situations still to be examined: models similar to STAR for other national markets, for allocation across national markets, and for derivative instruments. Finally, the parameters estimated for STAR are good, but nothing in their development shows that they are optimal.

Creating a framework in which STAR-like models can be specified, estimated and tested has made us aware of the similarities between elements in STAR and various knowledge engineering techniques. Changes have been made which draw on the methods used in that field. The rules have all been separated from the procedural code which evaluates their effects. They have been encoded in a way which allows them to be treated as data and dynamically modified. A formal structure has been devised which preserves the distinction between valuation rules and limit factors and permits an unlimited number of asset classes. The set of actions which result from the rules has been standardized. Much of the description of STAR in this paper was clarified by this work.

4.2 Estimation Using Genetic Algorithms

Testing different rule sets and determining precise parameter values was the most time-consuming part of STAR model development. The manual process used was the time-honored practice of generating a specification, testing it, examining the results for improvement, and repeating the cycle until satisfied. Rules which performed well served as the basis for further modifications. When the potential for improvement seemed to be exhausted, the model achieved its current form. Various alternatives are still tested, but there has been no compelling reason to change the models. On the other hand, there is no evidence that the model is optimal.

A **genetic algorithm** [5, 6] is an advanced optimization or search procedure that is based on the concept of natural selection. Solutions to a problem are treated as organisms whose fitness is judged by some measure of performance on the problem to be solved. Better solutions are more likely to be chosen for further modification, generating new solutions to be tested. Less fit solutions are discarded. Genetic search optimization is an automated version of the "generate and test" process used to create STAR, with random selection replacing human insight in creating new models. Academic study and industry experience have shown this method to be particularly useful for problems where the solution space is unstructured or poorly understood.

Genetic methods will help in several ways:

1. The existing STAR model can be examined to see what improvements are possible. Are the parameter values optimal?
2. A range of economic and financial indicators can be efficiently searched for more accurate valuation rules and limit factors.
3. STAR models can be developed for more asset classes and different markets.

Expanding the horizons of STAR brings its own problems. The strategies require asset classes which are tradeable and for which accurate data for a "long enough" timespan are available. The decision rules must be based and tested on *ex ante* data. Other than market price and volume, most data series are regularly revised and re-based. Even the announcement dates for economic statistics do not remain constant from year to year. Using what is **now known** in place of what was **then known** can lead to very different rules, and very different in sample and forecast accuracy. Preliminary work might be done on reasonably lagged data from published databases, but final versions of the models will depend on developing clean, "as-known" time-series.

Compound return may not be the most appropriate objective function for all situations. Its use is consistent with the behavior of a conservative, long-term investor, but other performance measures might be used to develop models for other investment strategies. An options-based strategy might only be interested in the number of correct moves--switches from one asset to another. A futures-based strategy would have to consider the effect of draw-down on the return. International strategies might depend on the use of currency hedging.

5. Conclusions

STAR is a successful asset allocation model developed to meet an investment need. Its rules show superior performance both in sample and forecast. While not designed with knowledge engineering methods in mind, reevaluating the STAR in light of this work has suggested ways to improve the model and expand its scope. In particular, using genetic algorithms to estimate rule-based models opens the possibility of rapidly producing customized decision rules for many investment situations.

We do not have the same experience or theoretically derived guidance for building rule-based financial decision models as we have for traditional statistical or econometric analysis. However, questions of significance, confidence, degrees of freedom and other information-based requirements cannot be ignored if forecast as opposed to in sample accuracy is to be obtained. For now we tread carefully, rely on our own judgement and look for better ways to judge a model.

6. References

[1] Ambachtsheer, Keith P., "Pension Fund Asset Allocation: In Defense of a 60/40 Equity/Debt Asset Mix," *Financial Analysts Journal*, September-October 1987.

[2] Bodie, Zvi, Alex Kane and Alan J. Marcus, *Investments*, Richard D. Irwin, Inc., Homewood, 1989.

[3] Brinson, Gary P., L. Randolph Hood and Gilbert L. Beebower, "Determinants of Portfolio Performance," *Financial Analysts Journal*, July-August 1986.

[4] Brinson, Gary P., Brian D. Singer and Gilbert L. Beebower, "Determinants of Portfolio Performance II: An Update," *Financial Analysts Journal*, May-June 1991.

[5] Davis, Lawrence, ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[6] Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[7] Jeffrey, R.H., "The Folly of Stock Market Timing," *Harvard Business Review*, July-August 1984.

[8] Kester, George W., "Market Timing with Small versus Large-Firm Stocks: Potential Gains and Required Predictive Ability," *Financial Analysts Journal*, September-October 1990

[9] Samuelson, Paul A., "Asset Allocation Could Be Dangerous to Your Health," *Journal of Portfolio Management*, Spring 1990, Volume 16, Number 3.

[10] Sharpe, W.F., "Likely Gains from Market Timing," *Financial Analysts Journal*, March-April 1975.

[12] Shilling, A. Gary, "Market Timing: Better than a Buy-and-Hold Strategy," *Financial Analysts Journal*, March-April 1992.

A NEURAL NETWORK SYSTEM FOR RELIABLE TRADING SIGNALS

by Marlowe D. Cassetti
ROCKET SCIENCE INVESTING (RSI)
P.O. Box 50276 Colorado Springs, CO 80949-0276
(719) 535-9125

INTRODUCTION A rocket science[1] approach (i.e., cutting edge technology) has produced a high performance trading system. Central to this system is a unique neural network that differs from the conventional, fully connected network. This configuration causes the network to recognize input patterns and then to associate those patterns with the desired output. The network also features data preprocessing, rapid training, and good trading performance.

A LOGICAL NEURAL NETWORK CONFIGURATION The network configuration identifies unique patterns in price momentum, price acceleration, volume movement, and price volatility. Refer to figure 1 for a simplified schematic of the neural network. A limited connection configuration accomplishes this. Pattern detector nodes contained in the first hidden layer connect to feature detector nodes. This layer feeds the output node that produces the single network output. This configuration is unique and differs from the conventional fully connected neural network. Because the network connects neurons logically, it forces the training to evaluate the inputs as distinct patterns and then to associate the features of the different input patterns. Inputs are not fully connected. Input pattern detectors are fully connected. The drawback of this configuration is that it introduces an additional hidden layer that, in turn, adds more weights and slows training time. As will be discussed later, training times have been reduced by other strategies. On the positive side, the configuration produces high correlation of forecast to target and hence, more reliable trading signals.

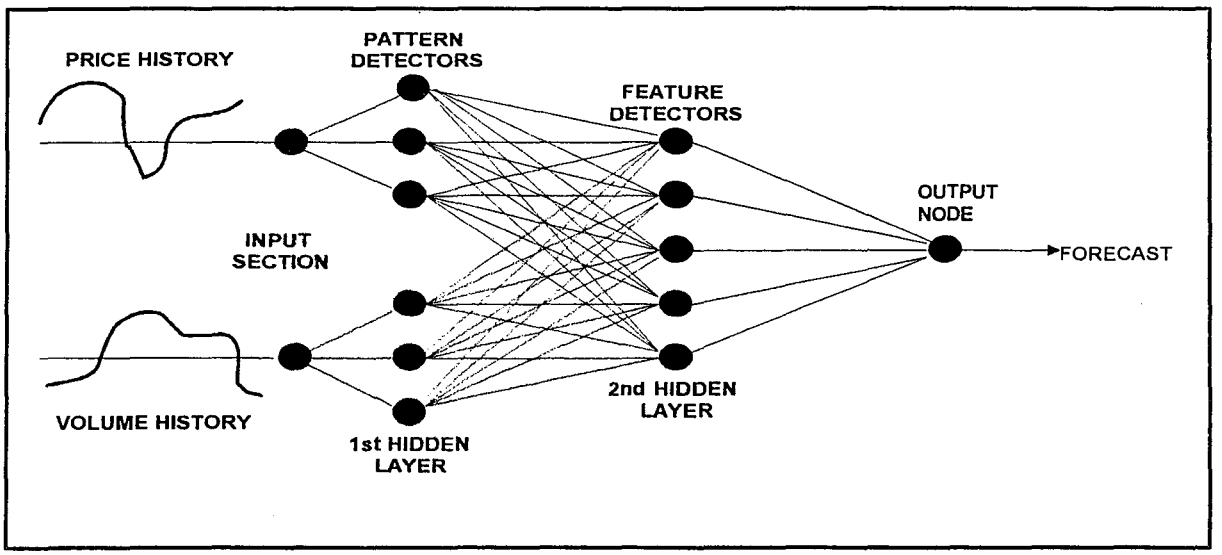


Figure 1.- Simplified Network Schematic

INPUT PROCESSING Preprocessing the inputs (price momentum, volume, and volatility) filter out noise and scale the values. Removing noise and smoothing the input signals reduce noise in the forecast predictions. This reduces false buy and sell signals, but introduces a lag or delay in the action signals. Action (i.e., buy and sell) signals that have too long a delay will also produce false buy and sell signals. The task of tuning the inputs is to strike a balance in these divergent requirements. Inputs that are presented to the network represent past historical sequences of price and volume movement. Analysis has shown that 300 to 400 weeks of past history are necessary to properly characterize the relationship of input patterns to output.

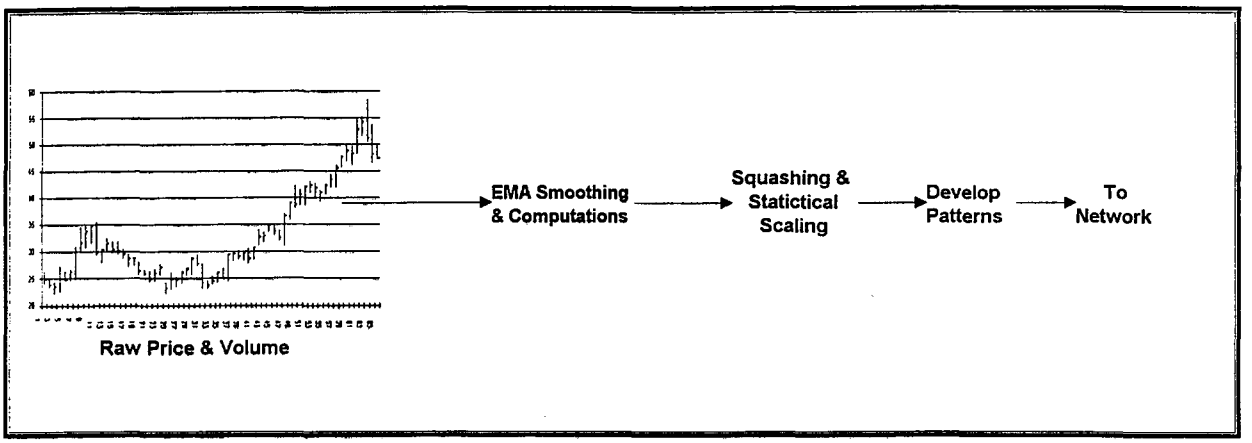


Figure 2.- Preprocessor Functional Flow

Exponential moving averages (EMA) perform the smoothing. Polynomial regression also yields excellent smoothing. The EMA, however, is more easily adapted to parameter analysis and tuning. After smoothing, the preprocessor squashes and scales the inputs and target vectors to lie in a range of +0.5 to -0.5. Since buy and sell signals occur near zero crossings, the squashing and scaling enhance the network's sensitivity at these action points. Figure 2 is a simplified representation of the preprocessor functional flow. Figure 3 is a typical input data stream that the preprocessor feeds to train the network.

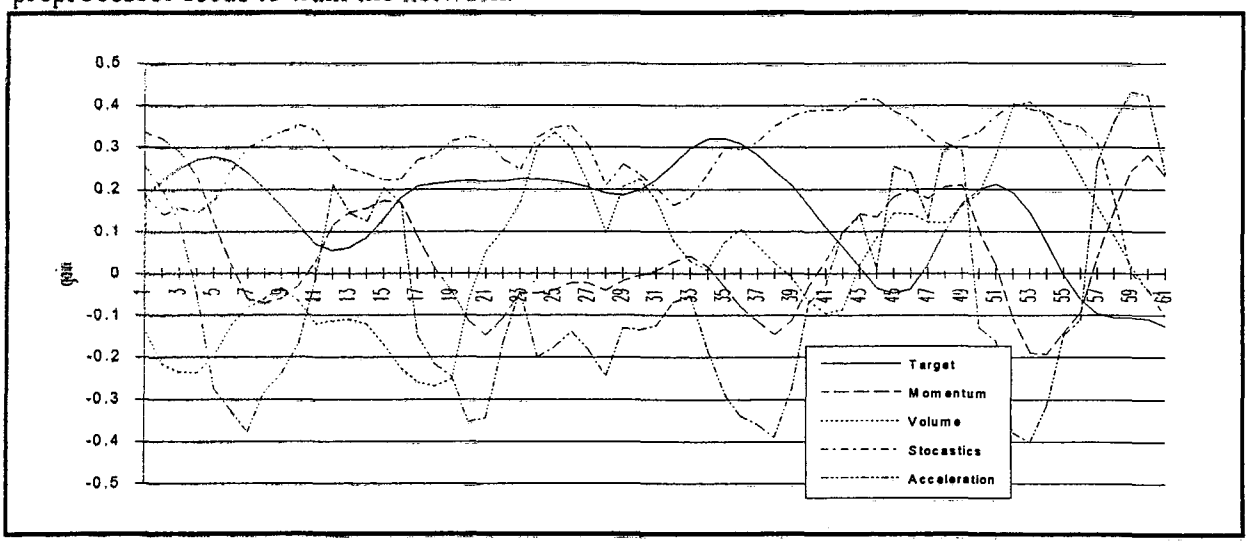


Figure 3.- Data Input Stream to Network

OUTPUT FUNCTION The target parameter utilized is a smoothed measure of the five week gain. This parameter gives good buy and sell signals with little probability of whipsaws. It does not get the investor in at the exact bottom or out at the exact top. It does a great job of riding sustained rises and avoids long declines. If we could accurately predict this parameter and trade near zero crossings, it would produce outstanding profits. Figure 4 shows the target function for a selected stock. Notice that it crosses into positive territory when it is a good time to buy and drops into the negative region when it is a good time to sell.

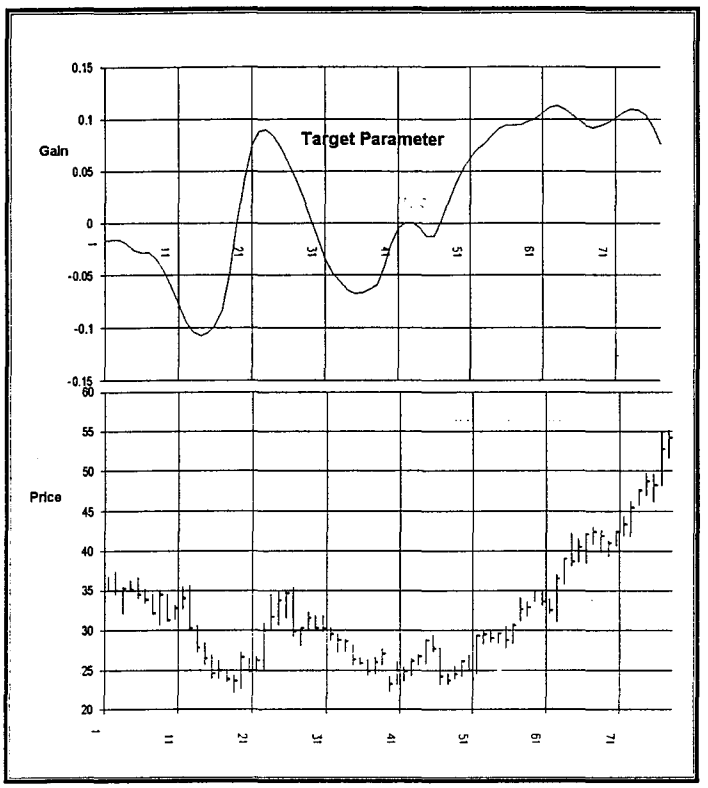


Figure 4. Target Function.

TRAINING STRATEGY This neural network utilizes Hebbian learning[2] with back propagation[4] of the errors to adjust the weights. Even the best designed neural network can give poor and unreliable results if not properly trained. Two factors are very important; 1) the ratio of training cases to network weights and 2) number of training iterations.

A neural network needs to be trained by presenting it a number of training cases. Training involves adjusting the weights to produce the correct forecast. Using too few training cases (number of cases < number of weights) will result in a process that will "curve fit" the weights to match the inputs. A good rule of thumb is to have four times as many test cases as weights. Thus, a network with 100 weights needs to be fed with at least 400 training cases. This requirement can result in large data matrices and very long training times. The size of the input data matrix can easily exceed the 64K byte restriction of PC DOS. There are ways to solve that problem.

Many training iterations, besides the frustration of long training time, can produce noisy and erratic results. This is because the weights fight each other as the network approaches the law of diminishing returns. This phenomenon is called network paralysis[2]. An inspection of the weights tells the tale of network paralysis. Large weights indicate a tug of war between competing weights.

One very effective way to reduce training iterations and eliminate paralysis is to start the network close to the right solution. Most commercial neural networks start training with an arbitrary set of starting weights. If the starting set is too far from the desired solution set, the net takes a lot of

training iterations before a solution is reached. In the extreme case the network never learns, and the operator has to intervene and restart the training session.

With a close starting point, the training easily slides into the best solution (high training correlation). This is accomplished by using a method called simulated annealing[2,3]. The method searches a vector space that is constantly shrinking. After the simulated annealing has done its job, the neural network starts training with a set of weights that are close to the "right" set. Training can then be accomplished with a small number of iterations. Figure 5 demonstrates the rapid convergence to a solution. Simulated annealing gets the back propagation training off to a great start.

Rapid training has many benefits. The total process takes less than 3 minutes on a PC with a 486DX33. The ability of the system to train 100 stocks with 500 weeks of data is not an unreasonable task. The ability to access results over a large span of consideration gives the investigator the ability to try a variety of network strategies and tune the system. This is a powerful advantage.

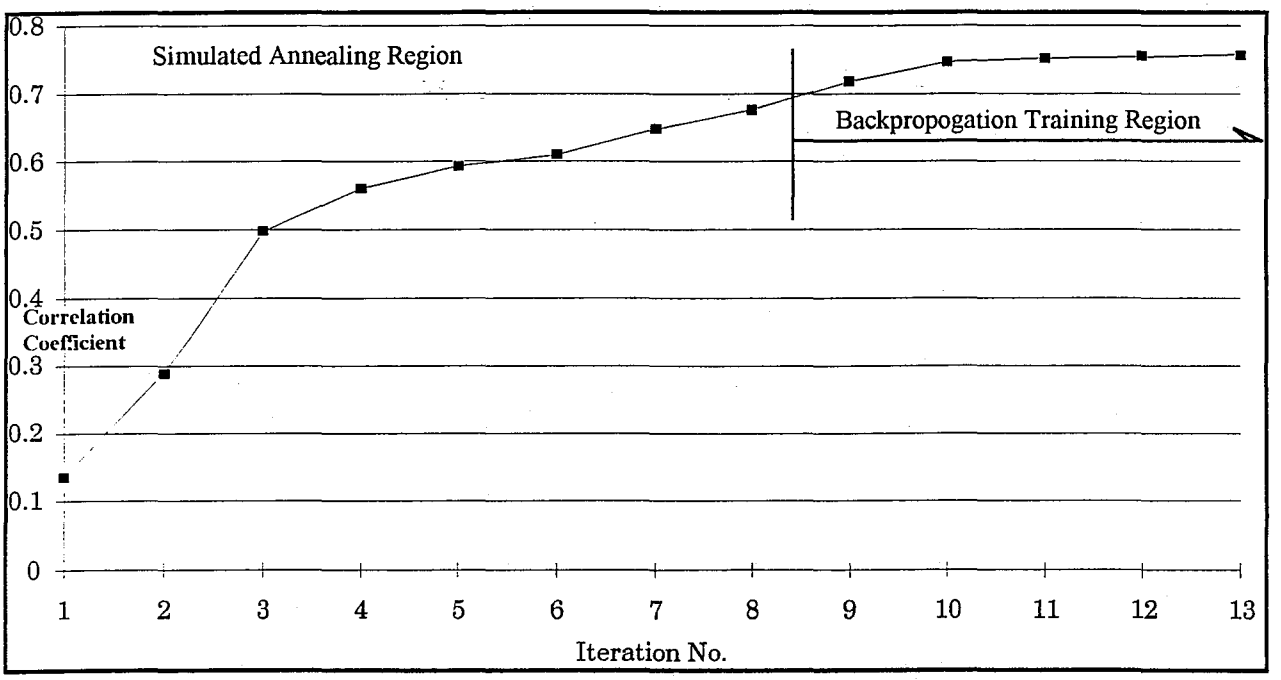


Figure 5.- Rapid Training Helped by Simulated Annealing.

TESTING RESULTS Testing is training over a span of data and then forecasting the five week gain. This process is advanced a week and repeated, to yield another forecast. A table of forecast gain, and the smoothed five week gain (the target), is generated. A typical graph of these parameters is shown in figure 6. Forecasts sometime lag the target. This effect tends to make the output of the network look like a trend following system. False signals are sometimes generated when the network prematurely generates a reversal signal. When the reversal does not happen, the network generates a whipsaw of action signals.

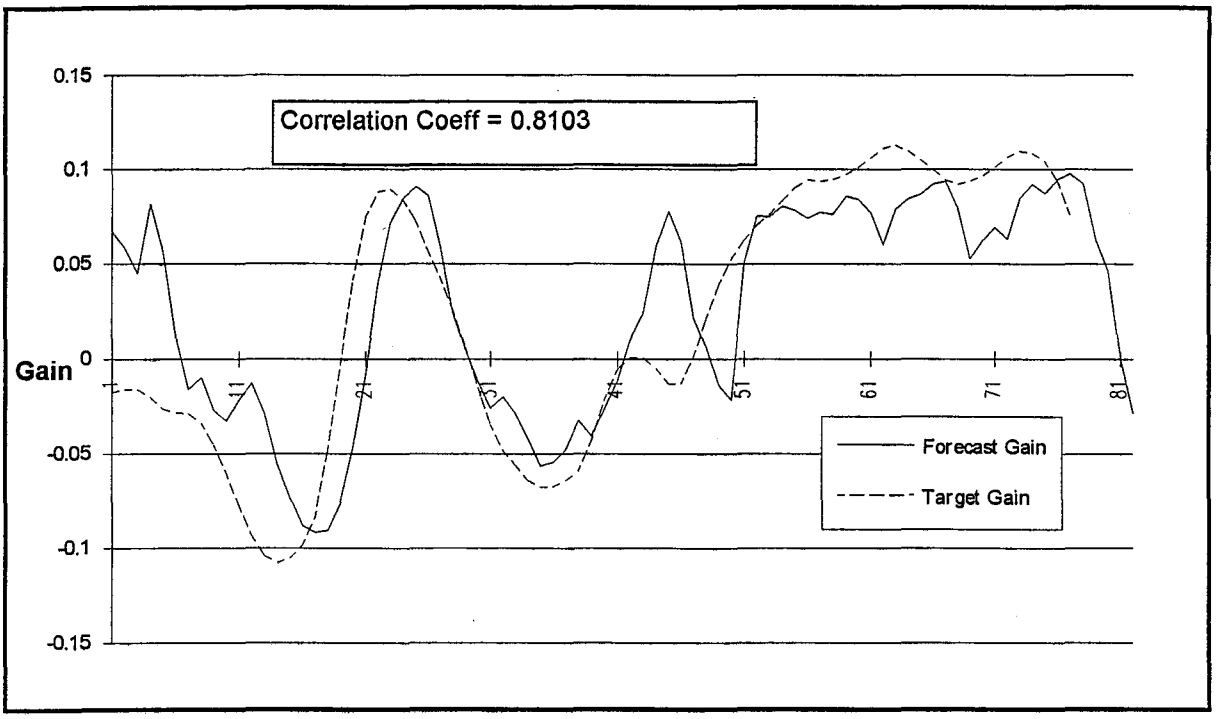


Figure 6.- Out of Sample Testing.

Reviewing the results for many stocks over many periods of time yields the following observations. High (>0.8) correlation coefficients can be achieved for some stocks. Stocks with high volatility correlate better than low volatility stocks. Stock indices (i.e., NYSE index) have low correlation coefficients probably due to low volatility. In spite of that effect, the network can predict the trend of the NYSE. Mutual funds that do not have volume have been evaluated. Limited testing has shown poor correlation and correspondingly poor forecasting. This antidotal evidence suggests that volume patterns are valuable in forecasting future price movement.

The author performed a trading test on 45 stocks. The training period utilized weekly data starting at 01/04/85. As described earlier, the network trains, then forecasts, and marches forward over the period from 08/09/91 to 02/26/93. The network generated buy and sell signals over the forecast period. From this universe, 127 buy and sell pairs were generated. A 1% slippage and commission factor were applied to all transactions. No short sales were evaluated. The histogram of this test is plotted in figure 7. Seventy percent of the action signals produced a gain and 60% produced a gain in excess of 10%. On the loss side, 20% of the cases had loses greater than 2% and few of the losses (7%) exceeded 10%. A stop loss strategy was not employed. This would have changed the shape of the histogram. Limiting this test to only high volatility stocks would show even higher performance.

Conclusion The high performance of this neural network is a result of its unique logical configuration, input preprocessing, and rapid training. The integration of the network into an automatic training system produces good buy and sell signals. It can not predict the future 100% of the time, however, it does a respectable job of generating reliable buy and sell signals.

Further research is under way to adapt this system to trading indices, commodity futures, and stock and future options.

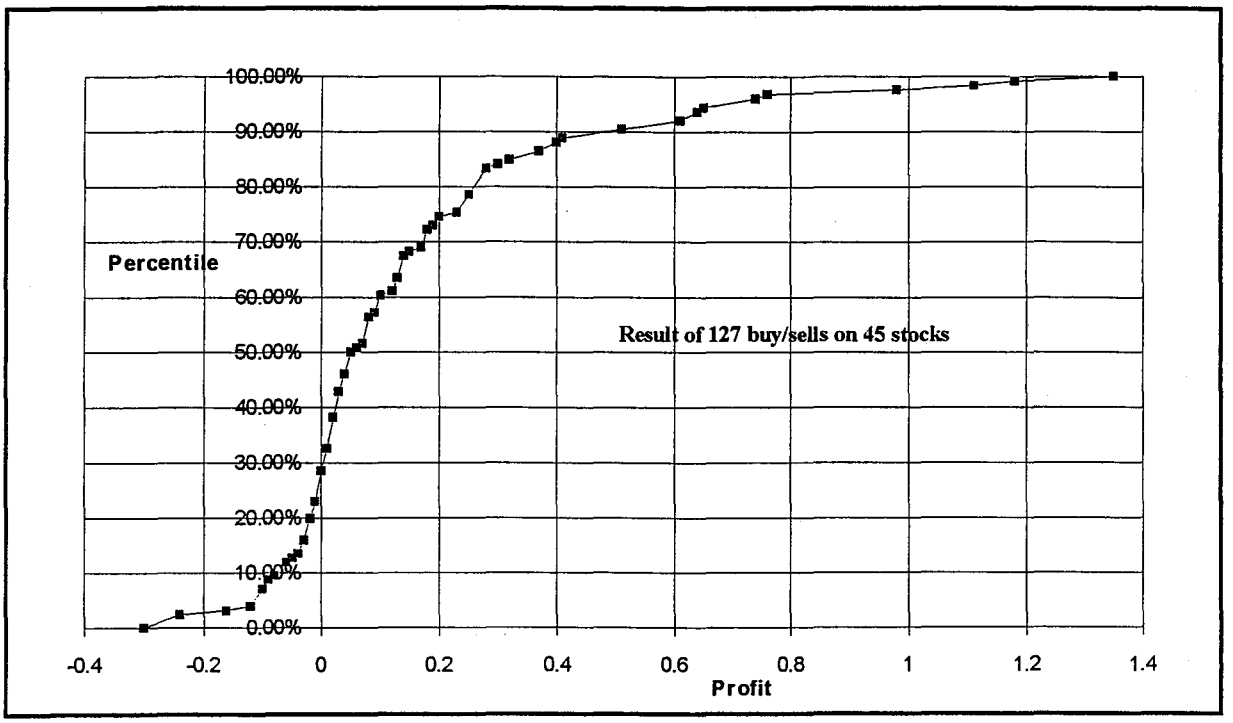


Figure 7.- Trading Results.

Marlowe Cassetti is editor of ROCKET SCIENCE INVESTING (RSI), *an investment newsletter that specializes in cutting edge technology for stock and futures investing. He is an actual rocket scientist, with a 36 year career in spaceflight research and operations. Marlowe is an avid follower of technical analysis and performs research in neural networks, genetic algorithms, chaos theory, and fuzzy logic. Contact* RSI *at P.O. Box 50276 Colorado Springs, CO 80949-0276 (719) 535-9125.*

REFERENCES:

1. Light, Larry et al, Special Report, "The New Rocket Science, Welcome to the Future of Finance", Business Week, November 2, 1992.

2. Wasserman, Philip D., "Neural Computing, Theory and Practice," Van Nostrand Reinhold 1989.

3. Khanna, Tarun, "Foundations of Neural Networks", Addison-Wesley Publishing Co., 1990

4. Caudill, Maureen, "Neural Network Primer", Reprinted from issues of AI EXPERT, Miller Freeman Publications, 1990

Invited Speaker: The Laboratory Market at the Taiwan Stock Exchange

Nai-Kuan Huang, Taiwan Stock Exchange

LABORATORY MARKET AT TAIWAN STOCK EXCHANGE

(Invited Speech)
Nai-Kuan Huang, PhD
Taiwan Stock Exchange
One Nan-Hai Road, Taipei, Taiwan, 100
Republic of China

ABSTRACT

Ever since its establishment in 1962, the Taiwan Stock Exchange is essentially a marketplace for equity stocks only. All other investment instruments listed on this exchange (excluding closed-end mutual fund) account for less than .5% of the trading. As new (derivative) products are to be introduced to the investing public, the exchange, which used to regulate a relatively simple securities industry, has a number of systemic homework to do before announcing the rules of the new game. A laboratory market is considered to be a useful tool for, among others' things, measuring the effect of different trading mechanisms on the conventional equity stock market, the new derivatives market, other financial markets, and the interaction among these markets. Laboratory findings further our understanding of the existing economic theories as well as their limitations.

1. Introduction

Historically, policy-making in China has primarily been a matter of ideology, rather than a result of in-depth examination of hard figures and facts. Consequently, the more eloquent party usually wins. By tradition, neither the ruling class nor the majority of the Chinese populace is conceptually and practically ready for a mathematically manageable modern society.

For a society to be mathematically manageable, three ingredients are indispensable, i.e., a system of up-to-date records of the business activities, a set of well-formulated theories that explain the records, and a proving ground that checks the validity of the theories. Laboratory market is the least costly form of such a proving ground.

Taiwan has transformed from an agriculture-based economy to a major productive industrial power in Asia. In the past twenty-some years, the per capita GNP has increased nearly four-fold; in the early seventies, the level was around US $2500, and now it is close to $10,000. The accumulated foreign exchange reserve exceeded $80 billion in 1990, comparable to that of Germany and Japan. Until recently, the savings rate had constantly been at 33% of the GNP. With a population of 22 millions, Taiwan is the twentieth largest economy of the world in 1992. For the first time in the Chinese history, a sizable political entity is almost mathematically manageable.

The Taiwan Stock Exchange in the past thirty years has essentially been an equity stock market. Government and corporate bonds (the only alternative investment instruments listed on this exchange till four years ago, when closed-end mutual funds were listed, and heavily traded) account for less than .5% of the trading. With such limited experience as a regulator and facilitator of the securities market, to offer derivative products for hedging and portfolio insurance is indeed an unprecedented challenge. Laboratory market helps fill in the gap, and examine the proposed new market structure and related rules.

We implement the laboratory market on a computer network, since we view the market, logically, as a price determination and dissemination system. [3] Laboratory market has been employed in the studies of organization behavior of auction markets [1] and elections. [2] Some recent work has incorporated automated trading [4] and artificial intelligence [5] into the experiments.

2. The Scope of the Project

To run the laboratory market in advance of releasing new products, our objectives are, in the order of their importance:
1) use the laboratory market as a regulatory proving ground for systemic, functional and technical tests;
2) extend the laboratory market as an educational and promotional environment, a training ground for the industry at large;
3) offer the laboratory market as a public service facility to conduct surveys and forecasts, by connecting it to the existing communication networks of the securities industry.

The major objective of this laboratory market is for internal experiments. The systemic tests include the effect of various market structures, such as an order driven system vs. a quote driven system, or a market with vs. without intermediaries. The functional tests include the market responses to all kinds of administrative options such as limits on margin trade, price movements and order volume in each business day. The technical tests include the system capacity, security and inter-operability with other systems.

To measure the effect of alternative trading mechanisms repeatedly under different economic conditions, a laboratory market equipped with expert trading systems has the advantage of speeding up the data collection process. A large number of controlled experiments can be conducted in a condensed time interval. The dynamics of the market over a week's time can be simulated in the laboratory market in less than an hour.

To extend this laboratory market for educational and promotional purposes, participants of the market gain equal access to the proposed new system. Before a fashionable category of products is officially listed and traded, potential investors can get themselves familiar with the new rules of the exchange and the characteristics of the novel instrument through this laboratory market.

The public service potential of the laboratory market can be considered as an added-value of the exchange, and a channel to communicate other added-values of the exchange to the retail investors, the institutional investors, the issuers and potential listed companies, and the brokerage community.

3. Future Plans and Conclusions

This project is far from completed. We are continually advocating the theory, enriching and enhancing the methodology, justifying the need of more resources, and equipping the laboratory with suitable artificial intelligence and expert systems. We are also considering joint research with the academics and other governmental agencies or semi-governmental institutions.

The stock exchange operates on public confidence. The ultimate responsibility of an exchange is to make sure that just and equitable principles of trade prevail, so as to maintain a fair and orderly market. It is thus unethical to let the market participants suffer from published rules that were not adequately contemplated, and tested. We hope a well-developed laboratory market will serve exactly this end.

4. References

1. Plott, C. R. and Smith, V.L., 'An experimental examination of two exchange institutions', *Review of Economic Studies*, 45 (1), February 1978, pp 133-153

2. Smith, V.L., and Williams, A.W., , 'Experimental market economics', *Scientific American*, December 1992, pp 72-77

3. Stern, R.L., 'The market is a price, not a place', *Forbes Magazine*, February 1990

4. Sunder, S., and Gode, D.K., 'Human and artificially intelligent traders in a double auction market: experimental evidence', CMU-GSIA Working Paper, October 1989

5. Sunder, S., and Gode, D.K., 'Allocative efficiency of markets with zero intelligence traders: market as a partial substitute for individual rationality', *The Journal of Political Economy*, 101 (1) February 1993, pp 119-137

Paper Session: Stock Market Prediction

Chair: Warren Kaiser, American Stock Exchange

A Neural Network Model for Stock Market Prediction

Davide Chinetti^*, Francesco Gardin°*, Cecilia Rossignoli^

° Department of Computer Science,
Università Statale di Milano
Via Comelico, 39/41
Milano, Italy

^ Department of Economics,
Catholic University,
Largo Gemelli, 1
Milano, Italy

* Artificial Intelligence Software SpA
Via Rombon, 11
Milano, Italy

Abstract

This paper describes our initial results in using Neural Networks for Stock Market Prediction. The aim of this project is to build a buying and selling timing prediction system for stocks listed at the Milan Stock Exchange, rather than for the specific behaviour of the stock itself. Our system is based on a combination of a Backpropagation Network which generates forecasts on the future market behaviour and an appropriate trading strategy. The system achieved good results in predicting the correct future market trend, and a simulation on stocks trading based upon predictions yielded results that, in term of profits, compare well with a good human trader. Ongoing work is focused on expert system based selection of specific trading strategies and use of meta-neural networks for the evaluation of the most appropriate specialized neural predictors trained for different market behaviors.

Introduction

Neural Networks [3] are being applied to a wide range of domains, because of their intrinsic ability to synthesise models which deal with fuzziness, uncertainty, incomplete data, errors, etc. [7]. This feature makes them suitable for domains lacking a well-defined underlying model or theory, because they do not require an a priori specification of a functional domain model.

For this reason neural network technology has being receiving considerable attention in the scientific community; it also represents a radically different approach to computation from the conventional algorithm model. The financial community is especially interested in this technology, for problems such as pattern detection, association and classification often arise in the areas of security investment and financial forecasting.

We believe that accurate stock market prediction is one such domain; we chose this application to find out whether neural networks could yield substantial help in designing and

implementing a successful short-term stock trading system. As Backpropagation (BP) is considered one of the best tools for most prediction applications, we based our system on a network trained with a standard BP algorithm. In the following section we outline the problem of forecasting in financial markets, and describe a specific neural-network-based weekly trading system for the COMIT index, a weighted average of market prices of all stocks listed on the Milan Stock Exchange. In ex-ante evaluation through a buying and selling simulation, this system outperformed a simple buy-and-hold investment strategy.

In order to improve the global performance of the system further, from the point of view of the profit generated on trading, we intend to add an additional module based on a simple rule-based expert system which selects appropriate trading strategies; meanwhile the simple fixed buy-and-sell strategy that is used at present already demonstrates the effectiveness of the combined approach. Further ongoing work is about automatic optimised selection of specialized neural predictors for different market behaviors using meta-neural networks [4].

1. Forecasting in financial markets

In order to buy and sell financial instruments successfully, predictions on their likely future prices have to be made. In the free market, investment is driven by movements and trends in prices, which determine the opportunity for traders to buy or sell according to their expectations about future market behaviour; the market dynamic has therefore inspired traders to seek a method of forecasting future market trends The approach applied to prediction in financial markets is generally twofold: Fundamental Analysis and Technical Analysis [3] The former attempts to predict stock prices by using macro economic data such as interest rates, income, money supply, and microeconomic data such as the status of the company. The latter uses a different approach: the analytical study of the dynamics of the market itself rather than the factors that affect it. In this case, predictions are made by analysing price and volume patterns, to discover the clues for future movements and trends: they are based on the belief that the markets behave in a recurrent manner, and that the past history of prices could help in predicting future market movements.

In our work we adopted a technical analysis view on the market. We use a neural model to correlate retrospective technical indices with the future trend of short-term price movements and the corresponding timing for buying and selling. We focus upon short term price fluctuations, trying to exploit the most recent past price and volume patterns in order to obtain accurate predictions of future market trends.

Because of the relative turbulence of prices in stock markets, it is impossible to achieve 100% accuracy in predictions from such a model; instead we expect to obtain predictions with more than 50% accuracy in identifying short term price trends.

As it is highly desirable that the predictions of the system should have an objective link with the daily activities of traders, the combination of buy-and-sell timing indications with related strategies seems to be the right approach in order to maximise the returns from the predictions supplied by the neural-network-based model.

1.1 Market Structure

In financial markets the leading model is the 'efficient market theory of prices', which implies that past price time series behaviour can provide no information about future prices. It is well known that the Random Walk Theory [8] claims that price history is not a valid indicator of future price trends, and that the price changes are independent and behave in a random manner, with the consequence that future price behaviour cannot be predicted with any accuracy. On the other hand, since inefficencies exist in the market, traders indeed try to exploit them, and in doing so they rely heavily on past time series behaviors. Whether this falsifies the theory or it is just a transient behavior of the 'eficient market' it is not for us to say. We can only state that these inefficiencies exist and that traders and market observers try to exploit systematically using some sort of relationship between past price patterns and future price trends to obtain profits; in other words they are implicitly exploiting a time inefficiency.

It has also been noticed that the movements of market prices are in turn non-random and seem to behave in a non-linear manner [10]: in fact they appear to be the output of a non-linear, dynamical process, whose equations are as yet unknown: in this case, financial time series may

have partly 'chaotic' character , in that there may be a more or less simple underlying relationship that generates price movements, although we may not know what it is. Because of the assumed market structure, we expect also that the prediction accuracy degrades in time, and that long term predictions with a high degree of accuracy and low risk are much more difficult to obtain. For this reason, we believe that the forecasting horizon has to be quite short, because of the chaotic structure of the market and the consequently dramatic increase of prediction errors with time.

Some experiments have also shown that for chaotic time series a neural network can give short-term forecasts with a high degree of accuracy and some interesting results [7]; a neural network can learn the underlying equation of a chaotic system without being given any advance knowledge about the structure of the system, by simply learning the relationship between input and output examples.

In our experiment we adopted a substantially technical point of view on the market: we therefore believe that there exists an inherent relationship between a set of technical indexes and the future trends of price movements, and that a neural network can successfully learn it, and provide useful predictions [1], [2], [5], [6], [12].

2. The Model

According to our technical point of view on the market, we believe that the past behaviour of prices and correlated information can help to predict future short-term price trends. Our aim was to build up a timing prediction system based only upon the past history of the market itself, which can adapt its structure to learn the right timing for buying and selling The prediction has to be made for one week in the future, using the retrospective features of the price walk.

As neural networks, with their unique self-adaptive capabilities, are able to perform pattern recognition in very noisy environments [12], they are ideally suited for learning patterns in the market and recognising them with a high degree of accuracy. For this reason our prediction system is made up of a neural network which has learned the relationship between a set of past technical indexes and a 'perfect indicator' which indicates when to buy and sell. Therefore the correlation is made with an abstract synthetic indicator directly related to the purpose of our prediction, i.e. the right timing for buying and selling.

2.1 Data Selection

In developing our prediction system, we decided to experiment with the Italian Stock Market. The goal was to build an accurate prediction system to forecast the timing of buying and selling for the COMIT index, and to generate a recommendation for the current week's position in the market. COMIT is used like the Dow-Jones average.

The raw data consisted of 1923 observations of COMIT's high, low, close and volume at the end of each trading day. In our system we sampled daily observations every 5 trading days to obtain a time series representing the COMIT's high, low, close and volume at the end of each week from January 1984 to August 1991, a total of 384 observations. As only 366 of these observations share the same time horizon, owing to the different time scope of each index, our system uses this subset of data for the training and testing phase.

As mentioned above, because of the market structure, the non-linearity of the prediction, and the great amount of noise, the time scope of the system must be short: data indexes calculated at week 't ' are used to forecast the future trend for the next week 't+1'. Of course our model could be used for different short-term time periods given a different stock data set.

2.2 Input and Output Data

According to our approach, the easiest way to implement a computer-based automatic trading system from the technical analysis point of view is to compute a set of various technical indicators, i.e. mathematical transformations of past price and volume data which are claimed to have some forecasting validity. Our system thus assumes that stock prices are predictable from temporal and spatial patterns in a set of technical indexes, such as moving averages and various oscillators. The system takes into

account only a limited set of quantitative indicators, selected from those used most often in technical analysis.

If we consider a generic week 't', raw data for the COMIT index form the array

$$COMIT(t) = (D(t), H(t), L(t), C(t), V(t)).$$

Where D is the date, H is high, L is low, C stands for close and V for volume.

Having collected a time series of weekly observations of type COMIT(t), we chose a set of functional transformations which can map the series COMIT(n), n = 1, 2, ..., N, on to a time invariant set of technical indicators I(t) for a particular point in time (week t). The resulting time series I(n), n = 1, 2, ..., N, represents the retrospective characteristics of the COMIT index in the form of a set of technical indicators which incorporate the more or less recent history of the index.

In training the network we chose a set of sample technical indexes, which represents its training set. Training data were built up from the values of 21 technical indicators at a particular sample point in time (week t), and a number, the so called "perfect indicator", which shows how the market had changed at the following sample point (week t+1).

Some of the technical indexes are:

- RSI
- Stochastic Oscillator
- PVI
- OBV
- %K stochastic
- etc.

The prediction model uses a moving average of each index, in order to minimise noise due to random walk, and to make the time series as regular as possible; because of the high irregularity and the different value scale, input indexes were also pre-processed (filtered), normalised and scaled in the [0, 1] range to become suitable for neural network processing. This is unavoidable, as the range of one indicator may be very different from that of others, and its changing in time may lower the influence of other important but smaller indicators.

The right timing for when to buy or sell, used as learning target output, is indicated as the 'perfect indicator'. The term 'perfect indicator' states that if we could exactly know this value, we would be able to trade with 100% accuracy.

In our system, given the value of COMIT index (COMIT) at time t, this measure is the COMIT weekly return (wr) at time t+1. Stated formally, the target output of our network is defined as follows:

$$wr(t) = (COMIT(t+1)/COMIT(t) - 1)$$

As for the input indexes, the raw data wr(t) were pre-processed to smooth out excessive irregularity due to high volatility, through the following transformation:

$$r(t) = MA_3 (wr(t))$$

where the operator MA_3 is the moving average of order 3. The resulting values r(t), the averaged weekly returns, which represent the network target output, were then pre-processed (filtered) and scaled accordingly, to obtain analogue values in the [0, 1] range.

3. Network Architecture and Learning

Our system is based on a feed-forward network, which uses a back-propagation algorithm [9] for the learning phase. Our choice is derived from the consideration that this algorithm is probably the best known and tested up till now.

The training strategy used the first 265 input-output examples for the learning phase, during which the errors are calculated and propagated to change network weights, until a reasonably low overall error was reached. The remaining data were used as test data for the prediction system: during the testing phase only forward processing was done and no error propagation occurred.

We used a standard commercial neural network development environment to perform the learning and the testing phases.

3.1 Architecture Parameters

In designing the BP network, we had to make decisions on the number and the size of network layers. As the input and output layer size is strictly determined by the corresponding vector length, the only choice is the number of hidden layers and their size. The number of hidden

layers in a BP network sholud be generally between one or two, as more than two layers do not increase the network's ability to learn [11]. Since it has been noticed that a single hidden layer is generally sufficient even for complex mapping, we adopted such a configuration for our model.

According to experimental results [1], the number of neurons should decrease from the input to the output layer; we started using 15 units in the hidden layer, and after a lot of trial and error we reduced the hidden layer size to 7 units.

The final architecture of the network is illustrated in Figure 1.

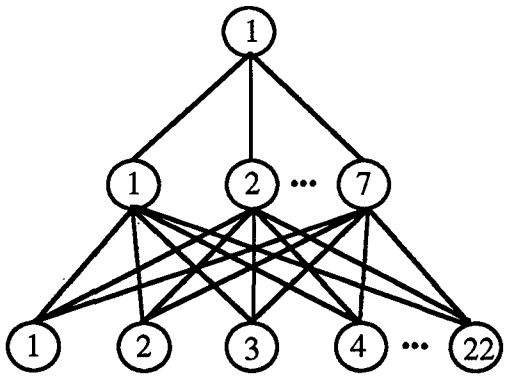


Figure 1

The resulting network consists of three layers: the input layer, one hidden layer, and the output layer. For our initial test the three layers are fully connected to each other to form a hierarchical network; we are now considering a pruning strategy to achieve partial connectivity, while maintaining fairly good accuracy in predictions.

The input layer is composed of 22 units, one for each indicator plus a bias unit. The output layer has only one neuron, which predicts the perfect indicator. For the hidden layer we opted for 7 hidden neurons, as mentioned above.

Each unit in the network performs weighted addition of its input, to calculate its activation potential. This value is then passed to a non-linear transfer function which maps its input onto an analogue value in the range (0, 1). As transfer function we used a standard sigmoid function asymtotically limited between 0 and 1.

3.2 Learning

We experimented to find the network with the highest performance and prediction accuracy, trying many network configurations which differed in the number of hidden layers and the number of neurons in each layer; the learning parameters were also applied in a wide range of different patterns, allowing us to discard many low-performance networks.

This activity was very time-consuming and required much trial and error to achieve reasonable results. We adopted a standard back-propagation algorithm for the learning phase, varying randomly the order of presentation of learning patterns so as to avoid undesirable memorisation effects, and to force the network to learn properly the underlying relationship between input and output. We also used different learning coefficient schemes: as learning progressed we lowered the learning rate and the momentum and training tolerances, to make the network reach a reasonably stable state. A small amount of random noise was added to input patterns to facilitate training and prevent the network from overlearning [11].

Finally, the network which performed best (with the highest prediction accuracy and the lowest error) was trained on 265 training input-output examples.

The training algorithm took about 4 hours of CPU time on a 386 PC at 25 MHz equipped with a math coprocessor to converge to a useful configuration, and about 140,000 iterations were necessary to reach a performing level of network convergence.

The resulting network is designed to run once a week to generate an input-output decision map for the end of the next week. This prediction is then compared to the actual target value, and an error is recorded. To verify the network's accuracy in making correct predictions, we initially computed a set of statistics on data examples that were previously seen by the network; the aim of this step being to verify that the network had learned the correct input-output mapping.

The resulting statistics are listed below.

$R = 0.7831$

$R^2 = 0.6133$

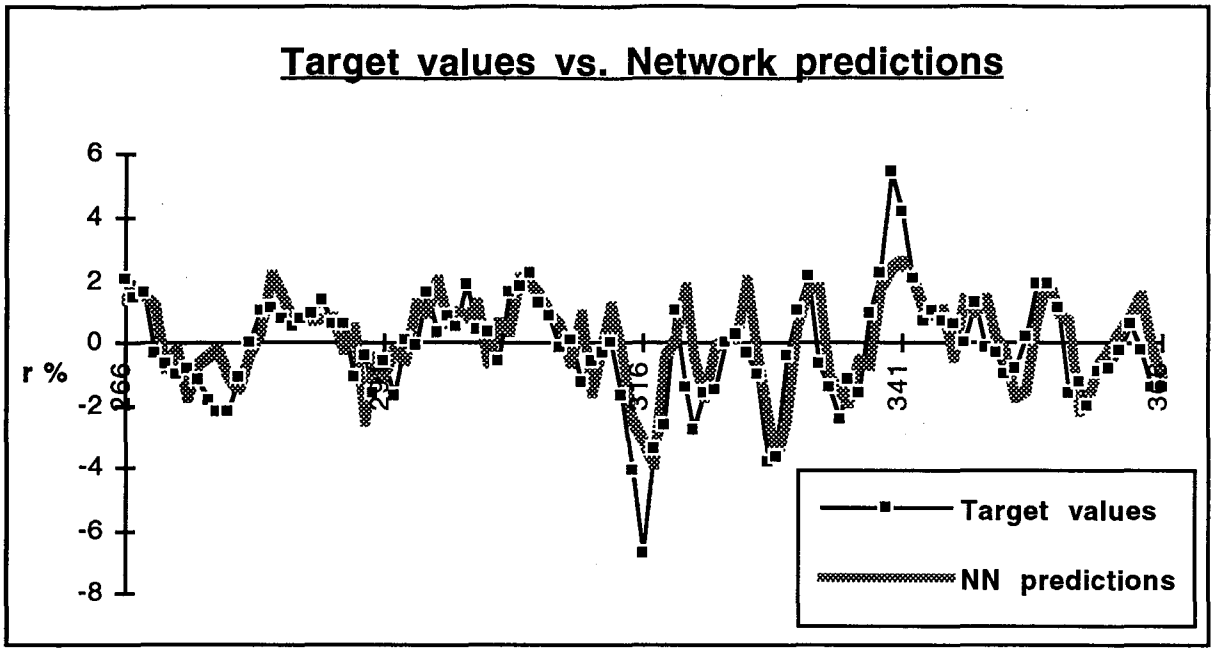


Figure 2

RMSE = 1.2824
%CT = 81.13

Where R is the correlation coefficient, RMSE is the root mean square error and %CT is the percentage of successful price trend prediction.

These statistics show that the network was able to predict the correct price trend for the coming week about 81 out of 100 times. In other words, at the end of the learning phase, the exact trend (sign of COMIT's averaged weekly return) was predicted with an accuracy of 81%, which is a fairly good result.

4. Evaluating Network Performance on Testing Set

The resulting network was applied to the remaining 101 data examples, which form the validation set. These data were not used during training, and during testing no re-learning was performed. The test phase is necessary to verify that the network is able to generalise well what it is supposed to have learned, and to determine whether it performs at a level that is useful for practical purposes (i.e. trading).

As in the previous test, we ran the network and recorded its output against the known target values for each of the examples.

Figure 2 shows a comparison between actual output and network prediction for the data in the validation set.

To verify the network's accuracy in forecasting the exact timing for buying and selling, we recorded some statistical coefficients of its performance:

RMSE = 1.211
R = 0.7292
%CT = 77.23

Synthetically, over the course of the validation set, the network predicted the trend (up or down) for COMIT prices 77% of the time. This result compares well with a skilled human trader.

5. Performance Evaluation with a Simulated Trading System

In order to translate these results into a more valuable form, we tried to verify the effectiveness of the prediction system using a simulation of buying and selling of stocks in the Milan Stock Exchange, following the system's recommendation. In this simulation, our aim was to determine whether the neural model

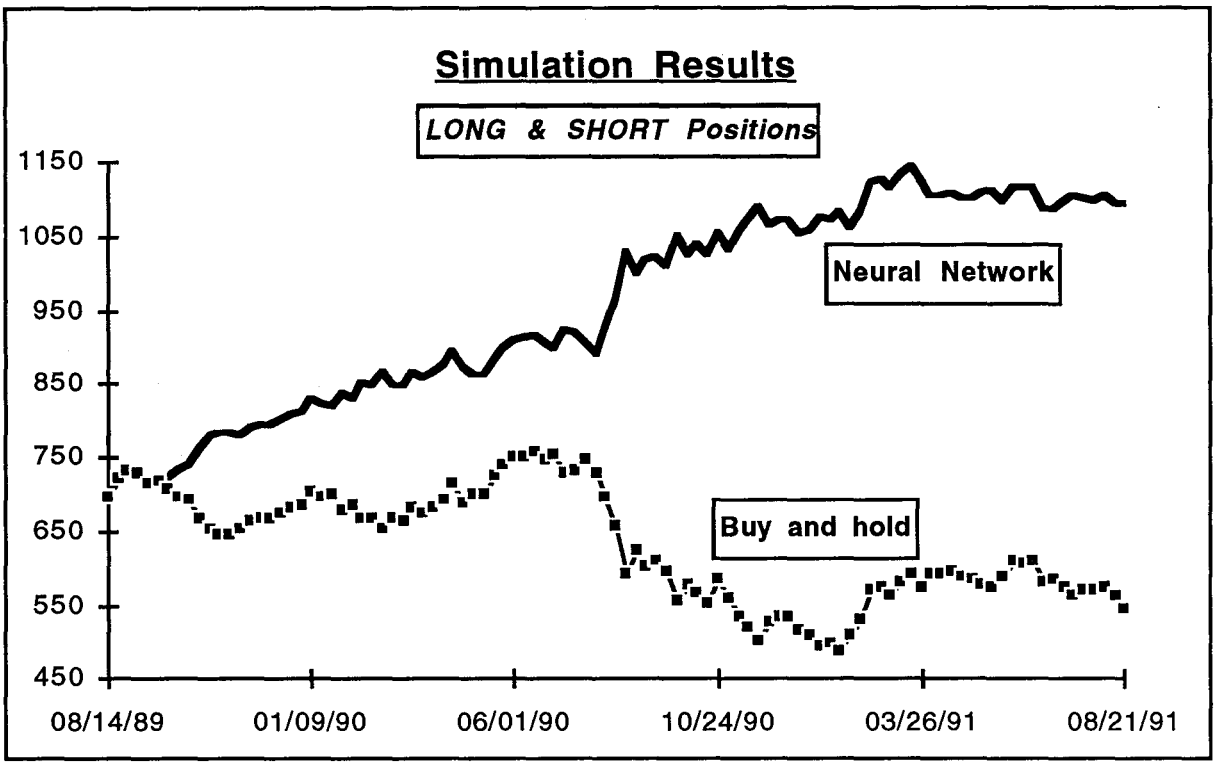


Figure 3

outperforms the simple buy-and-hold strategy, i.e. the index itself, over the same period.

The timing for when to buy and sell stocks is given by the network' s output: we considered an output above zero to be a buy signal and any other output a sell signal. According to this trading strategy, the network output could be considered a Long or Short recommendation.

During our simulation the prediction system always took a position in the market: it traded every week whenever an impending reversal in the market was detected, doubling up its position by going simultaneously long and on subsequent reversal of the market closing the opened long and going short. In other words, this fixed trading strategy means that the system is always in the market: it buys stocks long and then it sells them later going short at the same time. For the initial evaluation of the system, we decided not to use trailing stops to control losses and protect gains. The use of an appropriate stop size appears to be unavoidable to build up a simple money management rule, and is currently under investigation.

Performance of the model is reported in Figure 3. The lower diagram shows the buy and hold performance; the upper one shows the performance of the prediction system in terms of cumulative points gained from each trade.

Below are some statistics regarding simulated trades generated from the system:

Total Long Trades:	12
Profitable Longs:	5 (41.7%)
Total Short Trades:	11
Profitable Shorts:	8 (72.7%)
Biggest Gain (pts):	125.720
Successive Gains:	5
Biggest Loss (pts):	-18.900
Successive Losses:	4
Total Gain (pts):	398.580
Average Gain per trade (pts):	17.330
Total Gain on dealing capital %:	77.12

The buy and hold strategy yields a loss of about 11% (difference between initial and final invested capital), while the network yields a profit of about 77% on dealing capital invested over the same period.

Finally, we point out that actual simulated trading of the COMIT index using our prediction system achieved profits at about 33% per year on dealing capital invested, substantially outperforming the investment in the index.

Use of the system yielded a good overall profit despite events like the Gulf War which occurred during the test period.

Different buy and sell strategies might be more appropriate under different circumstances: this is currently being investigated.

6. Final Remarks

We have presented a neural-network-based prediction system which attempts to forecast the exact timing for buying and selling stocks, showing a good profit during simulated trading. A great deal of work remains to be done to turn these results into a useful working system for real trading, but we have already gathered evidence of the great potential of this technology for financial trading.

The two major directions for the development of the existing system are the expert system module for the selection of appropriate buy and sell strategies and the meta-neural network architecture for the automatic selection of the most appropriate neural predictors to apply to current market behaviors.

The expert system is needed because a single weekly trend is not a sufficient indicator of the best trading strategy: the trend must be interpreted in the light of previous trends. For example, an upward movement in the market which has been growing for several weeks has a different meaning from an upward movement in a market that has been falling.

The need for meta-neural networks, which proved to be successful for the problem of face-recognition [4], is motivated by the importance of the identification of the most appropriate neural predictors, under different stock market behaviors. Since specific, rather than general, neural predictors can be synthesised to deal with particular market behaviors, the use of meta-neural networks allows each time the automatic selection of the most appropriate one.

Both of the above improvements are expected to increase significatively the performance of the overall predictor.

References

[1] Bowen 91
Bowen, J.E.: "*Using Neural Nets to Predict Several Sequential and Subsequent Future Values from Time Series Data*", Proc. of the First Int. Conf. on AI Applications on Wall Street, New York, 1991.

[2] Colin 90
Colin, A.: "*Citibank STTM (Short Term Trading Model)*", Rocket Science Made Simple Forecasting & Optimisation in Financial Services, London, Oct. 1990.

[3] Eng 88
Eng, W.F.: "*The Technical Analysis of Stocks, Options & Futures*", Chicago, Probus Publishing, 1988.

[4] Gardin 92
Gardin, F., Flocchini, P., Mauri, G., Pensini, M., Stofella, P.: "*Combining Image Processing Operators and Neural Networks in a Face Recognition System*", in International Journal of Pattern Recognition and Artificial Intelligence", World Scientific Publishing Company, vol. 6, No. 2&3 pp. 447-467, 1992.

[5] Jang 91
Jang, G.S., Lai, F., Jiang, B.W., Chien, L.H.: "*An Intelligent Trend Prediction and Reversal Recognition System Using Dual-module Neural Networks*", Proc. of the First Int. Conf. on AI. Applications on Wall Street, New York, 1991.

[6] Kimoto 90
Kimoto, T., Asakawa, K.: "*Stock Market Prediction System with Modular Neural Networks*", Proc. of the IEEE Int. Conf. on Neural Networks, San Diego, vol. 1, 1990.

[7] Lapedes 87
Lapedes, A., Farber, R.: "*Nonlinear Signal Processing Using Neural Networks: Prediction and System Modelling*", Los Alamos National Laboratory report, Los Alamos, 1987.

[8] Malkiel 85
Malkiel, B.G.: "*A Random Walk Down Wall Street*", Norton, New York, 1985.

[9] Rumelhart 86
Rumelhart, D.E., Hinton, G.E., Williams, R.J.: "*Learning Internal Representations by Error Propagation*" in: Rumelhart, and al., Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1: Foundations, Cambridge Mass., MIT Press Ed., 1986.

[10] Skeinkman 89
Skeinkman, J.A., Lebaron, B.: "*Non-linear Dynamics and Stock Returns*", The Journal of Business, University of Chicago Press, vol. 62, n. 3, Mar. 1989.

[11] Wasserman 89
Wasserman, P.D.: "*Neural Computing - Theory and Practice*", New York, Van Nostrand Reinhold, 1989.

[12] White 88
White, H.: "*Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns*", Proc. of the IEEE Int. Conf. on Neural Networks, San Diego, vol. 2, 1988.

Neural Nets -- A Practical Primer or... What I Wished I Knew Four Years Ago.

James W. O'Sullivan

OBIL Neural Technology Inc.
Portfolio Management
Pequot Plaza, Southport, Connecticut 06490

Abstract

Neural network methodologies have spread rapidly over the past several years. One software house claims to have sold over 16,000 units. Experience suggests applicational success can be elusive. Consistency between the output task(s) of a network and investment objectives is critical. A rule-based expert system employing neural network forecasting modules can be powerful. An approach is outlined.

1. Introduction

Neural Network and related technologies have entered the lexicon of market participants as powerful and intriguing methodologies for investors -- as diverse as Shiller's [1] "Noise Traders" and "Smart-Money Investors".

Just a few years ago a listing of neural network technology suppliers would have been limited to pioneers such as California Scientific Technology, Neural-Ware and perhaps one or two others. A recent special issue of *AI Magazine* [2], listed no fewer than 44 providers of neural software and related services. There are now at least eight advisory services which offer Neural Network based trade advice. California Scientific's *Brainmaker* is now compatible with two INTEL neural chips and other accelerator boards are being offered. Frank Key of Structured Software [3] is engaged in an unusual effort to engage neural capacities on-line for intraday trading. Yale author, Paul Kennedy, ("The Rise And Fall Of The Great Powers") [4] alludes to global currency trading using intelligent neurons.

When OBIL first started to develop neural-network methodologies some four years ago, it seemed a lonely path to tread, now it is likely that most serious market participants have explored, at least tentatively, the use of neural networks.

Why the explosion of interest?

- The methodology is intuitively appealing -- the idea of "training" an adaptive system provides an assurance enriched by the promise of predicting the future. A powerful watchword in this business.
- The methodology eschews any need to understand the design mathematics, and respectably acts as an apologist for "Black-Box" explanations.
- Neural network applications provide opportunities for developers to work with many variables and formats. Genetic Algorithms applied to "training" variables are now offered which furnish satisfying feedback as networks train and converge.
- Nominal or apparent cost of entry is low. Network tools are offered for just a few hundred dollars. Access to price-data, technical indicators and fundamental data is cost effective and has become widespread.
- Network testing processes appear straightforward and credible. Investment system simulations promise a fairly quick means of assessing how a network is likely to perform in real-time.

Yet for all this interest there is a dearth of specific, credible application examples. Extravagant claims periodically surface. *Business Week* [5] reports that a $249 neural network software offers annualized returns in the fixed income market of 292%. Advertisements now routinely claim that networks can be designed and trained, ready for use in mere hours. Professional magazines include many examples of Trade Advisors claiming 90% successful trades and high confidence in predicted price-levels.

In the face of this, the appeal is understandable. However, in our view, widespread use will not so easily translate into legions of market beating systems. After over four years of effort we view the technology as tough, demanding and just as susceptible as other technical

systems to widespread misuse. The technology will do little to admonish those investment and trading habits which impede so many in their pursuit of market beating systems.

Yet for those who endure, applying consistent, disciplined effort opposite a clear objective the rewards can be worthwhile. Neural networks can and do work. They can be powerful tools in discerning relationships, they are dispassionate, usually consistent and projective. When combined with rule-based methodologies the alliance can address quite complex solution sets.

The technology remains stubbornly empirical, by repeated observation you distinguish between what works and what does not. Introductory user texts such as that by Jeannette Lawrence [6] are helpful, and provide a useful reference for designing a project approach and implementation guidelines.

Below we discuss a couple of the pitfalls we failed to avoid, and then offer an overview of our OBIL process for the deployment of neural and expert systems in our investment decision making. The overview tenders a fairly complete sketch of OBIL's process, for neural network development but extends little by way of specifics on network design. This is deliberate, not solely for reasons of proprietary caution, but because the process is far more important than the specifics of what indicators to use and how many time periods to train for. Our networks are surprisingly robust, we do employ proprietary inputs and fundamental data modules, but do not exercise any unusual training regimens, hidden neuron layers or learning strategies which might dictate network success or failure. Our core network structure is a simple back-propagation model. Designing and adhering to the development process, in the final analysis, is more demanding, requires more rigor and is easier to neglect than network configurations.

There are occasional hazards awaiting, and a few, perhaps worthwhile, implementation observations to remark on.

2. Common Pitfalls and Application Observations

2.1 Output Task(s)

A good place to start any design process is to define what it is that one wants the network to do. Most of us tend to focus on some description of a future price level as a desired output. It seems fairly obvious that if we know the price level in say ten days time, and if that projected price is higher than the current price, we should be able to profit from this knowledge. It's actually not that straightforward. If the current price is at 10 and projected to go to 12 in ten days time, but if en route to this projected level the price declines to say 7 by day six--what happens? Most participants would be stopped out close to the 7 level, and clearly it would have made more sense to buy at or close to 7 and await the move to 12. So a sure thing neural projected gain of 2, in practice translates into a loss of 3, and could have been a gain of 4 or 5 had we known that a price dip to 7 was probable. So, in practice, simply having access to a projected price level at some future time is usually not enough. Even if we projected the likely lowest and highest price level over the next ten days, we may not be able to profit -- since we need to know which comes first, the low or the high.

If in this example, our network had projected a low turning point in about six days with a projected low around 7, followed by a move towards 12, it is likely that many could profit from this information.

How we ask the network these output questions is also important. The example implies we have asked for absolute low and high price levels and the specific timing for a low turning point within the next ten days. Specific answers to specific questions -- we have tried to achieve this level of specificity, with at best mixed success. We have found that a more indirect or "subtle" approach is far more powerful. For example, we may frame the questions in the following terms:

- what is likely largest negative deviation from, say the linear regression value over next ten days? This output can be expressed in relative price terms or better yet in units of Standard deviation.
- what is likely largest positive deviation from the linear regression value over next ten days?
- which is likely to come first the largest negative or largest positive deviation?

In place of a simple linear regression future value, one may use other standard statistical projection measures, such as multivariate regression values, and so on, the important point here is to measure price-level departure from some expression of trend.

Mapping the output from these questions gives us essentially the same projections for the next ten days as in the direct approach. A high of about 12, a low of about 7 and the low being seen before the high. Our trading order now becomes, buy at 7 sell at 12, if filled stop at 5, and cancel order if 12 is seen before 7. The stop price will reflect the individual's risk management practice. In actual practice one would slightly adjust the projected prices for placing orders.

2.2 Omni-capacity

There is a great temptation to design a network which responds to all one would ever want to know about a market. Short-term prices, price targets, turning points, volume, highest highs, lowest lows and so on. The thinking being that one can just add more input data which seem to relate to all these questions, create larger training sets and let the network sort it all out. In fact we have found it necessary to deploy different networks to respond to different questions. We keep our questions fairly simple and, as above, we favor the indirect approach.

2.3 Network Testing

Many neural network software manuals advise retaining some percentage, such as 10%, of the training set as test data, to measure how well the network performs. Some software packages will even perform the task, sequestering at random say 10% of the training set into a testing file. This can be fatal for many time-series configurations. Consider a network trained to forecast an implied price level in say ten days time. If we isolate every tenth data set (day) for testing purposes, we will often find that the network tests well, almost as well as the training tolerance. In other words test performance shrinkage is nominal. When one employs the network in actual trading the results may be significantly worse than would be expected from "testing" -- high shrinkage. What has happened is that the training set includes data for all those future days leading up to and after the test day. The training network has "seen" the days before and after the test day. A quite different forecasting proposition than actual usage conditions.

If a network has been developed to forecast low turning points, it is important to test on data which includes such turning points as well as data which does not.

Network testing appears straightforward, but in fact can be quite complicated. Testing a network, trained on say 270 days of data, using a 30 day block of (unseen) data is not usually good enough. Tedious as it is, it can be necessary to cycle through the whole 300 days creating different training and testing sets. The one testing set of 30 days may show uniquely good or bad results. One 30 day period cannot describe or typify the markets.

Network testing must also emulate how you intend to deploy the network in practice. Consider: if one intends to use a network to forecast a price level in two days time, and if the production process involves retraining such a network each night or perhaps each week, then the testing protocol must mirror the expected usage process. A way of doing this would be to assemble say 600 days of data, train the first 400 days, test on the next 2 days, then retrain on either 402 days or (400-2)+2 days, and so on. It is cumbersome, but inattention to testing can be costly.

If in this example, the production process or usage environment, does not call for such frequent retraining, the training and testing discipline appears less demanding. However, the question then becomes how robust is the network, can it really retain it's forecasting vigor over an extended period, particularly if the character or state of a market is changing.

Our experience suggests there are a few short cuts in neural network development.

3. The OBIL Approach

3.1 Introduction

OBIL is a recently organized investment management firm, employing investment technology developed by the principals over the past four years. We have recently scaled up our applications capacity from 100 to 2000 products (effective May 1, 1993).

In the early phase of our neural development effort we succumbed to the pitfalls mentioned above and experienced all the frustrations of crafting a technology which seemed to promise more than we could extract from it. In many ways we failed to direct the technology, neural network behaviors and peculiarities were, in effect, channeling our efforts.

It was then we recognized the need to formulate an actionable relationship between our investment style, and a measurable view or model of how markets function. At which point we would be able to drive the technology to identify market opportunities flowing from our model, which satisfied our style and objectives. Shifting our emphasis from technology exploration to active technology engineering allowed us to focus on what we wanted from neural networks and expert systems, rather than what we thought we could get from their use.

3.2 Investment Objectives

Our investment style is compelled by product-sets and minimum trade or position standards.

The two product-sets are:

- *Positioning:* outright positions in equities, currencies and derivatives, including synthetic spreads.
- *Indexing:* yield-curve and stock-market index enhancement approaches.

Product coverage includes daily or weekly scanning of over 2,000 global products. An approximate breakdown would be:

-- 1200 USA equities,
-- 100 USA derivatives (including Futures),
-- 600 International equities, and,
-- 100 International derivatives.

Trade or Position standards embrace:

- *Trade Selection:* probability of strong trending behavior over a one to six month time-frame.
- *Volume:* always to the left of Vince's [7] optimal-f and related to risk. Ideally seek to enter and exit positions with no more than 25% of optimum volume. During trade, partial exit effected at target prices or trailing stops.
- *Timing:* use timing tactic for best entry price, and when to add or subtract volume.
- *Target Price:* must be able to identify a high probability target price which meets our return objectives.
- *Stop price:* applied to initial entry volume must pass a capital management (preservation) screen.
- *Trailing stop:* require process for deciding when to employ close-by or loose trailing stops.

3.3 Market-Model

Since we are investment management practitioners and not market theoreticians, we allowed ourselves the luxury of selecting those elements of existing market-models which seemed consistent with our observations and investment style.

The efficacy of our derived market-model was measured against two major criteria:

1. does it describe the market in terms we can employ in our investment decisions, and
2. can we deploy our neural and expert systems to identify and measure the various market behaviors?

We concluded that:

- Traditional models such as Capital Allocation Pricing Model (CAPM) and Efficient Market Hypothesis (EMH) do not work well and derived forecasts of market behavior have generally lacked empirical efficacy.
- Concepts of Equilibrium and Price-Efficiency are particularly difficult to relate to observed behavior.
- Our empirical view is that markets are non-linear dynamic systems (Peters' Chaos Theory [8]), characterized by:

  -- Long-Term correlations and trends (Market-Memory),
  -- Erratic behavior under certain conditions (multiple forecast solutions),
  -- Self-Similarity (Fractal behavior).

- Within this broad framework we utilize a variant of Vaga's [9] Coherent Market Hypothesis (CMH), and categorize markets as being in one of four states:

  -- Congruent ... strong trends, low risk,
  -- Transition ... directional bias, low-medium risk,
  -- Basing ... distribution/consolidation, high risk,
  -- Conflict ... frequent price-shocks, high risk.

  There are "fuzzy" boundaries between the Congruent and Transition states, and between the Basing and Conflict states.
- Risk for us is framed in terms of "State-Selection risk", i.e., what is the risk that we have mis-categorized a market-state.

OBIL's Congruent market state, our lowest risk and preferred investment condition registers high implied volatility and, within more traditional frameworks, would be accorded a "high-risk" ranking, with an associated high beta value (probably). Similarly our high-risk Basing state would enjoy a lower risk ranking.

To bring to bear a quantitative basis for defining our market states we use a whole range of technical and fundamental measures, many of which are in common usage. Two or our measures which are less commonly encountered include the Hurst Exponent [10] and the Stofko ratio [11]. We employ a modification of the Hurst exponent as below.

$$R/S = f(A*N*)^H$$

R/S = rescaled range,
A = a constant,
N = length,
H = Hurst Exponent.

The rescaled range can be computed by dividing a measure of historic range by the time-series standard deviation.

This measure is helpful in distinguishing Basing and Conflict market states, and on a lagging basis it relates the intensity of directional bias.

Another measure of directional bias, or potential onset of a Congruent market state, is given by the Stofko Ratio, which measures the ratio of long-term resistance to long-term support normalized for historic volatility. A low ratio is indicative of high trending potential.

$$SR = A*((R/S)/V)$$

SR	=	Stofko Ratio,
A	=	a constant,
R	=	resistance price-level,
S	=	support price-level,
V	=	historic volatility.

We have found that even our best technical measures and applications of fundamental data pose interpretation and timing challenges in terms of systematic trade identification and control objectives. However, they provide important inputs to our neural networks.

3.4 System Structure

OBIL's Investment matrix is shown below:

Market State	Freq. %	Risk	Price targets	Optimum Position Initial	Optimum Volume Mature
Congruent	10%	Low	Yes	25%	100%
Transition	20	Medium	Yes	10	50
Base	45	High	No	---	---
Conflict	25	High	No	---	---

(Frequency refers to percentage of time in which a given market state is observed for any given product.)

We deploy 3 neural networks embedded within expert systems incorporating about 1,000 lines of primary code and reliant on about 150 "rules or conditions". We can scan up to 2,000 products each evening, with a capacity to output a daily data sheet for each. Our system relies on an alliance of weekly, daily and hourly price data bars.

Early on in the daily process a set of screening algorithms pares down the initial product list, eliminating those who do not meet our key criteria (see below). Typically the system declares further interest in only about 150 to 200 products, and of these just 5 to 20 merit any action, such as adjusting volume or modifying stops.

In practice our output is limited to:

- A summary of all products which match our standards (congruent/transitional states),

and more details on:

- those products in which we have a position,
- potential new positions,
- selected indices, such as The Dow Jones Average,

The primary and supportive daily output elements and system source are:

Output Element	Source	Bar-length
Primary:		
Market State (Congruent etc.,)	Expert	Week/day
Position (long/short/flat)	Expert	Week/day
Position Volume	Expert	Day
Long-term price target	Neural	Week
Key trend prices	Neural	Week/day
Short-term turning point	Neural	Day
Supportive:		
Orders -- new/revisions	Expert	---
Short-term price target	Expert	Day
Next two day price bias	Expert	Day/hour
Today's price bias/range	Expert	Hour
Best position entry price	Expert	Day
Best position stop price	Expert	Day
Next 3 hour price bias	Expert	Hour
IntraDay Support/Resistance	Expert	Hour

Primary outputs decide investment positions while supportive outputs determine day-to-day adjustment and position control actions.

Examples of daily outputs are attached. Exhibit I, shows the output for a non-traded index as of March 23, 1993, and is an example of a "conflict" market state. It shows the Dow with a short-term or two week target of 3412 (prior close 3459), even though the long-term, or two month target is 3622. This typically suggests some choppy price action is likely.

Exhibit II, on the other hand shows a "congruent" market state for the German Mark June 1993 futures. The two week long target is identified as 0.6181 (prior close 0.6064) and the two month target is shown as 0.6470. There is also an interesting example of a projected "high range or turning point" projected for

March 24th or 25th. The position volume is shown as a total of four contracts per $100,000 of allocated trading capital.

Identifying system sources of information, such as neural and expert, illustrates an emphasis for the respective output element, however in practice the system is interdependent. Many of the Expert decision-sets depend on neural outputs as "if... then... else" conditions, and some of our neural inputs depend on rules derived from technical measures expressed in binary terms -- "on/off".

Our networks are based on back-propagation and are retrained weekly. Typically we employ a mix of technical and exogenous fundamental data. Product-specific fundamental data, such as p/e ratio and dividend yield, are captured within our expert systems.

The exogenous or environmental data is assessed in one "Utility" network which services all products and outputs data on global interest and exchange rates. (USA, Europe and Japan).

Product specific inputs are time-series technical measures typically expressed in ratio terms, but do also include binary "on/off" switches which reflect responsiveness, for example, to the output from our environment network.

Our expert rules are pyramidal, a large base builds to an apex or final "decision-set". Many of the rules draw on neural outputs and technical measures as well as product specific fundamental data.

A simple English example of a rudimentary apex rule would be:

IF, product is now in congruent long condition,
and, has short term target=A,
and, has long term target=B
and, product yesterday was not in congruent condition,
and, position in product is flat,
and, next two day price bias is down,
and, today's price bias is down,
and, there are no price-turn points projected for next X days,
and, earnings report is not due for at least Y days,
and, options not expiring for Z days.
THEN, place order at open to buy 0.25*C-shares/contracts at best entry price=D, stop=E.

While the rules are not actually written quite like this, it demonstrates the approach. In this, hypothetical and simplified example the initial "IF" statement, would itself be the product of several other outputs and rules.

4. Results

Our actual trading results, based on modest portfolios, have been encouraging. Our positioning account, has shown returns for 1992 in excess of 40%. Our equities based Indexing has significantly outperformed the SP 500 index. Scale-up simulations to 2,000 products have confirmed these performance levels, providing for greater month-to-month consistency as well as larger portfolio capacity.

5. References

[1] Shiller, R.J., *Market Volatility,* M.I.T. Press, 1989.
[2] AI Magazine, *Neural Network -- Special Report,* Dec., 1992.
[3] Key, F.S., *Structured Software,* Discussions with Author, 1992.
[4] Kennedy, P., *Preparing For The Twenty-First Century,* Random House, 1993, and discussions with author, 1992.
[5] Business Week, *Where Neural Networks Are Already At Work,* November 2, 1992, page 136.
[6] Lawrence J., *Introduction To Neural Networks,* California Scientific Software Press, 1993.
[7] Vince R., *Portfolio Management Formulas,* Wiley Finance Editions, 1990.
[8] Peters, E.E., *Chaos And Order In The Capital Markets,* Wiley Finance Edition, 1991.
[9] Vaga, T., *The Coherent Market Hypothesis,* Financial Analysts Journal, January, 1991.
[10] Peters, E.E., *ibid.,* page 62.
[11] Stofko, J., Discussions with Author, 1992.

EXHIBIT I

Daily Output for Non-Traded Dow Jones Average
Produced on March 23, 1993

OBIL NEURAL For next Trading Day after...930323 Dow Jones Indust Average

LONG TERM OBJECTIVE:

- - TODAY'S ACTION SCAN - -

LONG ABOVE 3339.8300 AND 3457.2700
LONG TERM TARGET = 3621.9700 [long]
SHORT TERM TARGET = 3412.6900 [short]

	2-Month	2-Week
POSITION:	Long	Short
PROBABILITY:	High	Low
QUALITY:	Medium	medium
Y- ZONE:	3339.8300	3457.2700
C -ZONE:	3353.9900	3406.4400
PRICE INDEX:		100%
PRICE INTENSITY:		Decreasing.

[conflict]

PRICE DYNAMICS:

UPTREND INDICATED ABOVE 3431.8500
and confirmed if above 3424.7000
*UPTREND CONFIRMED.

F O R N E X T T W O D A Y S : CLOSE = 3459.1600

Price-Phase Status:-
-No projections developed.

NEXT TWO DAYS:
If below 3450.2500 then DOWN BIAS

FOR TODAY...	Zones	Price objectives
UP, 55% if above	3444.5700 :	Bulls 3474.3900
DN, 75% if below	3438.8900 :	Bears 3443.9300

[SHORT STOP = 3487.5700]

BEST LONG STOP = 3430.7500
BEST LONG ENTRY = 3437.4800

LONG TERM OBJECTIVE:
MED TERM OBJECTIVE:
INTRA-DAY OBJECTIVE:

* LONG-TERM VOL:-
* MED-TERM VOL:-
* INTRA-DAY VOL:-

F O R N E X T 2-3 H O U R S:

POSITION : Short:
PROBABILITY: Med :
QUALITY : Med: [until below 3441.7300]

INTENSITY : Increasing short:
PRICE INDEX: 78
KEY NUMBERS: Y 3465.8100

INTRA-DAY RESIST/TARGET: 3484.1700
INTRA-DAYSUPPORT/TARGET: 3441.7300

* not solid short until below 3457.7000

EXHIBIT II

Daily Output for German Mark June '93 Futures
Produced on March 23, 1993

OBIL NEURAL For next Trading Day after...930323 German Mark

LONG TERM OBJECTIVE:

	2-Month	2-Week
POSITION:	Long	Long
PROBABILITY:	Medium	Medium
QUALITY:	High	High
Y- ZONE:	0.6064	0.6146
C -ZONE:	0.5944	0.5869
PRICE INDEX:		97%
PRICE INTENSITY:		Decreasing.

- - TODAY'S ACTION SCAN - -

LONG 3 ABOVE 0.5944
LONG TERM TARGET = 0.6470 [long]
SHORT TERM TARGET = 0.6181 [long]

[congruent]

PRICE DYNAMICS:

UPTREND INDICATED ABOVE 0.6007
and confirmed if above 0.6050
*UPTREND CONFIRMED.

F O R N E X T T W O D A Y S : CLOSE = 0.6064

Price-Phase Status:-
-High range or turning point today or tomorrow.
-Important phase count today or tomorrow.

*HIGH RANGE/TURN POINT TODAY-TOMORROW.

NEXT TWO DAYS:
Neutral price is 0.6006, expect FLAT ACTION

FOR TODAY...	Zones	Price objectives
UP, 55% if above	0.6029 :	Bulls 0.6095
DN, 75% if below	0.6009 :	Bears 0.6033

BEST LONG STOP = 0.6018
BEST LONG ENTRY = 0.6029

LONG TERM OBJECTIVE:
MED TERM OBJECTIVE:

* LONG-TERM VOL:- LONG 3
* MED-TERM VOL:- LONG 1

INTRA-DAY OBJECTIVE:

Piv = 0.5997 F O R N E X T 2-3 H O U R S:

POSITION : Long:
PROBABILITY: High:
QUALITY : High:
INTENSITY : Decreasing long:
PRICE INDEX: 34
KEY NUMBERS: Y 0.5951

INTRA-DAY RESIST/TARGET: 0.6090
INTRA-DAYSUPPORT/TARGET: 0.5997

* solid long while above 0.5951
though showing some loss of upward momentum. 0.6090 now key to continuing upaction.

Neural Nets in Investment Management: Multiple Uses

Dean S. Barr Ganesh Mani
LBS Capital Management
129 6th Avenue North
Safety Harbor, FL 34695
Phone: (813) 726 5656

Abstract

Neural nets have gained popularity in the last few years as powerful non-linear prediction tools. We describe multiple, related uses of a neural net in investment management with emphasis on forecasting the US equity market (the S&P 500 Index). LBS Capital Management has exploited neural nets to generate excellent predictions in terms of both magnitude and direction of price changes of the S&P 500 Index. In addition to forecasting, we discuss sensitivity analysis and rule extraction techniques. All the techniques discussed here are equally applicable to other markets.

1. Introduction

Investment managers often find themselves overwhelmed with large amounts of financial market information. Most of this information or data is numeric and often, also noisy. Typically, the investment manager processes a large subset of market data — both qualitative and quantitative — from various sources extracting relevant information to make investment decisions. These decisions frequently rely on the integration of statistical measures that attempt to compress much of the data and qualitative depictions such as graphs and bar charts with news events and other pertinent information such as analyst recommendations. Investment decisions usually involve taking into account significant non-linear relationships among the various components of the data. Even though investment managers and traders are able to factor in such relationships in the analysis, they may not be able to explain the decision and point out the significant factors and relationships contributing to the decision.

Computers, in general, are very adept at dealing with large amounts of numeric information. However, some clever algorithms are required to analyze and combine disparate information that can potentially impact security prices. By and large, Artificial Intelligence-based methods use such clever algorithms and rules of thumb (heuristics) in the decision-making process. Several expert system and neural network applications have been successfully deployed in the domain of Finance (e.g., see the Proceedings of the Innovative Applications of AI Conference over the last few years), including some in the area of investment management (e.g., see the Proceedings of the First International Conference on AI Applications on Wall St. and the other papers in these Proceedings). LBS Capital Management uses several expert systems and neural network models to manage portfolios totalling $300 million. Based on successful experiences with these systems, we elaborate upon several uses of a neural network in the investment management process. The discussion pertains to backpropagation implemented using the generalized delta rule [5] in a feedforward multilayer neural net.[1]

A feedforward network is not directly amenable to time series prediction problems since there is no obvious way to handle the temporal component (i.e., showing the network historical data in addition to data from the current time period for the various input indicators). We use a simple technique (explained below) to convert a time series prediction problem into a conventional prediction problem that employs cross-sectional data, by choosing certain historical time periods that are deemed significant by our pre-preprocessing routines.

[1]We used BrainMaker Professional from California Scientific Software and NeuralWorks Professional II/Plus from NeuralWare Inc. to simulate the neural nets.

2. Forecasting

In the last few years, neural nets have become popular for tackling problems ranging from protein secondary structure prediction to speech recognition. Such problems can be thought of as involving the prediction of the value of one variable based on several other variables that can impact it. In the investment management domain, proper neural network design is critical for generating good forecasts. It has been our experience that the more task-specific the neural net, the more effective it is. For instance, constructing a neural net to forecast the direction of a market trend may be more effective than forecasting both the trend and its amplitude due to noise and possible randomness inherent in financial market data. Other factors to consider in the forecasting process are—

a) the optimal time horizon for forecasting:
By this we mean whether to forecast the price of the S&P 500 Index a few minutes into the future, the next day, the next week, the next month, etc. Note that such an optimal time horizon changes over time necessitating constant re-examination and tuning of the predictive models and their inputs.

We have also usually found the change in the value of a composite index to be more predictable than the absolute value of the index. Currently, predicting the change in the S&P 500 Index 5 or 10 (trading) days forward appears to generate good results.

b) the set of inputs to use in building the neural net:
This involves multiple steps— 1) choosing the indicators (e.g., whether to use the Treasury Bill (T-Bill) yield as one of the indicators in predicting the change in the S&P500 Index), 2) deciding whether to use the change in the indicator from one period to another, and 3) determining the implied periodicity of the indicator (if any), and how far back in time the indicator should be examined. Thus, the search space of possibilities is potentially quite large. We have experimented with many of these possibilities. Currently, we employ a *telescoped time-delay* mechanism to generate a set of inputs based on one indicator. For example, in a number of models we have used the week-to-week (%) change in the T-Bill yield from 2 weeks back, 4 weeks back, 8 weeks back and 12 weeks back in predicting price changes to the major equity indices such as the S&P 500 Index. Each indicator may have its own optimal telescoping schedule. Note that such an approach converts time series (or panel) data to a cross-sectional format amenable to training a feedforward multilayer neural network using backpropagation.

The network currently used for predicting the change in the price of the S&P 500 Index employs a total of 21 indicators: 7 inter-market relationships derived from prices of other markets such as the CRB Index and Dollar Index, and 14 indicators based on price, volume and put-call ratio (on the OEX). A total of 126 inputs are derived from the 21 distinct inputs via telescoping and using data from 5, 10, 15, 20 and 25 periods back, in addition to the data from the current period. The telescoping periods are determined by pre-processing heuristics that attempt to maximize information content of the inputs. Each period corresponds to a training day in this case and the predictions are made 5 days forward. The network consists of 126 input nodes, 14 hidden nodes and 1 output node. An example of the network's forecast is shown in Figure 1 (the figure shows the actual value of the S&P 500 Index and the prediction from the LBS neural net for the 5 most recent trading days). The net for this prediction was generated using a data set of 182 trading days of which 164 were used in training. Data pertaining to the remaining 18 days was used in testing the network. The performance of the network on three trials with different train/test partitions is shown in Table 1 (the test set accuracy refers to the direction of change of the S&P 500 Index: up or down).

Trial Number	Test Set Accuracy	Number of Epochs
1	16/18	100
2	15/18	100
3	17/18	160

Table 1: Test set performance of a neural net that makes predictions of S&P 500 price changes 5 trading days (1 week) forward

We used a learning rate of 0.75 and a training tolerance of 0.1 while training the network. The network was tested every 10 epochs. In the process of arriving at the network reported here, we experimented with a number of configurations by varying the number of hidden nodes from 10 to 20

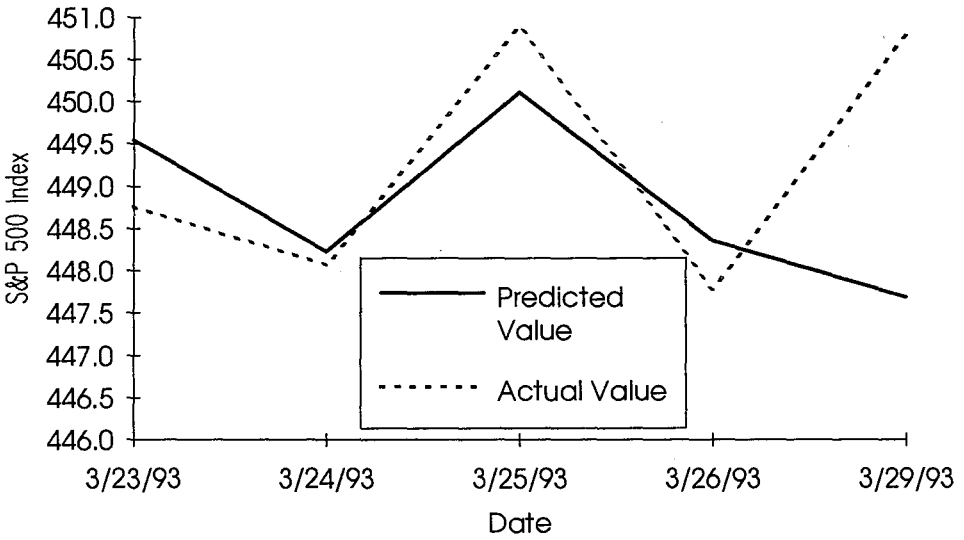


Figure 1: Predicting the S&P 500 Index: The dotted line shows the actual value of the index over the 5 most recent trading days while the solid line shows the LBS neural net prediction.

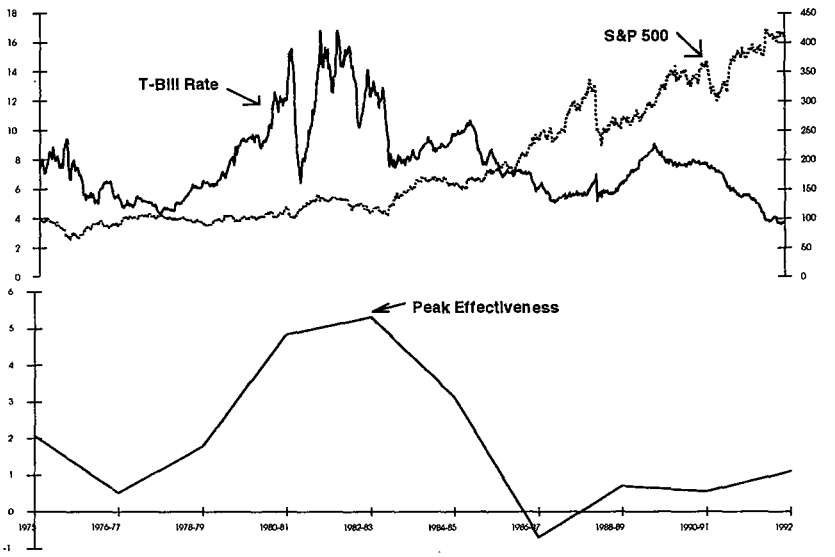


Figure 2: Sensitivity of (week-to-week % changes from 24 weeks back of) T-Bill Rates with respect to predicting the (% change 2 weeks forward of the) S&P 500 Index: The bottom graph shows the variation of this sensitivity from 1975 through 1992. The top graph shows the actual values of the T-Bill rate and the S&P 500 Index over this period.

(in steps of 2) and the learning rate from 0.5 to 1.0 (in steps of 0.25). The network described here is the best one (in terms of test set prediction accuracy) within this parameter space.

The prediction accuracy depicted in Table 1 refers to the direction of change in the S&P 500 and is derived from the actual predictions of the change in the S&P 500 Index. The average error of the predictions (taking into account the magnitude also) is around 0.08 while the RMS error is around 0.10 for the different test sets. The predicted values shown in Figure 1 are based on the magnitude predictions.

Training such a network takes about 3 to 5 minutes on an Intel 486-based PC. However, once the network is trained, predictions can be made almost instantaneously.

3. Sensitivity analysis

Some indicators have a larger influence on the variable being predicted (output) than others. One of the problems with predicting the price change of a composite index (such as the S&P 500 Index or the S&P 400 MidCap Index) is that a large number of indicators may be potentially relevant (from lumber prices to increases or decreases in jobless claims to changes in money supply). Note that with i indicators and t-way telescoping, ixt inputs are generated: While this may not be so large, the space from which the indicators and telescoping intervals are chosen is very large (exponential in both i and t). Determining a good set of indicators is crucial in generating good predictions as well as reducing training times to reasonable levels. Since a large number of indicators at first glance appear to be relevant, we have performed sensitivity analyses with respect to the different inputs. This involves noting the percent change in the output caused by a d percent change in one of the inputs (d is usually 5 or 10), keeping all the other inputs the same (dithering). Note that this is a weak sensitivity measure (since the possibility of non-linear interactions typically means that changes to two or more inputs in tandem can have a different effect from that of changes to one input alone).

An example of such sensitivity analysis is shown in Figure 2. The bottom plot depicts the % change to the output (which is itself the percentage change in the S&P 500 Index from one week to the next) in response to a 5% change to the week-to-week (%) change in T-bill rates from 24 weeks back. This was the most influential input from among other inputs derived from the T-Bill yield. In the top half of Figure 2, the actual T-Bill yield and the S&P 500 Index are plotted for reference.

One interesting aspect of this is that the sensitivity varies over time: the same neural net was used for the analysis but was trained on data from different time periods. Such an analysis can help prune the inputs (often sets of inputs, where each set is derived from a single indicator by telescoping).

Neural nets are often criticized for their black box nature: it is difficult to examine a trained net and infer rules from it. The sensitivity analysis provides one way of understanding a trained net. It provides the ability to see which inputs are being emphasized. For instance, by looking at a network trained to predict price changes to the S&P 500 Index, we can infer which investment theme is being emphasized in the equity markets in the time period under examination.

We have trained networks that attempt to capture the effect of different investment themes by using the S&P 500 price changes as the output and the price changes of a set of indices mimicking each theme as the input. The inputs for such analysis also includes price changes from a few periods ago. An example network involved using the following themes as inputs: Cyclical, Defensive, Growth and Financial.

For each theme, we created an index of four large stocks that fit the theme. The weekly prices of these synthetic indices were used along with two other price-derived indicators (a 4-period stochastic and a 4-week change in its value). A total of sixty inputs were generated from the 12 initial inputs by telescoping them to also consider values from 4, 8, 12 and 16 weeks back. From among a small number of different topologies we experimented with, the network with 8 hidden nodes generated the best performance (in terms of prediction accuracy). The network employed one output node. The accuracy in predicting the direction of the change in the S&P 500 was 69% on the test set. The average error was 0.0589 and the RMS error was 0.0798, taking into account the magnitude also on the test set.

Theme	Relative Influence
Cyclical	0.230
Defensive	-0.045
Growth	0.030
Financial	-0.090

Table 2: Relative emphasis on different themes for the four-week period ending 3/26/93

Once the network has been trained, the relative emphasis of different themes is computed as per the change each input has on the output (by dithering the input 10% in either direction). The relative emphasis numbers for the four-week period ending March 26, 1993 are shown in Table 2 and represent the change to the output (assuming other inputs are constant). A negative value for the influence denotes the existence of a negative correlation between the particular theme and the S&P 500 Index during that time period.

It is useful to know that the equity markets are emphasizing cyclical stocks at the expense of growth stocks. Such information can be used in tandem with a sector rotation strategy to realize potential profits.

4. Rule Extraction

The neural net can also be used as a tool for extracting rules that go into an expert system. Such a hybrid framework has the potential to augment rules and decisions supplied by human experts by providing rules that capture relationships that are not obvious to the investment manager, yet are captured by a neural net. Many such relationships may be based on correlations that are nonstationary (i.e. change over time). Since different investment managers may infer dissimilar information given the same data, temporary mispricings can occur in the marketplace. These mispricings provide the opportunity to generate profits until they disappear. The correlation-based rules inferred by the neural nets appear to be able to identify such short-term anomalies. Such rules can be used along with a conservative trading strategy, keeping in mind that the rules may stop working as the market environment changes. Also, by analyzing a rule implied on the basis of correlations, a human expert may be able to arrive at a more reliable or permanent rule.

Although extracting complete rules from a neural net is a difficult process, we can use sensitivity analysis to isolate subsets of indicators or factors that have the most impact on the output. These indicator sets can then be used to generate approximate rules. For example, we used a neural net to determine that the week-to-week (%) change in the S&P 500 P/E Ratio and the actual value of the current Bull/Bear Sentiment Indicator strongly influence the S&P 500 Index (% change) two weeks forward (see Figure 3). Based on this, we derived the following rule:

IF
P/E Factor * {P/E weekly change} + Bull/Bear Factor * {Bull/Bear value/1000} >= 0.05
THEN
Update Global Confidence Factor (SELL) by Adjustment Factor.

where

$$\text{P/E Factor} = \frac{\text{sensitivity P/E Ratio}}{\text{sensitivity P/E Ratio} + \text{sensitivity Bull/Bear}}$$

$$\text{Bull/Bear Factor} = \frac{\text{sensitivity Bull/Bear}}{\text{sensitivity Bull/Bear} + \text{sensitivity P/E Ratio}}$$

Such a rule can be used as a supporting rule to enhance the probability or confidence of a sell decision within the expert system. An example update uses the following formula:

New Global Confidence Factor =

Adjustment Factor + Global Confidence Factor

where

Adjustment Factor =

$$\frac{1}{\text{number of activated rules for SELL}} * \text{Global Confidence Factor}$$

Along with other hand-crafted rules (the current version of OMNI, LBS' expert system for market risk assessment constructed on the S&P 500 Index, has 268 rules), the signals generated by OMNI have resulted in a compounded annual return of 26.80% over the last six years (1987—1992) versus 14.04% generated by buy and hold on the S&P 500. For a discussion of some of the other rules employed by OMNI, see [1].

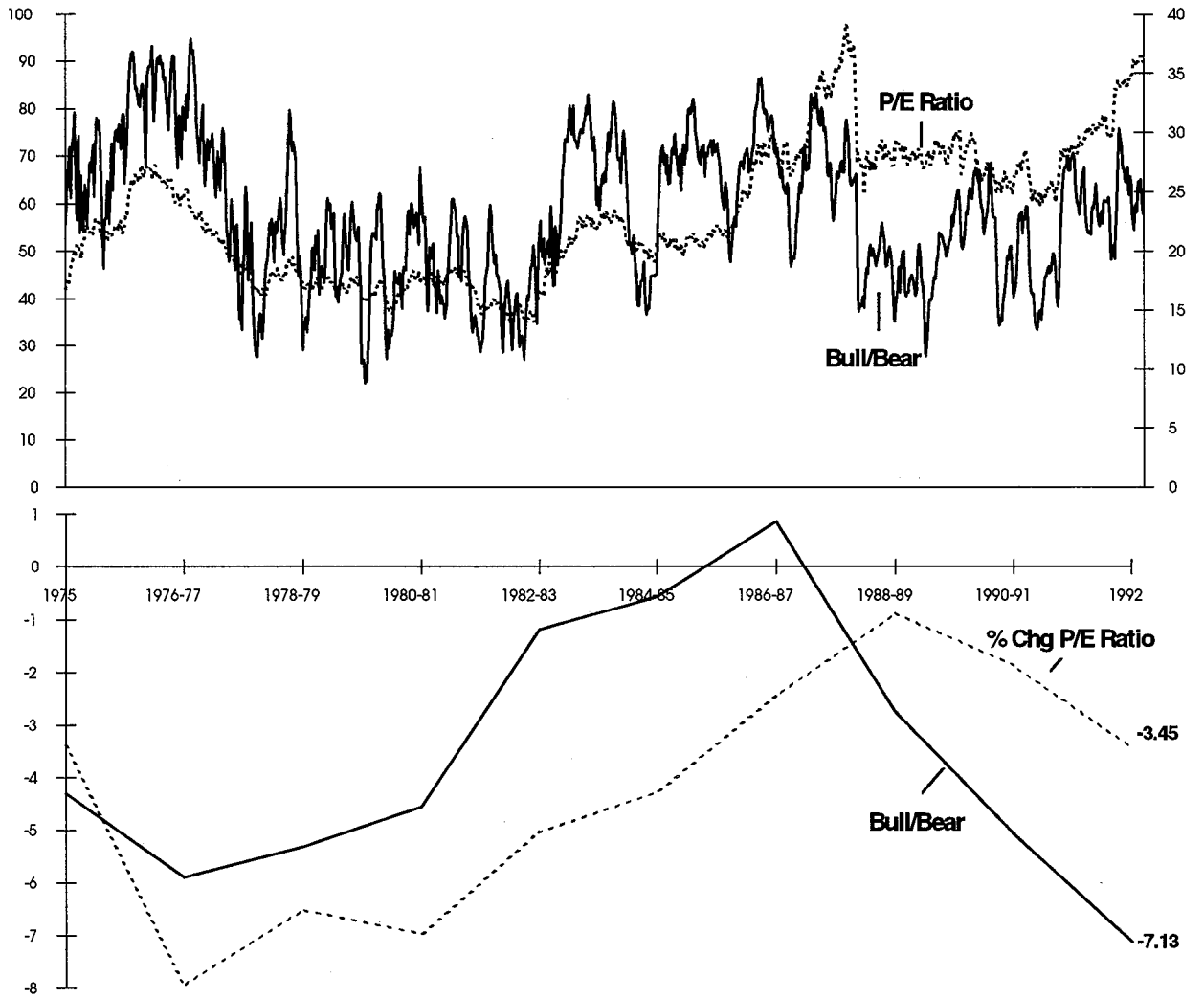


Figure 3: Examples of 2 indicators that are deemed to strongly influence the S&P 500 Index: The bottom plot shows their sensitivities over time. The top plot shows the values of the indicators for reference. The relative sensitivities (at the end of 1992) are used in constructing a rule (see text).

5. Discussion and Future Work

Financial data poses unique problems with respect to constructing predictive models. A large part of the effort in using a neural net for predicting market movements is determining which indicators to use and how their values are to be pre-processed. Using changes of prices rather than their absolute values, telescoping data to look at certain historical values and using sensitivity measures are some of the applicable techniques that we have discussed.

The search space of parameters for training the nets (e.g., learning rate, number of hidden nodes) is very large. A promising approach is the use of multiple nets (trained using different parameters with the same data) where the output of the nets is combined (also see [3]). Even a simple average of the outputs of the individual nets appears to work well in practice.

Another promising approach that we plan to focus on is endowing the net with some initial knowledge. This involves encoding approximate rules as weights in a neural net and training the net further using recent data [6]. This approach provides a way of combining a core set of rules with current data for dealing with changing market environments.

The use of recurrent nets [4] has been suggested for predicting temporal sequences, but it appears that they are not readily amenable to sensitivity analysis and extraction of approximate rules. Further research is needed to examine whether sensitivity analysis and extraction of rules can be adapted to recurrent nets which contain state information from previous time periods.

6. Conclusions

Neural nets appear to hold a great deal of promise in the investment management arena as borne out by experiments by other researchers and our models, examples of which were discussed here. Traditionally, prediction or forecasting accuracy is the only dimension along which neural networks are analyzed. We have described other perspectives from which to view the information captured by a trained neural network. These include sensitivity analysis to infer which of the inputs are deemed to be important by the net and the possibility of extracting approximate rules from a trained net. These techniques have been successfully exploited by LBS Capital Management to manage large client portfolios (totalling $300 million).

7. Acknowledgements

We would like to thank our colleague Grady Garrett for his help with the selection and pre-processing of inputs for some of the neural net models using the *CompuTrac* technical analysis package.

8. References

1) Fishman, M.B. and D.S. Barr (1991). A Hybrid System for Market Timing. *Technical Analysis of Stocks and Commodities*. Vol. 9, No. 8, August 1991.

2) Fishman, M.B., D.S. Barr and E. Heavner (1991). A New Perspective on Conflict Resolution in Market Forecasting. *Proceedings of the First International Conference on AI Applications on Wall St.*, October 1991, New York.

3) Mani, G. [1991]. Lowering Variance of Decisions by Using Artificial Neural Network Portfolios. *Neural Computation*. Vol. 3, No. 4.

4) Pineda, F. [1987]. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*. Vol. 19.

5) Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Rumelhart, McClelland, et al. (Eds.), Vol. 1, Ch. 8, Cambridge: MIT Press.

6) Towell, G.G., Shavlik, J.W. and Noordewier, M.O. [1990]. Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings of the Eighth National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, CA.

Intelligent Stock Market Prediction System Using Dual Adaptive-Structure Neural Networks

Gia-Shuh Jang, Feipei Lai

Dept. of Electrical Engineering & Dept. of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

TEL: 886-2-3635251, 886-2-3625252 Ext. 256
FAX: 886-2-3638247
E-mail: D78031@CS.EE.NTU.EDU.TW

Abstract

This paper presents an intelligent stock market prediction system that can generate timely stock trading suggestions according to the prediction of short-term trends of price movement using dual adaptive-structure neural networks. We have proposed a structure-level adaptive back-propagation learning algorithm that can automatically synthesize the structure of the neural network to fit the desired problem. The structure of the neural network can also be continuously adapted if the problem domain is changed later. We have shown that the proposed structure-level adaptive neural networks are superior to fixed-structure neural networks due to their compact size of hidden layer and improved generalization ability. Finally, in the testing period of 1990 to 1991, the annual rates of returns by using market prediction generated by the proposed system were larger than those using the buy-and-hold strategy.

1. Introduction

1.1 Stock Trading Using Neural Networks

Most stock trading decision processes involve primarily decision making by weighing evidence from various observations of the market and its environment. For example, a price trend prediction could be made by weighing evidence obtained through analyses of the monetary, political and economical fundamentals, technical indices, psychological factors, evaluation of the emergent news, and so on [4]. Besides, stock prices are often affected by ephemeral influences, including seasonal tax-motivated trading, singular international events, market psychology and the "madness of crowds." Also, the stock market is not stationary; its structure can be altered as regulatory, tax and trade policies change [3]. With these complications, the prediction accuracy of a stock trading model will be affected by the tolerance to ephemeral events influencing the market, the generalization ability of the model and the capability to continue learning when the market changes.

Several studies on the applications of neural networks on the stock trading decision processes have been proposed [5, 7, 8, 10, 16, 17]. Schoneburg analyzed the possibility of predicting stock prices on a short-term, day-to-day basis with the help of neural networks by studying three important German stocks chosen at random [16]. Fujitsu and Nikko Securities have developed a prediction system for the Tokyo Securities Exchange Stock Prices Indexes (TOPIX) [10]. This neural system is made up of several modular back-propagation neural networks that have been trained on various technical and macro economical time series data to produce as output the weighted average of weekly returns of the TOPIX. A 3-dimensional back-propagation model that simultaneously processes multiple records of time series data was proposed for capturing the temporal information in the input time series, and has been effectively demonstrated for time series forecasting by F.S. Wong [17]. This 3-D network has been applied to forecast the S&P500 weekly closing prices.

The volatility of the Taiwan stock market can be illustrated by a look at the annual turnover of the market as shown in Table 1. The highly speculative and volatile nature of the Taiwan stock market makes it a challenging target to be modeled using neural networks [7, 8].

Table 1.

Statistics of Taiwan stock market

Year	1987	1988	1989	1990	1991
Highest Daily TSEWPI*	4673	8789	10773	12495	6305
Lowest Daily TSEWPI	1063	2341	4873	2560	3316
Annual Closing TSEWPI	2339	5119	9624	4530	4600
Trading Value (US$ Billion)**	107	315	1016	761	387
Annual Turnover	2.67	2.95	5.24	4.59	2.85

* TSEWPI is the abbreviation of Taiwan Stock Exchange Weighted Price Index

** 1 USD = 25 NTD

1.2 Structure-level Adaptive Neural Networks

Recent developments in neural network theory have proven that multi-layer feed-forward neural networks with a sufficient number of neurons in one hidden layer can be used to approximate any multi-dimensional function to any specified degree of accuracy, if correct interconnection weights can be found [1]. Back-propagation algorithms are often used to determine the appropriate interconnection weights [15]. However, back-propagation algorithms that adjust the interconnection weights of fixed-structure neural networks suffer from some problems. First, the learning process sometimes gets trapped in a local minimum point of the error surface and the network can not produce the required accuracy. Second, the choice of an optimum structure of network has remained an art. For a given problem, one usually has no *a priori* information about the number of hidden units needed, so the structure of the neural network must be determined by trial and error. Third, even when a feasible structure of a neural network can be found in the beginning, if the characteristics of the problem change later, the original network structure may not be able to represent the underlying mapping between the input and output vectors and thus not permit the desired accuracy to be achieved through adaptation of interconnection weights. A straightforward solution to the problems addressed above is to design a network with a sufficiently large number of neurons in the hidden layer to handle all possible changes in the problem. This approach often leads to a network much larger than needed. Besides, a large neural network may be able to classify the training data completely but incapable of generalizing between input patterns that are minor variations of the training patterns. Furthermore, choosing an excessively large number of neurons in the hidden layer will occupy extra storage space and increase the computation time in both of the forward computation phase and the learning phase of the network [12].

Several researchers have studied the idea of *net pruning* [2, 9]. Sietsma *et al.* have proposed a method to prune a neural network with an excessively large number of neurons by examining all neurons under the presentation of the entire training set. They remove each excessive neuron and its interconnection weights that does not change state, or replicates another neuron [2]. Karnin proposed a procedure to estimate the sensitivity of the global error function to the inclusion and exclusion of each interconnection weight in the neural network. The neural network can be efficiently pruned by discarding the interconnection weight with the smallest sensitivity value [9].

The idea of learning through a dynamic network construction process that involves adjusting the interconnection weights, as well as the network structure has been proposed by several researchers [6, 13]. Hirose *et al.* presented a back-propagation algorithm that varies the number of hidden units. The structure of the neural network is adapted through adding a new hidden unit with random interconnection weights when the learning process becomes trapped in a local minimum, and deleting a randomly chosen hidden unit when the network converges to the desired accuracy [6]. Murase *et al.* proposed a modified back-propagation algorithm that adds a new hidden unit with random interconnection weights when the neural network can't converge, and discards the hidden units with the smallest amount of forward propagated signals when the neural network converges [13]. Lee proposed a general procedure for structure level adaptation for multi-layer feed-forward neural networks [12]. If after a certain period of training, the error is stabilized but is larger than a specified value, then a new neuron with interconnection weights duplicated from the neuron that contributes most to the output fluctuation will be generated. When the neural network converges, a neuron can be deleted if it is not a functioning element of the neural network or if it is a redundant element of the neural network. Lee's approach addresses the issues of when, where and how to generate or delete neurons of a neural network.

1.3 Stock Trading Using Structure-level Adaptive Neural Networks

The results presented above are encouraging and have led us to further research. A feasible structure of network found in the beginning may not be able to represent the underlying mapping between the input and output vectors and thus not permit the desired accuracy to be achieved through adaptation of interconnection weights, if the characteristics of the stock market change later. Therefore, we have examined the effectiveness of applying learning algorithms with structure-level adaptation ability to the problems of stock market prediction using neural networks.

In this paper, we are interested in the generation of stock market prediction utilizing dual adaptive-structure neural networks (*DAS net*).

In the following section, we present the stock data model used in our system. Section 3 focuses on how stock market prediction is done by the proposed dual adaptive-structure neural networks. Following that, in section 4, we will offer results of our analyses on the effectiveness of the proposed structure-level adaptive learning algorithm. A performance evaluation will be given in section 5. Finally, the conclusions will be given in section 6.

2. Stock Data Modeling

We have established the stock data model on the *DAS net* from the technical analysts' point of view. The stock data model is made by exploiting implications hidden in past trading activities by analyzing patterns and trends shown on the price and volume charts [14]. Prediction is

based on the rationale that history will repeat itself and that the correlation between price and volume reveals market behavior. It is expected that if a neural network model could be trained to simulate the experience-based intuition of a successful technician, it could result in a substantial increase in the number of stocks that could be analyzed in real time [5].

The input vector of the *DAS net* is consisted of technical indices preferred by human experts. The output vector, on the other hand, models the predictive short-term trends of price movement of the chosen stock. The discussion of retrospective input vector and predictive output vector are presented as follows.

2.1 Retrospective Feature Extraction

A technical view of a stock for the nth trading day can be defined as a 4-tuple $\underline{S}_n = <H_n, L_n, C_n, V_n>$, where n is the index of the nth trading day, H_n, L_n, C_n is the highest, lowest and the closing price of the nth trading day, V_n is the daily trading volume for the nth trading day.

Let $\underline{T}_{n,\alpha} = \{\underline{S}_i \mid i = n, n-1, ..., n-\alpha+1\}$, where α is the number of daily data $\underline{S}_n$ in time series $\underline{T}_{n,\alpha}$. Although $\underline{T}_{n,\alpha}$ is usually viewed as the retrospective characteristic of a stock for the latest α trading days, there are two shortcomings of this simple model that make it unsuitable to be used directly as input data for the neural networks. First, if $\underline{T}_{n,\alpha}$ is used as input data for a neural network, then there should be no less than 4α neurons in the input layer of the neural network. For a large α, the computation time for training the neural network will be lengthened by the large number of neurons in the neural network. Second, $\underline{T}_{n,\alpha}$ represents retrospective trading information of the latest α trading days. It has no memory of trading information out of the scope of α trading days. To extract temporal information from $\underline{T}_{n,\alpha}$, a transformation is used to figure out rates of changes of prices of consecutive trading days. We have developed a transformation F: $\underline{T}_{n,\alpha} \rightarrow \underline{R}_n$ to extract retrospective features from the time series data set $\underline{T}_{n,\alpha}$ to form a simple 16-tuple vector $\underline{R}_n$. The 16-tuple vector $\underline{R}_n = (r_n^0, r_n^1, ..., r_n^{15})^T$ is composed of sixteen technical indices chosen *a priori* by analysts. Keeping the retrospective characteristics of stock trading information in the sixteen-dimension vector $\underline{R}_n$, the number of neurons in the input layer of the neural networks is reduced from 4α to 16 by the transformation F: $\underline{T}_{n,\alpha} \rightarrow \underline{R}_n$.

2.2 Predictive Trend Modeling

The desired output vector $\underline{P}_n$ of the *DAS net* is chosen to reveal the trend of the price movement during the next six trading days. The value of traditional $\%K_n^k$ of the Stochastic Process invented by George Lane [11] reveals where the closing price of the current trading day stands relative to the retrospective fluctuation range of prices in the last k trading days. To represent where the closing price of the current trading day will stand in relation to the fluctuation range of prices for the consecutive k trading days, the FK_n^k, the forward-calculated K_n^k, is defined by equation (2-1).

$$FK_n^k = \frac{MAX_{i=n}^{n+k}(H_i) - C_n}{MAX_{i=n}^{n+k}(H_i) - MIN_{i=n}^{n+k}(L_i)} \qquad (2\text{-}1)$$

The FK_n^k, ranging from 0 to 1, is the only element of the output vector $\underline{P}_n$. Traders can take the value of the FK_n^k as the probability of a profitable trade during next k trading days from the position of buying stocks at the closing price of current trading day. A bullish trend can be identified by a very high reading of the FK_n^k (over 0.7), that puts the closing price of the nth trading day near the bottom of the price fluctuation range of the consecutive k trading days. Conversely, in down trends, the FK_n^k will have a low reading (under 0.3), and the closing price of the nth trading day will be located near the top of the price fluctuation range of the consecutive k trading days.

3. System Architecture

The problem contemplated here is to predict the trend of price movement, the FK_n^6, based on the retrospective feature vector $\underline{R}_n$. Dual modules of structure-level adaptive feed-forward neural networks are used to learn the mapping function between the input vector $\underline{R}_n$ and the desired value of FK_n^6 of the desired output vector $\underline{P}_n$. Two output vectors, the $\underline{O}_{n,S}$ and the $\underline{O}_{n,L}$, are generated by the short-term module and the long-term module of neural networks, respectively. The architecture of the proposed system is shown in Fig. 1.

3.1 Primitive Neural Network Modules

The *DAS net* used in the present system is composed

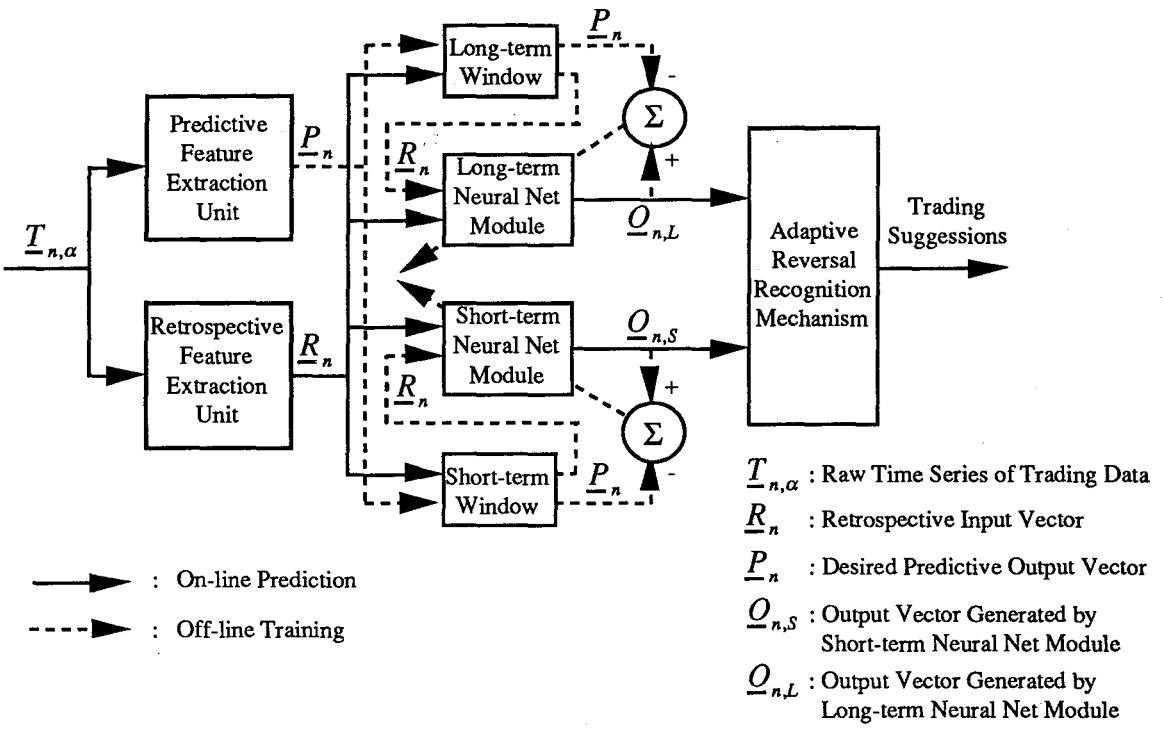


Fig. 1. System architecture

of two modules of multi-layer feed-forward neural networks. The function of this primitive neural network is to form a multi-dimensional mapping between the input vector $\underline{R}_n$ and the desired output vector $\underline{P}_n$. Each primitive network consists of three layers: the input layer, one hidden layer, and the output layer. The input layer is made up of sixteen units, each related to a projection value of the technical view of a selected stock on one of the sixteen axes spanning $\underline{R}_n$. The output layer contains only one neuron that generates the predicted value of the FK_n^6. The architecture of the primitive network is illustrated in Fig. 2.

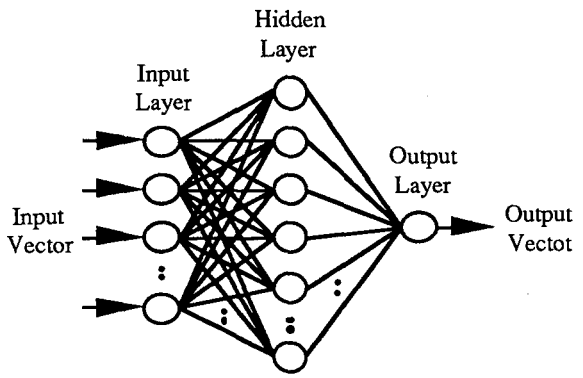


Fig. 2. Primitive neural network module of the *DAS net*

3.2 Learning Algorithm for Weight Adaptation

After the output vector $\hat{\underline{Y}}_i$ of the network is formed, it is compared to a desired output vector $\underline{Y}_i$ and the error vector $\underline{E}_i = \underline{Y}_i - \hat{\underline{Y}}_i$ is calculated. The learning of the neural network is done by adjusting the interconnection weights towards the direction of minimizing the mean square error $\varepsilon \equiv \left\langle \left\| \underline{E}_i \right\| \right\rangle$ using the back-propagation algorithm [15].

3.3 Feature Extraction and Training Scheme

3.3.1 Feature extraction

First of all, it is necessary to generate training patterns suitable for the *DAS net* architecture. Each training pattern is a 2-tuple $<\underline{R}_n, \underline{P}_n>$ which consists of the input vector $\underline{R}_n$ and the desired output vector $\underline{P}_n$. The retrospective feature extraction unit, as shown in Fig. 1, generates selected technical indices. This unit is used to establish the transformation F: $\underline{T}_{n,\alpha} \rightarrow \underline{R}_n$ that can extract retrospective features from the raw time series data set $\underline{T}_{n,\alpha}$ to form the 16-tuple input vector $\underline{R}_n$. The predictive feature extraction unit, designed according to equation (2-1), is used to

generate the desired output vector $\underline{P}_n$ of the training pattern.

3.3.2 Modeling criterion

For the output vector $\underline{\hat{Y}}_i$ generated by the *DAS net* model, the quantitative measure of the *goodness of the fit* for n training patterns is given by GF defined in equation (3-1).

$$GF = \frac{\sum_{i=1}^{n} \|\underline{Y}_i - \overline{\underline{Y}}\|^2 - \sum_{i=1}^{n} \|\underline{Y}_i - \underline{\hat{Y}}_i\|^2}{\sum_{i=1}^{n} \|\underline{Y}_i - \overline{\underline{Y}}\|^2} \qquad (3\text{-}1)$$

where $\underline{Y}_i$ is the desired output vector, and $\overline{\underline{Y}}$ is the mean of $\underline{Y}_i$ for n training patterns. Thus, GF measures the *goodness of the fit* between the *DAS net* model and actual short-term trends of the stock market in the sense that it gives the proportionate reduction in the sum of squares of deviations obtained using the *DAS net* relative to the naive predictor $\overline{\underline{Y}}$. Assume n is the number of training patterns in the training window. Then, GF can be used to evaluate the goodness of fit between n pairs of output vector $\underline{\hat{Y}}_i$ generated by the *DAS net* model and the desired output vector $\underline{Y}_i$.

The generalization mean squared error (*MSE*) between n pairs of output vector $\underline{\hat{Y}}_i$ of the network and the desired output vector $\underline{Y}_i$ is defined in equation (3-2).

$$MSE = \frac{\sum_{i=1}^{n} \|\underline{Y}_i - \underline{\hat{Y}}_i\|^2}{n} \qquad (3\text{-}2)$$

The generalization ability of the *DAS net* model can be represented by the mean squared error between n pairs of output vector $\underline{\hat{Y}}_i$ generated by the *DAS net* and the desired output vector $\underline{Y}_i$ when the *DAS net* model is tested with n test patterns.

3.3.3 Training scheme

A fixed stock market model may fail when the market behavior changes. To adapt the *DAS net* according to current market trading dynamics, a moving-window training scheme is used to tune the weights of the *DAS net* according to training data filtered by two fixed-width windows [7]. The short-term module of neural network uses a 24-day moving window to keep the latest twenty-four 2-tuple $<\underline{R}_n, \underline{P}_n>$ training patterns in the training set. This 24-day moving window can keep the short-term module sensitive to the latest changes in market behavior. The long-term module of neural network, on the other hand, concentrates on the training patterns collected from the latest seventy-two trading days. So the *DAS net* keeps track of both the short-term and the long-term views for the mapping between the retrospective technical indices and the short-term trends of price movements of the Taiwan stock market.

3.4 Structure-level Adaptation

The structure-level adaptation of the primitive neural networks is accomplished by adjusting the number of neurons in the hidden layer through two major procedures: neuron generation and neuron annihilation [12]. An auxiliary neuron randomization procedure is also included to encompass the conditions of trapping into the local minimum point. Fig. 3 shows the control flow chart of the proposed structure level adaptation techniques for multi-layer feed-forward neural networks.

3.4.1 Neuron generation

The basic criterion to generate a new neuron in the network is when the representation power of the network is insufficient. Hence, we can use the stabilized *goodness of fit* (*GF*) as an indicator to determine whether the network needs to generate a new neuron. If after a certain period of weight adaptation, the GF is stabilized, but is smaller than the desired value GF_{th}, then a new neuron can be generated to increase the discriminating power of the network. In other words, the network may have been trapped in a local minimum point. Therefore, adding a new neuron into the network will provide the network with a chance to escape from the local minimum [12].

Once a new neuron is to be generated, then it can be created with random interconnection weights [6, 13], or it can carry the same interconnection weights as one of the original neurons [12]. We use a modified version of Lee's approach to determine the interconnection weights of the newly generated neuron. Through monitoring the behavior of each neuron during the learning phase, we can find out the neuron with the least representing power and the largest influence to the final system error, hence the neuron can be used as a seed to generate the new neuron. The input interconnection weights of the newly generated neuron are duplicated from the neuron that contributes most to the output fluctuation of the network. To increase the probability of escaping from the local minimum, the output interconnection weights are generated randomly.

3.4.2 Neuron annihilation

After a certain period of weight adaptation, if the approximation GF is stabilized, and is larger than the desired value, then a neuron may be annihilated without affecting the performance of the network. For a set of training patterns, if the forward signal entropy of a particular neuron is very close to zero, then this neuron is not a functioning element in the network and it may be

annihilated without degrading the performance of the network.

3.4.3 Neuron randomization

If the stabilized GF of a network is not only smaller than a pre-defined value GF_{th}, but also smaller than the GF value before training, then we can determine that the performance of the network could not be improved through doing weight adaptation on current interconnection weights. Therefore, we introduce a weight randomization procedure into the process of structure level adaptation. When the performance of the network only gets worse after the normal weight adaptations, the input and output interconnection weights of the neuron with the largest influence to the final system error will be randomized. Then the training process can be re-started with modified interconnection weights.

The neuron randomization procedure alters the position of the standing point of the network's learning process in the hyper plane spanned by the interconnection weights of neurons. Therefore, the neuron randomization procedure may increase the probability for the learning process of the network to escape from the local minimum point.

3.5 Market Prediction and Generation of Stock Trading Decision Supports

The objectives of stock market prediction systems are extracting useful knowledge from a large-scale database containing time series price and volume data of stocks, and providing real-time stock market prediction. The retrospective feature extraction unit shown in Fig. 1 is used to extract useful feature $\underline{R}_n$ representing current market status from large databases of time series data $\underline{T}_{n,\alpha}$.

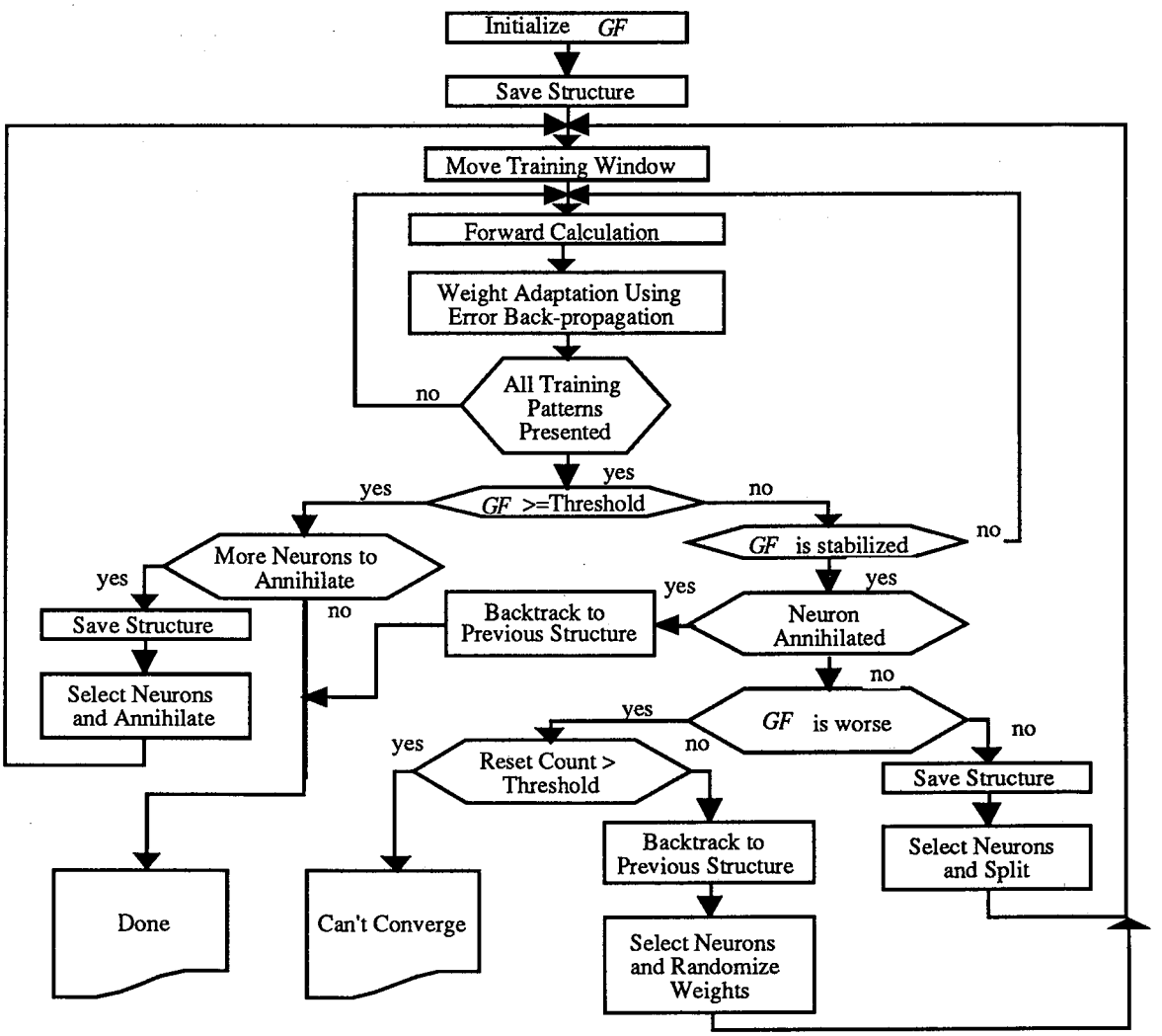


Fig. 3. Flow chart of structure-level adaptation learning algorithm

The *DAS net* architecture is used to build the correlation between the retrospective feature vector $\underline{R}_n$ representing market status and the future trends of price movements $\underline{P}_n$.

The *DAS net* can provide stock traders with the prediction of future trends of price movements. However, if we want to further automate the decision making process, it is necessary to provide stock traders with timely and accurate stock market prediction such as the recommendations for buying, holding, or selling certain kinds of stocks at certain times. We have developed an adaptive mechanism that can recognize reversals of price trends and generate stock trading recommendations for buying, holding, or selling according to the trend prediction generated by the *DAS net* [7].

4. Analysis

The 2-tuple $<\underline{R}_n, \underline{P}_n>$ patterns collected from 1987 to 1989 are used as the initial training set, while those collected from 1990 to 1991 are used as the test set. The number of neurons in the hidden layer required to accurately generate the predictive output is automatically adjusted using the structure-level adaptation algorithm. While there are six neurons in the hidden layer of the short-term module of the *DAS net*, the long-term module of the *DAS net* has eleven neurons in its hidden layer. To test the hypothesis that neural networks with extra neurons in the hidden layer do not generalize well, we use the generalization *MSE*, the mean of the squared differences between the outputs of neural network and the desired short-term trends of daily TSEWPI over the test set of 1990 to 1991, to evaluate the performance of neural networks with different structures.

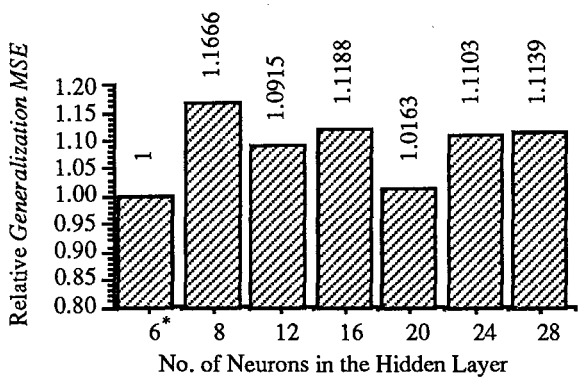


Fig. 4.a. Comparison of relative *generalization MSE* of neural networks with different number of neurons in the hidden layer (The *generalization MSE* of the short-term module of the *DAS net* with 6 neurons in the hidden layer is used as the comparison basis.)

In Fig. 4.a, the relative generalization *MSE* of predicting the TSEWPI from January 1990 to December 1991 using neural networks with 8, 12, 16, 20, 24 and 28 neurons in their hidden layer are compared to that of the short-term module of the *DAS net*. Fig. 4.b shows the generalization *MSE* of neural networks with 12, 16, 20, 24, 28 and 32 neurons in their hidden layer compared to that of the long-term module of the *DAS net*.

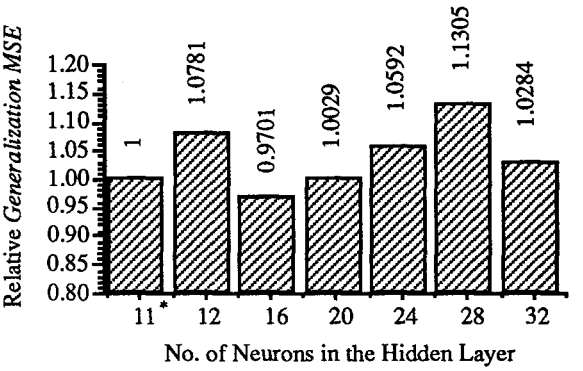


Fig. 4.b. Comparison of relative *generalization MSE* of neural networks with different number of neurons in the hidden layer (The *generalization MSE* of the long-term module of the *DAS net* with 11 neurons in the hidden layer is used as the comparison basis.)

From both figures we can see that the structure-level adaptable neural network with feasible number of neurons in its hidden layer generalizes better than other neural networks with extra neurons in the hidden layers. This phenomenon reveals the significance of the overfit problem of the neural networks when they are used to model the very complex stock trading dynamics. Furthermore, the feasibility of applying structure-level adaptation algorithm on the *DAS net* can be justified by the characteristics of smaller size of neural network, faster computation, and less generalization *MSE*.

5. Performance Evaluation

The performance of the intelligent stock market prediction system based on the *DAS net* is examined by simulating the buying and selling for the TSEWPI of the Taiwan stock market from 1990 to 1991 on the Sun SPARC Station 2 workstation. In this simulation, the initial training set is composed of the 2-tuple $<\underline{R}_n, \underline{P}_n>$ patterns generated from daily TSEWPI data collected from 1987 to 1989, while the test set consists of the daily TSEWPI data collected from 1990 to 1991. Some annual statistics of the TSEWPI, recorded from 1987 to 1991, are shown in Table 1 to illustrate the characteristics of the Taiwan stock market. The $MA_k(X_n)$ defined in equation

(5-1) is the k-day moving average of time series variable X_n of the nth trading day.

$$MA_k(X_n) = \frac{1}{k}(X_n) + \frac{k-1}{k}MA_k(X_{n-1}) \qquad (5\text{-}1)$$

The algorithm for generating buying and selling signals using the *DAS net* is listed as followed.

```
if (MA3(FK6n-1) < lower threshold of BUY and
MA3(FK6n) > upper threshold of BUY)
then {
    if (BUY existed)
    then
            HOLD;
    else
            BUY;
}
if (MA3(FK6n-1) > upper threshold of SELL and
MA3(FK6n) < lower threshold of SELL)
then {
    if (BUY existed)
    then
            SELL;
}
otherwise {
    HOLD;
}
```

The one-point buying and selling strategy is used in this simulation. One-point buying and selling means all available money is used to buy stocks at a particular time, and also means that all stocks held are sold all together.

The annual performance of buying and selling the TSEWPI from 1990 to 1991 is shown in Table 2.

All the rates of returns shown in Table 2 are calculated after taking the transaction cost of one percent for each transaction into consideration. To evaluate the quality of the proposed stock market prediction system, two categories of criteria, namely, the profitability and the consistency, are taken into consideration and the results are listed in Table 2.

The category of profitability criteria is composed of the total number of trades over the test period, the number of profitable trades, the percentage of profitable trades out of total number of trades, the average rate of return per profitable (unprofitable) trade, the average rate of return per trade, and the total rate of return over the test period.

The category of consistency criteria consists of the standard deviation of the rate of return per trade, the Sharpe ratio that can be calculated as the average rate of return divided by the standard deviation, the maximum rate of return per profitable (unprofitable) trade, the maximum draw-down that is the cumulative rate of returns of successive unprofitable trades.

Table 2.

Performance evaluation

Period	1990		1991	
Type of Neural Network	Fixed-structure Neural Networks	*DAS net*	Fixed-structure Neural Networks	*DAS net*
Total Number of Trades	13	4	5	4
Number of Profitable Trades	3	2	3	3
Percentage of Profitable Trades (%)	23.08%	50.00%	60.00%	75.00%
Average Rate of Return per Profitable Trade (%)	+ 20.81%	+ 22.23%	+ 13.40%	+ 14.77%
Average Rate of Return per Unprofitable Trade (%)	- 8.78%	- 8.93%	- 6.20%	- 13.75%
Average Rate of Return per Trade (%)	- 1.95%	+ 6.65%	+ 5.56%	+ 7.64%
Std. dev. of Rate of Return per Trade	0.1200	0.1608	0.1204	0.1035
Sharpe Ratio	- 0.1625	0.4138	0.4621	0.7380
Maximum Rate of Return per Profitable Trade (%)	+ 27.95%	+ 38.21%	+ 32.22%	+ 25.84%
Maximum Rate of Return per Unprofitable Trade (%)	- 14.92%	- 12.67%	- 8.10%	- 13.75%
Maximum Draw-down (%)	- 80.33%	- 17.85%	- 8.10%	- 13.75%
Annual Rate of Return (%)	- 30.66%	+ 21.61%	+ 25.76%	+ 29.20%
Annual Rate of Return of China Growth Fund (%)	- 39.52%		+ 23.42%	
Annual Rate of Return of Kwang Hua Growth Fund (%)	- 41.13%		+ 3.75%	
Annual Rate of Return of NITC Fu Yuan Fund (%)	- 45.21%		+ 28.30%	
Annual Rate of Return of Citizen Fund (%)	- 42.71%		+ 9.86%	
Annual TSEWPI Change (%)	- 52.93%		+ 1.56%	

Rather than being a real object for trading listed on the Taiwan Stock Exchange (TSE), the TSEWPI is a weighted index of prices of stocks of significant companies listed on the TSE. Therefore, buying and selling of the TSEWPI is equivalent to buying and selling a portfolio consisting of stocks of all significant companies listed on the TSE, of which the ratio of the investment amount on stocks of each company is the same as the weighting factor of its contribution to the calculation of the TSEWPI. The annual rates of returns obtained by two other trading strategies are also listed in Table 2. The annual rates of returns of the four closed-end funds issued in Taiwan are used to show the performance of trading decisions made by human experts. All of the four funds are domestic securities investment trust funds for domestic investors and invest mainly in stocks listed on the TSE. The principle objective of the funds is capital growth through investments in listed firms, subject to certain restrictions under the "Securities Investment Trust Contract" and related regulations. For example, the annual turnover of each fund is not allowed to exceed 60% of the annual average turnover of all stocks listed on the TSE. Under this restriction on the turnover of the funds, fund managers can not trade as frequently as using our one-point buying and selling strategy would require. However, they can increase the rates of returns by adjusting the portfolio to include stocks with high expected returns.

The annual rate of change of the TSEWPI is used to represent the raw annual rates of returns of trading on the TSEWPI according to the buy-and-hold strategy.

The performance of the proposed system is better than that of fixed-structure neural networks. Furthermore, the proposed system outperforms all the closed-end funds in the testing period of 1990 to 1991. According to Table 2, it is obvious that the annual rates of returns after the transaction costs by using the trading decisions generated according to the weighted output of the *DAS net* are larger than those using the buy-and-hold strategy over the testing period from 1990 to 1991. We can conclude that the intelligent stock market prediction system using dual-module neural networks can outperform the simple buy-and-hold trading strategy.

6. Conclusion

Dual adaptive-structure neural networks that can predict the short-term trends of price movement and generate trading suggestions, are utilized to develop an innovative, intelligent and profitable stock market prediction system for the Taiwan stock market. Transformations used to identify both retrospective and predictive features from raw data gathered from the market have also been presented.

Reinforcing the temporary correlation between the neural weights and the training patterns, structure of the *DAS net* is self-synthesized according to training patterns collected from windows of different sizes. The proposed adaptation techniques also allow the *DAS net* to continuously adapt its structure to follow statistical changes in the problem domain. We have justified the effectiveness of the proposed structure-level adaptive neural networks by comparing their generalization ability with that of fixed-structure neural networks. Finally, in the testing period of 1990 to 1991, the annual rates of returns by using market prediction generated by the proposed system are not only larger than those generated by using the buy-and-hold strategy, but also superior to all the closed-end funds managed under different disciplines by human experts. Due to the feature of above average rate of returns shown in the performance evaluation, an intelligent stock market prediction system can be realized using the structure-level adaptive neural networks.

Acknowledgment

The authors would like to thank the TAIWAN FUJI XEROX FOUNDATION for granting us a distinguished research award. We are indebted to Jack J.Y. Yeh, president of NATIONAL INVESTMENT TRUST CO., LTD., and Thomas Huang, manager of planning department of JIH SUN SECURITIES CO., LTD., for their technical discussions and criticisms of early versions of this system. This work is supported in part by the National Science Council under Grant No. NSC 82-0301-H-002-100.

References

[1] Cybento, G., "Approximation by Superpositions of a Sigmoidal Function," Mathematics of Control, Signals, and Systems, Springer-Verlag New York Inc. 1989.

[2] Dow, J.F., Sietsma, J., "Creating Artificial Neural Networks That Generalize," *Neural Networks*, Vol. 4, 1991, pp. 67-79.

[3] Elder, J.F. IV and Finn, M.T., "Creating 'Optimally Complex' Models for Forecasting," *Financial Analysts Journal*, January-February, 1991, pp. 73-79.

[4] Felsen, J., "Learning Pattern Recognition Techniques Applied to Stock Market Forecasting," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 5, No. 6, 1975, pp. 583-594.

[5] Hawley, D.D., Johnson, J.D. and Raina, D., "Artificial Neural Systems: A New Tool for Financial Decision-Making," *Financial Analysts Journal*, November-December, 1990, pp. 63-72.

[6] Hirose, Y., Yamashita, K., Hijiya, S., "Back-Propagation Algorithm Which Varies the Number of Hidden Units," *Neural Networks*, Vol. 4, 1991, pp. 61-66.

[7] Jang, G.S., Lai, F., Jiang, B.W. and Chien, L.H., "An Intelligent Trend Prediction and Reversal Recognition System Using Dual-Module Neural Networks," *Proc. First International Conference on Artificial Intelligence Applications on Wall Street,*

New York, U.S.A., 1991, pp. 42-51.

[8] Jang, G.S., Lai, F., Jiang, B.W. and Chien, L.H., "An Intelligent Stock Portfolio Management System Based on Short-term Trend Prediction Using Dual-Module Neural Networks," *Proc. International Conference on Artificial Neural Networks*, Finland, 1991, pp. 447-452.

[9] Karnin, E.D., "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 1, No. 2, June 1990, pp. 239-242.

[10] Kimoto, T., Asakawa, K., Yoda, M. and Takeoka, M., "Stock Market Prediction System with Modular Neural Networks," *Proc. IEEE International Joint Conference on Neural Networks*, 1990, pp. 1-6.

[11] Lane, G.C. , *Trading Strategies*, Future Symposium International, 1984.

[12] Lee, T.C., *Structure Level Adaptation for Artificial Neural Networks*, Kluwer Academic Publishers, 1991.

[13] Murase, K., Matsunaga, Y., Nakade, Y., "A Back-Propagation Algorithm Which Automatically Determines the Number of Association Units," in *Proc. IEEE International Joint Conference on Neural Networks*, 1991, pp. 783-788.

[14] Murphy, J.J., *Technical Analysis of the Futures Markets, A Comprehensive Guide to Trading Methods and Applications*, New York Institute of Finance, 1986.

[15] Rumelhart, D.E., McClelland, J.L. and the PDP Research Group, *Parallel Distributed Processing Explorations in the Microstructure of Cognition. Volume 1: Foundations*, The MIT Press, 1986.

[16] Schoneburg, E., "Stock Price Prediction Using Neural Networks: A Project Report," *Neurocomputing*, 2, 1990, pp. 17-27.

[17] Wong, F.S., "Time Series Forecasting Using Back-propagation Neural Networks," *Neurocomputing*, 2, 1991, pp. 147-159.

Paper Session: Trading Workstation Support

Chair: Yuval Lirov, Salomon Brothers

SIRIUS:

SWIFT's Intelligent Resource for International User Support

L. J. Thomae, D. Mukherjee, and R. Phelps

S.W.I.F.T.
P.O. Box 2005
Culpeper, Virginia 22701

Abstract

The SIRIUS project (S.W.I.F.T.'s Intelligent Resource for International User Support) is an intelligent, multifaceted expert system that will improve the productivity of S.W.I.F.T.'s User Support (or help desk) units. This will be done by lessening their dependency on other units within S.W.I.F.T.

SIRIUS will also assist the support staff in improving the quality and consistency of response to user problems and queries as well as providing proactive diagnosis of these problems. With SIRIUS, the Support Centers will become proactive as the expert systems within SIRIUS will warn the support coordinators when a customer is potentially having a problem. The Support Center can then handle the problem before it is realized by the customer.

Background and Need

S.W.I.F.T. (Society for Worldwide Interbank Financial Telecommunications) headquartered in La Hulpe, Belgium, operates a highly sophisticated, computerized international telecommunications network that provides members with fast, responsive and secure automated transmission of financial transactions.

Currently, S.W.I.F.T. has over 3,000 live users, mainly banks, in 89 countries, and processes well over 1.8 million transactions a day. Because of the high number of users and transactions, the User Support (or help desk) areas within S.W.I.F.T. are critical to the success of the company. There are currently three User Support Centres within S.W.I.F.T.: one located in the United States (in Culpeper, Virginia), one in The Netherlands, and a backup disaster center in Belgium.

In a support environment, and especially in a financial telecommunications company, one is faced with numerous telephone calls into the support area each day. These calls are from customers who are having problems using the S.W.I.F.T. network. Customer problems range from communications trouble at the user's end, difficulties at S.W.I.F.T.'s end, problems at an intermediate carrier's point, or simply informational queries such as "What is the currency code for Brazil?"

Since S.W.I.F.T. is a telecommunications company, one can obtain information about the network by way of "events". These events give information about the state of the network and any known network problems involving customers or S.W.I.F.T. components. For instance, an event is generated each time a user successfully logs on the network, is aborted off the network, or enters an incorrect password or command. These events can then be captured and fed into an expert system with appropriate reasoning knowledge.

SIRIUS Overview

SIRIUS is an intelligent, productivity enhancement project for S.W.I.F.T.'s User Support Centers. SIRIUS is comprised of real-time deductions, events processing, an event

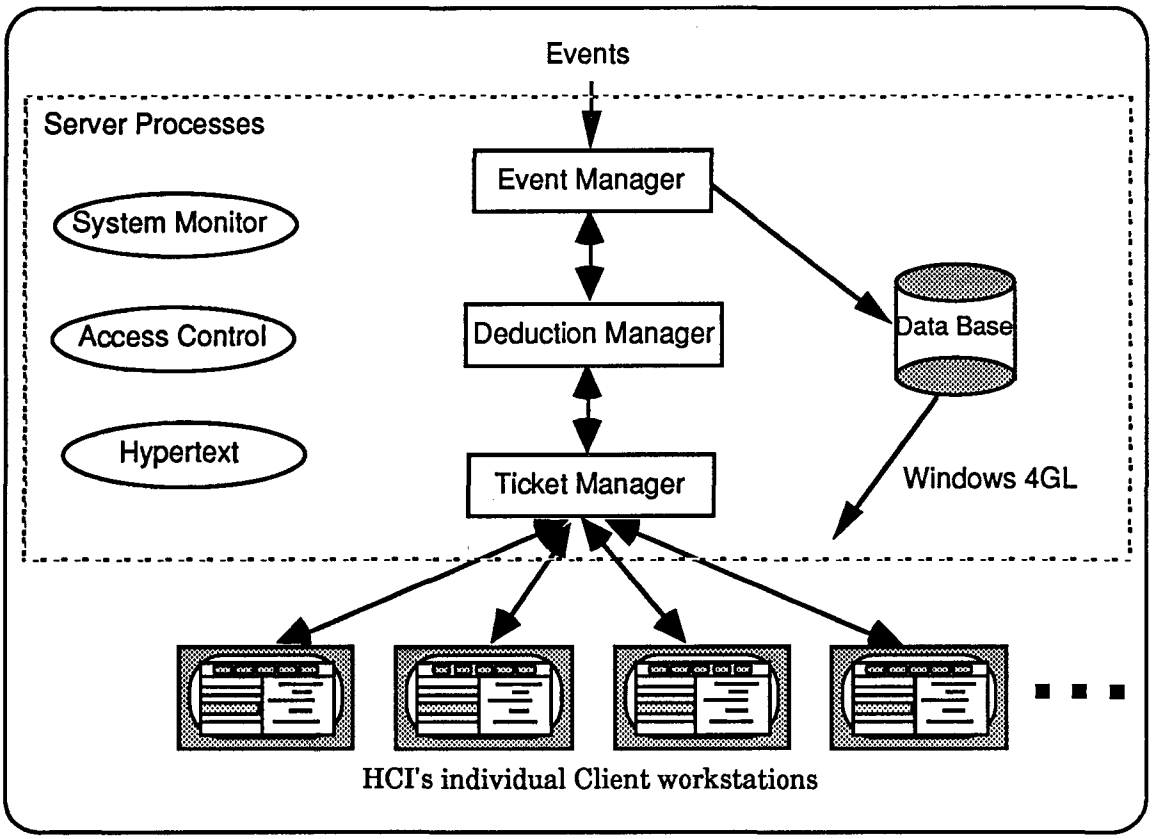


Figure 1. Simplified Flow of SIRIUS

manager, Hypertext manuals, electronic messaging calendars, a scheduler, a front-end interface to S.W.I.F.T.'s internal E-mail and problem management systems, a system monitor, a database for storage and retrieval, and a graphical user interface (figure 1). This paper will concentrate on the two expert system components of SIRIUS: the Deduction Manager (DM) and the Ticket Manager (TM).

SIRIUS Connectivity

The SIRIUS system is connected to the S.W.I.F.T. network via ANDES, a real-time network monitoring expert system which packages the events in the form of text strings. SIRIUS receives the S.W.I.F.T. network events into the SIRIUS server (figure 2). The SIRIUS server processes the received information, requests more information if needed, and sends the packaged information along to an Event Manager module within SIRIUS. The Event Manager parses the events and sends the parsed events off to the SIRIUS Deduction Manager. The Deduction Manager performs temporal reasoning and deductions on the events and produces "tickets" that are sent to the Ticket Manager module in the system. The Ticket Manager then sends the encapsulated ticket information to the SIRIUS workstations in the support area.

Each SIRIUS workstation is also connected to the company's E-mail system. The SIRIUS server is connected to S.W.I.F.T.'s Problem Management System (PMS) in order to automatically send problem statements based on the deductions and data that SIRIUS derives.

Inference Engines

SIRIUS contains two inference engines and knowledge bases. One knowledge base deals with the parsed events (the Deduction Manager or DM) and the other expert system deals with handling the tickets and the cooperative Groupware nature of SIRIUS (the Ticket Manager or TM). As each process, or module, within SIRIUS is totally independent of each other module, the DM and TM modules

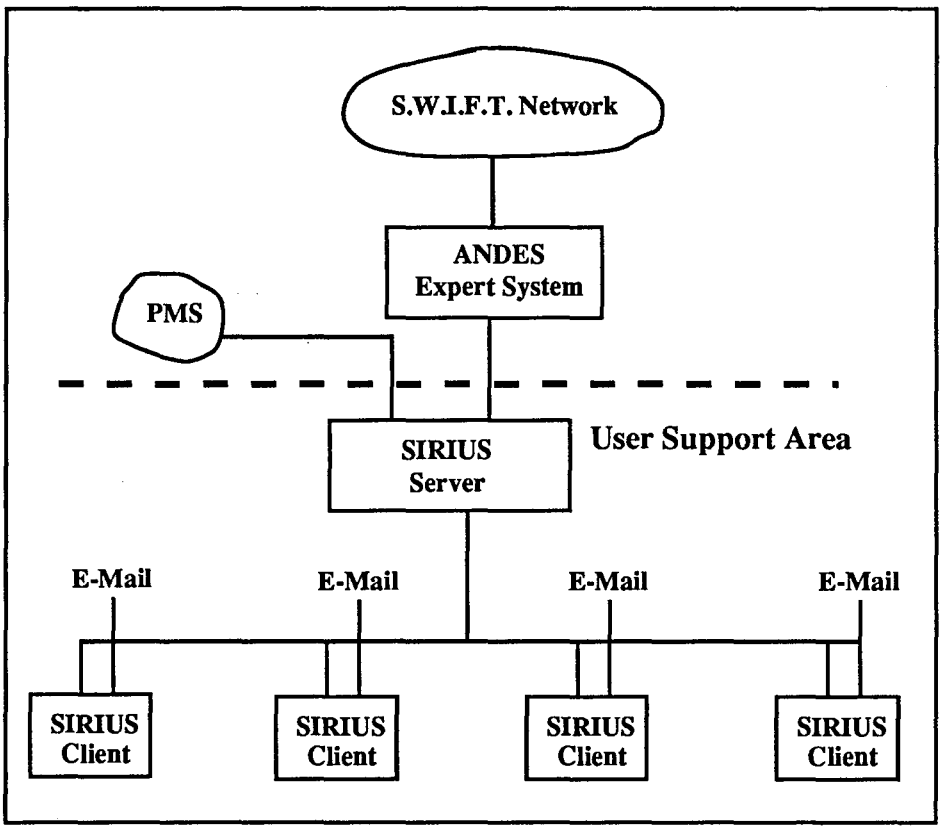


Figure 2.

communicate only via SIRIUS's custom IPC mechanism.

SIRIUS' IPC

SIRIUS uses a custom IPC (Inter-Process Communication) protocol that was built on top of Talarian's RTworks server. This IPC allows standardized structures to be passed from one SIRIUS module to another SIRIUS module independently of where the modules are located.

These messages contain the requesting datagroup (a particular HCI, TM, DM, etc.), the message type (for instance Ticket_update_msg), and the structure containing the message fields (figure 3).

Each module or process "registers" for messages, and each receives only the messages intended for the requester. This allows SIRIUS the flexibility of having its modules register for the messages of particular interest without worrying about the sender of the message.

Deduction Manager

The Deduction Manager (DM) is the portion of SIRIUS which performs deductions on the individual S.W.I.F.T. events and responses. This is the portion of SIRIUS that can be said to "reason" about the S.W.I.F.T. events and the ANDES nodal statuses.

The Deduction Manager specifically addresses the pro-active goal in SIRIUS to provide tools that will assist the Support Center coordinator in having an accurate prognosis of the customer's problem so that the coordinator will be able to provide the most accurate assistance possible.

As previously stated, one of the goals of the Deduction Manager is to allow S.W.I.F.T. to become more proactive in dealing with customer needs. If there is a problem impacting customers, the support coordinators should be aware of the problem prior to a customer calling.

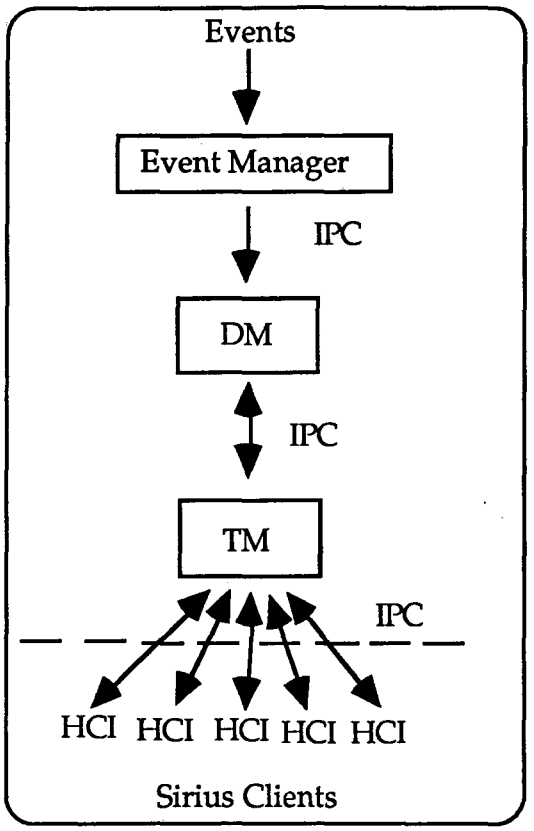


Figure 3.

To illustrate how SIRIUS will assist the coordinator, an example of a customer security violation will be used.

A S.W.I.F.T. user is only allowed to try to log into the S.W.I.F.T. system incorrectly X number of times. For security reasons the next (X + 1) time the user tries to incorrectly log into the system the user is inhibited (or locked out of the system) and needs to call S.W.I.F.T. to be reinstated. Currently, coordinators only know when a customer is inhibited when the customer reports the log in problem.

To assist with this problem SIRIUS, will automate the tracking of the number of times a customer logs in incorrectly and display a message, or "ticket", to the support coordinator when a customer has attempted to incorrectly log in the X + 1 time. The coordinator will then have the ability to immediately call the customer to investigate whether it is perhaps a security violation.

Implementation

The Deduction Manager is comprised of two parts: the internal model of the deducted flows and the high level rules of the deduction. The internal model and data structures used in performing deductions are kept in C in memory. The high level reasoning is contained in RTworks rules.

The deduction knowledge can be broken down into object types. There is a model of each bank or component. This is kept in shared memory (as well as in an Ingres database for crash recovery). The internal ticket structure contains an array of all possible tickets, or flows; bank objects on each flow for which a possible problem is relevant; symptom objects for each symptom, and finally ticket objects once the DM has concluded that a problem has occurred. By dynamically creating these objects as needed memory consumption is kept to a minimum (figure 4).

The knowledge in the Deduction Manager consists of domain dependent deductions concerning the events that are entering the SIRIUS system. These events are then reasoned on by using domain knowledge that was elicited from expert coordinators in the S.W.I.F.T. User Support area. The knowledge consists of what events or combination of events constitutes a problem or area of concern for the S.W.I.F.T. users and what is the appropriate action to take to resolve or rectify the problem(s). In some cases, for instance too many incorrect log in attempts, the correct solution is to call the user to verify that a security breach has not occurred. In other cases, a command must be sent to the S.W.I.F.T. network to resolve the problem. If a node needs to be activated, a command is sent out on the network to activate the node in the system.

The Deduction Manager also uses temporal reasoning to determine possible problems and outcomes. Each possible event, or symptom, for each ticket or flow type, contains a relative period of time for which the symptom is valid. For instance a flow might state that 5 NAK events are required within a 10 minute period, in order to state that a customer is having a problem of too many NAKs.

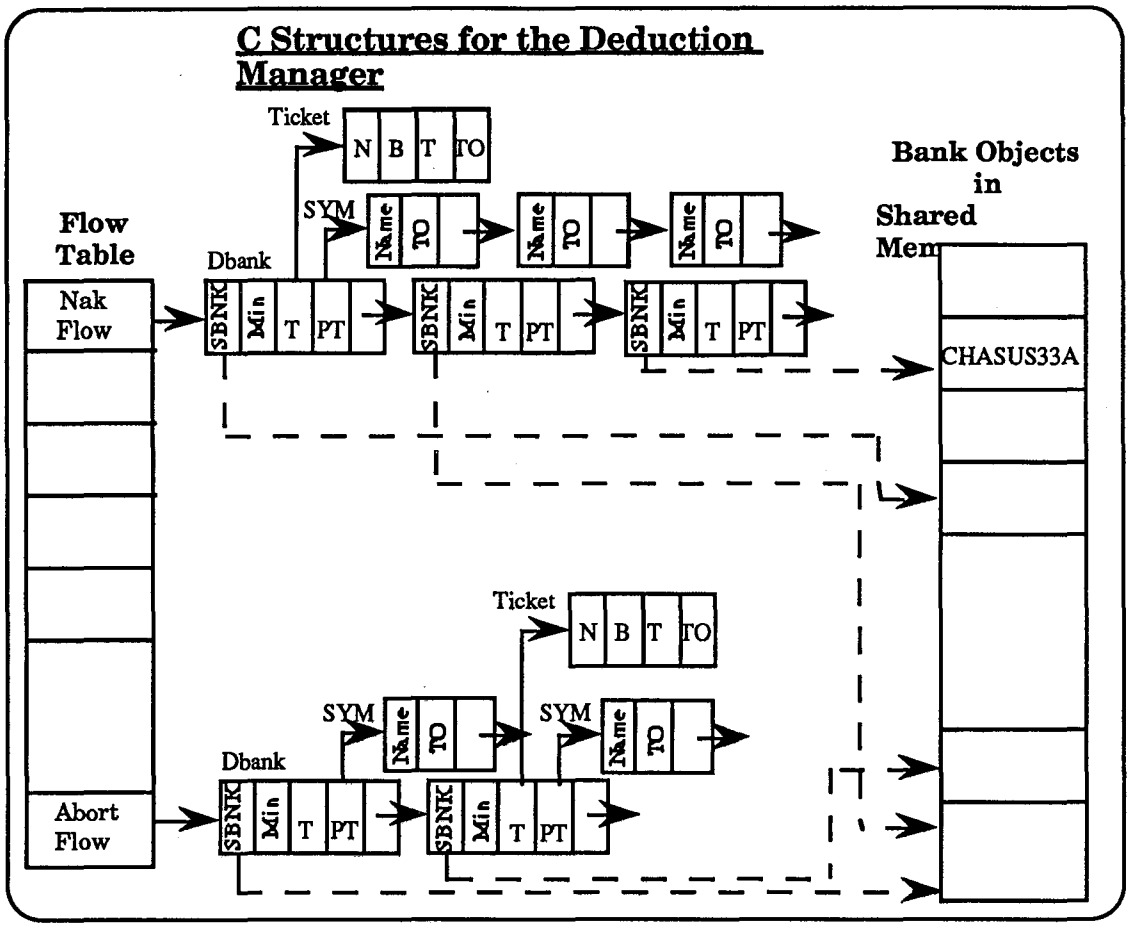


Figure 4

In addition to trapping the incoming S.W.I.F.T. events, SIRIUS, through the Deduction Manager, elicits information from the S.W.I.F.T. network. If the Deduction Manager feels that more information is needed to conclude that a problem has occurred, it will send out a command to the S.W.I.F.T. network requesting a report or more information than what was originally spontaneously generated by the system. The Event Manager then parses the returned information and sends it the Deduction Manager which will judge whether or not a problem is occurring, or if still more information is needed.

Ticket Manager

The Ticket Manager (TM) is the portion of SIRIUS which handles the deducted tickets. The DM communicates with the TM to send deduced tickets and the TM also communicates with all the SIRIUS HCIs in a groupware fashion.

The Ticket Manager (TM) is responsible for maintaining the set of open tickets known to SIRIUS. Tickets are generated either automatically by the Deduction Manager, or manually by the coordinators seated at the SIRIUS workstation. When a ticket is generated, the TM stores the ticket in Ingres for backup purposes and then makes the ticket available to coordinators by sending it to all the SIRIUS HCI's for display.

The Ticket Manager is written using a combination of the Talarian inference engine, rules, and C code. As each "ticket" is received by the Ticket Manager, the TM dynamically creates a ticket object. Each ticket object contains attribute slots to hold coordinator comments, commands sent to "solve" the problem, the

history of the ticket, the current state of the ticket, the "owner" of the ticket, etc. Rules and code fire off of and manipulate the "ticket" objects. Once a problem is solved, the ticket object is deleted from the TM and a message is sent to the DM to allow the it to remove the Deduction Manager ticket information.

Once displayed on a SIRIUS screen, a ticket is available for a number of operations by coordinators, including expanding the ticket, commenting the ticket, combining tickets, closing a ticket, sending the ticket off to the problem management system, printing a ticket, and updating a ticket with new information. It is the responsibility of the TM to coordinate these operations from multiple simultaneous coordinators while ensuring the integrity of the tickets.

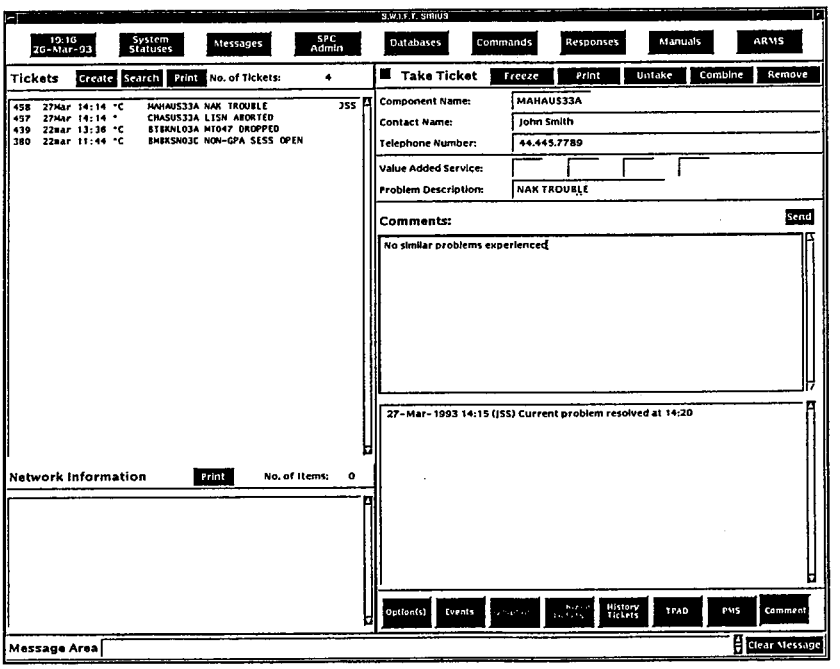
If a coordinator wants to display information about a ticket, the ticket is clicked on from a SIRIUS screen, an IPC message is sent to the Ticket manager stating that coordinator A would like information on Ticket X, the Ticket Manager sends back the information requested to the requesting HCI (the other HCI processes are only sent information stating the coordinator A has requested information on Ticket X).

The Ticket Manager also handles the HCI processes that join the SIRIUS system at staggered points in time. When an HCI first comes up (after rebooting of an individual workstation, for example), the HCI sends a message to the Ticket Manager requesting all current tickets. The Ticket Manager then sends back a message containing all current open tickets, and whether they are currently being handled by a coordinator or not. In this manner HCI's are always kept up to date with other HCI's.

Design Philosophy

The goal of the SIRIUS design is to hide the user from the architectural aspects of SIRIUS. The user need not be aware that there are separate HCI processes, or that there is an overall Ticket Manager or Deduction Manager. This philosophy encompassed one of the reasons the DM and TM were separated into two separate modules and two inference engines to cleanly distinguish between domain specific information (the DM rules), and a generic groupware ticket handler (the TM rules). When domain specific updates are needed, only the DM will need to be altered.

A View of the SIRIUS screen



Current Status and Conclusions

User Support currently works in a reactive mode. Coordinators answer incoming phone calls from the S.W.I.F.T. user community about a wide range of problems: from the simple queries concerning an error code or correct message format, to time consuming, complicated problems about an inability to connect to the S.W.I.F.T. telecommunications network. A user calls with a problem and the support coordinators try to resolve the problem using a variety of off-line programs, status reports, operation manuals, and learned expertise. One of the main goals of SIRIUS is to allow the coordinators to be proactive: to call the user and offer help before the user places a call to S.W.I.F.T.

SIRIUS is being written on HP workstations using Talarian's RTworks real-time expert system shell, C, UNIX processes, Ingres, and MOTIF. The system will be delivered to all three User Support Centers in September of 1993. A working prototype has been completed, and the project is in phase II and on schedule. Two AI engineers, three software engineers, and two User Support coordinators are assigned full time to the project to ensure that SIRIUS will, in fact, meet the needs of the S.W.I.F.T. User Support Centers.

ACKNOWLEDGEMENTS

E. Rajendiran, M. Manohar, T. Patel, and R. Chekaluk are gratefully thanked for their work on the SIRIUS project; J. Southard for his User Support expertise and S. Brady for her administrative assistance.

Intelligent Broker Station of the Near Future, A Rule-based Expert System.

Bernard Viau
Trois-Rivieres University
P.O. Box 5537, Station B, Montreal
P. Quebec, Canada, H3B 4P1

Abstract

Having in mind what should be an intelligent broker station aimed at the retail broker, we discuss the various sources of information and the way the system may address them. The brain of such a system must be a set of rules using the just-in-time manufacturing concept to control the broker's station. We use fuzzy logic terminology along with these rules. The final remarks discuss the concept of artificial intelligence and it's implications on broker's stations.

PART 1: WHAT SHOULD AN INTELLIGENT BROKER STATION BE?

With today's softwares, a financial advisor can do portfolio analysis, technical analysis and client prospecting. He has access to real-time quotes, Dow Jones news, research papers online as well as fondamental data on selected companies. Most firms also use electronic mail and have online databases on the clients. Problem is, all these systems and databases are unrelated so that the financial advisor has to learn several query languages and typically will make decisions without the full picture.

We estimate that 60% of the tasks described in this paper are already done by some systems, though not one company uses them all. More, these are not so intelligent systems because one must constantly input data and exchange informations with other programs.

An intelligent broker station must be able to think for itself. It must be able to interract with all the databases that a financial advisor might use daily. It must also be able to scan the market for trading opportunities and, whenever necessary, alert the financial advisor of a particular situation. The station, which we nickname Igor, must also be able to respond very fast to any request. A maximum of five seconds would be acceptable for most situations. The interface must be more than user-friendly, it must be childish. Typical financial advisors do not a lot of time to spend on learning, they are paid with commissions.

To monitor correctly the market, Igor is equipped with a set of rules concerning the economy, the fondamental and the technical. Decisions must be taken using the three aspects. For bonds, he should be capable, using rules, to select the best international bond market. If the client's objectives prohibit international investment, he must be able to think with the interest rate structure and suggest a switch between short and long term. For options and futures, he must be capable, using analysis from the mar-

ket, the fondamental and the technical aspects as well as studies on the volatility to think in term of buy write, bull spreads, bear spreads, tripod and so on and so forth. He must also think of the possibility to use convertible instruments whether warrants or convertible debentures.

The system has to be very fast because there will be a lot of number crunching involved. Access to large databases must be lightning fast. All modules must be made transparent so that information can flow between them. Rules must be developped to insure that good trading opportunities are found and that the financial advisor is aware of whatever he needs to know at the right time. We need access to internal electronic mail system as well as external capabilities. Finally, Igor will be able to talk to the financial advisor and to receive vocal orders; he must be capable to send orders directly to the floors when the financial advisor takes the decision; he must also be able to explain his reasoning as well as to learn from his mistakes in consultation with the house experts.

PART 2: WHERE DOES THE INFORMATION COME FROM?

Seven modules will pre-digest the information Igor needs to make decisions. These modules, in fact expert systems or neural systems themselves, will manage informations from: clients, fundamental analysis, technical analysis, derivatives instruments and market analysis, compliance and operation, fixed-income markets and currencies markets.

Inputs will come from four sources each with their own communication interface: news from Dow Jones News Retrieval or another system, live data feed from Reuters or another source, head office bond desk feed and finally, the broker's own input using a keyboard, a touchscreen, a tablet or simply his own voice.

2.1 Clients module

To organize the information on clients, we need more than a database management system, we need an expert system. Igor will have access to information as contained in client's application forms for all list processing activities but also to monitor investment's objectives. When opening a new account, he must decide whether documents are needed (special corporate accounts) and could also check credit ratings automatically. New account would be filed electronically with the head office with a scanning device. He must have access to prices in order to update portfolio's which must be evaluated every month, every night or live on demand. He must have whenever possible information on electronic addresses for those clients who wish to be kept in touch with their positions. Electronic account with service providers as Compuserve, Alex (Canada), Minitel (France) or videoway connections could be provided and partially paid for by the brokerage house for their best clients.

Prospecting can also be taken care of by the system and some programs already exist in the market which do that very well indeed.

Every night, the system would scan client's portfolios looking for trading opportunities: bonds renewal, profit taking situation, new issues which may be of interest for a particular client, or invest-

ment's objectives unrespected. Whenever an investment opportunities arises, the module would classify it in terms of priority and ask the central system for a process identification number for later processing.

2.2 Fondamental analysis module

Igor will have access to all financial data available for companies in the last five years. He must be able to read the informations from online news services and to recalculate financial ratios as earnings and other financial clues come online. He will have the capacity to present information in a graphical way, our brain being more an image processor than anything else.

This module will be an expert system whose tasks will be to identify investment opportunities based on financial analysis. As for the client's module, he will classify informations in terms of importance and ask the central system for a PID for later processing.

Known experts systems already on the markets are: EDGAR, ELOISE, FSA, ANSWER to name just a few **[1]**. Their output, instead of being reported on paper, is directed to Igor's main module which will decide of the importance at that particular moment of the day.

2.3 Technical analysis module

Using a direct online feed from the major markets, he will automatically perform analysis from the opening to the closing of the markets. Whenever a situation gets unusual, a request is made for a process identification number and the call is made to the central system which will perform other analysis to check the importance of the investment opportunities for the clients.

Many systems of technical analysis already exist on the market. Those of AIQ Systems are probably some of the most famous. As with all systems already discussed, limitations are that a signal is generated but without the whole picture. Is the company a sound financial investment? Is the market in general about to enter a correction so that we could buy it a lower price a few days from now? Which clients would be interested?

2.4 Derivative instruments and volatility analysis module

This module is divided into four parts: economic indicators, technical analysis on the market in general, using the Standard & Poors 500 index, future investments using fundamental and technical analysis, finally, options strategies.

The natural language processing layer is taking care of finding the right informations to update the database with economic indicators. Then, analysis is performed to find the impact of the informations. Various systems already exist: DOGFOX and GLOBAL REPORT from Citicorp for instance. **[2]**

Live data feed updates data on markets index. A technical analysis system like Index Expert from AIQ Systems or a neural system can make sense of the data and offer advice on the direction of the markets in general.

The same NLP layer used before also fetches data on fundamental news for commodities. Again, expert systems do already exist on the market. Other programs on the

market can analyse price movements on commodities contracts using standard technical analysis techniques.

The options module tackles strategies like covered calls, time spreads, bull and bear spreads, tripod and so on and so forth. Market Mind, from RBC Dominion Securities is an expert system on options and warrants. [3] Others exist.

2.5 Compliance and operations module

Igor has access to the house's compliance rules, he is able to calculate margins on stocks, options and futures in order to flag any problem. He will check data on dividends payable, interest coupons, warrants expiration date and rights as well. In all cases he will be able to communicate with personnel in charge using EMail.

Other functions could easily look for churning of accounts and inside trading. The Amex uses an expert system called Market Surveillance Expert to spot inside trading and the New York Stock Exchange uses ICAS, Integrated Computer Assisted Surveillance for the same purpose. [4]

2.6 Fixed-income module

Bond Yield Calculations from Intex Solutions of Massachusetts is a Lotus addins that perform like an expert system on municipal bonds as well as other fixed-income securities. [4] Such a system is to be incorporated so that Igor can connect it to other systems and then use it to make recommendations.

The head office bond desk and the news online will keep the system aware of new issues. The system will then ask the client's module for a list of potential buyers.

2.7 Currencies module

Manufacturers Hanover has a system named TARA used for currencies trading. [4] Neural systems could surely find some structure in the historical data. Some systems use standard technical analysis and could improve the recommendations.

Igor will monitor the informations from these systems in order to link and supplement them with other informations. A technical buy signal for Lockheed Corp. for instance, would trigger a request by Igor for the fundamental analysis on the company, then from the market's module to check general market conditions and finally from the client's module to list shareowners for possible put writing (if the account is option enabled) or to list of possible new clients in that stock. He would also send a request to the fixed income module to check if convertible debentures exist and if so, at the appropriate time, send a EMail message to the bond desk asking for a price.

The system would then, when the task becomes priority number one according to his set of rules, notify the retail broker: one screen with the list of shareholders who could be interested by shorting put and the list of possible new clients, one screen with charts of the stock and the debenture (if it exist). The broker's acknowledgement of this proposition would trigger outgoing calls by the system and some sort of calls queuing.

A natural language processing layer stands between the modules and the news feed from Dow Jones or Reuters. It has the task of perusing the news in search of keywords to update the databases in a manner similar to the ELOISE system developed by Arthur Andersen for the SEC. Any update of a database triggers a request to analyse the impact and establish the importance of the information right away and a request for a process identification number from the central system. Let us now look how it can all be tied together.

PART 3: HOW DOES IT COME ALIVE?

In the investment business, time is the essence of everything; every investment comes down to a question of time. In order for an expert in the investment field to be successfull, profitable that is, he must constantly make timely decisions, but considering the amount of informations coming in from sundry sources, the task soon becomes impossible to perform without a heap of missed opportunities; it is annoying, frustrating and can sometimes spells disaster. For Igor to be recognised as a true expert in the investment business he must be able to make timely interventions at all times.

When a fact or an information comes out in the investment world it carries one of three states: **1.** it is not to be acted upon right away, **2.** it might be interesting to look it over right now and, **3.** it is to be processed immediately. A no, a may-be and a yes; fuzzy logic terminology. Quarterly earnings are announced by a corporation in which we have no position or very small ones and these earnings are in line with the estimates: a no status; we shall see later when the markets are closed. Quarterly earnings are reported by a corporation in which we have large positions and these earnings are slightly different from what was expected: a may-be status, let's check the reactions of the market for the next hour, just in case. Finally, totally unexpected quarterly earnings are published by corporation XYZ in which we have large positions and the markets are reacting: a yes status. Does this change the long term prospects? Should we buy, sell, cover our positions or use options strategies? For which clients? Do we have instructions if they are not reachable by telephone. By the way, what are their telephone number? Would you put that order in for me? Was the stop loss-order cancelled yesterday for that stock? Could somebody help me for five minutes? Igor must behave timely, at all times.

To be timely, efficient and profitable, Igor need to have total control of the screen at all times leaving to the financial advisor the human relationship with the clients. We can expect a fair amount of resistance to change from the brokers but, as all comes down to a question of money, we are confident that Igor will win its case easily when the brokers will start to see commissions rolling in. In fact the brokerage houses will surely redefine the role of their financial advisors once they become aware of the true potential of such a system. A total control of the screen and in fact, of all the expertise by Igor will totally change the role of financial advisors at the retail level. Implications are far reaching and should be considered very serioulsy by the management.

To be timely in its operations, Igor will, firstly, have access to a database of all possible processes to be considered along with their states described in fuzzy logic term and, secondly, will use a set of rules to classify theses processes in terms of investment and profit opportunities for the clients. Let us consider the followings: client A needs to roll some maturing bonds; sugar contracts are heading close to their daily limit; company ABC has just announce a major acquisition; client B wishes to sell some shares and have the money sent to him; OEX options are behaving nervously because of an impending announcement; client C wishes to buy some shares for long term growth but does not know for sure if he should go with corporation BCD or else;and finally, options on stock CDE just moved 1/8 next to our stop position for that large portfolio. Nothing very special, just the usual but the broker would normally be aware of only two or three of these because, his attention would be focused on the phone and the screen; Igor, in contrast, can know that all seven processes have to be addressed at this very moment. Timely decisions can easily make the difference between $500 commissions and $5,000 commissions in the next half hour. Igor must be able to process these informations and rank them in order of priority. If I only have one sugar contract, obviously this, though important, is not a priority. The roll-over of bonds might be the first thing to do if the client has four millions coming to maturity.

All the facts to be considered will be stored in three databases according to their status (True, False, Unknown) along with a process identification number which will be assigned by a special central module so that Igor can chain the sequence of questions and answers to and from expert systems under him. Records for the three databases will be provided by the seven expert systems described in the second part of this paper. Each of the seven modules will keep a database wherein each record will include the process identification number and the description of the action to take or, of the information to be processed. The three central databases will only include the process identification and the status (True, False, Unknown) of the facts.

The heart of the whole system will be a Just-in-time module which will be responsible for the sequence of tasks to be adressed by the broker. The first character of the process identification number should identify the module. Depending on the hour, Igor will give priority to one module over the other. For example, between 8:00 and 12:00, the fixed income module will be more important, during market hours, signals from the technical analysis module and the volatility module will go first, between 17:00 and 8:00, jobs will be done for the compliance and operations module and for the fundamental analysis module. We need to have a ruled-based expert system which decides what to do and suggest to the retail broker the correct order to do the job.

Besides the just-in-time module, seven smaller central modules will be necessary; these modules will address security, telephone communications, broker's inputs, EMail, market's communications, keypad or voice interface and actions' follow-up interface. The security module will insure that the right

persons are using the system and that they cannot interfere with the expert system itself. One or two house experts will have full control on the system's maintenance and update. The telephone module could easily act as a screening device for incoming as well as outgoing calls. In fact, Boatmen Bank of St-Louis is already using such a system called ACT, Applied Computerised Telephony. [5] This does save a lot of time. The module for broker's inputs would take care of questions or information requests by the broker but Igor would manage to suggest tasks in order to minimize computermania. The EMail module would handle all internal memos and instructions to and from operations, compliance and other departments. This module would also be used by Igor itself to communicate problems, failures or questions he might have to the house experts. The market's module will transmit orders to the floors or the bond desk. Security and compliance would be especially tight with this module. Phibro Energy Inc of Connecticut, uses a keypad to enter trades directly to the floors [6]; the Sydney Futures Exchange uses voice input devices in the pits [7]. The voice interface module would be used to receive as well as to give informations. Voice signatures would be mandatory. Finally, the actions' follow-up module would, as it name suggests follow up the propositions made to the broker. When a task would be completed or cancelled (client is not there or refuses) this module would change the fact's status to False for later processing.

Obviously, Igor is going to need a lot of CPU time and a very efficient multitasking operating system. PC are definitely not fast enough. Workstation using Unix are better. If we think of all the number crunching involved, the fastest environment might be HP Apollo 9000 Serie 700 or NeXT machines which Phibro Energy Inc are already using [6]. But since a lightning fast response time is the essence of the investment business, the way to go is probably the one used by Prudential Bache which is to tie a supercomputer to one's network [8]. A five seconds response time for complex queries is a must in some areas of trading. A lot of Globex traders find the fifteen seconds paging procedure to be a major drawback, for instance.

The value of any expert system always comes down to the expertise level of the experts who built the system and to the money that can be saved or, in this case, earned by its recommendations. In our case, the heart of all the system will be Igor's just-in-time module. Everything will rely on the status assignment of a task (True, False, Unknown). Essential aspects to consider will be the dollar value of the trade to be done, the importance of the client's portfolio and the volatility of the markets involved to name just the most important ones.

PART 4: WHEN COULD WE REALLY TALK ABOUT ARTIFICIAL INTELLIGENCE?

The investment industry today is more modest with artificial intelligence; we are talking about knowledge based system rather than AI. I do think that probably 75% of human reasoning, logic and indeed behaviour can be done by computer systems using artificial intelligence.

Now, it is a fact that the best traders have problems to describe their way of reasoning and often refer to their intuition or gut's feelings. Neural computers are viewed by many as a gateway to intuition because they can learn and find logic in apparently odd or unintelligible data structures **[9, 10]**. But I believe that, given time, any sharp mind could also find the logic hidden in such data structures and, consequently, that this has nothing to do whatsoever with what we call gut's feelings or intuition.

I have traded OEX options for large institutional accounts for a few years. Now, everyone who has traded OEX can tell you that he had a system that worked. I had the same problem as most traders in explaining to my partner my way of reasoning. I was in fact using a theory in social sciences called structuralism. A lot of gut's feelings were involved because the model was referring to was it called a "lost form". For many years, I kept observing my own way of reasoning to understand its structure.

In my opinion, research on the process of intuition is the golden gate to true artificial intelligence. In 25 years from now, people will surely laugh about our pretentious AI programs. I believe that researchers on artificial intelligence should direct part of their efforts to understand the psychology involved with intuition. Probably the university of Sofia in Bulgaria would be the best place to start looking for such knowledge.

Going back to Igor, we describe it as an expert system in timely investments. If we look at the work involved at the retail level of the brokerage industry, probably 90% of the tasks could be improved. Bottom line is, the commissions earned by Igor could easily amount to three times those of the best broker. Now if we look at the decisions involved with top traders, Igor could take care of the menial jobs, the "cuisine" stuff much faster than any assistant would do. True intelligence in trading systems is not yet with us. Hence, it is important to realise that with any system actually, the final responsability rest upon the broker or the trader.

REFERENCES

[1] Liebowitz, Jay, *Expert Systems for Business and Management*, Yourdon Press Computing Series NJ, 1990

[2] Epstein, Cy, *Systems Strategy ensures the Citi never Sleeps*, Wall Street Computer Review, October 1988, p.12-15.

[3] Schmerken, Ivy, *Canadian Brokerage Hedges with on-line Expertise*, WSCR, December 1990, p.66-69.

[4] Francis, Ted, *Expert System Tools are Wall Street's Newest Creation*, WSCR, June 1989, p.27-40.

[5] Arend, Mark, *Computerized Telephony Debuts at Boatmen's Bank*, WSCR, June 1991, p.88-90.

[6] Michaels, Jenna, *Energizing the Trading Desk*, WSCR, March 1992, p.59-65.

[7] Radding, Alan, *The End of Paper Tickets*, WSCR, February 1993, p.45-48.

[8] Kulkosky, Victor, *Automation Strategies*, WSCR, October 1990, p.22-28.

[9] Chithelen, Ignatius, *New Technology learns Wall Street's Mindset*, WSCR, June 1989, p.19-22.

[10] O'Brien, Larry, *Throwing a Net over Wall Street*, AI Expert, April 1993, p.19-23.

Real-Time Object-Oriented Database Support For Intelligent Program Stock Trading

Victor Fay Wolfe & Lisa B. Cingiser

Dept. of Computer Science
The University of Rhode Island
Kingston, RI 02881

Kam Fui Lau

Dept. of Management Science
The University of Rhode Island
Kingston, RI 02881

Abstract

Expert systems for intelligent program stock trading require database management systems to support handling of large amounts of stock information. However, traditional database technology is not equipped to manage the time-constrained data of financial applications, such as "current" stock prices and volumes. Furthermore traditional databases are not equipped to schedule the time-constrained transactions of financial applications, such as transactions that must be performed within time bounds of guaranteed price quotations. This paper presents a model for real-time object-oriented databases that addresses these weaknesses and shows how this model can support a form of intelligent program stock trading called index arbitrage.

1 Introduction

Intelligent program stock trading [7] requires the management of large amounts of data where there are timing constraints that establish the validity of the data and timing constraints on the execution of transactions that access the data. For instance, "current" stock prices and volumes can only be considered valid for short periods of time and are thus time-constrained data. Similarly, transactions initiated by an expert system for program trading may be constrained to be completed before the deadline imposed by guaranteed price quotations from brokers. Computer systems typically use *database management systems* to manage large amounts of data as part of the controlling expert system. However, traditional database management systems are concerned with managing persistent data and providing good average response time to all transactions. They are not designed to support the time-constrained data and transactions found in intelligent program stock trading.

Recently, there has been limited research directed towards developing relational *real-time databases* [11] that can manage such real-time applications. Although the relational data model is useful for many applications, we believe that it is not as well-suited as an *object-oriented database model* (OODM) [15] for many financial applications that require complex data, complex relationships among data, and first-class support for timing constraints. This paper describes the model of a real-time object-oriented database (RTOODB) to support intelligent program commodity trading, and illustrates its use in a form of program trading called index arbitrage.

Section 2 describes the time-constrained data and time-constrained transactions of the index arbitrage application. Section 3 then presents our schema model for an object-oriented real-time database. Section 4 shows how the database can be used to support an expert system for intelligent program stock trading by focusing on its support for a simplified index arbitrage example. Section 5 summarizes and discusses future work.

2 Index Arbitrage Application

The illustrative application in this paper involves the trading of commodities by brokers who are licensed at commodity exchanges to represent customers. In particular, we demonstrate database support for *index arbitrage*, which involves taking a position in a stock index futures contract on an exchange, such as the Chicago Mercantile Exchange (CME), and simultaneously taking an opposite position in a basket of stocks that replicates the underlying index of the futures contract on a stock exchange such as the New York Stock Exchange (NYSE). We assume that the investor's computer database is tied directly to price

information from the stock market and the futures market through a Quotation Services Company's computer reporting system [12]. We further assume that the system is tied to an automatic order execution system (DOT and Super DOT are examples) that would accept orders in electronic form for the exchange-listed stocks[1];

Using the data stored in the database, the expert system computes the price difference between a futures contract at a futures exchange and the actual stocks at the stock exchange. If the price difference is more than the *carrying cost*, which is the cost of short term money less the dividends received during the holding period, the computer will initiate a transaction with brokers at respective exchanges. However, when the orders are actually placed, the prices for the stocks and the futures may have already changed. The price data used in index arbitrage is therefore called *perishable* data since it is only valid for a certain interval of time. If data is not used while it is valid, the investor may turn a profit opportunity into a loss.

In order to reduce the risk posed by perishable data, the investor must ensure that he can buy and sell the stocks and futures at the price he gets from his database. One way to facilitate ensuring a guaranteed profit is for the investor to obtain a *guaranteed quotation* from his brokers. A guaranteed quotation is a typical price quotation along with a guarantee that the quotation will be good for a certain time interval (*e.g.* from the time of the quotation for 30 minutes). Note that guaranteed quotations exist in the banking industry for foreign exchange [6], and that "guaranteed packages", which guarantee a price at the end of a trading day, are practiced at the NYSE. Furthermore, guaranteed quotations are not unreasonable, since if the commission is large, the broker may be willing to trade on his own accounts. Using guaranteed quotations, the investor can greatly reduce his risk as long as he completes the transaction within timing constraints. To do this, he needs database support capable of handling time-constrained data and time-constrained transactions.

3 Model of a Real-Time Object-Oriented Database

Our approach to providing real-time database support for intelligent program trading combines the data representation of the object-oriented data model [15] and the support for perishable data of a real-time database [11] into a model for *real-time object-oriented databases.* In our model, we capture the properties of databases, object-oriented databases and real-time databases. The main properties of databases are data representation, relationships among data, permanence of data, sharing of data, and arbitrary size of data. The properties of object-oriented databases include object identity, encapsulation of data and functionality, complex state (data), and inheritance. The properties of real-time databases are time-constrained data, time-constrained transaction scheduling, and predictability [13].

We use three components to model these properties of a RTOODB: *objects*, *relationships* and *transactions. Objects* represent database entities, such as individual stocks, futures, and exchanges. *Relationships* represent associations among the database objects, such as $PORTFOLIO$ that relates a collection of stocks. *Transactions* are executable programs which access the objects and relationships in the database.

3.1 Objects

An *object* is defined by $\langle N, A, M, C \rangle$. The component N is a unique name or identifier for the object. The component A is set of attributes, each of which is characterized by $\langle V, T \rangle$. V is a complex data type that represents some characteristic value of the object, and T is the time of the last update to V.

M is a set of methods which are the only means of accessing the attributes in the object. A method is defined by $\langle O, ATC \rangle$, where O is a sequence of programming language statements including: conditional branching, looping, I/O, and reads and writes to the object's attributes. Since our objects have both a value and a time field, read and write operations on attributes must read and write the time as well as the value. ATC is a set of *absolute timing constraints* on the execution of the statements in O; these constraints are formally defined in Section 3.3.

C is a set of constraints which specify the correctness of the object with respect to the system specification. A constraint is defined by $\langle Pr, ER \rangle$. Pr is a predicate represented in a boolean algebra with special atoms defined to signify the changing of an attribute (*change(a)*), the start time of an execution (*start(e)*), and the completion time of an execution (*complete(e)*). Execution e is a single executable entity such as a method invocation, or a simple read operation. Violating a constraint amounts to making the constraint's predicate false. The ER component of a constraint is an *enforcement rule* which is de-

[1] For automatic execution of over-the-counter stock trades, NASDAQ and Instinet will serve this purpose.

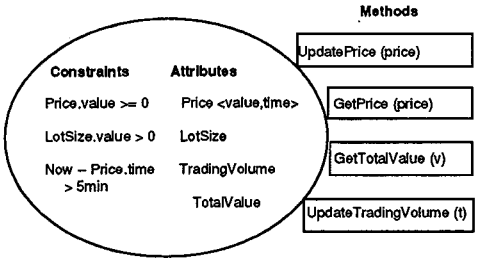


Figure 1: Example of a *Stock* Object

fined by $\langle O, ATC \rangle$. O is again a sequence of programming language statements including reads and writes on the object's attributes that is executed when the constraint is violated. ATC is a set of absolute timing constraints on the execution of these statements.

Figure 1 shows an example of an object representing a stock commodity in our database model. The attributes include a *price* which, like all attributes, has a *value* and *time* field. The price can be updated or retrieved using the methods *UpdatePrice* and *GetPrice* respectively. The stock object also contains constraints: Traditional data integrity constraints, such as constraining the price to be non-negative are supported; as are *temporal consistency* constraints, such as constraining the price of the stock to be valid for five minutes from when it is recorded.

3.2 Relationships

Relationships are objects that express associations among other objects; a relationship is defined by $\langle N, A, M, C, P, IC \rangle$. The first four components are identical to the same components in the definition of an object. P is the set of objects participating in the relationship. IC is a set of inter-object constraints placed on objects in the participant set. These constraints are defined similar to object constraints by $\langle Pr, IER \rangle$, but the predicate Pr is allowed to span multiple objects in P. IER is a set of inter-object enforcement rules used to maintain the inter-object constraints. These enforcement rules are defined as before by $\langle O, ATC \rangle$, however the programming language statements in O can now include invocations of methods of the objects in P. If any changes to the elements of P cause a constraint in IC to be violated, that constraint's enforcement rule is triggered.

In Section 4 we show an example of how the NYSE object is related to an object for Broker 1 by a *Licensed_At* relationship. Since relationships are objects, they may contain: attributes, such as the license number; methods, that may invoke methods of the related objects; and constraints that may span the related objects, such as a constraint that the broker may only trade on exchanges at which he is licensed.

3.3 Transactions

Updates and queries to the database are supported by transactions. In a traditional database system, a transaction is defined as a partial order of database operations terminated by a *commit* or *abort* operation. Typically, two desired properties of transactions are *atomicity*, which requires that either the effect of all actions in the transaction are reflected in the database or none are, and *exclusivity* which ensures that no other concurrent transaction interferes [1]. Our model of transactions is patterned after the RTC process model in [14] in that atomicity and exclusivity can be enforced through the use of constraints on parts of the transaction, but they are not required on the entire transaction.

In our model, a *transaction* is characterized by $\langle MI, C \rangle$. MI is a set of method invocations on one or more objects and/or relationships in the database. C is a set of constraints placed on the execution of these method invocations defined again by $\langle Pr, ER \rangle$. Some of the types of constraints that may be placed on a transaction follow.

- *Precedence* constraints define an irreflexive partial ordering on MI. That is, if $m_i, m_j \in MI$ such that $m_i \prec m_j$ then either m_i completes before m_j starts, m_i starts executing but m_j does not start executing, or neither method invocation is executed. The predicate is expressed as: $(complete(m_i) < start(m_j)) \vee (complete(m_i) = start(m_j) = \infty)$.
- *Exclusive* constraints specify that a set of method invocations, $M \subseteq MI$, of the transaction must be executed without interruption from any conflicting methods of other transactions. Let m be any method invocation in M, and n be any method invocation not in M that conflicts with a method invocation in M, the predicate of an exclusive constraint is expressed as: $(start(n) = \infty) \vee (complete(n) < start(m)) \vee (complete(m) < start(n))$
- *Atomic* constraints on a set of method invocations, $M \subseteq MI$, specify that every method invo-

cation in M is executed or no method invocation in M is executed. This requirement is expressed with the predicate: $(\forall_{m \in M} complete(m) \neq \infty) \vee (\forall_{m \in M}, start(m) = \infty)$.

- *Absolute timing* constraints specify that execution is to be performed within a certain time frame. We use a *temporal scope* to define absolute timing constraints: $\langle E, sa, sb, d \rangle$ where E is a set of executions, sa is an absolute earliest start time, sb is an absolute latest start time and d is an absolute latest complete time (deadline). The predicate for an absolute timing constraint is: $\forall_{e \in E}((sa \leq start(e) \leq sb) \wedge (complete(e) \leq d))$. In the case of absolute timing constraints on transactions or inter-object enforcement rules, E is a set of method invocations. For object methods and enforcement rules, E is a set of programming language statements including reads and writes of attributes.

- *Simultaneous* constraints specify that a set of method invocations, $M \subseteq MI$, must start executing within the same time interval. This constraint predicate is expressed as: $\forall_{m_i, m_j \in M}(start(m_i) = start(m_j) \pm \delta)$, where δ is a small time value.

Any collection of method calls generated by the investor's expert system or by the automated update facilities tied to the quotation company's reporting system can be realized in a transaction. In particular, Section 4 describes how the index arbitrage with guaranteed quotations can be supported by a transaction using this real-time object-oriented database.

3.4 Inheritance

In the traditional object-oriented paradigm, *inheritance* is the mechanism used to define objects as extensions of previously defined objects. A sub-type in the inheritance hierarchy inherits all of the properties of all of its super-types and may add its own properties as well. Traditionally, inheritance applies only to attributes and methods. In our model, we extend this notion to include constraints, enforcement rules, and the added features of a relationship. We also support another feature of inheritance known as *substitutability* [15], which allows us to treat an instance of an object as an instance of any of its super-types. In other words, an instance of a sub-type may be used in contexts where one of its super-types is expected.

We show in Section 4 how inheritance can be used to define *stock* and *future* object subtypes by inheriting from a general *commodity* type.

4 Using the RTOODB in Index Arbitrage

We now present a more detailed description of our RTOODB support for indexed arbitrage.

Type-Level Model. Figure 2 shows the types used in the database. Double rectangles are object types, double diamonds are relationship types. A relationship type links one or more object types together (it is shown in Figure 2 by using solid arrow pointing from the relationship type to the object type(s)). For instance, a $BROKER$ object type is related to a commodity exchange (COM_EX) object type by a licensed at (LIC_AT) relationship type. In addition to linking the two object types together, the relationship also stores information pertaining to the relationship, such as the broker's license number on the exchange. A relationship may exist among several instances of the same object type. For instance, $PORTFOLIO$ relationship type has just one link to the $STOCK$ object type, indicating that it is a relationship among $STOCK$ instances. A dotted line in the figure represents that one object type is derived from another object type using inheritance. For instance, a $STOCK$ object type is derived (indicated by IS_A) from a commodity $COMMODITY$ object type. The child object type ($STOCK$) inherits all of the attributes, methods, and constraints from the parent object type ($COMMODITY$); it also adds its own special properties.

A more detailed characterization of object and relationship types in terms of their attributes, methods, etc. are shown in Appendix A.

Instance-Level Model. In Figure 3 we show an object instance-level model representing brokers, commodities and exchanges in our local database to provide information for the index arbitrage. Ellipsoids represent object instances such as stocks $S\_1 \rightarrow S\_n$, brokers $B\_1 \rightarrow B\_m$ and $B\_1 \rightarrow B\_j$, futures $F\_1 \rightarrow F\_i$, and the $NYSE$ and CME exchanges. Each object instance is similar to the stock object shown in Figure 1 last section. Diamond shapes represent relationships. For instance the $NYSE$ object is related to object B_1 by a Licensed_At relationship called Lic_1. Other relationships are: Listed_At which has instances $L\_1 \rightarrow L\_n$ and $L\_1 \rightarrow L\_i$ between commodities and their exchange, and $S\&P500$ which is an instance of the $PORTFOLIO$ relationship type.

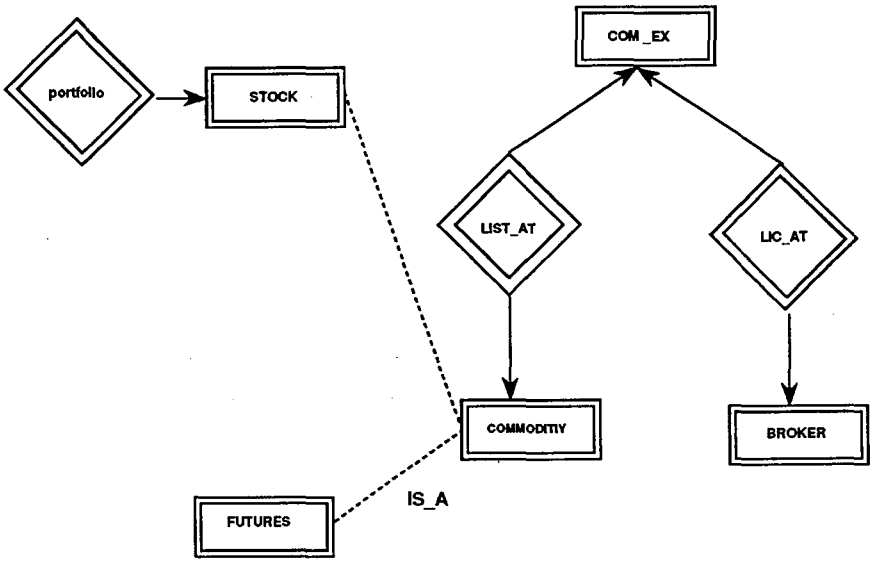


Figure 2: Type-level Model of Database for Index Arbitrage

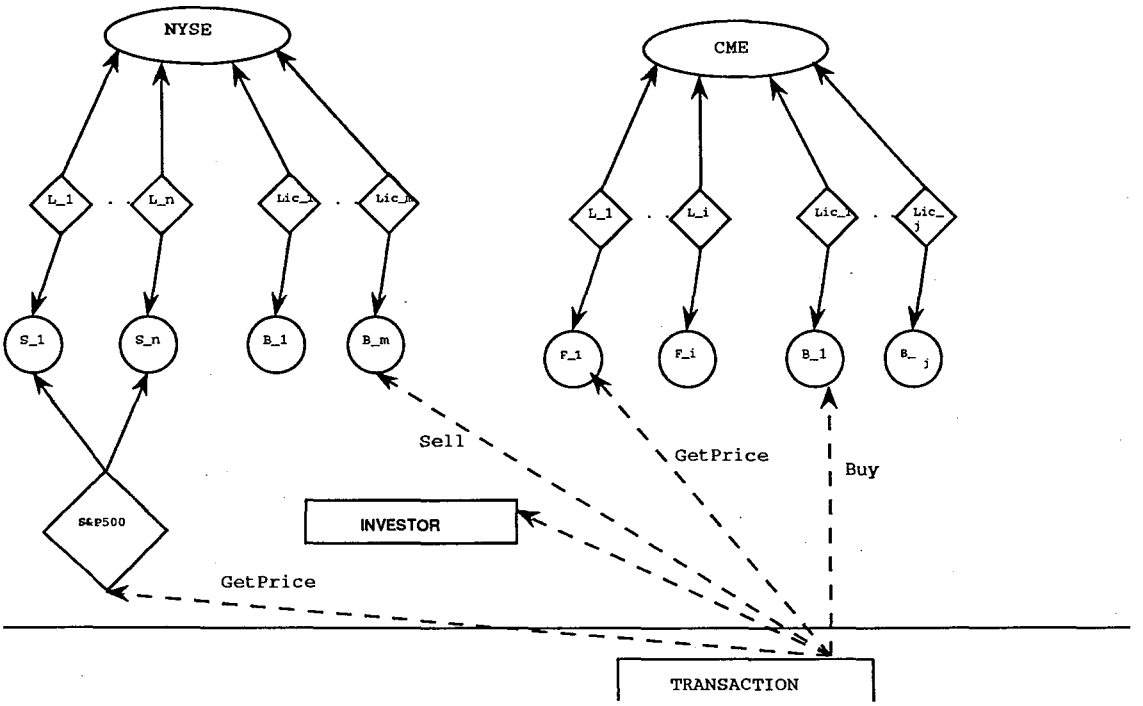


Figure 3: Object Instance Model For Index Arbitrage

Transaction Example. The actual index arbitrage is an example of a transaction in this real-time database. This transaction periodically reads both the CME and NYSE S&P500 object's price by invoking the appropriate methods as shown in Figure 3. If a difference is found and the profit is high enough, then the transaction obtains a guaranteed quotation from the brokers for a price and an interval of time for which the price is good. If the transaction still appears profitable and that it can be carried out within time, the transaction invokes *Buy* and *Sell* methods from the appropriate brokers. The transaction's execution is time-constrained by the constraints derived from the time validity intervals of the prices involved. To ensure that either the entire transaction is carried out within timing constraints or the transaction is aborted, we adapt a *timed atomic commitment* protocol [2], which sets intermediate deadlines on obtaining the guaranteed quotations, deciding on the course of action, and carrying out the actions decided upon.

5 Conclusion

This paper has described a new database technology: real-time object-oriented databases, and how it can be used to support an expert system for program stock trading. We have shown how the object-oriented data model is effective for modeling complex data types, such as individual stocks, and the relationships among data types, such as a portfolio collection. We have also shown how our model can effectively specify real-time constraints of perishable data such as "current" stock prices and volumes.

We have implemented a prototype database that supports objects and relationships such as those described in this paper. The SORAC (**S**emantic **O**bjects **R**elationships **A**nd **C**onstraints) database research project at the University of Rhode Island [8, 3] captures the objects and relationships of our model by combining features of the object-oriented [15] and semantic data models [10]. The SORAC data model maintains object-oriented principals: encapsulation of structure and behavior together in data objects; information hiding; and unique, but perhaps complex object identity [15]. It also adds the capability for user-defined relationships among object types [8]. To capture the model, we developed an object-oriented data definition language for defining objects and relationships [3]. A compiler translates these object and relationship definitions into code that executes on the Ontos [9] commercial object-oriented database system.

We are currently developing a transaction system for our prototype database. We are extending the standard SQL query language with capabilities to support objects [4] and real-time properties [13] as the interface to generate transactions. This interface will be used to implement a timed two-phase commit protocol, called timed atomic commitment [2], to design database transactions to perform the timed index arbitrage described in this paper. That is, the protocol will first determine if the transaction can realize a profit, and can do so within timing constraints posed by guaranteed quotations (Phase One). It will use the timing constraints expressed by the data model to facilitate this computation. The protocol will then either decide to abort due to lack of adequate profit or lack of time, or it will carry out both the buying and selling of stocks and futures (Phase Two). Although using such a system for actual indexed arbitrage may not be possible, we believe that the expression of complex data and timing constraints of our real-time object-oriented data model provides powerful general structure for the design of database components to support expert systems that perform program commodity trading.

Appendix A: Detailed Model of Example

This appendix describes types for objects in the index arbitrage example. A full description is impractical due to excessive details, but important features of the example are presented. Key: N = name of type; A = attributes; M = methods; C = constraints; P = set of participant types in a relationship type IC = inter-object constraints in a relationship type (See model of Section 3).

Exchange Object Type

N	:	Exchange
A	:	Total_Price
		Total_Volume
M	:	UpdateTotalPrice(P)
		UpdateTotalVolume(V)
		GetTotalPrice(P)
		GetTotalVolume(V)
C	:	TotalPrice > 0
		TotalVolume > 0

Portfolio Relationship Type

N : Portfolio
A : Price_of_Portfolio
Volume_of_Portfolio
M : GetPortfolioVolume(V)
UpdateMembership(Member)
UpdatePortfolioPrice(P)
GetPortfolioPrice(P)
UpdatePortfolioVolume(Volume)
C : PortfolioPrice > 0
PortfolioVolume > 0
IC : change(Stock Price) ⇒
UpdatePortfolioPrice(P)
change(Stock Volume) ⇒
UpdatePortfolioPrice(Volume)
P : {Stocks(n)}

Listing Relationship Type

N : Listing_At_Exchange
A : Listing_ID
M : GetListingID(Id)
UpdateListingID(ID)
IC : change(Stock Price)
⇒ Exchange.UpdateTotalPrice(P)
change(Stock Volume) ⇒
Exchange.UpdateTotalVolume(V)
P : {Exchange(1), Stock(n)}

Commodity Object Type

N : Commodity
A : Price
Label
M : UpdatePrice(Amount)
GetPrice(P)
UpdateLabel(Label)
C : Price > 0

Futures Object Type

N : Futures
A : Price
Label
UnitsPerContract
M : UpdatePrice(Amount)
GetPrice(P)
UpdateLabel(Label)
UpdateUnitsPerContract(Units)
GetUnitsPerContract(U)
C : Price > 0
UnitsPerContract > 0

Stock Object Type

N : Stock
A : Price
Label
LotSize
TradingVolume
M : UpdatePrice(Amount)
GetPrice(P)
GetVolume(V)
UpdateLabel(Label)
UpdateLotSize(Size)
UpdateTradingVolume(V)
C : Price.Value > 0
LotSize.Value > 0
Price.Time < Now - 5 min

Broker Object Type

N : Broker
A : Trading_Portfolio
Trading_Commodity
Trading_Portfolio_Commission
Trading_Commodity_Commission
M : GuaranteedQuote(Commodity,Price,Vol,V_deadline,Vc
UpdatePrice(Amount)
GetPrice(P)
Sell(Commodity,Vol,Price)
Buy(Commodity,Vol,Price)
ConfirmExecution(Commodity)
C : Commission ≥ 0

Licensed At Relationship Type

N : Licensed_At_Exchange
A : License_Num
M : UpdateLicNum(Num)
GetLicNum(Num)
P : {Exchange(1), Broker(n)}

References

[1] Phillip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison Wesley, New York, 1986.

[2] Susan Davidson, Insup Lee, and Victor Wolfe. Timed atomic commitment. *IEEE Transactions on Computers*, 40(5):573–583, May 1991.

[3] Michael Doherty, Joan Peckham, and Victor Wolfe. Implementing relationships and constraints in an object-oriented database. University of Rhode Island Dept. of Computer Science technical report URI-TR-93-218.

[4] Leonard Gallagher. Object SQL: Language extensions for object data management. In *Proceedings of International Society For Mini and Microcomputers*, Sept 1992.

[5] Market Volatility and Investor Confidence Panel. Market volatility and investor confidence - report to the board of directors of the new york stock exchange, inc. New York Stock Exchange, June 1990.

[6] G. Munn, F. Garcia, and C. Woelfel. *Encyclopedia of Banking andd Finance.* Bankers Publishing Company, 1991.

[7] J. Miller. *Program Trading – The New Age of Investing.* J.K. Lasser Institute, 1991.

[8] Bonnie MacKellar and Joan Peckham. Representing design objects in SORAC: A data model with semantic objects, relationships, and constraints. In *The Second International Conference on Artificial Intelligence and Design*, June 1991.

[9] Ontologic Inc. Ontos object database system reference for Unix systems. 1989.

[10] Joan Peckham and Fred Maryanski. Semantic data models. *ACM Computing Surveys*, 20(3):153–189, Sept. 1988.

[11] Krithi Ramamritham. Real-time databases. To appear in *International Journal of Distributed and Parallel Databases.*

[12] Spectrum Staff. How computers helped stampede the stock market. *IEEE Spectrum*, 18(12), Dec. 1987.

[13] Victor Wolfe and Lisa B. Cingiser. Issues in object-oriented real-time databases. In *Proceedings of the IEEE Workshop on Real-Time Operating Systems and Software*, May 1992.

[14] Victor Wolfe, Susan Davidson, and Insup Lee. *RTC*: Language support for real-time concurrency. *Real-Time Systems*, 5(1), March, 1993.

[15] Stanley Zdonik and David Maier. *Readings in Object Oriented Database Systems.* Morgan Kauffman, San Mateo, CA, 1990.

Embedded Artificial Intelligence for Trading Floor Support

Yuval Lirov, Ohad Aloni, Boris Grinfeld, and Alan McMichael

Salomon Inc

Rutherford, NJ 07070

Abstract

Real-time, distributed systems automation in support of financial applications requires the application of embedded artificial intelligence to cope with the urgency and volume of data in these environments. Conventional systems make excessive demands on the user. Syntax ridden input and unfiltered output reflect the lack of arbitrating intelligence in the design of these tools. This paper illustrates how modest application of advanced interface design and artificial intelligence techniques such as case-based reasoning, fuzzy logic, and expert systems can make these tools more responsive and contribute economies of scale.

1. Introduction

One of the most significant influences on the way financial markets (and their support systems) work is the effect of distributed computing technology. In an information era where transactions occur electronically, access to markets means access and control of the distributed automation that drives the process.

Like a cat chasing its tail, automation in trading floor support has become (in many cases) too complex and demanding for human operators. Paradoxically, this mitigates against the initial purpose of the automation.

This in turn has decreased productivity while increasing the expense of this vital function.

Technology sets its own agenda and trading support systems are not immune to it. Artificial intelligence and knowledge-based systems represent an effective hedge against being outrun by the very technology that was designed to streamline tasks [Melamed, 1991].

Because the design methodologies of these techniques take into account the needs of human operators and the effect that a system has upon them, knowledge based systems, fuzzy logic, and case based reasoning assure success by "re-enabling" the human operators of the system. The associated benefits of distributed applications, intuitive user interfaces, and other advanced computing tools create an effective synergy that redefines automation in useful, human terms [Lirov, 1992].

As illustrated in Figure 1, trading floor support is primarily an expense-driven component of trading. The trading that it supports is the source of income. There are exceptions to this dichotomy that are worth considering, such as external marketing of trading support software produced in-house, and the obvious link that this automation has upon productivity (i.e., income) of the traders. But the primary focus of this paper is not about the income side of trading automation (e.g., risk management, etc.), but the expense-based nature of trading floor support.

Accordingly, we will describe how this automation evolves along two principal paths: systems management and problem management.

Several examples of automation (fault and task management in vertical development and problem and service requests in horizontal development) will be illustrated for the purposes of linking cost/benefit justification and application of AI to the development of trading support software.

2. Evolution of Systems Automation

Evolution requires an operative position towards systems automation. It is concerned with automation software development and related product support responsibilities.

This position is focused in two distinct areas (along the paths illustrated in the following two tables).

2.1 Software Development "By Exception" (Horizontally)

An implicit goal of automation should be firm-wide repository of corporate knowledge. This goal has strategic importance because it links separate databases and enables an integrated view of projects and resources (see Table 1).

A lack of such systems precludes efficient cost management of several important and high-cost functions, e.g., help desk, systems troubleshooting, applications release management, and technology service requests. Such an integrated system would be invaluable at both improving services and identifying/controlling high cost items.

The usefulness of this repository hinges on the "interoperability" of databases and software development "by exception." Interoperability means that multiple heterogeneous applications must be able to interact.

Moreover, software should be developed only in exceptional cases. This is due to designing the majority of software so that it is reusable across applications.

2.2 Systems Management "By Exception" (Vertically)

The salient feature of successful systems automation is the capture and maintenance of long-term operational knowledge and its integration in real-time with the immediate situation. It focuses on aggregate services, evaluates service level in real-time, enables dynamic on-line configuration, automatically interprets data, issues troubleshooting advice, and even acts autonomously without having a fully specified plan in advance.

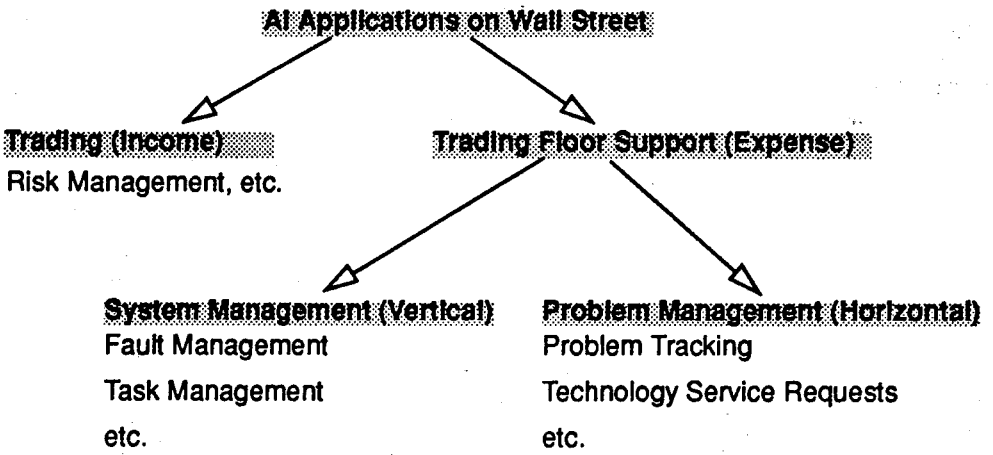


Figure 1: AI for reduction of expense

Conventional systems management tools are unable to meet cost-effectively these recent requirements.

This logically leads to the development of distributed operations productivity tools that promote the concept of "management by exception" (see Table 2).

These tools automatically perform the vast majority of systems management activities, including taking a desired corrective action and alerting the user only under exceptional circumstances.

They strike a balance between conflicting requirements to reduce operations costs while increasing operations responsibilities.

3. Case Study: Problem Management System (horizontal development)

3.1 PROTEUS Description

The Problem
Recording problem incidents within an organization provides support groups (help desks, etc.) with the information they need to identify and correct problems.

Manually recording these incidents on paper forms can be time-consuming and often inefficient.

The Solution
The solution to this problem lies in an automated method that offers ease-of-use and quick, accountable response. PROTEUS stands for PRObIem Tracking Expertise Utilization System.

Both users and support groups can create and manage problem incidents for further work and analysis, while PROTEUS safely stores the data and lets you define and print reports with both text and graphical output.

3.2 Case-based search in Problem Tracking Software

PROTEUS, the problem tracking application, relies on advanced searching algorithms to scan its database for instances that relate to the problem at hand [Riesbeck, 1988]. Since a database of problem reports may be quite extensive when employed for more than a few years, a powerful and efficient search heuristic should limit the "expense" in terms of operator time.

Table 1: Evolution of Systems Management Tools (Horizontal Development)

Aspect	Conventional Information Systems	Quality Tracking	Service Quality Control
Purpose	Monitor and communicate	Monitor, interpret, and communicate	Coordinate, monitor, interpret, and communicate
Focus	Process	Aggregate service	Aggregate applications
Output	Query and report	Query, report, alarm, and interpret	Query, report, alarm, interpret, and advise
Data Sources	Separate database	Multiple homogenous database	Multiple heterogeneous database
Scope	Single ticket per screen	Multiple tickets with single item per ticket	Multiple items per ticket
Configuration	Limited, global	Filter-based, global	Filter-based, task dependencies, local
User Interface	Uniform	Diverse	Diverse
Techniques	Conventional	Case-based reasoning	Case-based reasoning and rule-based expert system

PROTEUS extensively tests for similarity of the search string (automatically extracted from the current problem) to other occurrences in the database, but it limits this search by placing a higher weight on a category named "affected applications," which can be specified by the operator.

Herein is a hypothetical case of how cased-based search with PROTEUS might solve a dilemma on behalf of its mythological namesake.

The Problem

Proteus was a sea god, the keeper of the seals of Poseidon, with the ability to assume various shapes. Whoever could bind him during his noontime sleep to keep him from changing shape could oblige him to foretell the future.

Aristaeus was an uneducated guardian of herds and a bee-keeper. He experienced a dramatic incident and did not know how to behave. He turned to Proteus for advice, acted accordingly, and was rewarded. His session with PROTEUS could have been conducted in the following way (Aristaeus may have made a few spelling mistakes):

Aristaeus recorded his dilemma as a problem incident in PROTEUS, "I fell in love with beautiful Eurydice, wife of Orpheus, and tried to seduce her. As she fled from me, my bees chased her. She accidently stepped on a snake and died of its bite."

"The gods became enragged with me and kilt all my bees." Aristaeus then read a similar problem with a resolution that appeared on the PROTEUS screen:

Operator: Ethiopian Queen Cassieopeia

Problem: I boasted that my daughter Andromeda was more beautiful than the sea-goddesses Nereids. Poseidon, god of the sea and father of the Nereids, became enraged and sent a sea monster to kill all my people.

Resolution: I sacrificed Andromeda by chaining her naked to a sea cliff and Poseidon called off the monster.

The Conclusion

Upon reading the resolution of the second incident, Aristaeus decided to offer sacrifices and funeral honors to Eurydice. Soon, the new swarms of bees were generated.

PROTEUS finds automatically [Gick and Holyoak, 1980] that both stories have to do with something beautiful, that the gods are involved, somebody must be enraged, and somebody gets killed.

Note: Not only does PROTEUS disregard the spelling mistakes of Aristaeus, but it also makes conclusions without any human (or divine) help, e.g., without using keyword-based search requiring extensive pre-processing of text.

Table 2: Evolution of Systems Management Tools (Vertical Development)

Aspect	Conventional Information Systems	Quality Tracking	Service Quality Control
Purpose	Monitor and communicate	Monitor, interpret, and communicate	Coordinate, monitor, interpret, and communicate
Focus	Process	Aggregate service	Aggregate applications
User base	Low Hundreds	High Hundreds	Thousands
Output	Message, binary status	Message, status, troubleshooting advice	Message, status, advice, automatic execution of a program, action schedule
Data Sources	Single per message	Multiple per advice	Multiple per action
Configuration Scope	Static off-line, global	Dynamic, on-line, local	Adaptive, local
User control specification	Global	Filter based, global	Filter-based, task dependencies, local
User Interface	Uniform	Diverse	Diverse
Techniques	Object-oriented	Fuzzy logic, pattern matching, model-based reasoning	Rule-based expert system

3.3 Determining Cost/Benefit

A cost/benefit analysis of a project like PROTEUS juxtaposes the investment (man-months, hardware, software, place, etc.) spent on its development against productivity enhancements and savings gained through automation of manual labor, elimination of duplicate effort, and (optionally) against projected income from externally marketing the software.

Because the benefits of automation products cannot be easily measured based on sales, we have applied productivity measures based on user feedback. In the following cases, the term "headcount" is used as a hypothetical measure of composite expenses to maintain one employee.

3.4 Benefit analysis for PROTEUS

Assuming a user base of 100, and 10% improvement in trouble reporting/response by help desk and field support users (due to better information more readily available on line), and replacing phone calls by seamless access to a common database, we gain an estimated benefit of 10 headcount.

4. Case Study: Network Management System (vertical development)

4.1 DRMS Description

The Problem
Traditional network management systems cannot effectively handle very large-scale networks. Contradictory and excessive messages and alarms tend to confuse operators and prolong response.

The Solution
DRMS acts on the operator's behalf by using "fuzzy logic" to reduce the amount of traffic between the fault data collection and the message display components. The net effect alleviates the burden on both the operator and the network, thus making the system more responsive to the needs of the trading floor.

Deployment Results
Early results indicate that this system has been able to cut management work-load by half. Further operator work-load decrease (up to 90%) is possible through continued automation and pro-active trouble-shooting.

The improved troubleshooting that this systems provides operators not only improves service quality (i.e., reducing system downtime), but also positively affects systems operators and enhances their productivity by reallocating their time from mundane message interpretation to more demanding and productive work [Aloni, et. al., 1991].

4.2 Fuzzy Knowledge Base Design in Systems Management Software

Although fuzzy logic is typically associated with systems where a degree of ambiguity is inherent in the underlying problem, we have successfully integrated this technique within a more precise hierarchy of concepts (such as parent and child) to automate network diagnostics.

4.2.1 The Role of Systems Management Automation

Systems management fulfills two key functions: first, it monitors the network to identify deficient service, and second, it diagnoses the monitored data to ensure the timely and least costly resolution of an identified deficiency [Lirov and Yue, 1991].

These tasks are difficult to achieve concurrently because each is bound by an inherent conflicting constraint imposed by the network. The first is the data monitoring conflict where as much system fault data must be acquired and displayed to the operators as possible without unduly increasing system load.

The second is the data interpretation conflict where all possible service disruption causes together with the available evidence must be analyzed in order to indict the perpetrator of the disruption while remaining within time constraints.

4.2.2 DRMS Innovation: Efficient Application of Fuzzy Logic to Network Management

Fuzzy logic approach to automated network diagnostics resolves the conflict between the need to acquire vast amounts of detailed knowledge and the ability to act quickly with a sufficient degree of accuracy.

On the other hand, application of fuzzy logic to computer communications network management is complicated by the need to integrate precise information available about the network topology with the massive flow of time-dependent alarms. Our contribution consists of developing a simple, real-time, systems alarm diagnostic model.

4.2.3 Fuzzy Knowledge Base Design

In a fuzzy logic controller, the focus is the human operator's behavior. The systems control parameters are handled by a fuzzy rule-based expert system, a logical model of the thinking processes a person might go through in the course of manipulating the system.

To describe network service quality, we use fuzzy quantifiers. A fuzzy quantifier is a fuzzy number that fuzzy characterizes the absolute or relative cardinality of one or more fuzzy or non-fuzzy sets. A collection of fuzzy relationships in a database may describe the status of individual network components.

Additionally, we use test-score semantics [Zadeh, 1983], where the meaning of a semantic entity is represented by a vector of numbers in the unit interval. The elements of the vector serve a measure of the compatibility of the semantic entity with an instantiation of that entity in the database.

We automate network diagnostics in two phases: representation and inference. The first stage represents the topology in terms of a hierarchy (tree), that has a natural parent-child relationship. The second stage enables to infer status of a parent from states of its children by using a fuzzy rule:

IF **most** of the *children* are down for **long** time

THEN indict the *parent*

AND exonerate the *children*

In the previous example, the concepts **most** and **long** (shown in bold) are fuzzy concepts that are integrated with a structural and precise topology that uses the concepts of *children* and *parent* (shown in italics). This illustrates how two apparently incompatible systems can be successfully integrated.

5. Case Study: Batch Task Management System (vertical development)

5.1 JAWS Description

The Problem
A significant portion of network operations activities are devoted to managing batch tasks. Batch task management becomes intractable as the number of tasks grows and the operator must handle more tasks quicker [Stepanenko, 1987].

The Solution
JAWS explicitly captures and maintains global complex knowledge about task inter-dependencies using local job-specific descriptions, and it explicitly maintains the status of numerous tasks.

It integrates the long-term task inter-relationship knowledge with a global picture of instantaneous task status, thus facilitating dynamic task scheduling and efficient trouble-shooting.

Scheduling the jobs during execution and submitting the jobs for execution without having a complete schedule on hand is the most important feature of this new generation of systems support automation tools. This eliminates the burden of task execution by automatically following and adjusting the schedule, remotely running the scripts, and monitoring task execution.

Under specified exceptional conditions, JAWS automatically alerts the task manager and/or performs desired corrective actions.

It also offers several features to assure long-term batch cycle quality. In particular, its database of run logs maintains useful history and facilitates graphical statistical analysis.

5.2 Using an Expert System with Batch Task Management Software

The architecture of JAWS supports many "front ends" to the batch job scheduler (see Figure 2):

- multiple perspectives offered to each user (i.e., views of task, job, and groups of jobs)
- multiple applications running concurrently from each user
- multiple users, each running one or more applications.

The complexity of these intensive and interacting processes requires a high degree of inter-process communications and arbitrating intelligence.

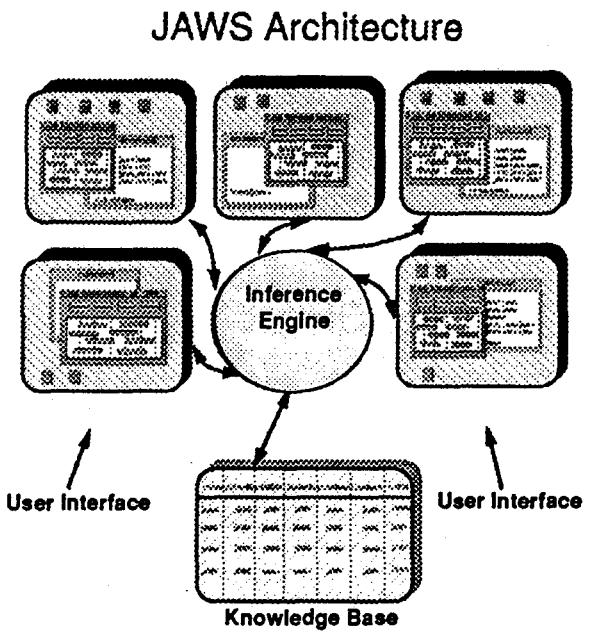


Figure 2: JAWS Architecture

5.3 Components of a traditional Expert System as employed in JAWS

JAWS uses three traditional expert system components within a production system whose complexity is largely transparent to the user (see Figure 3).

User Interface
A point-and-click, object-oriented graphical user interface makes describing task dependencies at the local level more intuitive.

While jobs are being monitored, this type of interface can also effectively communicate task status by displaying jobs with different status in different colors.

Inference Engine
The inference engine executes the rules and other data that are stored in the knowledge base. It performs global scheduling, monitors and executes tasks, and updates the user interface.

Knowledge Base
The data about task interdependencies is stored in a database using a Sybase server. The database maintains status and dependencies in the form of rules:

IF combination of events

THEN advise, execute, monitor, (change color), etc.

5.4 Benefit analysis for JAWS

A hypothetical application of 100 batch tasks without JAWS requires 3 to 6 months of chron files development and 1 operator for manual execution of every 100 tasks.

Thus, an installation with 10,000 tasks using JAWS saves 100 operators.

6. Summary

As illustrated in these three examples, systems automation in support of trading can achieve two-fold benefit from the appropriate application of artificial intelligence. Firstly, the operator-oriented focus of AI software development ensures operator productivity. In turn, this greatly reduces the expense of the development when measured in productivity gains.

AI has come out of the labs and into the mainstream of business applications. It brings unanticipated benefits to organizations whose development costs have spiraled, and a tangible sense of support to those whose job it is to support.

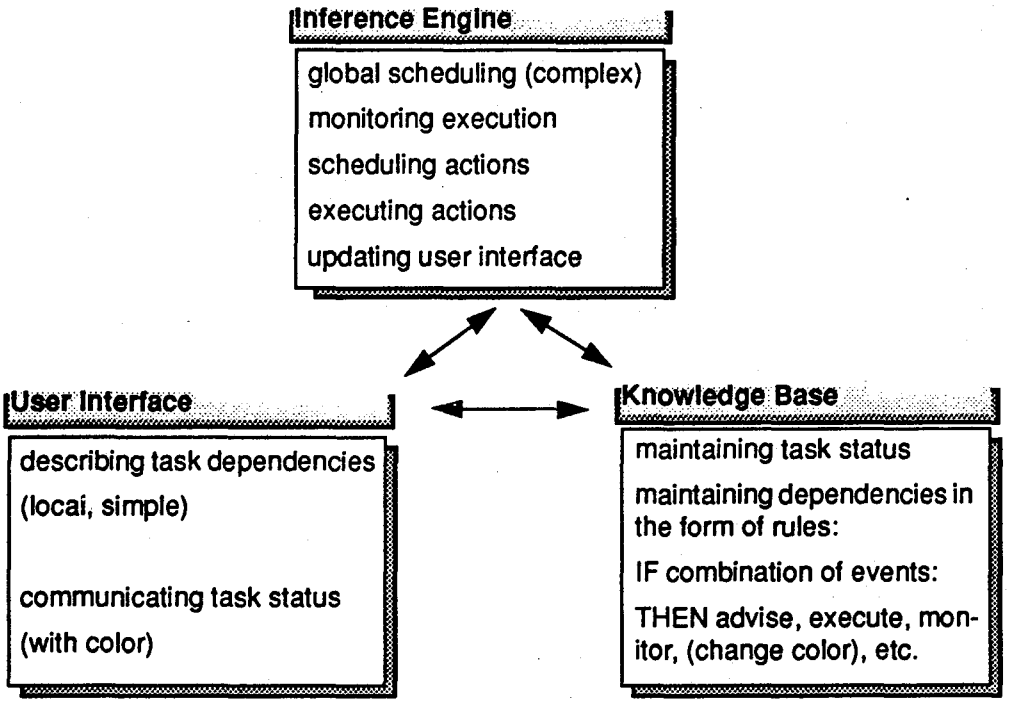


Figure 3: Expert System Components in JAWS

7. Acknowledgments

Peter Bloom, Bob Cassiliano, Kathleen DeGuilme, Mike DeMarco, Tom Lee, Kelly McGowan, Stu Sugarman, and Jim Traverso for their continued encouragement and support.

Jiyang Xu and Dany Breslauer, for their contribution to the design of JAWS.

Paul D. Henry, for editing this paper.

8. References

O. Aloni, Y. Lirov, A. Melamed, and F. Wadelton, "Performance Analysis of an Expert System For Network Management," World Congress on Expert Systems, Orlando, 1991.

M. Gick and K. Holyoak, "Analogical Problem Solving," Cognitive Psychology, 12:306-355, 1980.

Y. Lirov, "Gaining Strategic Advantage with Real-Time Distributed Artificial Intelligence," Plenary Talk at 1992 Expert Systems Applications Conference in Paderborn, Germany.

Y. Lirov and O. Yue, "Automated Troubleshooting Knowledge Acquisition for Network Management Systems," Applied Intelligence, 1:121-132, 1991.

A. Melamed, "Distributed Systems Management on Wall Street - Technology Needs," AI Applications on Wall Street, New York, October 9-11, 1991, pp. 206-212.

C. Riesbeck, "An Interface for Case-Based Knowledge Acquisition," Proceedings of DARPA Workshop on CBR, ed. by J. Kolodner, Morgan Kaufman, San Mateo, CA, 1988, pp. 312-326.

A. Stepanenko, "Network Task Scheduling Service Protocol," Automatika i Vychislitelnaya Technika, 21(3), pp.18-24, USSR, 1987.

L. Zadeh, "Commonsense Knowledge Representation Based on Fuzzy Logic," Computer, October 1983, pp. 61-65.

Paper Session: AI and Modern Portfolio Theory

Chair: Ross Miller, GE Corporate R&D

Tactical Asset Allocation Using Fuzzy Logic and Mean-Variance Optimization

Ypke Hiemstra
Information Systems Department, Faculty of Economics and Econometrics
Vrije Universiteit Amsterdam
De Boelelaan 1105 , 1081 HV Amsterdam, The Netherlands
Email YHiemstra@econ.vu.nl

Abstract

Tactical asset allocation adjusts the diversification of a strategic (i.e. long term) investment portfolio among stocks, bonds and cash in order to respond to fluctuating short term market conditions. The hardest part of successful tactical asset allocation is to predict the stock market. Experts predict the stock market using complex and partially non-numerical data, while their knowledge is imperfect and uncertain. As stock market prediction involves imprecise concepts and imprecise reasoning, fuzzy logic appears to be a natural choice for knowledge representation. This paper discusses a rule-based fuzzy logic stock market predictor. In addition, the paper presents an optimization model based on the popular mean-variance portfolio selection model to determine the optimal tactical portfolio. An empiric test indicates the performance of both models. Fuzzy logic and mean-variance optimization appear to form a powerful combination for tactical asset allocation.

1. Introduction

This paper presents a rule-based fuzzy logic stock market predictor and an optimization model based on the mean-variance portfolio selection model to determine the optimal tactical portfolio. Tactical asset allocation (TAA) adjusts the diversification of a strategic (long term) investment portfolio among stocks, bonds and cash in order to respond to fluctuating short term market conditions. As stock market prediction involves imprecise concepts and imprecise reasoning, fuzzy logic (Zadeh [12]) appears to be a natural choice for knowledge representation. Fuzzy logic has proven to be succesful in industry (Kandel [5]). Two examples of fuzzy logic applications in portfolio management are Wong et al. [9] and Yasunobu et al. [11].

First the paper introduces TAA. In section 3 the paper introduces an approach to stock market prediction, which corresponds to the way experts work and which can be implemented in a natural way using fuzzy logic. Then the paper briefly discusses fuzzy logic, and discusses in section 5 the application of fuzzy logic to represent knowledge to predict the stock market. Section 6 presents the optimization model. Section 7 discusses the implementation and evaluates the system's performance, and section 8 contains the conclusion.

2. Tactical asset allocation

Sharpe [7] presents an integrated view of various types of asset allocation. Figure 1 presents a general view of TAA. Short term market behavior and the investor's risk attitude determine the optimal portfolio, under constraint of the strategic portfolio. Confidence in TAA is limited due to the risk associated with predicting short term market behavior. Using the strategic portfolio as a constraint limits the deviation from the long term optimal portfolio, ensuring that adjustments correspond to the confidence put in TAA. TAA policy is evaluated by comparing the optimal portfolio's performance to the performance of the strategic portfolio.

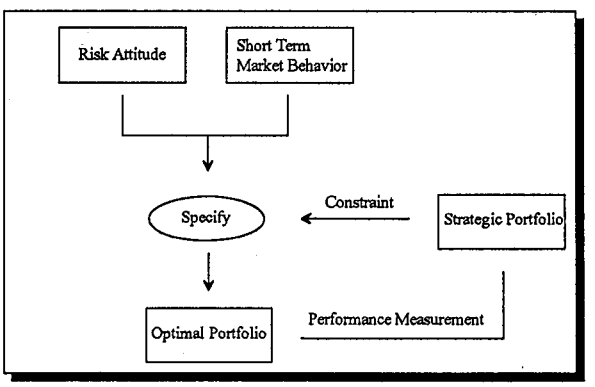


Figure 1, general view of TAA

3. Predicting the stock market

There is a wide variety of approaches to predicting the stock market. Examples are time-series analysis, scenario-based methods, and dividend discount models. This study adopts an approach to predicting the short term stock market return which relates fundamental information about macroeconomic and market conditions to the market excess return, the market return minus the risk-free rate of return. The approach is fact-based as opposed to forecast-based in that no attempt is made to make explicit forecasts of any other variables than the stock market return.

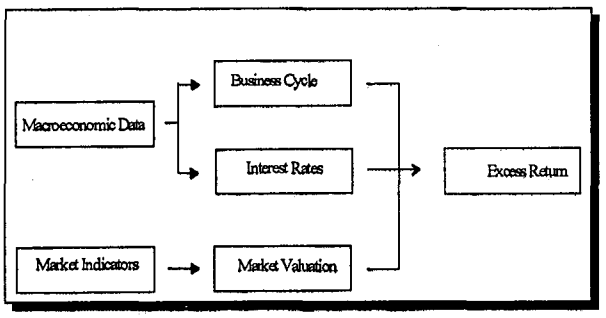


Figure 2, stock market prediction scheme

Figure 2 illustrates the approach. The stock market return is predicted using market valuation and two macroeconomic factors, the business cycle and interest rates. The current state the economy is in together with information on interest rates characterizes fundamental investment conditions. Market indicators like the price-earnings ratio are used to determine the level of market vaulation. The better fundamental investment conditions, the higher the excess return. The higher market valuation, the lower the excess return.

4. Fuzzy logic

The hardest part of successful TAA is to predict the stock market. Experts predict the stock market using complex and partially non-numerical data, while their knowledge is imperfect and uncertain. As stock market prediction involves imprecise concepts and imprecise reasoning, fuzzy logic Zadeh [12] appears to be a natural choice for knowledge representation.

Fuzzy logic treats vague concepts as "linguistic variables": variables whose values are not numbers but words. These values are defined using fuzzy sets. A fuzzy set represents the meaning of a vague concept in a precise mathematical way by a membership function, which expresses the degree to which an element of the universe of discourse belongs to the fuzzy set. If X is a linguistic variable in a universe of discourse and L a linguistic term defined by a fuzzy set, the membership function $f_L(X)$ specifies the degree to which X belongs to the fuzzy set denoted by L. This degree is the truth value of the fuzzy proposition p: X = L, stating that L applies to X.

Fuzzy inference relates fuzzy propositions by applying fuzzy production rules. The rule's antecedent expresses one or more conditions in terms of a fuzzy proposition. Fuzzy operators compute the truth value of a compound condition. The rule's conclusion likewise is a fuzzy proposition, and application of the rules in the knowledge base creates a fuzzy set representing the conclusion. A "defuzzification" transforms this fuzzy set into a concrete statement. An example of a defuzzification method is the center of gravity method.

5. Fuzzy logic to represent knowledge to predict the stock market

Knowledge about the concepts used in the approach to predict the stock market as presented in figure 2 can be represented as fuzzy propositions, while knowledge to relate these concepts can be represented by fuzzy rules. A number of cases for the NYSE for the period 1972-1991 were created in order to identify relevant knowledge. Interviews with several experts guided the formulation of fuzzy sets and rules.

Excess return, market valuation, the business cycle, and interest rates are considered linguistic variables. The rules relate fuzzy propositions about the business cycle, interest rates and market valuation to the excess return. For example, the following rule concludes that the excess return will be very high.

```
if
BusinessCycle:Phase = Boom and
InterestRates:Pressure = Down and
Market:Valuation = Low
then
Market:ExcessReturn is VeryHigh
```

Figure 4, rule concluding that the market excess return is very high

The membership functions are of a simple symmetrical kind. The 'and' operator used in the rules computes the truth value of the condition by taking the lowest of the truth values of the components. All rules have three components, referring to the three input variables.

Note that the rule combines information that has a quantitative character and information that is purely symbolic. Input for the variable Market:Valuation for example is a number, while input for the variable InterestRates:Pressure is a symbol.

The system corresponds to the way portfolio managers work, and as such offers interactive problem solving. Benefits are that the portfoliomanager can formalize and refine vague knowledge, analyzing the consequences of adjustments of particular rules or vague concepts.

6. Specifying the tactical portfolio

The mean-variance portfolio selection model (see for example Haugen [3]) is in wide use for configuring stock portfolios, and this section demonstrates how to apply it to TAA.

The mean-variance model characterizes portfolios in terms of expected return and risk. Given the investor's risk attitude, the model selects the optimal portfolio from the set of efficient portfolios. Efficient portfolios are those portfolios that have minimum risk for a particular expected return. Portfolios that are not in this category are discarded, since a portfolio with identical expected return but higher risk is inferior to a portfolio with lower risk. In order to determine the efficient set, the model needs expected returns, standard deviations and covariances of the investment alternatives as input. The optimal portfolio is the portfolio with the highest utility (the most preferred combination of expected return and risk).

If short term expectations of returns differ from long term expectations, the mean-variance model will calculate an efficient set for the near term which is different from the long term efficient set. The strategic portfolio will no longer be the optimal portfolio. The principle of using the mean-variance model for TAA optimization is to balance additional expected utility and tracking error variance, which is the variance of the return on the actual portfolio minus the return on the strategic portfolio. Adjusting the portfolio to resemble more closely the optimal portfolio adds to the portfolio's expected utility, but introduces the risk of underperforming the strategic portfolio. This risk is expressed by the variance of tracking error and can be viewed as active risk: the risk taken on purpose to achieve better investment results. The investor should specify preferred combinations of additional portfolio optimality and tracking error variance, in order to determine the portfolio with optimal tradeoff.

Figure 5 shows a graphic which the optimization component shows to illustrate the tradeoff among portfolio optimality and tracking error variance in a particular problem setting. The strategic portfolio lies to the left, with tracking error variance = 0. The graph shows that after an initial increase, portfolio optimality starts to drop. Portfolios beyond this point are clearly not optimal.

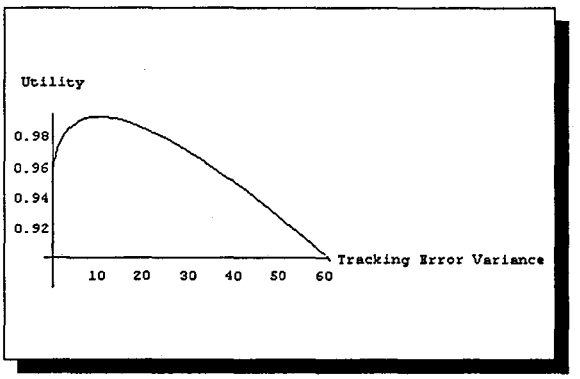


Figure 5, Tradeoff among portfolio optimality and tracking error variance

To facilitate the analysis, portfolio optimality is normalized by relating it to the strategic portfolio's optimality under long term conditions. Optimality of the strategic portfolio higher than 1 means favorable short term investment conditions, an optimality lower than 1 means poor short term investment conditions.

Since utility nor tracking error variance are linear functions, balancing utility and tracking error variance is a complex optimization problem. In the case of asset allocation, where the number of investment alternatives under consideration is limited, a simple solution is to enumerate portfolios near the strategic portfolio, and to determine the one with the optimal tradeoff.

7. Implementation and performance

The major part of the system is implemented in Mathematica, a general programming environment offering numerical, symbolic and graphical computing (Wolfram [8]). The basic prediction sequence is data input, fuzzy inference, and finally defuzzification. Performance of the fuzzy logic predictor was evaluated by feeding the system the set of cases, and by comparing

the results to historical returns on an annual basis (main data source: Ibbotson and Sinquefield [4]).

Overall performance was evaluated by comparing the risk-return properties of the strategy suggested by the optimization model to the results obtained with a rebalancing strategy. One option is to compare the results to a rebalancing strategy which maintains the strategic portfolio. An alternative is to compare the standard deviation of the suggested strategy to the standard deviation of that particular rebalancing strategy which would have generated the same return. This way it is immediately clear if the results of the suggested strategy indeed originate from tactical adjustments, rather than from another diversification. This is important since a biased TAA system blurres strategic and tactical asset allocation. For example, if the system consistenly overestimates an asset class return, it will be biased towards this class, and effectively change the long term diversification. As TAA aims at optimizing the portfolio's risk-return properties, obviously a biased system which chooses systematically for a biased asset mix is of little use.

For reasons of simplicity, cash was omitted from the analysis. The input for the optimization model was generated as follows. The fuzzy logic predictor provided an estimate of the return on stocks. Yield to maturity was used to predict the return on bonds. Standard deviations and the covariance between stocks and bonds were estimated using the historic data. A standard utility function specified portfolio optimality, and was used to determine the strategic portfolio. This portfolio had a 0.4 stock weight. A function similar to the utility function specified the preferences as to additional portfolio optimality and variance of tracking error. The historic data covered the period 1972-1987.

information coefficient	0.23
TAA policy return	9.09
TAA policy standard deviation	13.4
rebalancing standard deviation	13.4

Table 1

Table 1 shows the results. The correlation on an annual basis between predicted excess return and realization, the information coefficient, is 0.23, which is not particularly high (DuBois [1]). This is reflected in the standard deviation of the suggested strategy. A rebalancing strategy yielding a 9.09% annual return had the exactly the same standard deviation.

The test points out that in its current state the system does not outperform a rebalancing strategy. However, the test indicates minimum performance of the system. Since the fuzzy logic predictor can be refined, and the inclusion of other estimates should improve performance, the system promises potential to add value to TAA.

To improve performance, the fuzzy logic predictor can be refined in various ways. Membership functions can be refined, rules can be added and more advanced operators to combine rule conditions can be used. Another way to improve performance is to feed the optimization model with more accurate estimates of standard deviations and covariance. Options are to use implied volatilities, or estimates based on recent market volatility. In addition, in order to estimate the short term return on bonds, yield to maturity is convenient, but not the most accurate.

8. Conclusion

The paper presented a fuzzy logic system to predict the stock market excess return. The system offers interactive problem solving and supports knowledge refinement. The system shows acceptable performance for its life cycle stage. Further development depends upon the quality of expert knowledge, as well as the success in eliciting and modeling this knowledge. Constructing a fuzzy logic system does not make much sense if the required knowledge is not available or too complex to model, or if learning techniques (e.g. Kosko [6]) perform better.

The optimization model for TAA offers an elegant framework for adjusting the strategic portfolio. However, the model is as good as its input. The quality of the suggested strategy depends heavily not only on forecasts of returns, but also on estimates of standard deviations and covariances.

References

[1] DuBois, Charles H. (1992), "Tactical Asset Allocation: A Review of Current Techniques", in: Robert D. Arnott and Frank J. Fabozzi (eds.), *Active Asset Allocation*, McGraw-Hill, London

[2] Hammer, David A. (1991), *Dynamic Asset Allocation*, John Wiley and Sons, New York
[3] Haugen, Robert H. (1990), *Modern Investment Theory*, second edition, Prentice Hall, Englewood Cliffs, NJ
[4] Ibbotson, Roger G. and Rex A. Sinquefield (1989), *Stocks, Bonds, Bills, and Inflation: Historical Returns (1926-1987)*, The Research Foundation of The Institute of Chartered Financial Analysts, Charlottesville
[5] Kandel, Abraham (ed.) (1992), *Fuzzy Expert Systems*, CRC Press, Boca Raton
[6] Kosko, Bart (1991), *Neural Networks and Fuzzy Systems*, Prenctice-Hall
[7] Sharpe, William F. (1987), "Integrated Asset Allocation", *Financial Analysts Journal*, 43, September-October, pp. 25-32
[8] Wolfram, Stephen (1991), *Mathematica*, second edition, Addison Wesley, Redwood City, CA
[9] Wong, F.S., P.Z. Wang, T.H. Goh and B.K. Quek (1992), "Fuzzy Neural Systems for Stock Selection", *Financial Analysts Journal*, January-February, pp. 47-52
[10] Yager, Ronald R. and Lofti A. Zadeh (1992), *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Publishers, Dordrecht
[11] Yasunobu, C., M. Kosaka, K. Yokomura and K. Honda (1992), "A Decision Support System Building Tool with Fuzzy Logic and its Application to Chart Technical Analysis", *Proceedings of the 1992 BANKAI Workshop*, S.W.I.F.T. (ed.), Elsevier Science Publishers
[12] Zadeh, Lofti A. (1965), "Fuzzy Sets", *Information and Control*, 8, pp. 338-353

USING GENETIC ALGORITHMS TO SOLVE FINANCIAL PORTFOLIO PROBLEMS RELATED TO OPTIMAL ALLOCATION, PORTFOLIO INSURANCE, AND PERFORMANCE PREDICTION

M. L. GARGANO
COMPUTER SCIENCE

PAULA CHAMOUN
COMPUTER SCIENCE

D. L. von KLEECK
MANAGEMENT SCIENCE

PACE UNIVERSITY, New York, N.Y. 10038

ABSTRACT

There are many real world financial optimization problems that are considered innately difficult to solve. Genetic algorithms are biologically inspired tools that borrow concepts from natural selection to aid in solving such problems. These genetic concepts are survival of the fittest, crossover, and mutation. Three applications using these powerful techniques are presented here. First, we show how coarse grained space search type solutions of discontinuous and/or highly multimodal portfolio optimization problems can be closely estimated using the genetic algorithm paradigm. Then, we use the genetic algorithm methodology to solve a portfolio insurance problem. The genetic algorithm paradigm can be applied in the construction of a portfolio with an investment mix that will closely approximate a predetermined payoff initially desired by the investor. The final application deals with portfolio prediction.

1. INTRODUCTION

There are many real world financial optimization problems that are considered innately difficult to solve. Genetic algorithms are biologically inspired tools that borrow concepts from natural selection to aid in solving such problems.

These genetic concepts are survival of the fittest, crossover, and mutation. Three methods of using these powerful techniques are presented here. First, we show how coarse grained space search type solutions of discontinuous and/or highly multimodal portfolio optimization problems can be closely estimated using the genetic algorithm paradigm. Then, we use the genetic algorithm methodology to solve a portfolio insurance problem. The genetic algorithm paradigm can be applied in the construction of a portfolio with an investment mix that will closely approximate a predetermined payoff initially desired by the investor. The final application deals with portfolio prediction.

2. OPTIMIZATION PROBLEMS

We will first describe a genetic algorithmic approach to solving optimization problems by considering the following illustrative examples.

EXAMPLE I

An investor has D dollars to allocate amongst various investment alternatives with the objective of maximizing annual return. The constraints might consist of: the minimum total investment to be made, a limit on the average risk assumed, maximum and/or minimum limits set on particular investments (e.g., government bonds, money market funds, high tech stocks, municipal bonds, etc.), and so forth.

This problem can be modeled as follows:

MAXIMIZE

F(f1,f2,...,fn) = Annual Return

subject to:

Ri(f1,f2,...,fn) <= Ci $1 \leq i \leq m$

Rj(f1,f2,...,fn) >= Cj $m+1 \leq j \leq p$

(f1,f2,...,fn) >= O

where the fi are the portfolio allocations and the inequalities represent the investment constraints.

EXAMPLE II

An other investor also has D dollars to allocate amongst various investment alternatives. This investor, however, has the objective of minimizing total risk. The constraints might consist of: the minimum total investment

to be made, minimum level of return desired, maximum and/or minimum limits set on particular investments (e.g., government bonds, money market funds, high tech stocks, municipal bonds, etc.), and so forth.

This problem can be modeled as follows:

MINIMIZE

F(f1,f2,...,fn) = Total Risk

subject to:

Ri(f1,f2,...,fn) <= Ci 1 <= i <= m

Rj(f1,f2,...,fn) >= Cj m+1 <= j <= p

(f1,f2,...,fn) >= O

where the fi are the portfolio allocations and the inequalities represent the investment constraints.

The difficulty with these types of problems is that in the real world the functions and inequalities one must deal with are very often of the form such that traditional analysis based on mathematical programming methodologies will not work.

An optimizing genetic algorithm can oft times be used to solve such problems. Although, optimal results are not guaranteed, the results are usually very good.

Our genetic algorithm requires an initial population of feasible points, an evaluation function, conventions for creating potential new members of the population by mating or random mutation, and a grim reaper mechanism to delete poorly performing members so as to make room for new ones.

In both of these optimization problems the population POP = { (f1,f2,...,fn) } is a large subset of all the possible feasible points. POP is initially chosen randomly and consists of a large but finite number of points. If each fi is encoded in whole number form, then we can view each member of the population as a long string of decimal digits of length (k * n). For example, if n = 3 and k = 8 then (f1,f2,f3) = (250000, 1000000, 600000) can be coded as 00250000010000000 600000 a string of length 24.

In these problems we will use the objective function F(f1,f2,...,f3) to evaluate the performance or worth of individual members of the population. Only the better members will survive and help in continuing the search for an optimal solution.

The mating convention is as follows. Two points P1 and P2 are randomly chosen from the population to be mated. One point, say P1, is selected from the high performers while the other point P2 is chosen randomly. The genetic algorithmic mechanism of crossover now can be used to replace two low performing points Q1 and Q2 with the offspring C1 and C2 which are the result of mating the parent points P1 and P2. The deletion of Q1 and Q2 from the population is handled by a grim reaper mechanism ..

Let:

P1 = v1 v2 v3 . . . va va+1 . . . vb vb+1 . . . vkn

and

P2 = w1 w2 w3 . . . wa wa+1 . . . wb wb+1 . . . wkn

First, we randomly choose the crossover positions, say a and b. The offspring will then be created by swapping substrings from each of the parents. The parents will continue to remain in the population. The children created by swapping the parents substrings is displayed below.

C1 = v1 v2 v3 . . . va wa+1 . . . wb vb+1 . . . vkn

and

C2 = w1 w2 w3 . . . wa va+1 . . . vb wb+1 . . . wkn

The offspring C1 and C2 are then checked for feasibility. The offspring that are feasible replace the low performing Q1 and Q2 in the population.

Mutation is the genetic algorithmic mechanism where we randomly choose a point P in population then change one randomly chosen digit in its string. This is not done very frequently, nevertheless, it is useful in creating new areas to search. If the mutation P" results in a feasible point it then replaces P in the population.

We can now state the genetic algorithm.

Step 1 Initialize a large feasible population.

Step 2. Evaluate any member which has not yet been evaluated.

Step 3. Randomly choose and mate P1 and P2 creating C1 and C2

Step 4. Replace badly performing members of the popula tion with the feasible offspring.

Step 5. Infrequently mutate a random member of the population.

Step 6. If time is up then return the best solution found and stop else return to Step 2 and continue.

3. PORTFOLIO INSURANCE

Let S(t) be the total amount to be invested at time t. Consider the payoff functions of n mixed portfolios P1, P2, ..., Pn which have been randomly selected (see figure 1). Each payoff function Pi evaluates the payoff of portfolio Pi given the value S(T) of some financial index at time T (typically t plus one year). If pi * S(t) are the amounts allocated to each portfolio then the resulting effective payoff is P(T) which is the sum of the pi * Pi as i ranges from 1 to n (see figure 2).

A portfolio insurance strategy attempts to find an investment mix that closely approximates a predetermined payoff function D which is desired by the investor. The payoff function D typically has very limited loss potential while also providing for very large potential gains (see figure 2). This would of course be very attractive to many investors.

To solve this problem using a genetic algorithm methodology we require an initial population of points POP = { (p1,p2,...,pn) } where pi is the proportion of the investment S(t) allocated to portfolio Pi. This implies the the sum of the pi equal s 1.

As an evaluation function, we can find how closely a member of the population approximates the desired payoff function D. This might be accomplished by calculating the square root of the definite integral of the square of the difference of the desired function D with a payoff P(T) (which represents the effective payoff of that member of the population).

The operations of crossover and mutation are similar to those previously discussed. However, a normalization process must take place after each operation to insure the sum of the values in a potential population member totals 1.

We can now state the genetic algorithm for the portfolio insurance problem.

Step 1. Initialize a large feasible (i.e., normalized) population.

Step 2. Evaluate any member which has not yet been evaluated to see how closely it approximates D.

Step 3. Randomly choose and mate P1 and P2 creating C1 and C2 by normalizing the results of the crossover operation.

Step 4. Replace badly performing members of the population with the offspring.

Step 5. Infrequently mutate a random member of the population and normalize the results of this operation.

Step 6. If time is up then return the best solution found and stop else return to Step 2 and continue.

4. PORTFOLIO PREDICTION

Again, let S(t) be the total amount to be invested at time t and consider the payoff functions of n mixed portfolios P1, P2, ..., Pn which have been randomly selected. Since each payoff function Pi evaluates the payoff of portfolio Pi given the value S(T) of some financial index at time T (typically t plus one year), a forecast of S(i) might be desirable over the time period t+1 to T.

There are many quantitative forecasting models to choose from, for example, moving averages (ma), exponential smoothing (es), least squares (ls), regression analysis (ra), adaptive methods (am), etc. Some models may perform better than others over the time period of interest. It is therefore desirable to find an optimal sequence of forecasting methods which when applied to certain time intervals would do a better overall job at predicting than any single method could do.

A genetic algorithm can be applied to a population of forecasting method sequences where POP = { (fm(t+1), fm(t+2),...,fm(T)) }. A sample sequence (es(pi), ma(pj), ma(pk),...) would tell us to use an exponential smoothing model with parameter set pi to forecast S(t+1) from S(t) and any prior values, then to use a moving average model with parameter set pj to forecast S(t+2) from S(t+1) along with S(t) and any prior values and so forth.

By applying each member of the population to similar past time series values, we can evaluate which sequences are the better performers. The mean absolute deviation (MAD) can therefore be used as a simple evaluator function to test the fitness of every member of the population.

The operations of crossover and mutation will create new forecasting sequences which are potentially better predictors than members of prior generations of the population.

When the genetic algorithm returns what it has found to be a good predicting sequence, we can use it to

forecast S(T).

We can now state the genetic algorithm for the forecasting problem.

Step 1. Initialize a large population of forecasting sequences.

Step 2. Evaluate any member which has not yet been evaluated to see how well it performs using MAD.

Step 3. Randomly choose and mate P1 and P2 creating C1 and C2 by using the crossover operation.

Step 4. Replace badly performing members of the population with the offspring.

Step 5. Infrequently mutate a random member of the population

Step 6. If time is up then return the best solution found and stop else return to Step 2 and continue.

5. CONCLUSIONS

We have been experimenting with these genetic algorithmic models on small problems and have been encouraged by the results thus far. We feel that the genetic algorithm paradigm is both powerful and flexible when trying to deal with portfolio pr problems.

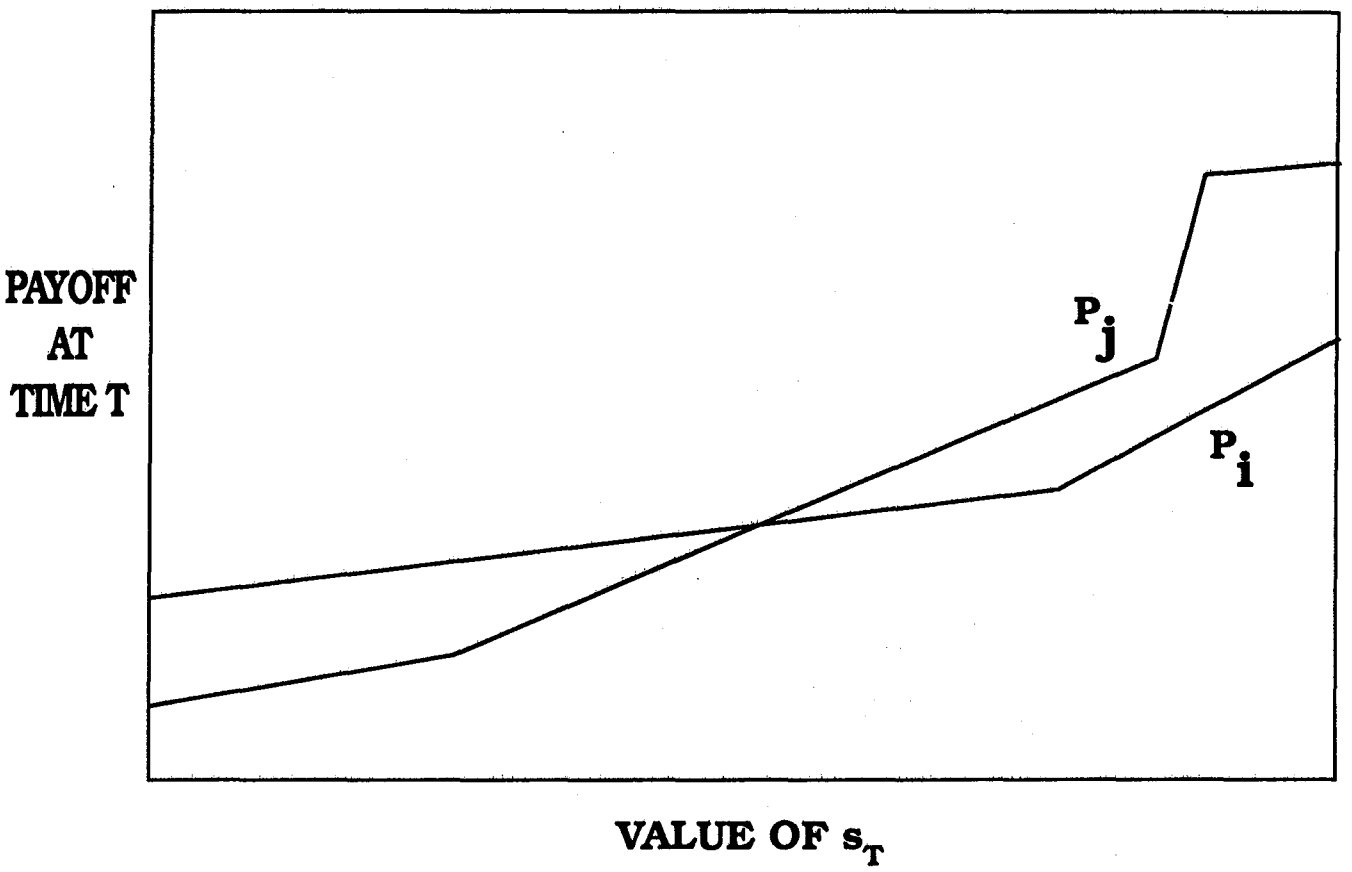


FIGURE 1

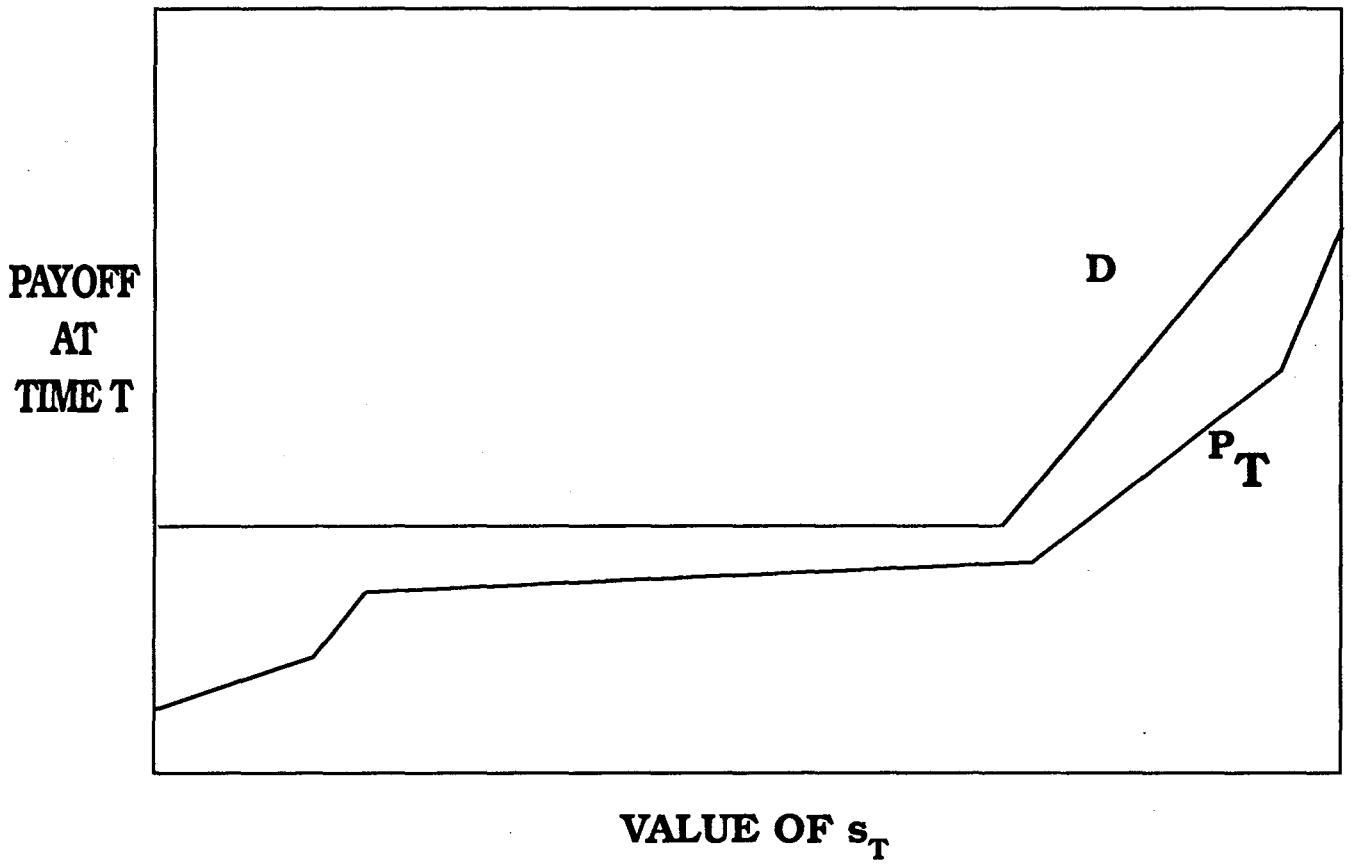


FIGURE 2

REFERENCES

DAVIS,L.
(1989) Handbook of Genetic Algorithms Van Nostrand Reinhold Publishing

DUFFIE,D.
(1989) Futures Markets Prentice Hall Publishing

GARGANO,M.L.
(1990) Classifier Voting in Neural Networks Proceedings of the Second International Neural Network Society (INNS) Conference

GARGANO,M.L., MAROSE,R.A., von KLEECK,D.L.
(1990) An Application of Neural Networks to Software Development Proceedings of the Eight International Conference on Cybernetics and Systems

GARGANO,M.L., MAROSE,R.A., von KLEECK,D.L.
(1991) An Application of Artificial Neural Networks and Genetic Algorithms to Personnel Selection in the Finance Industry Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street

GARGANO,M.L., von KLEECK,D.L.
(1992) A Proposed Application of Artificial Neural Networks and Genetic Algorithms to Measuring Corporate Social Performance (submitted to Computers and Society a publication of the ACM SIG on Computers and Society

GOLDBERG,D.
(1989) Genetic Algorithms Addison Wesley Publishing

PIASCIK,C.
(1992) Applied Mathematics for Business and the Social and Natural Sciences West Publishing

SIGMA PLUS INC.
(1989) An Example of ADAP at Work

WASSERMAN,P.D.
(1989) Neural Computing Van Nostrand Reinhold

Detecting Anomalous Risk Behaviors in Portfolio Management Strategies

Earl Cox
CEO/Chief Technical Officer
The Metus Systems Group
1 Griggs Lane, Chappaqua, New York 10514
(914) 238-0647

Abstract

Strategies for minimizing the risk of an investment program while maximizing the potential revenue have been developed that concentrate on the variability of the stocks or on the predicted revenues flow. This strategy does not address another problem: detecting cases where the portfolio mix strategy has a risk level inconsistent with good management. This occurs either because the manager has failed to consider some aspects of the market or a deliberate increase in risk has been assumed. This paper addresses an approach to this problem based on an analysis of "behavior" patterns within the portfolio. The method using a non-parametric anomaly detecting system that uses both the idea of peer group analysis and the law of large numbers to find instances where the behavior of a portfolio manager is at significant variance from his/her peers.

1. Overview

The Risk Detecting System takes a new approach to isolating probable anomalous behaviors with increased risk. This approach uses a fuzzy nonparametric anomaly detecting algorithm. In essence we look to the portfolio itself to find unusual behavior by considering peer versus individual manager behavior. The degree of variance provides a metric in discovering possibly high risk or inappropriate management. We have identified nearly 90 activities that could indicate possible high risk or unusual management practices. Of these, we selected 27 activities, called "Behavior Patterns" that (1) could be reasonably automated and (2) would uncover a significant portion of the problem behavior. High risk behavior is detected by comparing individual managers of a specific type for a specific behavior pattern against the behavior of his/her peers of the same type in a single portfolio type for that behavior. We Are also interested in the predictive nature of the model in isolating behaviors that are trending toward the anomolous frontier. This means we want to examine a time series dimension in the model to detect such conditions as,

if Δ(BP[t],BP[t-1]) is significant
then Risk is Increased

Of course, in reality, such changes between neighboring periods are rarely significant, we only observe important anomolous behavior when it movesmonotonically (or "nearly" monotonically) across a larger time frame. Thus the square of the variances might prove a netter metric,

$$\sum_{i=0}^{n}(BP(t_i)-BP(t_{i-k}))^2$$

In this paper, however, we will examine the model analysis in a single time window.

Basically, the system develops a detailed statistical profile of peer behavior for each pattern. The degree to which a manager varies from this distribution function is used to find probable high risk activities.

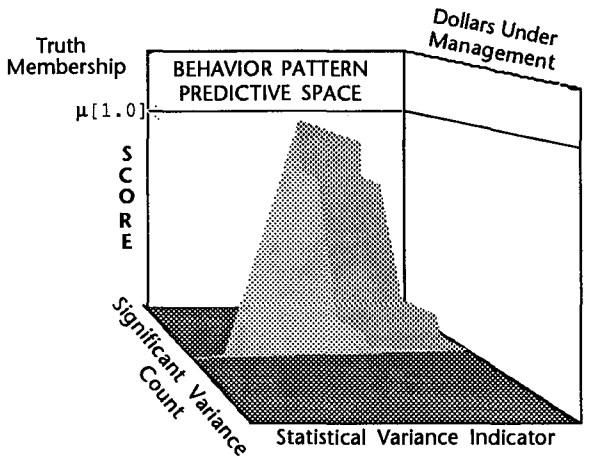


Figure 1. The Risk Detection Space

2. Model Philosophy

Each behavior pattern is associated with an "information cohesion index" indicating the statistical variance from the peer norm. This number is then plotted against the total dollars; that is, the effective dollar amount under management. We view this in terms of a zero-sum game strategy where the manager is "paid" the amount under management. Thus, variances against high dollars will be the primary selector for unacceptable risk. This appears as,

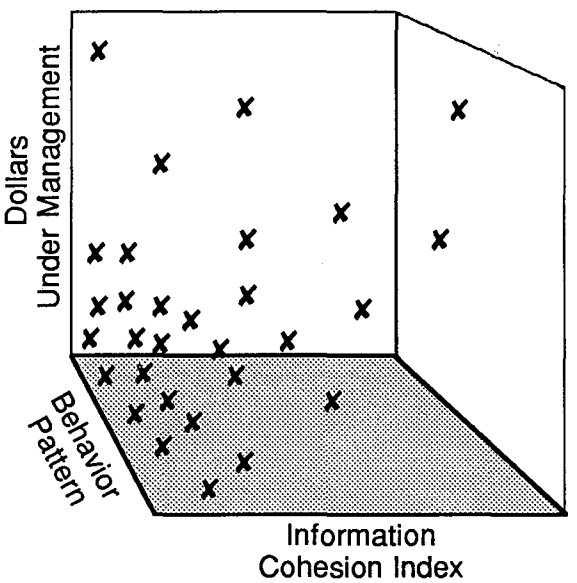


Figure 2. Behavior Pattern Dispersal

Once the information cohesion indexes have been arrayed across the behavior patterns and dollars under management, the expert system identifies the probable risks by establishing and slicing off points that fall on a set of outlier frontiers. As an example, the first outlier level might appears as

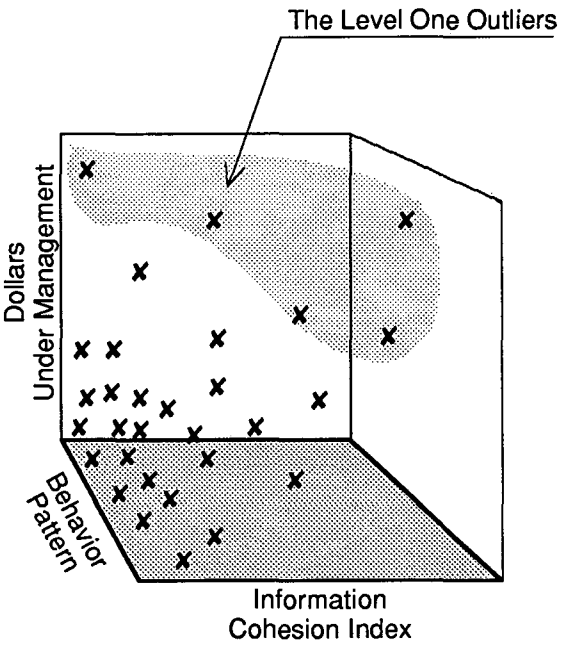


Figure 3. Finding the highest anomaly level

Once these points have been selected and counted, the system looks at the second frontier. These are managers that fall somewhere between the first frontier layer and the central mass of normal behavior points. Note that frontiers are not parametric. A frontier layer is developed in the same way that peeling an onion results in layers.

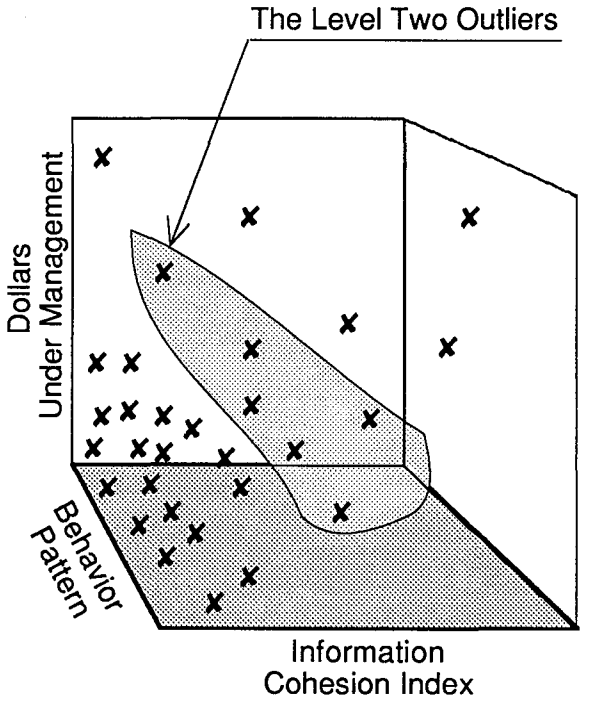


Figure 4. Finding the minimal anomaly level

3. Detecting Anomalous Behavior

The expert system counts the number of behavior patterns that are in the frontier for each manager. This count is the "risk indicator" number. The higher this number, the more times a manager was statistically outside the norm for his/her peer group. Managers are ranked by this risk indicator number. In a population of portfolio managers we would like to see a distribution that represents a cohesive clustering of behaviors around the mean for each behavior pattern, somewhat like that seen in Figure 5.

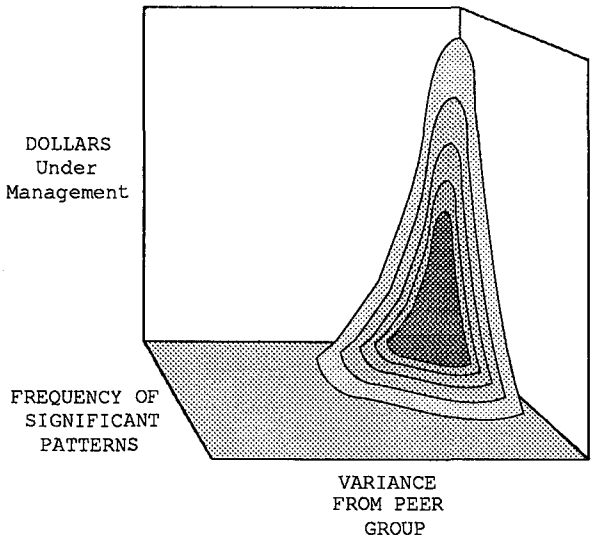


Figure 5. The Space manifold distribution for Managers

The concept of peer anomaly detection means looking at individual managers across each of the appropriate behavior patterns. When we do this, we find a temporal drift away from the peer mass for those managers whose behaviors are significantly different. Figure 6 illustrates (somewhat amplified) how these populations become distinct and detectable.

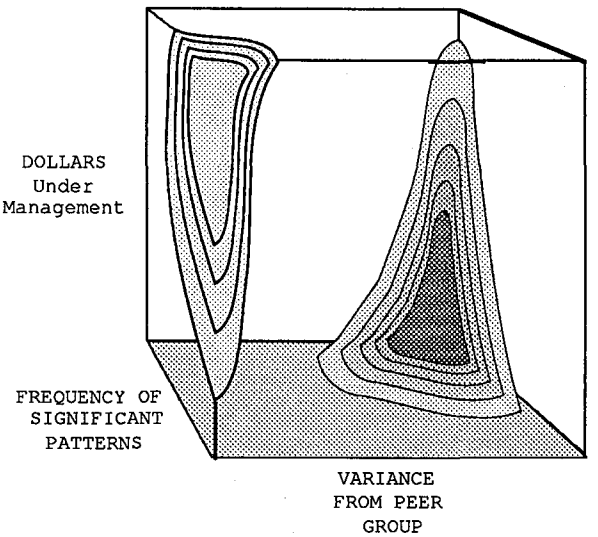


Figure 6. Distribution of Anomalous Behaviors

4. Fuzzy Expert System Design

In addressing the risk assessment and isolation problem, we have taken a quite different approach. Using fuzzy reasoning capabilities we want to evaluate all the cohesion index number and dollar values as factors effecting the general concept "manager is a risk". Additionally, we decided that the number of behavior triggers (x out of y) that are "significant" would also influence the risk concept by increasing the manager's membership function. This linguistic space bounded by three fuzzy sets, establishes the degree to which a manager is a risk. This appears schematically as,

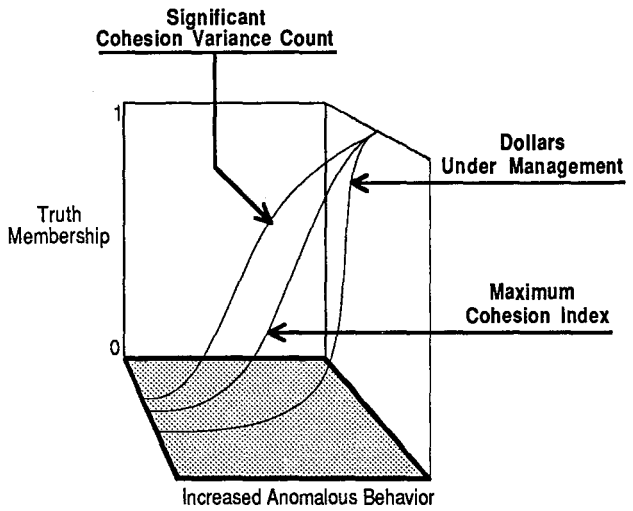


Figure 7. Underlying Fuzzy Descriptions

Each behavior pattern contributes to the surface of the consequent fuzzy set through the application of a single rule. The rule uses the Yager geometric compensatory AND operator to balance the truth values between the various conditions. The rule appears as,

```
if (cohesionIndex is high
      and dollars are high)
      and sigBPcount is significant
      then Risk is increased;
```

5. Calibrating the Risk Metric

As these behavior patterns are processed, a single fuzzy set is created reflecting the degree to which the manager is exhibiting anomalous (perhaps risky) behavior. This final surface is "defuzzified" to derive an overall score.

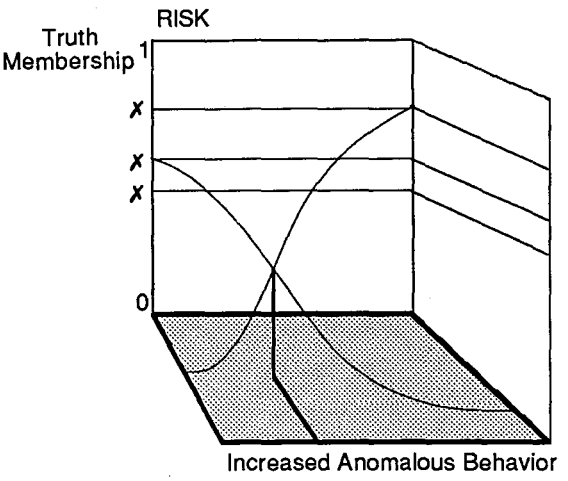


Figure 8. Defuzzification to find Risk Level

It is this risk index number that indicates whether or not the manager is engaging in possible high risk activities. By ranking these numbers we produce a list for the Unethical activities or unacceptable risk detection unit.

6. Conclusions

The risk analysis system provides a powerful and flexible method of evaluating imprecise anomalous behavior in a population of similar activities. The use of the fuzzy approximate reasoning engine allows the sensitivity of the system to be calibrated for a wide range of behaviors. The ability to tune a fuzzy logic system in several degrees-of-freedom—the rules, the shape and properties of fuzzy sets, the implication method, hedge surface transformations, operator interpretation, and defuzzification—means that such systems can become highly responsive to complex, poorly understood business applications.

7. References

[1] Cox.E.. Adaptive Fuzzy Systems, IEEE Spectrum Magazine, Vol. 30, No. 2, February, 1993, pages 67-70.

[2] ——. Applications of Fuzzy System Models, AI Expert Magazine, Vol. 7, No. 10, October, 1992, pages 33-45.

[3] ——. Fuzzy Fundamentals, IEEE Spectrum Magazine, Vol. 29, No. 10, October, 1992, pages 58-61.

[4] ——. Integrating Fuzzy Logic into Neural Nets, AI Expert Magazine, Vol. 7, No. 6, June 1992, pages 42-47

[5] ——. Solving Problems with Fuzzy Logic, AI Expert Magazine, Vol. 7, No. 3, March 1992, pages 28-37

[6]——, Myths of Fuzzy Logic, AI Expert Magazine, Vol. 7, No. 1, January, 1992.

[7] Dubois, D.J. and Prade H. Operations in a Fuzzy-Valued Logic, *Information and Control*, vol. 43, no. 2 (November 1972), 224-240

[8] ——, *Non-Standard Logics for Automated Reasoning*, New York: Academic Press (1988)

[9] ——., *Possibility Theory - An approach to Computerized Processing of Uncertainty*, New York: Plenum Press (1988)

[10] Gaines, B.R. *Foundations of Fuzzy Reasoning, in Fuzzy Automata and Decision Processes*, Gupta, Saridis, and Gaines (Eds.), New York: Elsevier North-Holland, (1977), 1975

[11] Hoon, K. and Vachtsevanos, G. Adaptive Fuzzy Logic Control, In Proceedings of IEEE International Conference on Fuzzy Systems 1992, pages 407-414, San Diego, CA. March 1991.

[12] Huntsberger, T. and Ajjimarangsee, P. Parallel Self-Organizing Feature Maps for Unsupervised Pattern Recognition, Int. J. General Systems, 16, pages 357-372, 1990.

[13] Klir, George J, and Folger, Tina A. *Fuzzy Sets, Uncertainty, and Information*, New Jersey: Prentice Hall (1988)

[14] Kosko, B. *Neural Networks and Fuzzy Systems — A Dynamical Systems Approach to Machine Intelligence*, New Jersey: Prentice Hall (1992)

[15] Lee C.C. Intelligent Control Based on Fuzzy Logic and Neural Net Theory, In Proceedings of the International Conference on Fuzzy Logic and Neural Networks, ed. T. Yamakawa, Iizuka, Japan, pages 759-764, 1990.

[16] ——, A Self-Learning Rule-Based Controller Employing Approximate Reasoning and Neural Net Concepts, Intl. Jour. Intelligent Systems, 6, pages 71-93, 1991.

[17] Lee S.C. and Lee, E.T. Fuzzy Neural Networks, Mathematical Biosciences, 23, pages 151-177, 1975

[18] Wee, W.G. and Fu, K.S. A Formulation of Fuzzy Automata and its Application as a Model of Learning Systems, IEEE Trans. Syst. Sci. and Cybernetics, SSC-5, pages 215-223, 1969

[19] Zadeh, L.A. Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges, *Journal of Cybernetics*, vol. 2, no. 3 (1972), 4-34

[20] ——, Fuzzy Sets versus Probability, Proceedings of the IEEE, vol. 68, no. 3 (March 1980), 421ff

[21] ——, Quantitative Fuzzy Semantics, Information Sciences, vol. 3, no. 2, (April 1971), 159-176

[22] ——, Fuzzy Sets, Information and Control, vol. 8, (1965), 338-353

[23] Zimmermann, H.J, *Fuzzy Sets, Decision Making, and Expert Systems*, Norwell, MA: Kluwer Academic Press (1987)

[24] ——, *Fuzzy Set Theory — and its Applications*, Norwell, MA: Kluwer Academic Publishers (1985)

A Comparison of Stochastic Search Heuristics for Portfolio Optimization

Roy S. Freedman & Rinaldo DiGiorgio
Inductive Solutions, Inc.
New York, NY 10280

Abstract

Modern portfolio theory is based on a rational investor choosing the proportions of assets in a portfolio so as to minimize risk and maximize the expected return. In this paper, we investigate the applicability of different stochastic search heuristics to the problem of finding the optimum portfolio. We compare their performance on two problems with known solutions.

1. Portfolio Optimization

Given a set of assets, what is the optimum proportion of each that is required to achieve an investment objective? Modern portfolio theory is based on a rational investor choosing these proportions so as to minimize risk and maximize the expected return. If risk is measured in terms of the variance of the resulting portfolio, then portfolio optimization is reduced to a "means-variance paradigm" — the optimal portfolio can be derived by knowing the expectations of returns and correlations of returuns for all assets.

Using vector notation from linear algebra, in its simplest form, the portfolio problem is to

Find a vector $\mathbf{w} = (w_1, \ldots w_n)$
that maximizes the return/risk ratio

$$\frac{[\mathbf{r} \bullet \mathbf{w}]}{(\mathbf{w}^T \bullet \mathbf{C} \bullet \mathbf{w})^{1/2}}$$

subject to constraints

$$m_k \le w_k \le M_k, \; k=1,\ldots,n$$
$$\text{and } (|w_1| + |w_2| + \ldots + |w_n|) = 1$$

The vector **w** defines the portfolio: the set of weights that represent the proportion of each asset. Vector **r** is the vector of expected returns for the assets. **C** is a symmetric n-by-n matrix that represents the covariance beteween the returns: c(i,j) is the return covariance between the returns of asset i and asset j:

$$c(i,j) = \sigma_i \, \sigma_j \, \rho_{ij}$$

where ρ_{ij} is the correlation coefficient between the returns for asset i and asset j, and σ_i is the standard deviation of return on asset i. These quantities must be measured statistically. If short sales are allowed in this portfolio then the mimimal constraints mk can be negative. In this case, the absolute value constraints follow the definition of Lintner short sales [1]. Any portfolio **w** that satisfies the constraints is a *feasible solution* to the problem. The goal is to find the best feasible solution.

It is important to note that the above problem can be generalized to include "nonstandard" measures of return or risk, such as geometric return or partial moments, and other types of constraints, such as asset sector limitations.

The portfolio problem represented in the above form is a quadratic programming problem, which can be solved by traditional methods which are variations of the revised simplex linear programming algorithm or gradient-search (hill-climbing) techniques. For some portfolios, these methods may take a long time to solve and, because of the constraints imposed by the problem, may be awkward to initialize to a feasible solution. For example, the simplex algorithm for linear programming exhibits exponential complexity for the worst-case, and polynomial complexity for the average case [2].

Stochastic search algorithms are useful for applications where stable and acceptable (ie, near optimal) answers are desired quickly. In general, stochastic search algorithms do

not require knowledge of a derivative (as in gradient-search methods) and perform best with highly nonlinear or highly combinatorial problems. Even though their worst-case behavior is also exponential, their average performance to yield acceptable answers may be quadratic [2]. In this paper, we investigate the applicability of three different heuristics, some having been inspired by biological processes, and compare their performance on two problems with known solutions.

2. Example Problems

Problem A is a simple textbook problem with three assets [1]. There are three assets: the returns and covariance matrix are

	Covariance			
Returns		A1	A2	A3
14	A1	36	9	18
8	A2	9	9	18
20	A3	18	18	225

The constraints are $0 \le w_k \le 1$ together with the summation equality constraint $\Sigma w_k = 1$. The theoretical optimum solution is $\mathbf{w}$ = (14/18, 1/18, 3/18) with Return/Risk ~ 1.66.

Problem B was discussed in [3] and reflects a real example for allocating assets among US and foreign bonds. The returns and covariance matrix for these assets are

		Covariance						
		U.S.	Canada	German	Japan	U.K.	Dutch	French
9.75	U.S.	126	115	64.93	56.7	69	63.5	48.8
10.03	Canada	115	195	94.34	71.7	106	85.4	72.7
9.81	German	64.9	94.32	61.8	179	161	236	201
15.42	Japan	56.7	71.71	78.7	300	154	169	166
12.57	U.K.	69	106	160.9	154	335	149	127
10.48	Dutch	63.5	85.42	36.1	169	149	230	191
10.09	French	48.8	72.72	01.2	166	127	191	200

The constraints are $0 \le w_k \le 1$ together with the summation equality constraint $\Sigma w_k = 1$. An optimal solution, computed by traditional methods is $\mathbf{w}$ = (0.55, 0.0, 0.0, 0.34, 0.09, 0.0, 0.02), with Return/Risk ~ 1.1054.

3. Genetic Algorithm Solution

Genetic algorithms are biologically inspired maximization stochastic heuristics, based on a method that randomly selects two potential solutions from a population of potential solutions, and "breeds" them to create children solutions. A "steady-state" genetic algorithm keeps the population a fixed size: the "worst" solutions in the population are then replaced by the children solutions. Genetic algorithm techniques and "breeding" operations were originally defined to maximize "fitness" functions of binary arguments. Many of these breeding operations can be extended to other representations, such as integers, real numbers, and permutations.

For the portfolio problem, the population consists of a set of vectors {$\mathbf{w}$}, each $\mathbf{w}$ corresponding to a feasible portfolio. The genetic algorithm operations for real numbers (based on [4]) are crossover, coarse tuning, fine tuning, average, and mutation, each with fixed operator probabilities and fixed probabilities that are used to determine which set of weights that the operator will be applied. The operators are adjusted so that they generate a random feasible solution. After each iteration, one or more of the breeding operators are selected, and applied to a two randomly selected parents, whose selection probability is proportional to the Return/Risk ratio that is to be maximized (the "fitness" or objective function).

Problems A and B were both solved using GenSheet, a commercially available genetic algorithm package. For Problem A, the population started converging to the optimal solution (to two decimal place accuracy) in 10 generations. For Problem B, after generating the following randomly generated population...

U.S.	Canada	German	Japan	U.K.	Dutch	French	RET	RISK	Ret/Risk
0.06	0.09	0.185	0.2	0.21	0.12	0.14	11.628	12.748	0.9122
0.11	0.03	0.193	0.09	0.17	0.15	0.24	10.992	12.626	0.8705
0.06	0.21	0.084	0.26	0	0.18	0.2	11.483	12.152	0.9449
0.03	0.04	0.058	0.22	0.34	0.25	0.05	12.197	13.489	0.9042
0.16	0.11	0.129	0.16	0.15	0.11	0.17	11.263	11.755	0.9581
0.18	0.01	0.226	0.19	0.17	0.12	0.11	11.438	12.392	0.923
0.1	0.26	0.067	0.23	0.26	0.07	0.02	11.913	12.02	0.9911
0.19	0.16	0.041	0.18	0.26	0.11	0.06	11.665	11.65	1.0013
0.19	0.1	0.092	0.01	0.27	0.09	0.24	10.76	11.709	0.919
0.1	0.15	0.235	0.18	0.01	0.06	0.28	10.96	12.262	0.8939
0.19	0.050	2	0.22	0.2	0.14	0.01	11.687	12.206	0.9575
0.07	0.24	0.209	0.2	0.2	0	0.08	11.527	12.058	0.9559

...the population started converging to the optimal solution (to two decimal place accuracy) in 100 generations:

U.S.	Canada	German	Japan	U.K.	Dutch	French	RET	RISK	Ret/Risk
0.56	0	0	0.27	0.17	0	0	11.776	10.751	1.0954
0.53	0.03	0	0.32	0.03	0	0.09	11.692	10.638	1.099
0.56	0.01	0	0.25	0.14	0.03	0.02	11.562	10.57	1.0938
0.56	0.01	0	0.24	0.16	0	0.03	11.582	10.597	1.0929
0.6	0.02	0	0.26	0	0.09	0.03	11.311	10.459	1.0814
0.48	0.04	0	0.23	0.14	0	0.11	11.522	10.551	1.092
0.6	0.02	0	0.26	0.1	0	0.03	11.506	10.486	1.0973
0.6	0.02	0	0.26	0.1	0	0.03	11.506	10.486	1.0973
0.6	0.02	0	0.26	0.1	0	0.03	11.506	10.486	1.0973
0.48	0.04	0	0.23	0.14	0	0.11	11.522	10.551	1.092
0.48	0.04	0	0.23	0.14	0	0.11	11.522	10.551	1.092
0.56	0.01	0	0.25	0.14	0.03	0.02	11.562	10.57	1.0938

4. Simulated Annealing

The Metropolis algorithm [2] simulates the evolution of a physical system in contact with a heat-bath as it is observed at random times. Lowering the temperature results in freezing the randomized behavior of the physical system, so it approaches a ground or "annealed" state. In operation, the Metropolis algorithm assumes an "energy" function f(x) is defined on a state x. We generates a new state x_i from an old state x_j randomly, and accept the new state if $f(x_j) - f(x_i) \leq 0$; otherwise we accept x_i with probability P, where

$$P = \exp(-[f(x_j) - f(x_i)]/T)$$

This expression is based on a heuristic derived from the Boltzmann probability distribution of thermodynamics. As T approaches zero, the "randomization" becomes frozen: the optimal final state is reached. In the Metropolis simulated annealing algorithm, this is accomplished with an "annealing schedule" that for each iteration, generates a sequence T_k that converges to zero. At zero temperature, there are no random solutions generated.

For the portfolio problem, the simulated annealing states correspond to feasible portfolios. The function f is just the Return/Risk ratio. At each iteration, the algorithm randomly selects a set of asset weights to change, and generates a random feasible solution. The initial fixed temperatures updated by the annealing schedule T := .9*T.

After each iteration, new solutions are evaluated and accepted with respect to the Boltzmann probability distribution.

For Problem A, the portfolio converging to the optimal solution (to two decimal place accuracy) in 56 iterations. For Problem B, the portfolio started converging to the optimal solution (to two decimal place accuracy) in 379 iterations.

5. Dynamic Search Space Reduction

This heuristic finds the optimal solution to a nonlinear optimization problem by reducing the size of the search region dynamically. After generating a random feasible solution **w**, the algorithm, generates a new vector $\mathbf{w}_{new}$ from a random number R, with

$$\mathbf{w}_{new} = \mathbf{w} + \mathbf{g}$$

where $\mathbf{g} = [\mathbf{M} - \mathbf{m}]*R$

and $\mathbf{M} = (M_1, ..., M_n)$, and $\mathbf{m} = (m_1, ..., m_n)$.

If $\mathbf{w}_{new}$ is not feasible then a new $\mathbf{w}_{new}$ is generated.

For Problem A, the portfolio converging to the optimal solution (to two decimal place accuracy) in 18 iterations. For Problem B, the portfolio started converging to the optimal solution (to two decimal place accuracy) in 115 iterations.

6. Discussion

Simulated annealing was the poorest performer of the three stochastic search methods for the portfolio problem. This may be because of too much "randomization" required in the algorithm: the weights are too sensitive to randomized changes because of the constraints. The genetic algorithm and dynamic search space reduction algorithm had similar performance. Both heuristics are similar and are more sensitive to the constraints. Because of its emphasis on population convergence, the genetic algorithm identified alternate solutions better than the other two algorithms.

In general, the difficulty in all optimization techniques concerns the constraints, ie, in generating a feasible solution. Stochastic search techniques seem to live up to their promise in generating acceptable solutions quickly, even with complicated constraints and objective functions.

For Problem B, both algorithms took less than a second to execute. Informal studies indicate that for larger portfolios (200 to 1000 assets), execution times for acceptable non-optimal solutions scale up quadratically. We are now investigating parallel implementations of stochastic search algorithms for very large portfolios.

7. References

1. Elton, E., Gruber, M., 1991. *Modern Portfolio Theory and Investment Analysis*. John Wiley: New York p. 69.

2. Bonomi, E., Lutton, J., 1984. The N-City Traveling Salesman Problem: Statistical Analysis and the Metropolis Algorithm. *SIAM Review, 26:4, pp. 551-568.*

3. Jorion, P., 1992. Portfolio Optimization in Practice. *Financial Analysts Journal*, January-February, pp. 68-74.

4. Davis, L., 1989. Adapting Operator Probabilities in Genetic Algorithm. In *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann: San Mateo, pp. 61-69.

5. Guggenheimer, H. 1989. *Lecture Notes in Numerical Analysis*, Polytechnic University.

Paper Session: Marketing and Business Strategies

Chair: Susan Garavaglia, Dun and Bradstreet Information Services

USING AI TO PLAN A BUSINESS STRATEGY

Phil Baugh, M.A, Head of International Business, Lancashire Business School, University of Central Lancashire, PRESTON. PR1 2HE. UK.
Alan Gillies, M.A., PhD*, Reader in Business Information Management, Lancashire Business School,University of Central Lancashire, PRESTON. PR1 2HE. UK
Piotr Jastrzebski, MSc, c/o Information Technology Institute, University of Salford, M5 4WT. UK.
Prof. Peter Smith, BSc, PhD. School of Computer Science and Information Systems, University of Sunderland, Sunderland UK

Abstract

AI can no longer be regarded as a new technology, its origins being in the 1950s. However, in spite of the vast potential for the application of AI in domains where information is a high value resource, such as business planning, the uptake of AI is still relatively slow. The authors have identified the following bottlenecks in the adoption of AI:

- *Availability of high quality knowledge*
- *Validation of knowledge where available*
- *Access to information held in existing databases*
- *Maintenance and adaptability in the face of changing business needs*

The authors propose an approach based around established business techniques and an expert system linked to multiple databases in a moderately coupled arrangement. The approach has been adopted and a system implemented.
This paper outlines the approach and the implemented system.

USING AI TO PLAN A BUSINESS STRATEGY

1.The barriers to the adoption of AI in high value business domains

Most businesses have information available that is not being used. In financial companies this information is a valuable resource that is being wasted. AI offers one way of releasing the value tied up in this redundant resource. Unfortunately, the uptake of AI is held back by a number of bottlenecks including:

- Availability of high quality knowledge
- Validation of knowledge where available
- Access to information held in existing databases
- Maintenance and adaptability in the face of changing business needs

1.1 Limited availability of high quality knowledge

Since the potential benefits that may accrue from expertise in these domains are high, the knowledge itself is considered to be of high value and the basis for the competitive edge of a company. Much of the desired knowledge is therefore not available in the public domain. This is restrictive both to academic and commercial developments which will be limited to 'in-house' knowledge.

1.2 Validation of knowledge where available

Where knowledge is available, it is often difficult to validate it and therefore to test and quality assure the system effectively. It is therefore understandable that companies are reluctant to base crucial decision making processes upon an unvalidated system, although the corresponding human judgements are seldom validated either.

1.3 Access to information held in existing databases

One of the crucial factors in a successful system is the access to existing information stored in databases. Until recently, a stand-alone culture has been prevalent in the AI community which has been unhelpful to this objective. This has changed in recent years as the importance of integration with databases has been appreciated.

1.4 Maintenance and adaptability in the face of changing business needs

One of the weakest aspects of AI has been its ability to adapt to changing business needs. This is due to its failure to adopt the structured programming paradigm or to provide an alternative to ensure that the implemented is easily changed to take account of errors or changes in business needs. The success of XCON,[1] probably the first commercial expert system to based upon a structured programming paradigm, suggests that the long held view that the ill-structured nature of AI problems is incompatible with structured programming is incorrect.

2. AI in business planning

Business planning has many attractive features for the expert system designer. It requires the use of statistical techniques but many practitioners are not trained in statistics. Thus the problem is characterised by scarce expertise. It represents quite a narrowly defined problem domain. These are the classical features of a problem amenable to a knowledge-based system solution. The problem for the system developer is that whilst selection of methods and interpretation of results requires symbolic computation, the actual statistical processing is clearly numeric in nature. Thus an integrated solution is required, and the approach adopted is based upon Gillies' JKBS approach.[2,3].

* *Dr Alan Gillies is the author to whom correspondence should be addressed*

Since market research is costly at every stage in the process, whether it be data collection, analysis or interpretation, there is a need to explain buyer behaviour and predict patterns of consumption of the population as a whole from as small a sample as possible. Unfortunately, within many organizations, there is limited knowledge and understanding of the mathematical and statistical techniques employed in market research putting expertise at a premium. Even where knowledge of the techniques is available there are problems for the statistician concerning the interpretation of results i.e. putting them into a form which has meaning for market researchers and company executives. Further, where such techniques can be applied, and this is true of any rule-based system, important issues of problem context still arise since irregularities in a market research data set can only be interpreted by contextual knowledge.

3. Development Of The System

3.1 Designing the solution

The solution proposed is designed in accordance with the principles of integration described in Gillies[3]:

- The `problem drives the solution' principle states that the problem should determine the type of solution.
- The `means to an end' principle states that anything employed is only important as far as it contributes to the effectiveness of the overall solution.
- The `toolbox' principle states that an IT (information technology) problem solver has at his disposal a series of technologies which may be likened to a set of tools within a toolbox.
- The `targeting' principle states that an effective solution requires the targeting of each technology to the parts of the problem where it can be most effectively employed.
- The `clipping principle' states that each contributing technology must only be used within its designed operating range.

The process of design has been described elsewhere [1]. The concept of the solution is an expert system combining data from a number of existing sources to advise upon a business strategy, as shown in Figure 1.

3.2 Implementing the solution

The system was implemented with the following three features to address the four bottlenecks identified above:

- Knowledge drawn from existing business models
- Multiple and Hybrid coupling interface to databases
- Structured and modular implementation

a) Knowledge drawn from existing business models

In order to address the validation and availability issues for the expert system, it was decided to adopt an existing business model as the basis for the system's conclusions. Whilst this is by no means a guarantee of perfection, it does enable the user to make reasoned judgements about the veracity of the conclusions, based upon their perception of the business model. It enables the system to be validated against a known standard. Thus the system's veracity becomes it faithfulness to the model on which it is based. This may be measured.

After much investigation, the rules encapsulated within the system are based upon the Profit in Marketing Strategy (PIMS) project[4], enriched by the use of portfolio analysis[5]. This was decided because the PIMS project was based upon data drawn from a large number of companies and the findings were presented in the form of rules. A sample of the rules employed in the expert system are given below in Table 1.

Since PIMS findings can be presented in the form of rules, they proved suitable for this project. In fact these rules have formed the core of the knowledge base used in this system, and portfolio analysis was used only to support PIMS findings. To use any of these rules it is necessary to prepare financial data about a company, and its product line. It is also important to have all the market segments well defined. Every product in the company is treated as a part of a portfolio. The rules seek to encapsulate all the facts concerning a specific project.

The first source of data is based upon customer questionnaires, usually distributed at time of sale and linked to registration for warranty purposes. Since all these questionnaires include the name of the product purchased, it is possible to count how many items of this product have been sold in every segment of the market (all the points which form this segment are known - they were used in the clustering package to find what the segments are). The number of items sold in every segment allow the determination of which segments form the market for the particular product, what is its market share etc..

The second database contains data about different products. This can deliver data like: variable cost, marketing effort (funds dedicated to promote this product), price...All these variables are essential to assess the financial position of the product line. A third database contains special properties of the products (strengths, and weaknesses). When the market of the product is found it is possible to compare the description of this market with the strengths and weaknesses of this product.

This also helps to describe the product's position. Apart from the financial analysis of each product line, the overall strength of the company and that of competitors is also assessed, based upon an index (Z) derived by Slatter[6]:

$$Z=1.2*x_1 + 1.4*x_2 + 3.3*x_3 + 1.0*x_4 + 0.6*x_5$$

Where

x_1=working capital/total assets
x_2=profit retained/ total assets
x_3=profit gross/ total assets
x_4=total sales/ total assets
x_5=market value of equity/total liabilities

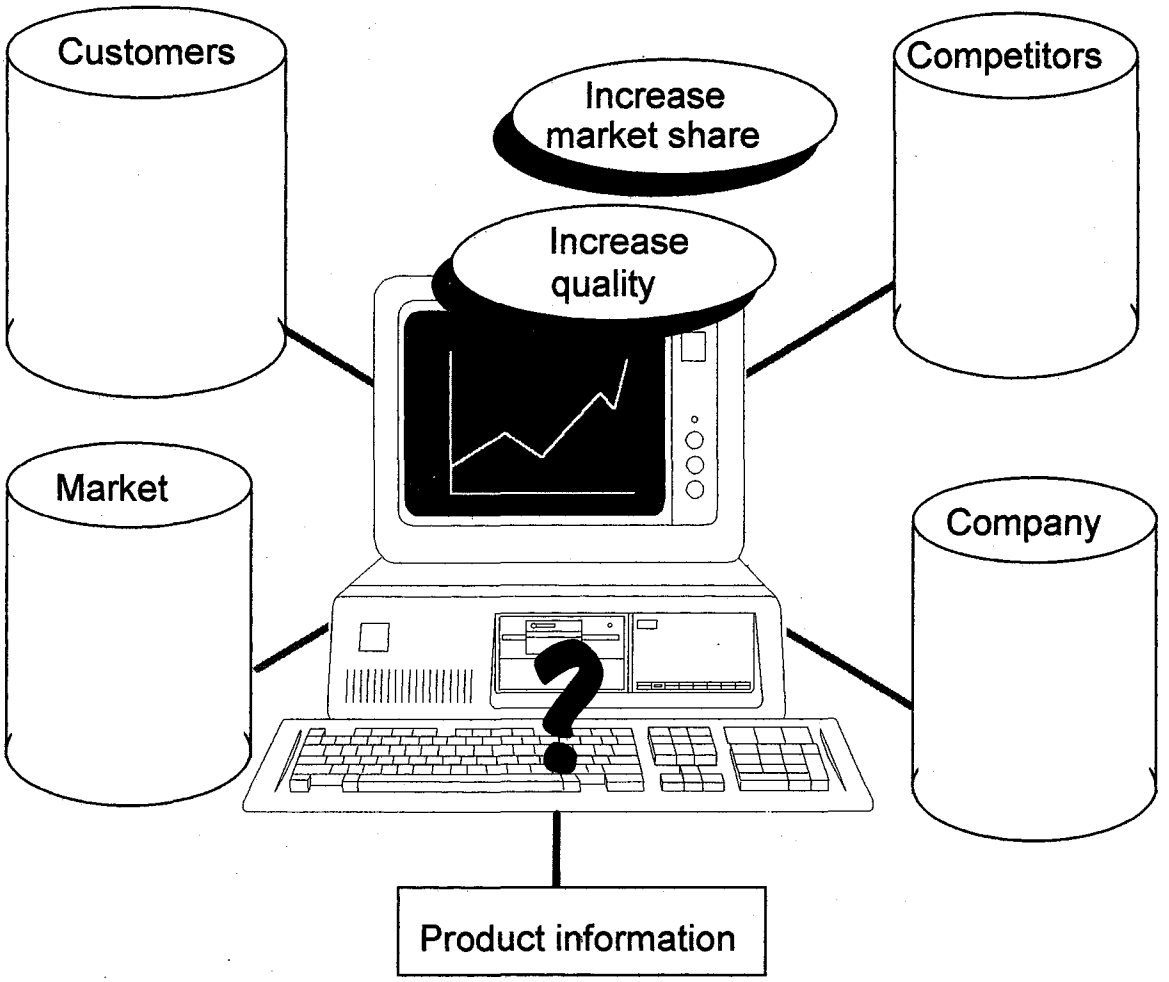


Figure 1 Solution concept

Table 1 Sample of rules relating to the action of 'Reduce Investment'

3. There is a strong recommendation (P = 0.7) to reduce investment if • there is prediction of low rate of return (ROI)
4. There is some indication to reduce investment if • there is low growth in that sector
5. There is strong recommendation to reduce investment if • there is low growth in that sector • and, at the same time competitors are strong.
6. There is a recommendation to reduce investment if • there are cash problems
7. There is a recommendation to reduce investment if • there is low growth in that sector. • there is small market share
8. There is a recommendation to reduce investment if • there are indications to implement a safe strategy.

If the number of Z is smaller than 1.8 then the situation of the company can be called as critical. If Z is bigger than 3.0 it is a sign of prosperity. The index Z is created from data which can be taken from the balance sheets, and yearly reports of the company. These balance sheets are stored in a separate database, and are accessed by the system.

b) Multiple and hybrid interface

The system was initially conceived as a tightly coupled system in the terms of Torsun and Ng[7]. All variables of all modules would reside permanently in the memory. There would be no problem with data exchange on any level. The benefits of this approach can be easily seen in the increased efficiency, speed, and memory usage.

However, it was found to be necessary to compromise the strength of coupling, principally because of the need for flexibility. A more loosely coupled system allows greater flexibility in design and use. This allows the word processing and database functionality to be handled by proprietary products. Each module in a loosely coupled system can be optimised for a specific function. For example, with the aid of C libraries to access DbaseIII files it has proved possible to incorporate the facilities of a proprietary database management system within the overall system context. This gives the user flexible, and convenient access to the database.

The result is an interface which is pragmatically optimised for the needs of the problem, rather than constructed according to a particular scheme. This is in accordance with the 'problem drives the solution' principle of integration stated above.

c) Structured and modular implementation

The system is implemented in modular fashion, illustrated in Figure 2.

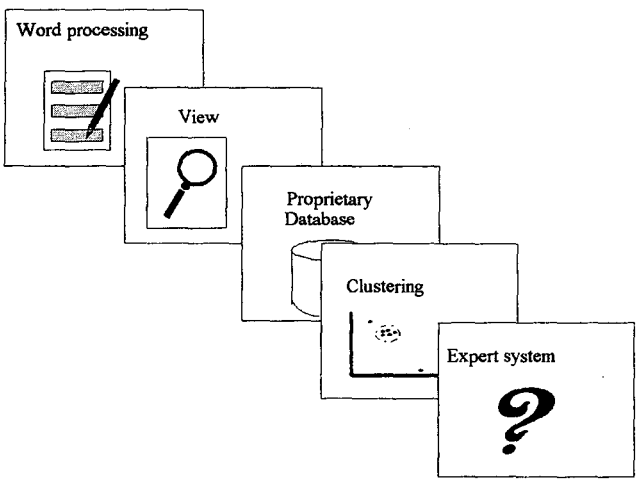


Figure 2

The system was conceived to support the whole process from drawing the form, collecting data, clustering to the conclusions at the level of strategic management. There are three modules designed to gather and collate the raw customer data, described in Baugh et al[8]. These modules are selected from an initial menu. The system was planned as a set of five sub-programs. . All five modules can be invoked form the main menu of the system, but also each of the modules can work as a stand alone system, and can be run from DOS. All the modules work with the same data, with the same files.

Apart from two modules which are just a part of standard business software (the word processor, and Dbase package) the rest has been written especially for the project in C++.

In order to provide customer data in appropriate form for the expert system, the raw data is first clustered. The structure of the clustering programme is shown in Figure 3. The resulting clusters form the data stored in the customer database.

This highly modular and structured design and implementation is designed to provide easy access for maintenance, adaptation and enhancement. The use of C++ for the dedicated modules ensures that the structured paradigm is implemented down to the code level.

4. Conclusions

This project has shown that a system can be implemented which makes use of AI in the context of a high value business domain. Further, by careful structured design of the system., the required virtues of auditability and maintainability may be incorporated. The use of an established model for the knowledge base enables the knowledge base to be verified as self-consistent and true to its origins.

In doing so, it rejects a considerable amount of conventional AI thinking and practice. However, it conforms to a growing number of applications which suggest that AI can no longer afford to ignore the wisdom of conventional software development.

5. Acknowledgements

Much of the work in this paper was carried out at the IT Institute at the University of Salford. The authors are happy to acknowledge this. Piotr Jastrzebski was supported by a TEMPUS grant and the University of Salford for the duration of this project, and this support is gratefully acknowledged by the authors.

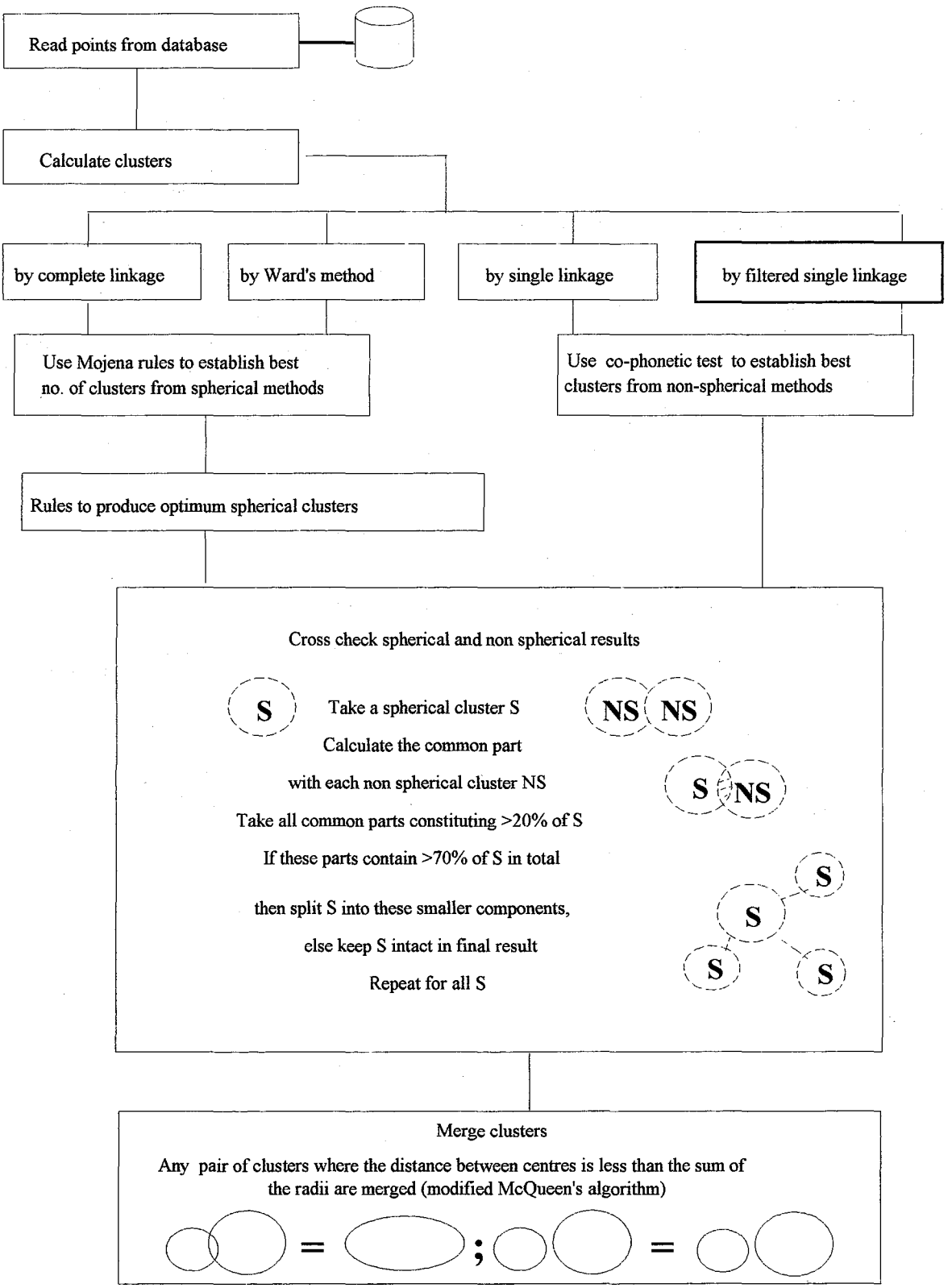


Figure 3

Sources: McQueen[9]; Complete linkage/co-phonetic test [10]; Ward's method[11]; single linkage methods[12];Mojena rules[13]

6. References

[1] McDermott, J. `Doing R1 with style', 2nd conference on AI, Miami, 1985.

[2]Gillies, A.C. and Baugh, P.J. `A journey in methodology from fringe pattern analysis to statistical forecasting', *Heuristics*, (in press).

[3]Gillies, A.C. *The integration of expert systems into mainstream software*, Chapman and Hall, London,1991.

[4]Ham, The profit impact of market strategy in the insurance industry, Best's Review, vol 79, no 8,(1978).

[5]Henderson, `The product portfolio', *Perspectives No 66*, BCG (1970)

[6]Slatter, *Corporate recovery*, Penguin, London (1984)

[7]Torsun, I.S. and Ng, Y.M. TCEDI: an expert database system interface, *Proc ES '88*, CUP, 210-223, (1989).

[8]Baugh, P.J. Gillies, A.C and Jastrebski, P. Combining Knowledge-Based And Database Technology In A Tool For Business Planning, *Information and Software Technology, in press*, (1993).

[9]McQueen *Some methods for classification and analysis of mulitvariate observations*, AD669871, University of California Press, Berkeley (1967).

[10]Sokal and Michener `A statistical method for evaluating systematic relationships', working paper, *University Of Kansas Bulletin*, (1958).

[11]Ward `Hierarchical grouping to optimize an objective function', *Jou. Amer. Statistical Assoc.* (1963).

[12]Sneeth `The application of computers to taxonomy', *Jou. Gen. Microbiol*, 17, (1957).

[13]Mojena `Hierarchical grouping methods and topping rules: an evaluation', *Computer Journal*, 20 (1977).

Developing a Neural Network for Selection of Sales Promotion Instruments in Different Marketing Situations

Jack Kluytmans
Berend Wierenga
Mathijs Spigt
Erasmus Universiteit Rotterdam
Heemraadssingel 276B, 3021 NA Rotterdam, The Netherlands

Abstract

Effectivity and flexibility in the choice and application of marketing techniques become more and more important aspects for companies to obtain and maintain a competitive advantage. Marketing managers are experts in this field and carry a large package of knowledge with them. But it's all between the ears. Problems arise when the marketing manager leaves the company. A valuable amount of expertise will be lost. This, however, can be prevented by creating a knowledge base. The knowledge base contains the expertise of the marketing manager and can be consulted at any moment, this instead of the "hard to reach" manager. A technique that has qualities to function as a knowledge base is a neural network. The expert knowledge can be stored in the values of the weights of the network, when the network is given the right architecture and when it's trained on suitable data. Sales promotion is one of the marketing-fields on which a marketing manager may have extensive expertise. To develop our neural network application we choose sales promotion, because of its bounded and relatively structured characteristics.

1. Introduction

The experienced marketing manager may have build up a valuable amount of know how during his years with a specific company. Although this know how is an important asset to the company, it is not easy to make it explicit. The expertise is only available to the manager himself. For an other member of the company it is only possible to use this know how by consulting the marketing manager. This paper comprises research on the possibility to develop a neural net based on expert-knowledge, which can suggest suitable sales promotion instruments after being given the importance of certain marketing-objectives. The function of this neural system can be the support of a marketing manager's decision concerning the choice for a sales promotion technique in a specific marketing situation. This knowledge base can be available to all members of the organisation in the future and will thus have a knowledge keeping and knowledge distributing function for the company.

The research was motivated by the fact that neural nets are a relatively new and promising technique for computer learning and that there are not many applications known in the field of marketing, especially concerning sales promotion.

2. Description and Validation of the Fields of Research

2.1 Sales Promotion

Decisions on sales promotion are mostly of a semi-structured kind and made on a tactical level in an organisation. Neural nets might be suitable for applications in this area. On a strategical level problems are relatively unstructured, complex and situational. Personal interpretation is very important by solving these kind of problems. In contrast, on an operational level decisions are structured and bounded. People can make routine-based decisions and existing information systems are adequate on this level.

According to Lilien and Kotler [5] sales promotion comprises a wide variety of tactical promotion tools of a short-term incentive nature, designed to stimulate earlier and/or stronger target-market response. This definition stresses the variety of existing sales promotion instruments, their joint short-term qualities and the diversity in marketing objectives that can be achieved by using sales promotion.

In exhibit 1 we give an inventarisation of some well-known sales promotion instruments. These are also the sales promotion instruments that we used in our research.

Examples of Sales Promotion Instruments

- Contest
- Sampling
- Self liquidating premium
- Stamps
- Cash refund
- Price discount
- Product plus
- Premium (in-,on-,near-pack)
- Coupon

Exhibit 1

Some often occuring marketing objectives to achieve with a sales promotion action are listed in exhibit 2. Based on articles in scientific and professional journals we considered the here listed objectives as the most important ones.

Marketing Objectives with Sales Promotion Instruments

- Generation of awareness.
- Attraction of new costumers.
- Increase use of current costumers.
- Increase loyalty of current costumers.
- Support advertising campaign.
- Potential to improve image.
- Improvement of trade relations.
- Improvement of display/shelf-position.
- Increase of distribution.

Exhibit 2

Once it is decided to use sales promotion as an instrument to achieve certain marketing objectives, the choice has to be made which specific tool to select out of the wide variety of possibilities. Additional circumstances have of course an important influence on the decision the marketing manager has to make. Influences on the effect of a sales promotion action are the magnitude of the incentive, the specific group of people at whom the action is aimed, the starting-date, the method of distribution, how long the promotion will last, the available budget and the situation of the product in the market. However in most situations the greatest influence on the marketers choice comes from the specific marketing objectives which are to be achieved.

We assume that there is a link between the importance of a specific marketing objective or a specific set of important objectives and the suitability of a specific sales promotion instrument to achieve these objectives.

2.2 Neural Networks

The neural network theories are based on connectionism, which is subject of research in the area of artificial intelligence. The connectionist tries to design models for the behaviour of man, which are based on the structure of the brain. The brain exists of a large amount of small elements (neurons). These neurons are connected to each other in countless ways. A network of neuron-like elements can be called a neural network. Individual elements in such a network are excitated and inhibited. When a certain threshold is exceeded, the element will be activated itself and excitate or inhibit other elements via its connections. The network will function according to the strenght of the connections and the way the elements are connected to each other.

The knowledge of the network is in the connections. Every connection can play a role in many kinds of knowledge and every part of the knowledge can be distributed over many connections. The connections that contain the knowledge can change in time. This happens in a continuous way, which causes learning. Learning-rules can be brought into the network. When the neural network is being shown learning-examples it may be able to learn a desired input/output-pattern and may develop generalising capabilities, which make the network useful to judge new examples.

Thus, a neural net, having generalising capabilities, might be a suitable technique to choose the most preferable sales promotion instrument from a set of instruments just by being given the importance of different marketing objectives. We used this assumption as a starting point to develop a neural network for the selection of sales promotion instruments in different marketing situations. The next part will be about gathering and constructing the data.

3. The data

The learning-data for the neural net were gathered by using an attribution matrix (figure 1). Horizontally we see different sales promotion instruments, vertically there are different marketing objectives. This matrix was filled in by a marketing expert. The figures are on a scale from 1 to 5 and express the suitability of a certain sales promotion instrument to achieve a certain marketing objective. "1" Stands for extremely unsuitable, "5" stands for extremely suitable.

Attribution matrix	Con-test	Sam-pling	Self liqu. prem.	St am ps	Cash re-fund	Price dis-count	Pro-duct plus	Pre-mium	Coupon
Generation of awareness	4	5	3	1	1	1	1	3	4
Attraction of new costumers	3	5	2	1	4	2	1	3	4
Increase use of current costumers	1	1	1	3	3	4	1	1	1
Increase loyalty of current costumers	2	1	1	5	1	1	4	1	1
Support advertising campaign	4	4	3	1	1	1	1	3	2
Potential to improve image	3	1	5	1	1	1	1	4	2
Improvement of trade relations	1	1	2	1	2	3	1	2	1
Improvement of display/self-position	1	1	4	1	2	2	4	4	1
Increase of distribution	1	1	1	1	2	3	2	1	4

Figure 1

The training- and test-data for the research were composed by giving randomly measures of importance to the different marketing objectives on a scale from 0 to 5 and multiplying these measures with the figures in the matrix. In this way we get, for every set of randomly selected measures of importance for the different marketing objectives, a set of measures of suitability for the different sales promotion instruments. To make pairs of data comparable each measure of suitability was divided by the sum of measures of suitability of every datapair.

The composed data may look like this (2 examples):

	pair 1	pair 2
Inputdata:		
Generation of awareness	1	2
Attraction of new costumers	2	3
Increase use of current costumers	4	1
Increase loyalty of current costumers	2	5
Support advertising campaign	3	3
Potential to improve image	5	5
Improvement of trade relations	5	0
Improvement of display/shelf-position	0	0
Increase of distribution	3	0
Outputdata:		
Contest	11	14
Sampling	10	11
Self liquidating premium	13	13
Stamps	9	11
Cash refund	10	8
Price discount	12	6
Product plus	7	9
Premium (in-,on-,near-pack)	12	13
Coupon	11	10

The "1" in the first row and first column is the measure of importance of the first marketing objective in the attribution matrix, namely "generation of awareness". The "11" in the tenth row and first column is the corresponding measure of suitability of the first sales promotion instrument in the attribution matrix, namely "contest". In the first pair of data the marketing objectives "potential to improve image" and "improvement of trade relations" are valued with a "5". This means that these objectives are considered very important in this situation. In contrary, the marketing objective "improvement of display/shelf position" has the value "0", which means that this objective is considered very unimportant in this particular situation. When we look at the values of the sales promotion instruments in the first datapair, we see that the instrument "self liquidating premium" has the highest value, namely "13". This means that this instrument is considered the most suitable one in the given situation. By using a computerprogram many pairs of data can randomly be composed.

Advantage of this way of data-gathering is that it is clear and not time-consuming. Disadvantage of using the attribution matrix is the fact that individual suitability figures are, probably incorrectly, being summed to collective suitability figures. This disadvantage can be avoided by using inquiries to gather data. Although this will cost more time and money, it is a recommendation for future research.

4. Developing the neural net.

For developing the network the Back Propagation paradigm was selected. Considerations made for selecting this paradigm are the relative simplicity and the supposed power of this paradigm. A hetero-associative memory, like Back Propagation, is suitable for classification tasks with presentation of the output on a continuous scale.

The learning proces takes place under supervision. This means that the network must be trained on a selection of cases containing input/output-pairs. In this research the input-pairs consist of the first 9 variables in the data-pairs, the output-pairs consist of the last 9 variables in the data-pairs. After the netwerk is trained to an acceptable level, it has to be tested on a test-set of input/output-pairs. Both the training- and test-set are randomly composed. In our sets there were 187 pairs of data composed for the training-set and the test-set consisted of 40 pairs.

The first step taken in designing the neural net is deciding on the complexity of the network. This concerns the amount of hidden layers and the amount of hidden units in every hidden layer of the network. In this case the input layer as well as the output layer consisted of 9 units. By experimenting there was found that one hidden layer with 9 hidden units did best on the data. We experimented with 0, 3, 6, 9 and 12 hidden units. The next step in the designing proces concerns the connections in the network. Here all input units were connected to all hidden units and all hidden units were connected to all output units. The hidden and output units were connected to a bias. The hyperbolic tangens function was used as transferfunction in the whole network and for learning the normalised cumulative delta-rule was used. The epoch size was set to the amount of data-pairs on which the network was trained. During training data-pairs were randomly presented to the network. During the training the learningparameters were adjusted after fixed amounts of learn counts. They were given smaller values the longer the learning proces lasted. The training was stopped when the error level did not change any more.

5. The Results

As mentioned above, the best results were obtained with a 3-layer network and 9 units in every layer. The performance of the network was measured by comparing the order of suitability of sales promotion instruments given by the network with the desired order of suitability given in the training- and test-cases. Especially the order of the three most suitable instruments per case was considered, because these are the important instruments between which the marketing manager finally has to decide and to which attention really is being given. For the first three instruments also the amount by which they were found more suitable than the others was considered to measure performance.

The performance on the training data was excellent. In all cases the suitability of the sales promotion instruments was given in the right order. On the testing data in 37 of 40 cases the desired most suitable instrument was actually indicated by the network as most suitable. For 30 testcases the numbers one, two and three were indicated in the correct order. In the 10 cases where mistakes were made the indicated suitability-value never amounted more than 3% of the desired suitability-value. Thus mistakes are being made, but they were never disastrous.

An example of correct output-data:

	desired	actual
Outputdata:		
Contest	11	11
Sampling	10	10
Self liquidating premium	13	14
Stamps	9	8
Cash refund	10	10
Price discount	12	12
Product plus	7	6
Premium (in-,on-,near-pack)	12	12
Coupon	11	11

An example of less correct output-data:

	desired	actual
Outputdata:		
Contest	14	14
Sampling	11	9
Self liquidating premium	13	14
Stamps	11	14
Cash refund	8	8
Price discount	6	6
Product plus	9	10
Premium (in-,on-,near-pack)	13	13
Coupon	10	9

In the last example the sales promotion instrument "contest" is the desired most suitable instrument (value "14"). The network also gives "contest" the highest suitability value, however the instruments "self liquidating premium" and "stamps" reach the same actual score, "premium" comes next with value "13". So the network classifies the instrument "stamps" incorrectly in the top 3 and also overvalues it. But still the best and worst suitable instruments, "contest" and "price discount" respectively, are correctly classified.

We conclude that the overall results are quite promising. It is possible to store expert knowledge in a Back Propagation network. However, many interesting possibilities to experiment remain.

6. Further Research

Recommendations for further research include the use of survey-data obtained from more marketing experts, the use of other neural paradigms and the use of different network-architectures.

With survey-data the expert can judge more complex situations than with the use of an attribution matrix. The expert can even judge the same situation several times. Maybe his judgement for the same situation may differ at different times. It may be interesting to see how a neural net will handle this. Of course it is also possible to get information about the same situation from several experts. The task for the neural net will be to give a neutral advise for situations on which experts disagree and a clear advise for situations on which experts do agree.

A survey can be constructed by randomly composing sets of degrees of importance of different marketing objectives. These sets can be judged by marketing experts, whose task it is to give a value to every sales promotion instrument for the suitability of the instruments in the given marketing situation. This will be the next step in this research.

In the present research only the Back Propagation paradigm was used, but there are of course many other neural paradigms that can be used for the selection of sales promotion instruments. One of them is the Learning Vector Quantizer paradigm which gives binary outputs and in this way can judge a certain sales promotion instrument as being suitable or not suitable in a certain marketing situation.

Another experimentation possibility is constructing a modular network. When all hidden units are connected to at most one output-unit spatial crosstalk can be avoided. The supposed advantages of an architecture of this kind are the increase in the speed and in the generalising capability of the network [3].

7. References

[1] Bailey D., D. Thompson, *How to develop neural network applications*, tutorial, 1991.

[2] Caudill M., C. Butler, *Naturally Intelligent Systems*, MIT Press, Cambridge Mass., 1990.

[3] Jacobs Robert A., Michael I. Jordan, *Task decomposition in a Modular Connectionist Architecture: The What and Where Vision Tasks*, Cognitive Science 15, p.219-250, 1991.

[4] Keon J.V., J. Bayer, *An expert approach to sales promotion management*, The Journal of Advertising Research p.19-27, June/July 1986.

[5] Lilien G.L., P. Kotler, *Marketing Decision Making: A Model Building Approach*, Harper & Row, New York, 1983.

[6] Little J.D.C., L.M. Loddish, *Commentary on Judgment Based Decision Models*, Journal of Marketing 45 (Fall), 24-9, 1981.

[7] Lorin Cook R., J.M. Schleede, *Application of Expert Systems to Advertising*, Journal of Advertising Research p.47-56, June/July 1988.

[8] Rumelhart D.E., J.L. McClelland, *Parallel Distributed Processing*, MIT Press, Cambridge Mass., 1986.

[9] Schreinemakers J.F., *Pattern recognition and symbolic approaches to diagnosis*, Eburon, Delft, 1991.

[10] Spigt M.H., *Neurale netwerken in management omgevingen*, Stichting Syllabi, Rotterdam, 1991.

[11] Verbeke W., A.P. Mosmans, *Marketing-communicatie en chaos*, Kluwer Bedrijfswetenschappen, Deventer, 1991.

[12] Wierenga B., *Knowledge based systems in marketing: Purpose, performance, perceptions and perspectives*, Management Report Series Rotterdam School of Management, nr.112, Rotterdam, 1992.

Direct Mail Response Modeling Using a Counterpropagation Neural Network

Susan Garavaglia
Dun & Bradstreet Information Services, N. A.
C. U. N. Y. Graduate Center

Abstract

The counterpropagation neural network introduced by Hecht-Nielsen [4] is applied to predicting the likelihood of a positive response to a business-to-business direct mail campaign. The results are compared to those obtained using logistic regression. Model performance is evaluated in terms of response lift, which is the percentage improvement over mailing to the population at random. Model stability, rank-ordering power and practical application of the neural network model are discussed.

1. Characteristics of the Response Modeling Problem

Direct mail advertising is a widely used tool in both consumer and business markets. For example, businesses sold about $53.3 billion of goods and services to other businesses in 1990 via direct mail campaigns[1], and the outlook is that this form of advertising will continue to play a major role. Although direct mail is an attractive advertising medium to many businesses, it is also estimated that 98% of direct-mail items are discarded.[2] As competition forces more product differentiation and market segmentation which raises the cost of selling, there will be increased emphasis on accurately targeting customers to counteract the expense of more elaborate catalogs, list rentals, and so forth.

The task of predicting the most likely to respond is complicated by a number of factors. Direct mail response rates are very low percentages; a rate as low as 1.5% could be considered a marketing success for some campaigns. This makes data collection for modeling difficult for two reasons; 1) a large amount of data must be available to get a sufficient number of respondents to build a model and 2) the modeler must address the issue of how large a percentage of respondents to include in the data in order to have sufficient discrimination and validation. If, for example, the population response rate is 1%, and the modeler creates a model development data set with a 20% response rate, the proper validation of the model would require some sort of weighting procedure to compare the model's results with a random selection. The weighting technique used is discussed in Section 5.

Marketing data is often described as "soft" in that individual data elements tend to be less predictive than those in credit data, as a comparison. This is to say that generally, a data element such as "Average Days to Pay," which might be found in a credit scoring model, tends to be statistically more significant than "MSA Code" (for Metropolitan Statistical Area), which is a typical data element in response models, often represented by a series of dummy variables. Another contributor to the data "softness" is the fact, that in a response campaign, the decision is made by the buyer, as opposed to a credit decision, which is made by the seller. When one agent, the seller, makes all the decisions, some consistency is expected. When many agents, such as individual buyers, make the decision, each agent may have applied a unique set of criteria to the decision process. Arguably, both models predict the behavior of buyers, but the seller exerts more restrictive conditions on creditors than on prospects.

2. Response Model Data Overview

The purpose of this business-to-business response model is to predict response to a direct mail campaign which offers a corporate credit card for gasoline purchases and other related items. Data supplied came from a specific mailing campaign with a response rate of about 3.7%. The company, a major oil company, supplied data to identify the businesses and indicate whether or not they responded to a particular campaign. This data was matched to data owned by Dun & Bradstreet's (D&B) Analytical Services department, which is used by the department as predictive data in models. The original campaign included 3.1 million businesses, of that group, a sample of 327,000 was

[1] Source: *Direct Marketing*, July, 1991, p. 31.
[2] Source: Editor's Note, *American Demographics*, September, 1992. p. 2

supplied to D&B. About 59.9% of the sample was able to be appended with D&B modeling data, and became the final sample. From this sample, a development data set of 118,173, including 19,783 (16.7%) respondents and 98,390 (83.3%) non-respondents, and a holdout data set of 79,246, including 10.506 (13.26%) respondents and 68,740 (86.74) non-respondents was created. The development data set was originally used to create a model with logistic regression, and the holdout data set was used to validate that model. The exact same data sets were used for the neural network development and validation. The logistic regression model was developed first and implemented; the neural network model was developed later for the purposes of introducing neural network technology to the D&B Analytical Services group.

A list of the variables used in both the logistic regression and the counterpropagation network models is in Appendix A. Out of the 28 variables, 23 are *dummy variables*, meaning their value is either 0 or 1, depending on the premise that they represent. For example, the variable PUB_NO has a value of 1 if the observation is a privately held company. Many of the variables represent the membership of the observation in decile or quartile response groups. For example, SIC1_90 indicates if the observation belongs to a SIC[3] code 90% percentile, i. e., the SIC code of the observation belongs to a group that was in the top 10% of response rates. Seven of the 28 variables are of this type; this reflects the difficulty in finding data elements that are truly predictive for this model. In addition, this technique could not be used for a new marketing campaign, where no response rates were yet determined.

3. The Counterpropagation Neural Network

A standard counterpropagation network was built using the same 28 variables as the logistic regression model. The counterpropagation network used the 28 variables in exactly the same form as the logistic model. The output was represented by a layer containing two processing elements; one representing response and the other non-response. The output vectors are {1,0} and {0,1} respectively. The network contained one hidden layer of 301 PE's. The total number of weighted connections is (28 + 2)*301, or 9,030.

[3] SIC is the Standard Industry Classification code for a industry name, and is a system used extensively by government and business to classify industry groups.

Counterpropagation was chosen from a set of categorization-type networks, including Learning Vector Quantization (LVQ) and a Self-Organizing Map (SOM) variant with a categorization layer (for detailed descriptions of these networks, see [4] and [6]). In counterpropagation, the hidden layer units each learn to identify a class by adapting their weights to the means of the data vectors though Kohonen learning. Kohonen learning calculates a new weight as a function of the old weight and a weighted difference between the old weight and the input value. The Kohonen learning law is:

$$W_{ij}^{new} = W_{ij}^{old} + C * (x_{ij} - W_{ij}^{old})$$

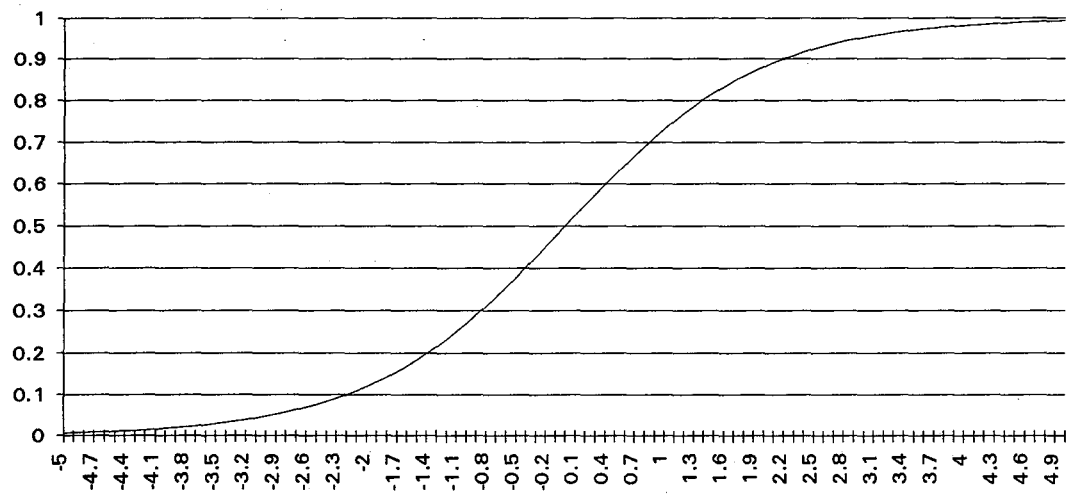
where W_{ij}^{new} is the new weight from input unit i to hidden unit j, W_{ij}^{old} is the previous value of the weight, C is a learning coefficient, or constant, and x_{ij} is the input value being passed from input unit i to hidden unit j. When there are more hidden units than classes, the hidden units may learn a subset of a class.

Although LVQ and SOM networks were trained with the same response model data, they did not appear to learn to discriminate between respondents and non-respondents after a large number of learning trials (over 500,000)[4] LVQ is a more rigid network than counterpropagation, in that it only produces output values in the set {0,1}, the SOM network is more suited for data clustering applications than strict categorization. Counterpropagation produces output values in the interval [0,1] and the range of values can be used for ranking the output (See Section 4.3). In addition, a standard backpropagation network was built and trained with the data set; however the average RMS error, based on an epoch size of 11,800 (10% of the training data) was .7745 after 778,800 training iterations, and no further analysis was performed due to the large RMS error.

Although it is generally advisable to build a counterpropagation network using a development data set comprising equal numbers of examples from each class, useful results were obtained even with the unequal distribution of respondents and non-respondents.

[4] Failure to learn was assessed by examining the root mean square (RMS) error after each presentation of the entire data set as a single epoch. If the RMS error did not remain significantly below 0.5, it was decided that there was insufficient discrimination. Epoch-level convergence was not attained or expected for this type of application.

Figure 4-1 The Logistic Function (EXP(X)/(1 + EXP(X)))



4. Comparison to Logistic Regression

Logistic regression[5] is a maximum likelihood estimation (MLE) method, and it produces a probability of response for each observation. The principle of MLE is that a set of parameters is computed that maximizes the likelihood of obtaining the set of sample observations from the population. If the model form is $Y = X\beta + e$, the probability of response (i. e. $Y = 1$), is calculated as:

$$p(Y = 1 \mid \{X,\beta\}) = \frac{e^{X\beta}}{1 + e^{X\beta}}$$

The logistic function is shown in Fig. 4-1. Note that the logistic transformation is a non-linear transformation of a linear function.[6]

Prior to the estimation using logistic regression, a univariate analysis is performed to determine which variables should be included in the logistic regression. Logistic regression is a very computationally resource intensive method, and it is generally not feasible to include more than 30 to 40 variables maximum. In addition, if the final model contains a large number of variables, this increases the overall cost of running and maintaining the model. Therefore, a goal in the process is to produce an effective model using the fewest number of variables.

A logistic regression produces a range of probabilities, which may or may not range from 0 to 1, depending on the data. The holdout data set produced a range of probabilities from 0.000 (0%) to .446 (44.6%). The percentage of actual respondents increases monotonically with the assigned probability.

The counterpropagation network also produces a range of values between 0 and 1, and the breadth and continuity of the values depends on the characteristics of the data set. For this response model the 301 output values ranged from -0.00005 to 0.649001. There was not a strict monotonic relationship between the output unit values and the percentage of actual respondents assigned to those output values. This suggests a high degree on non-linearity in the relationship between the data and the likelihood of response. In both cases, the distribution of respondents over the probabilities or the output unit values was not uniform.

The next section discusses performance measurement and comparison.

5. Response Lift

The response lift is expressed as the percentage improvement over a random selection in capturing potential respondents. It is a useful way to measure response model performance. The formula is:

$$\frac{M}{R} - 1$$

[5] Recommended references on logistic regression include [5], [7], and [1].

[6] Note: The logistic function, the sigmoid function, and the hyperbolic tangent function all have the same "s" shape and are commonly used as transfer functions in neural networks.

where M is the response rate obtained by using a model, R is the random selection response rate, and 1 is subtracted to give the net improvement. For example, if 1,000 observations from this sample were selected at random, the expected response rate would be 3.7% or 37 respondents. If 1,000 observations were selected by the model as having a response rate of 10%, or 100 expected respondents, the response lift would be:

$$\frac{0.10}{0.037} - 1 = 1.703 \text{ or } 170.3\%$$

Because both model development and direct mail campaigns are expensive processes, users of models have high expectations for improved response rates. Generally, the lift should be at least 50% at the fifth decile, but most of the interest is usually in the first two deciles.

In order to present the response lifts in the context of a real marketing campaign, the data must be weighted to reflect the actual response rate of 3.7%. The holdout sample response rate was 13.26%. To bring the response rate down to 3.7%, the non-respondents were each weighted by a factor of 4.[7] Generally, weighting the category with the higher proportion of observations is recommended , so that all the observations from the other category will receive their full weight of 1.

Table 4-1 (Tables follow the References section) shows the logistic response model cumulative lift by centiles. At the 1%, 5%, 10%, and 20% levels (cumulative percentage of total) the lifts are about 347%, 250%, 196% and 129% respectively (outlined in table). As expected, the cumulative lift gradually declines by centile. From a marketing decision standpoint, if potential customers with the same characteristics as those in the top two deciles were used for the next campaign, the expected response rate would be 8.42%, or about 129% better than the overall average of 3.7%. In other words, over twice as many customers would be expected to respond.

In Table 4-2[8] , the response lift by descending hidden unit value is shown. At the 1%, 5%, 10% and 20% levels, the cumulative lifts are 303%, 231%, 159%, and 112% respectively (outlined in table). Viewing the results this way, the logistic model demonstrated superior performance to the counterpropagation network. However, since the output unit values are discrete groupings of the data, it would be possible to take the best performing output unit values, and choose the prospects with the same characteristics as those in the best performing groups. Table 4-3 shows the cumulative response lift, sorted in descending order, by best performing output unit values. At the 1%, 5%. 10%, and 20% levels the response lifts are about 582%, 278%, 185%, and 123% (outlined in table). At the 1% and 5% levels, the response lifts surpass those of the logistic regression model, and are closer to it at the 10% and 20% levels.

These results suggest that, although the counterpropagation network output did not outperform the logistic regression at the first and second deciles, it was able to identify smaller groups that were very likely to respond. Although it is not listed in Table 4-1, the logistic regression response lift at the 0.1% level was 454.7%. This compares to the neural network response lift at the 0.13% level of 490.92% (Table 4-2) and 2167.37% at the 0.1% level when considering the highest performing output unit values (Table 4-3). In terms of application to marketing strategy, the counterpropagation network might serve well where the objective was to identify a relatively small group of potential customers with a very high likelihood of responding. For example, if the per customer mailing costs were very high, as would be the case if merchandise samples were mailed.

6. Model Stability

Model stability is not strictly defined, but it refers to the difference in effectiveness between the model's performance with the development (training) sample versus the holdout (testing) sample. The same principle applies to both regression models and neural networks. A significant change in performance for the holdout sample may indicate overlearning the development sample, or suggest that these two samples are not from the same populations. Therefore, the model's performance on out-of-sample data would be unpredictable.

Overlearning is often attributed to an insufficient number of hidden units, but the real problem is due to too few training examples. In a neural network,

[7] The weight of 4 was calculated in the following manner: out of a total of 79,246 observations, 10506, or 13.26% were respondents and 68,740 were non-respondents. Therefore, there were 13.26%/3.7%, or 3.58 times as many respondents as the population and 86.74%/96.3, or .9007 times as many non-respondents. To bring the number of non-respondents up to 96.3 of the holdout sample, they should be multiplied by 3.58/.9007, or 3.98.

[8] Tables 4-2 and 4-3, do not include all 301 output unit values. They just include the output unit values under discussion.

the coefficients are the actual function estimators, and if there are insufficient degrees of freedom, as measured by the number of observations less the number of coefficients, then the estimation process will break down (see [5]).

Because the development sample was over 118,000 observations, and the total number of weighted connections was 9030, a stability problem was not anticipated, as there are over 100,000 degrees of freedom. In addition, a means and variance analysis was performed on the data when the logistic model was developed, to verify that the development and holdout samples represent the same population. For a more detailed discussion of model stability, see [2].

7. Summary and Conclusions

Direct mail response data often is characterized by complex and difficult to identify relationships that exist between likelihood of a response and the independent variables. This seems to suggest a high degree of non-linearity, which would lend itself to a neural network model. Logistic regression is a powerful tool for developing a robust response model. However, practical implementations of logistic regression still force the data into basically linear relationships which are transformed into probabilities through the non-linear logistic function. The process of developing a logistic model involves many steps in including and excluding variables and sometimes making some arbitrary decisions, such as increasing significance level cutoffs to force inclusion of certain variables.

The counterpropagation network yielded some very high performance groupings that demonstrated response lifts that exceeded the logistic model. The logistic model produced a smoother ranking of the response rates and performed better at the first and second decile levels. The trade off between the two models is that the logistic model performs more consistently for all deciles while the counterpropagation network shows significantly superior performance at the 0.1% and 1.0% levels. This is more of a reflection of the nature of the data than of counterpropagation networks in general Garavaglia [3] discusses a counterpropagation network that behaves very much like a logistic regression in terms of ranking output.

This research has applicability to any model that has categorical output. Although applications for financial markets typically require a continuous random variable output, classification and categorization techniques can be helpful. For example, in financial markets, it may be useful to classify securities or the overall condition of the market as volatile or stable, based on price movements, volume, and velocity. It might also be useful for classifying investments for a portfolio inclusion. An interesting application would be forecasting responses to news, e. g., which securities prices would rise or fall on some given news item which is not strictly financial, such as a political change in government. For these and other classification tasks, the counterpropagation network is a relatively straightforward paradigm that can be very effective.

8. References

[1] Amemiya, T. 1985. Advanced Econometrics. Cambridge, MA: Harvard U. Press.

[2] Garavaglia, S. 1993a. Desperately Seeking Stability: Neural Networks with Insufficient Data. Proceedings of the Second International Conference on Artificial Intelligence Applications on Wall Street. New York April 19-22, 1993. Gaithersburg, MD: The Software Engineering Press.

[3] _____. 1993b. Interpreting Neural Network Output. *Advanced Technology for Developers*, 2 (February).

[4] Hecht-Nielsen, Robert. 1990. Neurocomputing. Reading, MA: Addison-Wesley.

[5] Kennedy, Peter. 1985. A Guide to Econometrics. Second Edition. Cambridge, MA: The MIT Press.

[6] Kohonen, Teuvo. 1989. Self-Organization and Associative Memory. Third Edition. Berlin: Springer-Verlag.

[7] Maddala, G. S. 1983. Limited Dependent and Qualitative Variables in Econometrics. Cambridge U. K.: Cambridge University Press.

[8] NeuralWare, Inc. 1991. Neural Computing. , Pittsburgh, PA: NeuralWare, Inc.

Table 4-1. Logistic Model Expected Response and Lift Distribution

Cumulative % Of Total	Expected Response Rate %	Lift %	Cumulative % Of Total	Expected Response Rate %	Lift %
0.999	16.449	346.67	42.000	5.968	62.06
1.999	14.976	306.67	43.000	5.899	60.19
3.000	14.234	286.52	44.000	5.824	58.15
3.999	13.516	267.02	45.000	5.753	56.22
5.000	12.907	250.49	46.000	5.685	54.37
5.999	12.387	236.37	47.000	5.627	52.80
7.000	11.953	224.58	48.000	5.551	50.74
7.999	11.612	215.32	49.001	5.485	48.94
9.000	11.252	205.54	50.000	5.4325	47.52
*9.999	10.902	196.04	50.999	5.3755	45.97
11.000	10.551	186.51	52.000	5.3111	44.22
11.999	10.207	177.17	53.000	5.25	42.56
12.999	9.891	168.59	54.000	5.1988	41.17
13.999	9.602	160.74	54.507	5.171	40.42
15.000	9.373	154.52	72.000	4.5393	23.26
16.000	9.202	149.88	73.001	4.5073	22.39
16.999	9.003	144.47	74.000	4.4773	21.58
18.000	8.783	138.50	75.001	4.4465	20.74
18.999	8.577	132.91	76.000	4.4092	19.73
19.998	8.42	128.64	77.001	4.3824	19.00
20.999	8.263	124.38	77.989	4.3449	17.98
22.000	8.11	120.22	79.001	4.3128	17.11
22.999	7.951	115.91	80.000	4.2852	16.36
24.000	7.804	111.92	81.001	4.2569	15.59
24.998	7.652	107.79	82.000	4.2272	14.79
26.000	7.516	104.09	83.001	4.1965	13.95
26.999	7.394	100.78	84.000	4.1671	13.16
28.000	7.26	97.14	85.000	4.1403	12.43
29.000	7.156	94.32	85.996	4.1124	11.67
30.000	7.05	91.44	86.989	4.0771	10.71
30.998	6.95	88.73	88.001	4.0466	9.88
32.000	6.861	86.31	89.000	4.0161	9.06
33.000	6.759	83.54	89.997	3.9837	8.18
33.999	6.659	80.82	91.000	3.9552	7.40
35.000	6.568	78.35	92.001	3.9259	6.61
35.999	6.459	75.39	93.000	3.8942	5.75
37.000	6.364	72.81	94.001	3.8625	4.89
37.999	6.279	70.50	94.997	3.8338	4.11
39.000	6.198	68.31	96.001	3.8057	3.34
39.999	6.119	66.16	97.000	3.7763	2.54
41.000	6.045	64.15	97.999	3.7475	1.76
42.000	5.968	62.06	99.001	3.717	0.93
43.000	5.899	60.19	100.000	3.6826	0

Table 4-2 Counterpropagation Network Response Rates and Lift (By Descending Hidden Unit Value)

Hidden Unit Value	Expected Response by Hidden Unit	Lift By Hidden Unit	Cumulative Response	Cumulative Lift	Cumulative % of Total
0.649001	45.95%	1141.78%	45.95%	1141.78%	0.01%
0.567468	19.03%	414.41%	21.86%	490.92%	0.13%
0.561815	18.37%	396.41%	21.08%	469.65%	0.16%
0.526389	4.00%	8.11%	20.21%	446.10%	0.17%
0.515113	35.48%	859.02%	21.11%	470.58%	0.18%
0.506442	28.00%	656.76%	21.72%	486.90%	0.20%
0.487566	0.00%	-100.00%	20.84%	463.29%	0.21%
0.482154	21.57%	482.94%	20.90%	464.80%	0.23%
0.482045	19.41%	424.70%	20.46%	452.91%	0.32%
0.472345	11.11%	200.30%	19.70%	432.45%	0.35%
0.442956	0.00%	-100.00%	19.24%	419.97%	0.36%
0.438125	17.41%	370.51%	18.88%	410.35%	0.45%
0.419607	31.43%	749.42%	19.22%	419.43%	0.46%
0.418459	13.40%	262.22%	18.82%	408.57%	0.49%
0.395248	18.37%	396.41%	18.79%	407.77%	0.53%
0.393931	15.25%	312.28%	18.53%	400.81%	0.57%
0.378226	10.34%	179.59%	17.98%	386.03%	0.61%
0.377654	7.77%	110.05%	16.13%	335.82%	0.74%
0.373545	15.79%	326.74%	16.09%	334.98%	0.82%
0.369478	18.37%	396.41%	16.14%	336.25%	0.83%
0.369084	9.68%	161.55%	15.90%	329.68%	0.87%
0.367916	11.43%	208.88%	14.91%	303.03%	1.11%
0.364469	12.28%	231.91%	14.74%	298.25%	1.19%
		(. . . . .)			
0.272071	11.35%	206.79%	12.83%	246.64%	4.51%
0.271683	6.66%	79.91%	12.24%	230.94%	4.98%
0.270944	20.00%	440.54%	12.26%	231.31%	4.98%
0.266984	6.83%	84.61%	11.75%	217.62%	5.50%
0.265386	7.09%	91.53%	11.28%	204.79%	6.12%
		(. . . . .)			
0.22577	6.30%	70.20%	9.95%	169.05%	9.17%
0.224376	6.06%	63.91%	9.84%	166.03%	9.44%
0.224375	5.78%	56.08%	9.57%	158.52%	10.13%
0.222413	5.88%	58.98%	9.56%	158.35%	10.15%
0.220778	8.65%	133.80%	9.54%	157.87%	10.35%
		(. . . . .)			
0.167238	1.85%	-49.95%	8.16%	120.62%	17.87%
0.167215	5.66%	53.05%	8.09%	118.56%	18.43%
0.166109	4.63%	25.14%	7.86%	112.33%	19.74%
0.166041	4.19%	13.17%	7.75%	109.55%	20.31%
0.163741	5.88%	58.98%	7.75%	109.49%	20.34%
		(. . . . .)			
0.148977	7.28%	96.89%	7.51%	103.03%	21.94%

Table 4-3 - Counterpropagation Network Response Rates and Lifts (By Best Performing Hidden Unit Values)

Hidden Unit Value	Expected Response by Hidden Unit	Lift by Hidden Unit	Cumulative Rates Total %	Resp. Rate	Lift %
0.283334	96.17%	2499.25%	0.00%	96.17%	2499.25%
0.287182	55.06%	1388.00%	0.10%	83.89%	2167.37%
0.649001	45.95%	1141.78%	0.12%	79.70%	2054.09%
0.515113	35.48%	859.02%	0.13%	75.96%	1952.87%
0.419607	31.43%	749.42%	0.14%	72.07%	1847.83%
0.506442	28.00%	656.76%	0.16%	67.18%	1715.78%
0.482154	21.57%	482.94%	0.18%	62.55%	1590.54%
0.282957	21.09%	469.96%	0.23%	53.16%	1336.72%
0.351747	20.00%	440.54%	0.24%	51.93%	1303.48%
0.270944	20.00%	440.54%	0.24%	50.79%	1272.62%
0.250382	20.00%	440.54%	0.25%	50.35%	1260.88%
0.482045	19.41%	424.70%	0.34%	41.75%	1028.42%
0.567468	19.03%	414.41%	0.46%	36.02%	873.63%
		(. . . . .)			
0.418459	13.40%	262.22%	1.01%	25.25%	582.48%
0.361764	13.30%	259.47%	1.22%	23.17%	526.08%
0.326895	13.20%	256.66%	1.34%	22.28%	502.09%
		(. . . . .)			
0.179954	8.97%	142.31%	4.81%	13.94%	276.76%
0.306465	8.84%	139.01%	4.91%	13.83%	273.88%
0.171239	8.77%	137.08%	4.95%	13.79%	272.77%
0.220778	8.65%	133.80%	5.16%	13.59%	267.31%
0.121992	8.26%	123.36%	5.32%	13.42%	262.73%
		(. . . . .)			
0.171308	7.17%	93.71%	9.32%	10.85%	193.17%
0.171255	7.13%	92.66%	9.50%	10.78%	191.35%
0.265386	7.09%	91.53%	10.12%	10.55%	185.20%
0.168548	6.98%	88.78%	10.51%	10.42%	181.57%
0.260886	6.98%	88.56%	10.53%	10.41%	181.44%
		(. . . . .)			
0.130014	4.69%	26.84%	19.31%	8.34%	125.34%
0.209828	4.67%	26.17%	19.45%	8.31%	124.62%
0.179655	4.63%	25.22%	19.81%	8.24%	122.80%
0.166109	4.63%	25.14%	21.13%	8.02%	116.71%
0.137065	4.56%	23.35%	23.02%	7.73%	109.05%
0.126784	4.51%	21.97%	23.61%	7.65%	106.88%
0.157028	4.51%	21.87%	23.87%	7.62%	105.94%
0.171645	4.47%	20.70%	24.05%	7.60%	105.30%

Appendix A
Independent Variables Used in the Response Model

The signs in parenthesis indicate whether a positive value in the variable results in a greater (+) or lower (-) likelihood of responding.

SLNGPCT2 - Percentage of total payment experiences that are slow or negative (+)
AGE - Age of company (years since most recent change in control) (-).
SALES - Annual sales (-).
TOTEMPL - Total number of employees (-).
PAYDEX1 - The most current D&B Paydex score (a payment promptness indicator) (-).
ZCHGPOP - Zip Code area population change (+).
PDX1MIS - An indicator if PAYDEX1 is missing (-).
VAR6MIS - An indicator if the variance of the 6 most recent Paydex scores are missing.(+).
ZCHPOPLW - An indicator that the Zip Code area population change is missing (+).
SLNGPMIS - An indicator that the percentage of slow or negative payment experiences is missing.(-)
NEW_FIRM - An indicator that the firm is new to the D&B database (+).
STAT_BR - An indicator that the status of the firm is a branch (-).
SUB_YES - An indicator that the firm is a subsidiary (-)
REC_ITEM - An indicator that the firm has one or more suite, liens, or judgments filed against it or the firm has filed for bankruptcy in the past (+).
AGEMISS - An indicator that the age of the firm is missing (-).
VAR611- An indicator that the variance in the most current 6 months of Paydex scores is greater than 11 (+).
ZDEN_HG - An indicator that the population density in the zip code is greater than 12,501 people per square mile (-).
SIC1_90 - An indicator that the response rate for this SIC code is in the 90th percentile of all SIC's (+).
SIC1_10 - An indicator that the response rate for this SIC code is in the 10th percentile of all SIC's (-).
SIC1_75 - An indicator that the response rate for this SIC code is in the 75th percentile of all SIC's (+)
SIC1_25.- An indicator that the response rate for this SIC code is in the 25th percentile of all SIC's. (-)
MSA_75 - An indicator that the response rate for this MSA (Metropolitan Statistical Area) is in the 75th percentile of all MSA's (+).
MSA_25 - An indicator that the response rate for this MSA is in the 25th percentile of all MSA's (-).
MSA_10 - An indicator that the response rate for this MSA is in the 10th percentile of all MSA's.(-).
LT1_4 - An indicator that the firm owns 1 to 4 light trucks (+)
PUB_NO - An indicator that the firm is NOT a public company (+).
STATE_BD - An indicator that the firm is located in Maryland, Missouri, Virginia, Illinois, Michigan, New York, Wisconsin, Louisiana, Pennsylvania, or the state is missing (-).
SIC_TRCM - An indicator that the firm is in the transportation, communication, or utilities industry (+).

A METHOD TO PREDICT THE TELEVISION AUDIENCE BASED ON NEURAL NETWORKS.

Eloy Parra Boyero
Dpto. de Informática
Radio Televisión de Andalucía
Crta. San Juan - Tomares Km. 1.3
41920 San Juan de Aznalfarache
Sevilla, SPAIN

Francisco A.Triguero Ruiz, and
J.L. Pérez de la Cruz
Dpto. Lenguajes y Ciencias de la Computación
Facultad Informática, Universidad Málaga
Plaza El Egido s/n
Málaga. SPAIN.

ABSTRACT.

This article describes a system to predict the Television (TV) behaviour by using Retropropagation neural networks.

The system allows us to find out the share of the market that each of the Television Channels in Spain holds, during each fifteen minutes period, as well as the total television viewership.

The system is specially useful to improve the TV programming and to work out the purchasing of the advertisement time.

The main problem found when developing a system to predict the television audience behaviour is the wide variety of television programmes, even within the same field, and the great quantity and complexity of factors that affect the audience.

1. RESEARCH ENVIRONMENT.

THE USE OF THE SYSTEM IN "WALL STREET".

The system designed allows us to anticipate the number of people who are going to watch each of the channels every fifteen minutes.

The system is especially useful for the television stations and for the advertising companies designing and purchasing advertisement time.

Thanks to this system, television networks can test and select the best choice of programming and therefore can improve their audience shares. The improvement in the audience shares implies getting higer incomes from the sale of advertisement time.

Besides knowing the audience shares beforehand, it also allows the television network to sell their advertisment time more easily.

The businessess that design and buy the advertising time, can clearly evaluate the expense of a certain advertisement as they know the number of people the advertisement wil reach, and in this way, they can be more competitive.

The system is efficient, not only for the predicction concerning the population; but also for the different social groups such as adults, house wives, children, etc. This is relevant when buying the advertisement time which is directed at a specific social group.

THE DIFFICULTY OF THE TASK.

There are many factors which affect the audience.
Apart from the programming itself and competition from other stations, political events, sport specials, trends, weather, the station reputation, the time of the programme, previous advertising of the programme, number of publicity spots, etc, all have a role to play.

And in respect to the programmes it is necessary to consider the kind of programme, the age of the program, if it is black and white or color, previous audience shares on the same programme or in the same type of programme, etc.

All this has made the creation of methods to **predict** the audience behavior a difficult task.

ACTUAL METHODS MEASURING PAST AUDIENCE.

There are systems wich measure recent audiences based in audiometres.

Audiometre is a mechanism that automatically picks up the TV channel that it is tuned into, and they also get either manually, or automatically the presence of the different members of a family who are watching TV.

These audiometres are installed in a sample number of homes which alternate all the time.

Everyday the audiometres unload the data they have got, by means of a modem. The family which has been selected benefits from free telephone calls while the mechanism is installed at their house.

The data is processed and offered to the businesses involved. The businesses that buy this service, get the data everyday through a modem.

The audiometres are an efficient system of analysis, but not for the prediction of a future day. For example, just think that the broadcasting of a good programme in any of the other channels can ruin a programme on another cannel which used tc have high audience shares in previous broadcastings.

This system will provide us with most of the data that wil be used to train the network.

We have got a file with the information relating to the programmes that have been broadcasted and the audience shares for each channel every fifteen minutes.

OTHER TV SYSTEMS OF AUDIENCE PREDICTION.

They are several audience predict systems.

On the one hand, we have these which are based on public opinion polls. These are nor quick enough and they are very expensive, as they are based on public opinion polls.

The use of Panels of experts are also being used succesfully. They show the average audience shares got from the predictions made by a group of experts in the field. This method requires a big human effort and therefore it is very expensive, especially if we take into account that are looking for the predictions for each of the television networks everyday and every half an hour or every fifteen minutes.

Several mathematical models, based on temporal series are also used. The system that we have developed has been compared to the best mathematical model used in Spain, the "Mitra" system. The error rate acumulated while testing using the mathematical model during a certain period of time was 1.2 %, while the error rate in the system based on neural networks was 0.03% during the test week, that is, it was 40 times lower.

AUDIENCE CONCEPTS.

Before we go on, the basic elementary concepts related to the television audience will be defined, as they are going to be used below.

Total television viewership.

Total number of people watching TV.

Average Audience.

Total number of people watching a certain TV channel.

If during a fifteen minute period of time, different people have been watching a channel, for five minutes each, it is counted as if a single person had been watching the channel continuously.

Audience Share.

Percentage of people watching a certain television channel in relation to the total televison viewership.

Weighted average Share.

It is the weighted average share in relation to time and to the total television viewership.

The formula for the weighted average share for n shares c_i during a period of time lasting d_i, and with a total TV consumption during that period of time C_i is:

$$S=\frac{\sum c_i \times d_i \times C_i}{n \times \sum d_i \times \sum c_i}$$

IN SEARCH OF A SOLUTION.

We consider that the population tastes concerning audiences are quite regular and defined, because we usually have the same tastes.

From this idea we decided to create a system capable of finding the audience behaviour patterns when faced wiht a choice of programmes, talking into account the programming of the competition and all the parallel factors affecting the audience.

Knowing that certain neural networks algorithms have been used to recognize patterns, and after studying the alternate choices, the Retropropagation paradigm was chosen.

A system which allows us to obtain the market share for each TV channel against other competitive channels will have gathered the audience taste patterns or the audience behaviour offer towards a TV.

The system we will describe, allows us to get each of the TV channels Market Shares, every fifteen minutes from 14.30 to 24.15 p.m.

We will also describe the method followed to calculate the Total Viewership every fifteen minutes.

This allows us to calculate the Average Audience by applying the Shares to the Total Viewership.

2. METHODOLOGE.

If we do a synthesis of the research works already carried out can be divided into the followins sections:

Step 1. **Selection of the Network Architecture.**
Step 2. **Selection of the tool.**
Step 3. **Selection of the data.**
Step 4. **Collection and preparation of the data.**
Step 5. **Network training.**
Step 6. **Network Test.**
Step 7. **Repetition of the last four steps.**

The selection of the data as an important key phase is incorporated into this methodology.

Step 1. **Selection of the Network Arquitecture.**

We will use the Retropropagation paradigm. The reasons for this are:

- We have the data required to do supervised training, which is the one needed by the Retropropagation networks.
- The algorithm is used for general purposes and its convergence has been proved.
- It is the most commonly used paradigm for practical purposes, and it has been succesfully used to solve problems of the recognition of patterns and prediction, which is the matter we are dealing with.
- Other kinds of classifying algoritms require a previous classification which is really expenive in our case because of the high number of possible exits from the network.

Step 2. **Selection of the tool.**

A study of the different neural networks packages on the market was done and the NeuralWorks was selected. Although it is easy to program the nucleus of the network, it is very expensive to program the user interface in relation to the change of the parameters, the definition and modifications of the network, graphics analysis, the general operations of the networks, etc. So for the creation of practical applications it is interesting to have the tool already constructed.

Step 3. **Selection of the data.**

Obviusly, The networks can not work magic and they can not derive nor find number relationship between output data and input data, hence the great importance of the selection of the data the network is going to be fed with.

After the first result, the selection can be improved in different ways:

- By studying the importance of the input data to the final results.
- Eliminating the input data that has connections with very low weights in the next layer.
- Analysing the errors in the network in order to deduce which data is missing is excessive or is incorrect.

This step has been one of the hardest ones in this work.

Step 4. **Collection and preparation of the data.**

In this case, the collection of data was an easy task as we had most of it kept in files DBASE, comming from the audiomeasurement businesses. Programmes in Clipper were developed to deal with the data bases.
The recodification of the data is show below (later in this paper).

Regarding the preparation of the data for the training and test, we have to try to get good training data, which is representative of the universe we are working with. The difficult categories of input data have to be represented by an analogous number of cases during the training time in order to avoid that the network to learns more about certain categories or kind of data, to the detriment of others.

In our case, it was impossible to make a balanced selection of the different categories of input data, as there were many posibilities and the possible rules of grouping gathering all the cases in a few ones are unknown. We chose then, to provide the network with all the data

gathered, and we hope that this was complete and representative enough.

Step 5. **Network training.**

A computer with an operting system MS DOS and a 80486 25 MH procesor was used.

The network is trained by varying the different parameters and functions, structures and ways of training as well as the input data and the different variations of the selected algorithm, in order to get the most accurate results.

Step 6. **Network Test.**

The ability of the network is tested over a week and its results are shown on graphics.

The output information is analysed, and hypothesis to correct the defects are set up.

We **repeat** the previous steps from step 3 until we get satisfactory results (Step 7).

3. TECNIQUE.

We can describe the tecnical solution found from:

- **Selected output data.**
- **Selected input data.**
- **Determined network architecture finally gained, training parameters, and the network paradigm that has been used.**

3.1. SELECTED OUTPUT DATA.

Originally, we tried to get the average audience, but in the best solution reached, the Shares and the total Viewership are calculated separately, and from this the average audience is obtained.

It is convenient to take the total TV viewership and the market Share as separate output for the following reasons:

- The total television consumption is a more regular piece of information, and therefore, it is easier for the networks to predict it.
- The Share has a direct relation with what is broadcasted, so it will be easier for the network to obtain the share from the programmes in competition rather than obtain the Average Audience.
- The network does not have to calculate the Average Audience and we can do the job for it.

Besides, in order to simplify the network work, we make it predict the Share of a single channel each time, and this way, we get very good results in the convergence of error. This is due partly to the definition of the half squared error itself.

3.2. SELECTED INPUT DATA.

The defined and selected input data is the followin:

* TIME.
* STRENGTH.
* QUALITY.
* TYPE.
* PREVIOUS PROGRAM FORCE.
* FOLLOWING PROGRAM FORCE.
* LENGTH OF TIME.

This information will be prepared for each quarter of an hour.

Time.

We introduce the time expressed in minutes at the beginning of the 15 minutes period we are studying.
We intend by this piece of information, to provide the network with a variable from which it can derive the different behaviour depending on the different periods of time. This different behaviour does exist, as the kind of television viewers changes from those who watch TV after lunch or those who watch in the evening or in the night.

This piece of information will occupy a neuron in the input layer.

Strength.

Programmes are classified depending on groups and subgroups. For example, horror films or tennis sport.
We define the strength of a programme as the weighted average Share, that the group and subgroup of programme it belongs to obtained from the previous year and during a period of time lasting from 13 to 25 hours.
We consider that this is a basic fact that justifies the behaviour of the audience.
5 Neurons in the input layer will be used for this piece of information, one for each of the main five TV Spanish channels.

Types.

Programmes have been reclasified in the folowins types:

Tipo 1. Cinema, Theater.	1000000
Tipo 2. Series.	0100000
Tipo 3. Quizes	0010000
Tipo 4. Variety shows.	0001000
Tipo 5. News.	0000100
Tipo 6. Sport and bull fight.	0000010
Topo 7. children's programmes.	0000001
Tipo 0. Others (Religion,...).	0000000

We try to provide the network information with the nature (main characteristics) of the programme so it can pick up the preferences of the audience towards the different kind of programmes.
35 Input neurons are used, 7 for each of the channels. The column on the right shows the codes used.

Quality.

Knowing the wide variety within certain programmes belenging to the same group and subgroup, it is advisable to categorize the programme itself.
For certain programmes, three qualities have been used: Good, Medium and Bad. Categories such as Medium-Good or Medium-Bad, have been used for programmes which were difficult to classify in the main categories.
This categorization has been made with films, football and soap operas.
It also has specially to be done with any particular programme which stands out among the ones belonging to its same group and subgroup.
This categorization of the broadcasting has been made by an expert in Audience.
5 Input neurons are used to get this information.
To provide the network with this categorization, we codify the categories from Bad to Good giving them values from 0 to 4.

The Strength of the previous and the following programme.

The previous and following programme strength is the same concept which has been defined above, this time related to the programme that preceeds and the programme which follows the programme which is broadcast during the fifteen minutes we are dealing with.
This variable is used because it is known that the previous and following programmes do affect the audience.
10 input neurons are needed to provide the network with this information.

Length of time.

The network is also provided with the length of time of the programme expresed in minutes.
This information is basically provided so that the network can grasp the influence of the previous and the following programme depending on the length of time of the programme which is broadcast.
5 input neurons are used to get this piece of information, one each of the channels.

Others.

A bias neuron has been used with a constant value of 1.

Others pieces of information which have not been selected in the final procedure are: typical deviation of the strength, whether,it is the week-end or not, closeness to the previous and following programmes, the day of the week, the strength calculated for each programme, the language, the season, whether it is a repeat or not, if it is revival or recorded, etc.

We want to emphasize that the DAY of the week is not provided, despite being a very important piece of information, in order to intend for the network to use the information that we have mentioned above and by inter-relating this information, thereby providing us whit an accurate account of audience tastes.

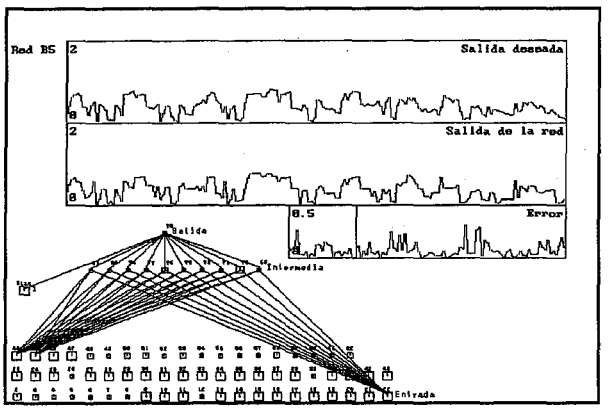
In order to train the network, we used the information from the month preceeding the week or the period of time the preditions are going to be made.

3.3. FINAL DETERMINED ARQUITECTURE OF THE NETWORK.

3.3.1. PREDICTION OF THE SHARE.

For the prediction of the Share we use a different network for each of the TV channels.

The following picture shows the representation of the network used for the prediction of the share.



Number of layers and neurons.

A network whith 61 input neurons will be used. They are used as follows: 1 for the time, 5 for the strength, 5 for the quality, 35 for the type of programmes, 10 for the previous and following programme strength and five for the length of time.
In the hidden layer the most appropiate number of neurons is 10.
The output layer has a single neuron where we will get Share.
The transfer function used is hyperbolic tangent.

Learning and calculating Algorithm.

The algorithm used is normalized accumulative Retropropagation.
The presentation of the training couples has been made in an aleatory way.

Parametres and training plan.

The epoch, the error coeficient so that the learning is stopped, learning coeficient (coef.), and the momentum term are the parametres to be defined.

These parametres, except for the error coeficient, vary depending on a training plan based on the number of learning iterations for the network training:

Iterations	5000	15000	25000	35000
Learning Coef.	0.15	0.075	0.01875	0.00117
Momentum	0.4	0.2	0.05	0.00313

The **error coefficient** used is **0.001**. The **epoch** is equal to **4** while training time, and if the convergence has not reach in this time, we change the epoch to 1.

We randomize the strength of the connection weigths before the training.

The number of iterations was aproximately 36000, and the convergence was always obtained.
The training times was around 20 minutes.

3.3.2. OBTAINING THE TOTAL TELEVISION VIEWERSHIP.

In order to predict the total viewership, we use to networks, one for the weekend another similar for the other days of the week.

Input Data:
HOUR, STRENGTH, and QUALITY.

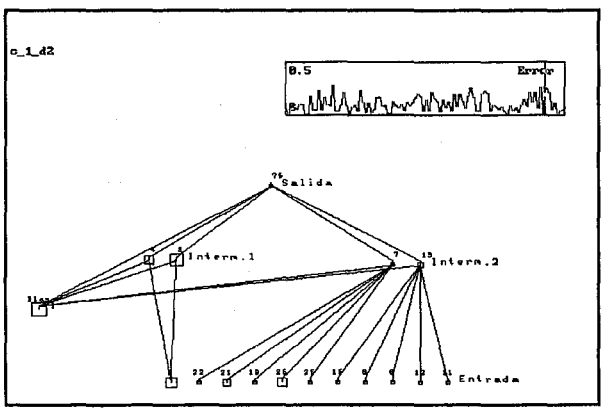
Algorithm: Accumulative normalized Retropropagation.

Parametres.

Half squared Error equal to 0,0011.
The training plan, the leaning coefficients and the momentum are the same as were used to calculate the Shares. They were shown above.
The number of learning iterations was around 19000 for the weekend days, and about 42000 for the rest of days of the week.

Number of neurons and layers during the week.

The following picture show the network architecture.



An 11 input neuron layer has been used, two hidden layers with two neurons each, an output layer with only one neuron, and a bias neuron.

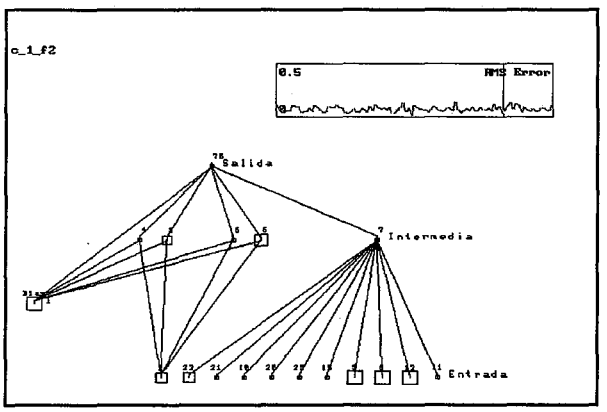
The graph shows the connections which are established.

The input data referring to time, is linked to one hidden layer. The input neurons referring to the different channels strength are linked to just one neuron, and the quality of the programmes is linked to another neuron placed in the second hidden layer.

The hidden layer transfer function is the sine funchtion. The hyperbolic tangent function is used in the second output layer.

The training is carried out in steps, first training the network without the second hidden layer, and then we put the iterations counter to 5000 and follow the training with the second hidden layer.

Number of layers and neurons for the weekend.



One input layer, three hidden layers, an output layer, and a bias neuron are used.
The information concerning time is linked to the first two hidden layers. One of them has the sine function as the transfer function, the other has the hyperbolic tangent function.

The values of the strength and quality programmes of the various channels are linked to the third hidden layer wich has a single neuron with a hyperbolic tangent transfer function.

It is trained in steps as with the previous network. The different layers are swiched on in sucession.

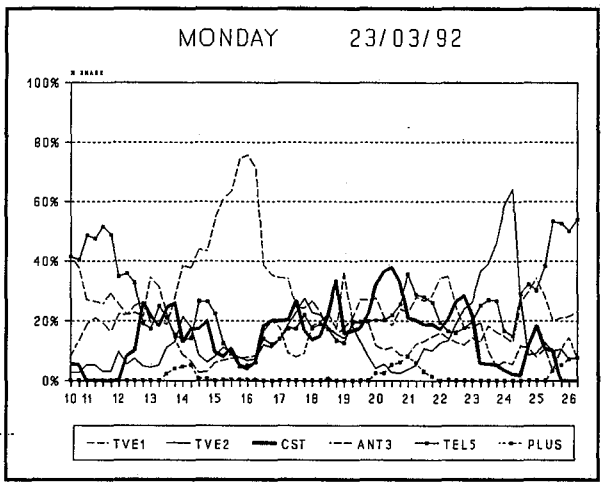
4. RESULTS.

The results wil be given first though graphs and later in an analytical way.

Due to limitations of length, only the Share and total television viewership for the first day of the testing week in CST (Canal Sur TV, the TV broadcast channel of Radio TV of Andalucia) wil be shown. And finally, the result referring to the Average Audience and total viewership for the whole testing week in the same channel, wil be shown in small graphs.

4.1. SHARES TO BE ESTIMATED.

A graph showing the "real" data found on the first day of the week is included (23/03/92), as an example of the large share variety.



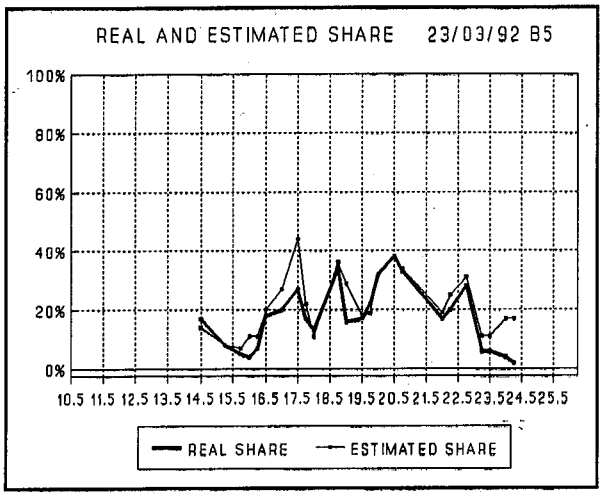
The "real" data has been obtained from an audiometer company, and it is at the same time a valuation of reality.

The abbreviations TVE1, TVE2, CST, ANT3, TEL5, y PLUS, correspond to the different Spanish TV Channels.

4.2. SHARE PREDICCTIONS FOR "CANAL SUR".

The following graphs show both the real data and the data estimated by the network for the first day of the testing week.

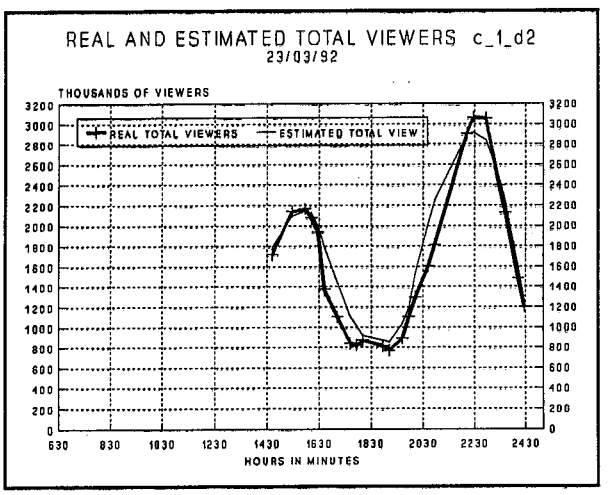
The network was trained with data from previous weeks to the testing week.



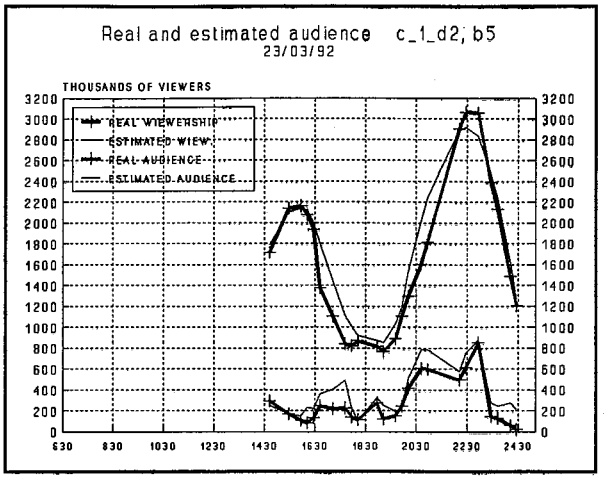
Note the change produced at 15:30 on Monday, Tuesday, and Wednesday, wich corresponds to a soap opera. This change is due to the end of a quite succesful soap opera (during the training period), and the beginning of an unsuccessful one during the training period also.

4.3. TOTAL TELEVISION VIEWERSHIP PREDICTION.

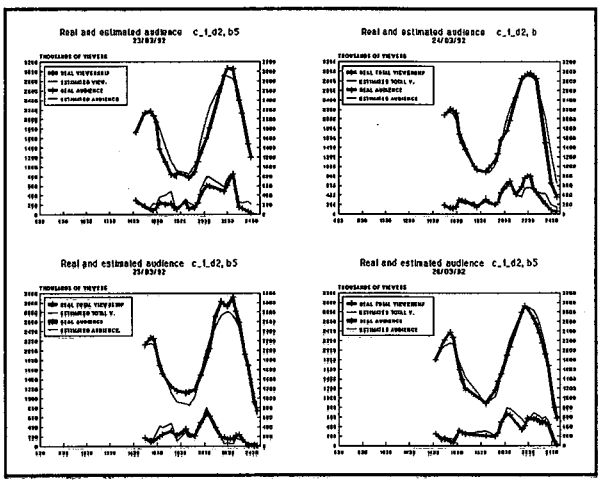
The results for the total television viewership are the following:

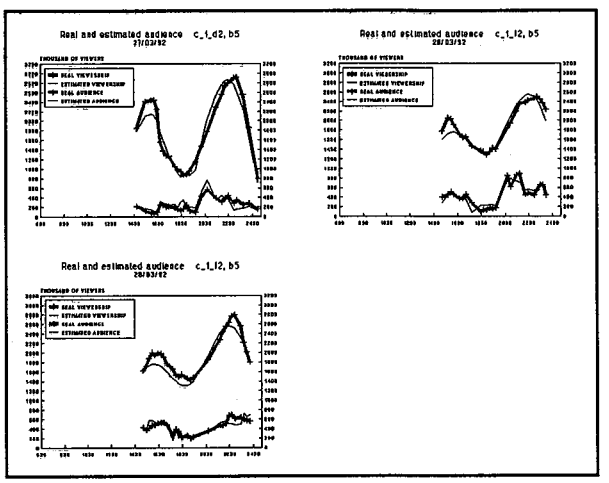


If we combine the last two graphs we can obtain the Average Audience.



Now, the whole testing week results are shown, including the first day of the week (23/03/92), which has just been represented.





4.4. ERRORS.

After a comparative analysis of the data obtained by the network and the real data, we find the following errors for each of the channels:

Share errors:

	A.E.	A.E.A.
CS	-1.84	4.40
TV1	0.37	6.01
TV2	0.94	4.23
AN3	0.66	3.62
TL5	-0.10	4.62
Totals	**0.03**	**4.58**

The acumulative error (A.E.) is the sum of the errors of each estimation, and the average error in absolute value (A.E.A.), is the average of the absolute value of the errors.

Average Audience errors:

	A.E.	A.E.A.
CS	-23.58	79.93
TV1	0.70	130.76
TV2	15.78	72.93
AN3	10.83	62.61
TL5	-0.93	88.75
Totals	**2.8**	**86.99**

The Share error values are expresed by percentages and the average audience errors are expressed by the number of TV viewers.
The value of the average error in absolute value gives us the average error measure wich is made.

5. STATISTICS.

The method has been developed throughout the year 1992.

Three complete loops of steps 3,4,5 and 6 were carried out.

During the second loop, 36 networks were trained for periods of time lasting from 15 to 45 minutes.

In the third loop, about 130 networks were trained . The average training time was 25 minutes long and the work was completed throughout one month, all day long.

The steps which took the longest were steps 3 and 4, that is the selection and the preparation of the data.

The computer used in the first loop was a 386sx computer with mathematics coprocessor with 16 Mh (Mega Herzios), and for the other loops a 486 at 33 Mh computer were used.

BIBLIOGRAPHY.

* Claude Cruz.
  Understanding Neural Networks: A Primer.
  Cutter Information Corp. 1991.
* Flach Peter A., Meersman Robert A..
  Future Directions in Artificial Inteligence.
  IFIP TC12 Founding Workshop Collected Papers.
  Ed. Nort-Holland. Elsevier Science Publishers B.V.
  1991.
* Herzt, Krogh y Palmer.
  Introduction to the theory of Neural Computation.
  Addison-Wesley 1991.
* James A. Freeman, David M. Skapura.
  Neural Networks.
  Algorithms, Applications, and Programming Tecniques.
  Addison-Wesley 1991.
* NeuralWare
  Neural Computing.
  NeuralWare, Inc. 1991.
* Tarun Khanna.
  Fundations of Neural Networks.
  Addison-Wesley 1990.
* Several Authors.
  Neurocomputing.
  Fundations of Research.
  Ed. James A. Anderson, Edward Rosenfled.
  MIT Press. 1998.

Invited Speaker: Neural, Genetic, and Fuzzy Approaches to the Design of Trading Systems

Guido Deboeck, The World Bank

Neural, Genetic, and Fuzzy Approaches to Design of Trading Systems

Guido J. Deboeck
World Bank,
1818 H Street NW, Washington DC 20433
e-mail : gdeboeck@worldbank.org

"Any sufficiently advanced technology is indistinguishable from magic"
Arthur Clark

Abstract

"When the only tool you have is a hammer, everything begins to look like a nail". This Lotfi Zadeh quote underscores the need for knowing and using multiple tools, especially when trying to improve trading, risk and portfolio management. This paper focuses on differences between neural networks, genetic algorithms and fuzzy logic; it reviews strengths and weaknesses of various advanced technologies, and suggests what developments are needed to make the design of trading systems more effective. The paper also suggests that advanced technology will have a substantial impact on trading, risk and portfolio management only when advanced techniques are better understood, made easier to use, and made more accessible to many more users. [1]

1. Introduction

Over the past few years trading technology has moved from video to digital, from stand-alone to integrated networks of workstations, and from voice to image processing. The next step is machine learning and related techniques for pattern recognition, optimization and support of trading decisions. Neural networks, genetic algorithms, and fuzzy logic are techniques to improve performance and reduce risks in trading, risk and portfolio management.

The advent of advanced technology always scares people. When "mechanical horses" first appeared around the turn of the century lots of people were opposed; when large computers emerged in the forties some could only see them used for census surveys; when microcomputers emerged in the seventies they were advertised for checkbook balancing... In the early nineties traders, portfolio managers and systems managers are apprehensive, fearful, or alarmed, when they read about neural networks, genetic optimization, or fuzzy control.

Many traders have difficulty coping with the idea that a model can achieve better hit ratios. Portfolio managers, even those who believe in the value of trading models for disciplined trading, are struggling with how to integrate computer and human-based trading strategies. Systems managers with responsibility for trading technology, prefer to ignore the new analytics and focus their attention on familiar grounds of selecting workstations, networks, or operating systems.

Despite all this, applications of neural networks, genetic algorithms, fuzzy systems, nonlinear dynamics and other techniques have penetrated trading floors and are no longer the sole hobby horses of rocket scientists. Applications of these techniques appear in respectable journals, even the popular press. Tools have become available that allow individual users, without the financial and computing resources of large organizations, to apply these techniques from within common spreadsheet environments.

This paper starts out by outlining shortcomings of traditional approaches to trading (section 2). Section 3 discusses neural net approaches to the design of trading models. Section 4 outlines how genetic algorithms can be used to avoid problems of training or configuring neural nets. Section 5 demonstrates how fuzzy logic systems can be deployed. Finally, this paper points to key strengths and weaknesses of each of these approaches; and what is needed to get the magical solutions everyone is expecting from advanced technology.

2. Flaws of Traditional Models

Traditional fundamental and technical analyses make lots of simplifying assumptions to make the design of models more manageable. Many traditional models are simple rule-based systems based on "what if" scenarios. Some of these rule-based systems have less than 10 rules, use moving averages or other technical

[1] The ideas expressed in this article are those of the author and do not necessary represent those of his current employer. Guido J. Deboeck is Manager of Advanced Technology in the Investment Department of the World Bank. Queries on this article can be addressed to: gdeboeck@worldbank.org

analysis indicators.

Rule-based systems are static and deal only with symbolic information where inputs, thresholds and decision-rules change in discrete steps. Rule-based systems cannot deal easily with non-linearity; they need to be tuned based on changing market circumstances. Rule-based systems assume all factors are independent; that influences of various factors on one another can be separated. They orthogonalize inputs; filter out dependencies; and assume Gaussian distribution of price or yield changes.

The problem with these assumptions is that many factors are treated as unimportant. It is assumed that when an event perturbs the system, it will revert to equilibrium after a short time span. It is also assumed that traders react to the same information in a linear fashion; that groups of traders are not prone to fashion; that in the aggregate all traders are rational (even if they do not behave rationally as individuals); the main assumption is that trader skills and experience are normally distributed !

Given these shortcomings more and more financial institutions in Wall Street, London, Franfurt and Tokyo are studying market behavior in new ways. At the World Bank in Washington, an R&D project on the use of advanced technology for design of trading systems was started in the summer of 1990. Over the past 3 years this project has produced new insights on the behavior of financial markets and the use of advanced technology for trading operations. Annex 1 contains a list of publications, workshops and tutorials developed at the World Bank since the summer of 1990.

Table 1 (on the next page) provides an overview of the new insights into the behavior of financial markets.[2] Techniques from signal processing, nonlinear dynamics, fractal geometry and/or chaos theory were used to test new market hypotheses concerning the behavior of the US Treasury markets that contradict traditional views of market behavior. This table categorizes market hypotheses based on whether market signals as stochastic or deterministic; and classifies trading strategies and modeling approaches based on whether they are linear or nonlinear.

If one assumes that market signals are damped harmonic oscillators with random noise or Gaussian distributions of returns then **markets can be efficient.** In this case predicting the market is difficult or impossible. This leads to the suggestion that an index or benchmark is the only way to produce returns equivalent to market returns. If temporary market inefficiencies are assumed then it is possible to accept a **trending market hypothesis.** In this case historical analysis can be used to put together a clever benchmark that outperforms a market index by benefiting from temporary inefficiencies. As temporary inefficiencies disappear, benchmark compositions need to be updated. Models of limited complexity produce value added by capturing on-going trends for short periods of time.

If on the other hand market signals are damped anharmonic oscillators with random noise (that is only approximately Gaussian) then a quite different market hypothesis needs to be considered. Peters and others have suggested a **fractal market hypothesis.** The fractal market hypothesis entails that markets are nonlinear dynamic systems with non integer fractal dimensions of which the predictability can be estimated. The nonlinear and fractal nature of financial markets is illustrated by Figures 1 and 2.

Figure 1
Phase Plot of 2 year Treasury yields plotted over time

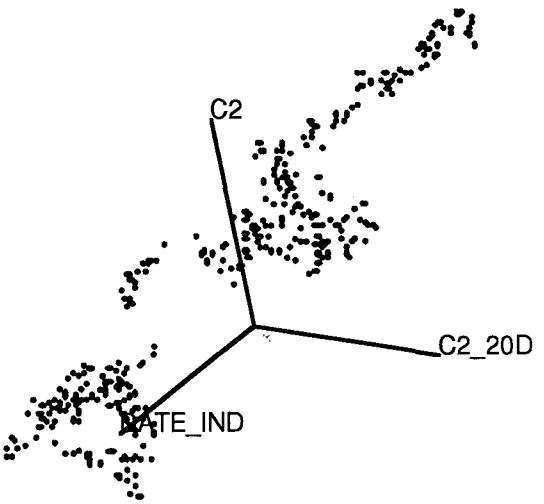


Figure 1 shows a phase space of two year Treasury notes; figure 2 shows the spread of fed funds versus 30 year bonds over time. The fractal market hypothesis permits the dimension of the probability distribution of price or yield changes to range between 1 and 2 and to be either a fractional or integer value. The efficient market hypothesis assumes that the probability density function of the price or yield changes equals 2 exactly. A further refinement, the **coherent market hypothesis,** advanced by Vaga [Vaga90] suggests that there are at least four different market environments : coherent bull markets, coherent bear markets, transition markets, and random walk markets. Each requires a quite different investment strategy and risk management. For example, in coherent bull markets a buy/hold strategy may have a better reward/risk ratio than any other strategy.

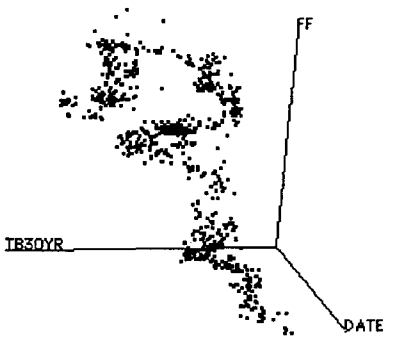
[2] This table is derived from the work on "Chaos and Order in Financial Markets" by Edgar Peters the Coherent Market Hypothesis by Tonis Vaga, and the work of many research groups who over the past 2-3 years have demonstrated nonlinear properties of financial markets.

In sum, the fractal and coherent market hypotheses provide more general paradigms of the behavior of financial markets.

Table 1 : Trading Strategies, - Models, and their Applicability

		Linear Models	Non-Linear Models
Stochastic Market Signals	Market Signal	I. Damped harmonic oscillator + random forces (Gaussian distribution of returns)	IV. Stochastic Chaos : damped anharmonic oscillator + random forces
	Market Hypothesis	**Efficient Market Hypothesis**	**Coherent Market Hypothesis**
	Trading Strategies	Noise Trading	- Dynamic Neural Net Based Trading based on fractal dimensions and Lyapunov exponents
	Applicability	- Random Walk Markets	- Coherent Bear /Bull Markets - Trading Markets and - Random Walk Markets
Deterministic Market Signals	Market Signal	II. Damped harmonic oscillator + random forces (Gaussian distribution of returns)	III. Deterministic Chaos damped anharmonic oscillator + random forces
	Market Hypothesis	**Trending Market Hypothesis**	**Fractal Market Hypothesis**
	Trading Strategies	- Fundamental & Technical - Trend Following - Rule-based Models	- Static Neural Net-based Trading ; non-linearities
	Applicability	- Bull/Bear Markets only	- Bull/Bear and Trading Markets

Figure 2 :
Phase Space of the spread of Fed Fund rates versus 30 year Bonds plotted over time



3. Design of Neural Net Models

If financial markets are nonlinear systems influenced by many variables with complex relationships or behavior of market prices with chaotic attributes then trading these financial markets requires much more sophisticated market intelligence than the indicators commonly displayed on most trading workstations.

The advent of more powerful workstations and analytical techniques specifically the advent of machine learning techniques such as neural networks, provide enhanced capabilities for design of models to provide improved trading recommendations .

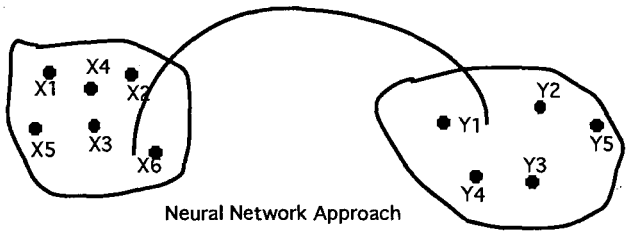
Neural networks are excellent at estimating non-linear relationships; in dealing with many different market indicators and different desired results; at detecting complex patterns. Neural nets require, however, specialized knowledge and skills. Training neural nets i.e. finding weights that incorporate the non-linear relationships remains more an "art" than a science.

Neural networks can be thought of as a method of estimating classes of nonlinear functions e.g. f: X->Y where in the case of trading systems, X are market indicators and their transformations and Y is an output signal that can be translated into a trading recommendation. As such, neural networks do nothing more than project the input space unto a lower dimensional space in which classification of patterns or nonlinear relations may be realized more effectively.

Figure 3 shows the neural net mapping of x, y pairs. In sum, a neural net based trading system requires the specification of a nonlinear dynamic system, availability of sufficient representative numerical training samples, and the encoding of these training examples in the dynamical system by repeated learning cycles.

The class of neural networks commonly used are machine learning tools which use a gradient descent algorithm to minimize the Euclidean distance (or error) between the output given the input, and the desired output.

Figure 3 : Neural Network Approach to Design of a Trading System : Value Mapping



The first step in design of a neural net trading system is to define the objectives. The objectives for design of a neural trading system can be anything from the recognition of profitable patterns, avoidance of high risk situations, achievement of a certain level of profitability with low risks to the development of a fully automated system for trading to achieve a combination of goals including a desirable trading frequency.

Common inputs for the neural net trading models are the daily closing prices or yields (potentially the open, high, low and close), *spreads* between an instrument and a benchmark, index or risk-free rate; *trend* and *volatility* indicators. Trend indicators can be computed in many ways : moving averages, slopes or regressions coefficients which provide level independent indicators. A simple trend indicator is the difference between the current price and the price 10, 20, or 40 days earlier. Volatility indicators can be standard deviations or averages of daily changes in returns.

The architecture of a neural net trading model is usually very simple. It consists, in general, of an input layer with 1 to i inputs, a hidden layer with 1 to j processing elements and an output layer with one or more outputs, all nodes being fully connected between adjacent layers.

The most common learning algorithm adopted for trading systems is backpropagation or a variant of backpropagation which uses a modified gradient decent learning algorithm to dynamically evolve a relationship between inputs and the output. For example, extended-delta-bar-delta (EDBD) which automates the adjustment of the learning and momentum rates and may or may not use weight memory.[3]

Experience from the design of numerous neural net models has demonstrated that the selection of a neural net architecture and learning algorithm is far less important than the selection and identification of appropriate samples for training a neural net. The sampling technique(s) for generating the training samples have a critical influence on the performance of the neural net trading system.

A valid approach for deciding when to stop training of a neural net is to look at the performance of the neural net on a test data set and to stop training when the performance based on the test data set decreases. It should not be based on root square error between actual and desired output(s). Valid criteria to be used are financial and not statistical criteria of performance on the test data set.

Profitability and associated risks can be measured in different ways. To judge the profitability and risks we often use the ratio of average profit over 2-3 years divided by the maximum drawdown (or the maximum of the successive losses) over the entire period. This is a fairly rigorous evaluation criteria. A detailed discussion of a framework for evaluation of neural networks is provided in [6].

All models exhibit trade-offs between average profit, maximum drawdown and frequency of trading. In addition, networks trained with trend and volatility indicators have generally larger returns than those trained on volatility or trend indicators only.

Neural models for individual securities can be combined in a portfolio. The actual weighting of each model can be done on the basis of various approaches. One approach is to weight the models so as to give equal risk to each model in the portfolio. Another approach is to look at the volatility of the instruments and take into account the covariances of price movements among different securities.

To test the robustness and stability of the neural net portfolio results the average return and the maximum drawdowns should be considered for moving windows: for example, all three, six or twelve month moving windows over 2-4 years. This provides performance

[3] We found that the EDBD algorithm is sensitive to the learning rate parameters, and tends to saturate into an unusable state if the number of hidden processing elements is too small and/or the learning rate is too high with memeory off, but too slow to converge using weight memory.

estimates for approximately 1000 overlapping periods.

Growing sophistication in the design, configuration, testing and use of neural networks is leading to better documented results and a better appreciation by traders, portfolio managers and systems people of the true capabilities of neural networks. As a result it is not surprising to find that neural networks are increasingly used for trading foreign exchange, commodity, stock and fixed income markets in many institutions around the globe.

4. Use of Genetic Agorithms

Several papers have been written on how to facilitate finding the weights of a neural nets using genetic algorithms. Genetic algorithm are good at finding solutions for problems with multiple parameters, and an objective which may be subject to one or more hard or soft constraints.

A genetic algorithm (GA) is a procedure that generates a problem solution for a multi-parameter problem by considering in parallel, say, 30-50 potential solutions and measuring the goodness-of-fit of each against the user-defined objective or fitness function.

A problem solution can be a neural net with a particular set of weights, processing elements and connections. A GA selects the best solutions from a collection of problem solutions and applies reproduction, crossover and mutation to generate a new population of problem solutions. Reproduction takes the best solutions and considers them as "parents" to generate "children"; crossover mixes the attributes of the parents; and mutation introduced attributes that may not yet be represented. The new collection of problem solutions is cycled through the same selection or "survival of the fittest process", until a solution meets particular criteria. Thus, a GA can be used to find neural net weights that optimize the user-defined performance objectives and meets user-defined constraints or risk limits.[4] GA's can also be used to find the optimum neural net architecture e.g. the optimum number of processing elements at the hidden layer, the connectivity among different levels and/or the best transformation function. GA's are also used to find the best combination of inputs.

To setup a genetic optimization problem one needs
(i) an objective or fitness function;
(ii) potential constraints to be met;
(iii) the parameters that are subject to change; and
(iv) the ranges for the adjustable parameters; potentially
(v) the number of problem solutions in each population and
(vi) the criteria for stopping the process (e.g. number of generations, time available to optimize, and/or the criteria for stopping the process based on the reduction of level of improvements in successive generations -for example, stop the process when the best solutions no longer change by more than X points between successive generations-).

The objective function for a GA used to find the weights of a neural net for trading can be any one or set of financial criteria e.g. the average return. Constraints can be the risks measured by the maximum drawdown over the entire period.

The adjustable parameters are the neural network weights and the biases. Setting the ranges of the adjustable parameters too narrow will accelerate the GA process but may constraint the GA of finding a solution in a small part of the hyper-dimensional solution space.

To monitor the progress of the GA one should follow the evolution or selection of the best solutions and the increase (or decrease) in the quality of solutions in each set of problem solutions (population). At any point in time the GA process can be paused or stopped. The best solution or the ten best solutions (networks obtained) can be retrieve and stored in a file.

In sum, genetic algorithms can be very useful for avoiding the training process of neural networks. They can be used for selection of inputs, the configuration of neural networks as well as finding the optimal set of weights or collection of best neural networks for a particular problem case. The main drawbacks of using GAs to optimize neural nets for trading is that they may produce results that are undesirable (e.g. low frequency of trading) or do not generalize properly (e.g. by reducing the variance of solution to the objective function at the expense of introducing more bias). The end product from a GA process will largely dependent on the specification of the objective function.

5. Fuzzy Trading

Neural networks can be trained to trade financial markets but have limited explanation capabilities. They are subject to performance degradations over time. GA's can facilitate training of neural nets given a specific objective function, constraints and parameter ranges; the solution GA's produce may however lack smoothness and thus provide poor generalization. In consequence, the last technique we will explore to design of trading systems is fuzzy logic and fuzzy systems design.

A simple model of a fuzzy trading system consists of one or more inputs (e.g. trend and volatility measures), a fuzzy rule base, and one or more output variables (e.g. desired trading patterns). Such a model

---

[4] An easy way to implement a GA in a spreadsheet, without programming macros or scripts such as those described in [11]) is to use of Evolver™ from Axcelis Inc. Evolver™ is an inexpensive extension to both Microsoft Excel™ (under Windows) and WingZ™ on the Macintosh. (see also [13])

requires that inputs are "fuzzified" , membership functions are created, that "fuzzy associations" between inputs and outputs are defined, i.e. fuzzy rule bases are established, and that the fuzzy outputs derived from the system are "defuzzyfied" into crisp trading recommendations.

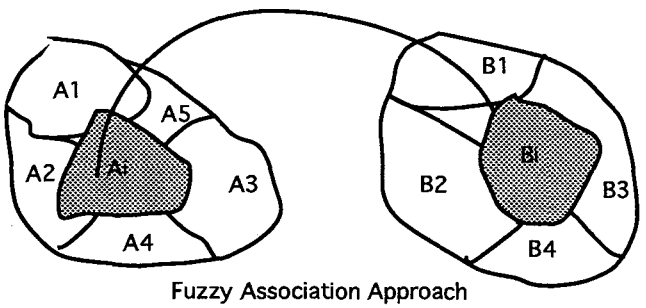
A fuzzy approach to the design of a trading system can use an adaptive clustering algorithm that drawsfuzzy associations between inputs A_i and the outputs B_j. This results in fewer different rules than input antecedents. Figure 4 shows the mapping of fuzzy set A_i into fuzzy set B_j. A comparison with figure 3 shows the main differences between neural nets and fuzzy systems. A fuzzy associative approach to the design of trading system requires only a partially fill of a rule matrix.

Fuzzy rule bases can be designed by experts or can be estimated from trading samples. A fuzzy set sample encodes a structure that can be mapped as a minimal fuzzy association of part of the output space with part of the input space. A fuzzy associative memory or FAM is a map of fuzzy sets to fuzzy sets.

The first step in design of a fuzzy trading model is converting inputs into fuzzy representations or collections of membership functions. Membership defines the degree of adherence rather than the probability of an event. In essence, continuous inputs are divided in discrete sub-ranges which overlap. Within each range the membership measures the degree of adherence to the particular segment attributes.

For example, if the trend of a security measured by a moving average is chosen as an input, we start out by finding the minimum and maximum of the trend over a historical data series. The range thus defined is called the "universe of discourse". This range is divided into subranges that are labeled, say, negative big , negative small, average, positive small, positive big. For each subrange one computes the starting, center and end value. Figure 5 illustrates the concept of a membership function.

Figure 4 : A fuzzy logic approach to Design of a Trading System: Range Mapping



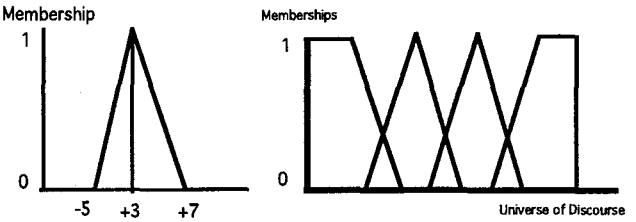
Fuzzy Association Approach

In fuzzy systems the fuzzyfied inputs are used in place of the original values as inputs. In neural networks, this technique is commonly known as a 1-of-N-code.

The K-mean clustering algorithm for defining bin sizes is not the only way in which membership functions can be defined. The main advantage of the approach outlined is simplicity; the main disadvantage it that it does not provide for different membership shapes (e.g. Gaussian, trapezoid shaped memberships or a mixture of them.)

In sum, fuzzy sets omit the need for defining precisely the change from one state to another. The fuzzification of inputs and data transformations has profound implications for design of trading models. It is at the heart of the power and flexibility of fuzzy logic. Fuzzy logic does thereby allow to write trading rules in terms of imprecise ideas of what constitutes the state or change in the variables.

Figure 5 : Membership Functions and the Universe of Discourse



The next step in the design of a fuzzy model is to create a fuzzy rule base. Fuzzy rules can be constructed using the membership functions of inputs and output variables to make connections between antecedents and consequences. Use of membership functions facilitates rule extraction and generalizations.

A rule base can be constructed by one or more experts or can be extracted automatically from sample data. Most important of all, both approaches can be combined, which provides fuzzy model design an edge over neural net-based models.

An example of a fuzzy rule is

IF the trend of x is RISING RAPIDLY AND the volatility of x is LOW
THEN GO LONG
or
IF current position is LONG
THEN increase position A LOT",
which is a more natural way of expressing a trading rule, and is less prone to changes in market regimes. Thus fuzzy rules are expressed in English with a syntax similar to mechanical rules. The general format for fuzzy rules is

IF <fuzzy proposition>
THEN <fuzzy proposition>

or in the case of multiple antecedents

IF <fuzzy proposition 1>

AND (or OR) <fuzzy proposition 2> THEN <fuzzy proposition>

where a fuzzy proposition is of the form "x is Y" or " x is not Y", x being one of the original scalar variables (e.g. the trend of a security) and Y being a fuzzy set associated with the variable e.g. negative big.

Fuzzy rule implementation implies that rule firing will depend on the fuzzy sets of the trend and volatility i.e. on the rule that ties the inputs to the output properties.

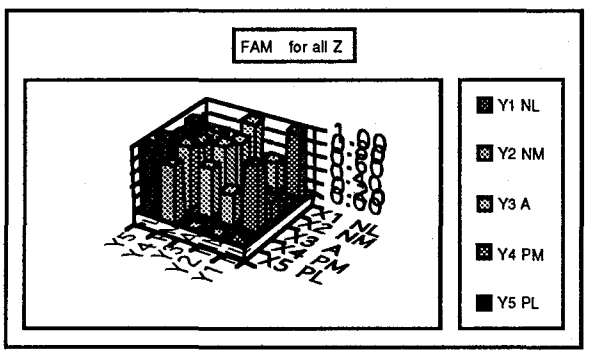
A group of fuzzy rules forms a fuzzy associative memory. When a set of input values are read, each rule that has any truth in its premise will be fired.

Generally the number of rules a system requires is related to the number of control variables. The rules or fuzzy associations represent knowledge that may be important for the system to be able to respond to all possible combinations of the inputs. The incorporation of structured knowledge into a fuzzy rule base by consultation with experts overcomes one of the main disadvantages of neural network based trading models.

Another method of obtaining fuzzy rule bases are automated extraction of rule bases from a representative data samples.

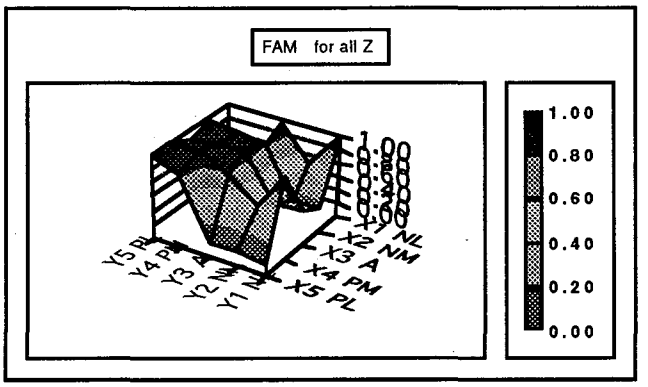
A fuzzy rule base can be extracted automatically from say historical data. The performance of such systems will depend on the initial inputs, the definition of the membership functions, as well as the rule extraction itself. B. Kosko has demonstrated how differential competitive(DCL), supervised (SCL) and unsupervised competitive (USL) neural learning can be used to extract fuzzy rule bases from training data set. On-going research in this area involve the use of genetic algorithms to optimize the performance of a fuzzy trading systems subject to minimizing fuzzy rules (complexity) and optimizing the shape of the membership functions.

Figure 6 : Graph of the composite FAM between trend, volatility and trading positions



Fuzzy associative memories can provide the fuzzy relationships between each of the inputs and the output variable. The cells in these matrices represent the membership intersections or clipped fuzzy sets between the fuzzy sets of the trend, volatility and recommended trading positions. A 3D bar representation of a composite matrix for all fuzzy sets on the trading positions is shown in Figure 6. This figure provides the membership functions (or activation levels in neural network terms) for every combination of the fuzzy sets of the trend and volatility information. A 3D surface representation of the same composite matrix or a control surface for the trading system output is shown in figure 7.

Figure 7 : Control Surface of a Trading System based on Trend and Volatility inputs



The proceedings of the LIFE (Laboratory for International Fuzzy Engineering in Japan) conferences include several examples of fuzzy control system. F.S. Wong et.al. have developed examples of fuzzy neural system for stock selectionandHideo Yuize et.al. have applied fuzzy reasoning for a decision support system for foreign exchange trading .

6. Conclusions

Neural networks and related advanced technologies provide new opportunities for automating trading, risk and portfolio management. The use of machine learning techniques, genetic optimizations, and fuzzy control in the design of trading systems, is the next step in the evolution of trading technology.

Neural nets are very good at finding relationships in a huge input-output product space even if the patterns are ill defined. The performance of a neural net trading system will depend however on what information flows into the system; how the input information is preprocessed; what the desired output is; and on the "quality" of the neural net design itself. The neural net design itself depends on the appropriateness of the neural net architecture, the learning algorithm, the adequacy of neural net training, and the relevance of the training samples. In consequence, no matter how sophisticated the neural net technology the design of a neural net based trading system remains an "art".

The art of designing neural nets, especially of training and configuring neural nets for trading, can be simplified by the use of genetic algorithms. GAs are powerful tools for finding optimal parameters for complex problems with given objective functions and one or more constraints. GAs can be used for selection of inputs, the configuration as well as finding optimal weights or the best neural nets for a particular problem. The main drawbacks of using GAs to optimize neural nets for trading, is that they may produce results that are undesirable or do not generalize properly The end product from a GA process will largely dependent on the specification of the objective function.

The drawbacks of neural-based trading systems are (i) lack of explanatory capability; (ii) difficulty to include structured knowledge; and (iii) bias towards quantitative data. In sum, the lack of confidence intervals, capability to trace back outputs, or to even include structured knowledge prevent rapid diffusion and easy acceptance of neural net technology.

A solution to this is to integrate fuzzy logic and neural networks in the design of a trading system. In the case of trading systems, several of the input variables are continuous; mathematical models of the interaction between inputs and desired trading strategies do not exist; elaborate expert systems for supporting trading decisions are too complex to be evaluated for real-time operation; high ambient noise levels in market signals must be dealt with; expert traders are available but have seldom time; and rules underlying system behavior are poorly defined. Fuzzy logic and mathematical techniques can improve the effectiveness of neural networks for the design of advanced trading systems. They can imbed structured knowledge about financial markets, have more explanatory capability, and are inherently more stable.

* * *

What impact can advanced technology have in the future ?

Neural networks are used for credit card fraud, bankruptcy predictions, bond ratings and -pricing. Several banks use neural nets for trading foreign exchange, commodity, stock and/or fixed income markets. Fuzzy systems are used for portfolio management, trading and commercial applications. Genetic algorithms are used for stock selections, asset allocationa and portfolio composition. GAs are also used in design applications, for work scheduling, and planning. Empirical findings from nonlinear dynamics and chaos theory have made the efficient market hypothesis and Capital Asset Pricing Model obsolete.

To speculate that all these won't have an impact is like dreaming about a world without cars, television, computers, or electronic mail. How much impact advanced technologies will have, depends on how the technologies are integrated and made accessible to many users. Ease of use has always had a dramatic impact on the adoption of technology.

Tools for design of neural networks, genetic algorithms, and for building fuzzy systems are at present still too difficult to use. Some are prototypes, standalone C routines or products with a poor or non-standard human interface. Most software products are single function tools (like hammers) aimed at similar functionality (e.g. training neural nets; finding nails ?).

Multi-purpose software tools that provide easy access to neural net capabilities, genetic algorithms, fuzzy logic, including automated extraction of fuzzy rule bases, have yet to be developed. Until someone does, many applications will continue to be the result of painstaking trial and error, rather than engineering.

Finally, most commercial products for applying advanced technology are too expensive in order to be adopted and widely distributed in large organizations. Under these circumstances it is unlikely that advanced technology will become as popular as Excel, WingZ or Lotus 1-2-3.

In sum, "sufficiently advanced technology will continue to look like magic" (especially to all those who have yet to discover these frontiers), however, this does not mean that all advanced technology applications should be accepted as magical solutions.

References

[1] Takashi Kimoto, Kazuo Asakawa, Morio Yoda, Masakazzu Takeoka : "Stock Market Prediction System with Modular Neural Network", Proceedings IJCNN San Diego, 1990, pp I-1-6.

[2] Gia-Shuh Jang, Feipei Lai, Bor-Wei Jiang, and Li-Hua Chien, "An Intelligent Trend Prediction and Reversal Recognition System Using Dual-Module Neural Networks, " Proc. First International Conference on Artificial Intelligence Applications on Wall Street, New York, 1991, pp. 42-51.

[3] Andrew Colin: Neural Networks and Genetic Algorithms for Exchange Rate Forecasting, IJCNN Beijing 1992

[4] Paul De Grauwe, DeWachter H., Embrechts M. : Exchange Rate Theories : Chaotic Models of Foreign Exchange Markets, (forthcoming)

[5] Lawrence Davis, Handbook of Genetic Algorithms, VanNostrand Reinhold, New York, 1991.

[6] G. Deboeck :"Preprocessing and Evaluation of Neural Nets for Trading", Adv. Technology for Developers, August 1992.

[7] Edgar Peters : Chaos and Order in the Capital markets : A new view of cycles, prices and market volatility, John Wiley & Sons, Inc, 1991; see also "A Chaotic Attractor for the S&P 500" in Financial Analysts Journal, March-April, 1991, p 55-81; and "Fractal Structure in the Capital Markets" in Financial Analysts Journal, July-August 1989, p 32-37

[8] M. Larrain : "Testing Chaos and Non-Linearities in T-Bill Rates", Financial Analysts Journal, September-October 1991, p 51-62

[9] David E. Goldberg, Genetic Algorithms: In search, optimization and machine learning, Addison-Wesley Publishing Inc., 1989.

[11] Geoffrey F. Miller et.al.:"Designing Neural Networks using Genetic Algorithms", Proceedings of the Third International Conference on GA, George Mason University, Morgan Kaufmann Pub, 1989.

[12] Bart Kasko, Neural Networks and Fuzzy Systems, Prentice Hall, 1992.

[14] Michio Sugeno et al. "Fuzzy Systems Theory and its Applications, Tokyo Institute of Technology ,1992

[13] F.S. Wong, P.Z. Wang, T.H. Goh, and B.K. Quek, "Fuzzy Neural Systems for Stock Selection," Financial Analyst Journal, Jan.-Feb. 1992, pp. 47-52.

[14] F.S. Wong, P.Z. Wang : A Fuzzy Neural Network for FOREX Forecasting, Fuzzy Engineering toward Human Friendly Systems, Proceedings of the International Fuzzy Engineering Symposium '91, November 13-15, 1991, Yokohama, Japan

[15] Hideo Yuize, et.al. : "Decision Support System for Foreign Exchange Trading : Practical Implementation", Fuzzy Engineering toward Human Friendly Systems, Proceedings of the International Fuzzy Engineering Symposium '91, November 13-15, 1991, Japan

Annex 1: Publications, Internal Documents, and Workshops

The following is a list of publications and internal documents produced by the Advanced Trading Laboratory of the World Bank, including a list of tutorials and workshops organized since 1990.

A. Publications:& Internal Documents

Fuzzy Trading : an Object Oriented Programming Approach, by D. Benachenhou, and G. Deboeck, Internal Document, World Bank, April 1993 (forthcoming).

Chaos in US Treasury Markets : Implications for Trading and Modeling : by G. Deboeck, M. Cader, M. Embrechts, Internal document, World Bank, March 1993, incl. annexes

Trading US Treasury Securities with a Portfolio of Neural Net Models, by Guido Deboeck and Masud Cader, Internal Document, World Bank, April 1993

How to build a Hybrid Trading System in a spreadsheet...in five easy steps, by Guido Deboeck, Internal Document, World Bank, December 1992

Basic Techniques of Fuzzy Model Design, by Guido Deboeck, Advanced Technology for Developers, November 1992

Design Principles for Neural and Fuzzy Trading Systems, by G. Deboeck, H. Green, G. Jang, M. Yoda, Proceedings of International Joint Conference on Neural Networks, Beijing, November 3 to 5, 1992

Implementation of a Neural Trading System, by D. Benachenhou, M. Cader, G. Deboeck, Proceedings of International Joint Conference on Neural Networks, Beijing, November 3 to 5, 1992

GenNet: Genetic Optimization of Neural Networks for Trading, by Tony & Guido Deboeck, Advanced Technology for Developers, October 1992

Nonlinear Dynamic Analysis Techniques for Preprocessing of Data for Neural Nets", by G. Deboeck, Advanced Technology for Developers, September 1992, pp 1-10

IJCNN'92 Baltimore : Back-to-Office Report, Internal Document, World Bank, June 15, 1992, pp. 7.

Nonlinear Dynamics In US Treasury Markets : Implications for Trading Strategies and Neural Network Models, by G. Deboeck, A. Polymenopoulos, M. Cader, M. Embrechts, Internal document, World Bank, June 25, 1992, pp. 31 plus annexes

Preprocessing and Evaluation of Neural Nets for Trading", by G. Deboeck, Advanced Technology for Developers, August 1992, page 1- 13.

Financial Neural Net Applications in Asia : Back-to-Office Report, Internal Document, World Bank, December 23, 1991, pp.4.

Genetic Optimization of Model-based Trading, by G. Deboeck, Internal Memo, World Bank, October 15, 1991.

Advanced Trading Systems Using Neural Nets, by B. Benachenhou, M. Cader, G. Deboeck, J. Loofbourrow, Internal document, World Bank, September 1991, pp 7.

FY91 Neural Net Research Project : Neural Net Model Results, Internal Memo, World Bank, July 1, 1991. pp. 6.

B. Workshops & Tutorials:

Applications of Neural Nets and Other Advanced Technologies in Financial Markets, Interdisciplinary Center for Neural Networks, Catholic University of Leuven, Belgium, February 17-18, 1993

Neural Networks and related advanced technologies, World Health Organization, Geneva, February 16, 1993

Tutorial : Financial Applications of Neural Networks, International Joint Conference on Neural Networks, Beijing, November 3 to 5, 1992, pp.125

Advanced Technology Seminars : (I) Financial markets, Forecasting and Advanced Analytical Techniques; (II) Design of Financial Neural Nets, Information Technology Institute, Institute of System Science, and Japan Singapore AI Center, Singapore October 29, 1992

Nonlinear Dynamic Analysis and its importance for Neural Network Design, IBC Conference : "Data Mining in Finance and Marketing", September 23-25, 1992, London, pp. 28.

From Technical Analysis to Nonlinear Dynamics : Evolutions in Trading and Investment Strategies, Society of Market Technicians, August 6, 1992, Washington DC, pp. 5.

Information Biases Financial Markets...until the bias is changed, IBC's Third Annual Conference on "Advanced Trading Technology", July 20-21, New York.

Applications of Neural Networks, Fuzzy Systems and Genetic Algorithms, Leading Edge Technology Workshop Series, World Bank, January 15 to April 1, 1992, Washington DC (10 sessions)

Financial Neural Nets : Practical Suggestions, IBC Conference on "Rocket Science Made Simple", October 9-10, 1991, London, pp. 21

Injecting New Technology into a large organization : How many smoke signals does it take ? Second Annual Conference on Expert Systems and Neural Networks in Trading, January 23-24, 1991, New York

Neural Networks and Fuzzy Logic : Applications in Finance, Advanced Analytics Seminar for Treasury Management, World Bank, March 1990, Washington (3 volumes)

Paper Session: Fixed Income and Bond Ratings

Chair: Ken Kleinberg, New Science Associates

FORECASTABILITY OF RETURNS WITH NEURAL NETWORKS: AN APPLICATION TO SPOT AND FUTURES ITALIAN BOND MARKETS

Emilio Barone[1], Andrea Beltratti[2], Sergio Margarita[3]

[1] IMI and LUISS University, Viale dell'Arte 25, Roma, Italia, fax +39 6 59592300

[2] Fondazione ENI Enrico Mattei of Milano and Istituto Prato, Università di Torino, Via della Cittadella 10/E, 10122 Torino, Italia, fax +39 11 541497

[3] Istituto di Matematica Finanziaria, Università di Torino, Piazza Arbarello 8, 10122 Torino, fax +39 11 544004

Abstract: we discuss the possibility of applying neural networks for the analysis of financial markets. We consider simple forecasting and more complicated devising of trading rules. The former can be performed with a single neural network or with a combination of neural networks, while the latter can be performed with an unsupervised learning methodology that applies genetic algorithms to neural networks. We apply these techniques to the Italian bonds spot and futures markets. We find few signs of predictability out-of-sample, with a high (more than 30%) degree of explanation in-sample.

1. INTRODUCTION

This paper has the purpose of discussing the possibility of using very general nonlinear models for forecasting purposes in financial markets, and of presenting a specific application to the Italian bond futures traded at the LIFFE and to the Italian market for spot bonds. There are three key elements in this statement: the forecasting purpose (the problem), the nonlinear models (the methodology), the particular financial instrument (the application).

The problem is obviously relevant to practitioners, and has also interested many in the academic profession, especially in the last 20 years. Scientific research has addressed the forecasting job mostly with linear models, where the future change in prices of a specific financial asset is regressed on current and past values of some information variables that are known to the market before the change in price. A very general survey of the methodology and of the results can be found in two papers by Fama (1970, 1991). Most of the results show that a large part of the variation of returns is not predictable on the basis of current and past information, apart from forecastability of multi-period (2 years to 10 years) returns, where the explained percentage of the variance can be as high as 40%. Research on the last finding is however not concluded, and it is likely that practitioners are mainly interested in predicting one-day or one-week rather than ten-year returns, so that much remains to be explored on the issue of short-run forecasts.

This moderate failure of the linear model provides a motivation for the use of more sophisticated models. There is a small (in relative terms) but growing literature on the application of specific non-linear models to explanation of price movements, see for example Priestley (1988) and Mills (1990), or on the search for general non-linearities that may be used for prediction purposes, see Scheinkman and LeBaron (1989) and LeBaron (1992). Notable in this recent literature is the use of neural networks, see for example applications of White (1988), Weigend, Huberman and Rumelhart (1992), and theoretical papers on the econometric content of neural networks collected in White (1992).

It is fair to say that this literature is not extremely successful either. LeBaron (1992) acknowledges that "Several recent papers have proceeded to test for nonlinear forecast improvements in several economic time series. All these have found almost no evidence for any improvement using standard nonparametric estimation techniques. These results could indicate that the previous tests are just picking up nonstationarities in the series tested, and there is little evidence for any nonlinearity which stays stable over time". He then goes on to show that taking volatility as an index for detecting breaks in the stock returns series it is possible to obtain "significant but extremely small"

forecast improvements. His conclusion is that "stock returns remain, as they should, a relatively difficult series to forecast". Similar conclusions are obtained with neural networks; Weigend, Huberman and Rumelhart explain no more than 5% of the variance of changes in exchange rates, while White obtains similar results for stock returns.

In this paper we pursue this line of research by exploiting the performances of neural networks and genetic algorithms. At first we use standard neural networks trained with back-propagation to forecast the change in the prices of bonds and of futures on bonds. We evaluate the effectiveness of the predictions by means of simple statistical measures like the coefficient of determination. We then change the focus of our research in two directions: we embed predictions in a more general structure of trading rules that are studied by the simultaneous use of many neural networks, whose outputs are then combined by means of a large network. Finally we combine neural networks and genetic algorithms to devise a general global maximization search. We apply these nonlinear techniques to the study of Italian bond futures traded at the London LIFFE market and of bonds prices in the Italian spot market.

The organization of the paper is the following: section 2.1 describes the forecasting use that can be made of statistical instruments, and contrasts it with the need to devise operative trading rules to take positions in the market. Section 2.2 and 2.3 then describe how neural networks and genetic algorithms may be used both for forecasting purposes and for practical decision-making. Section 3 considers the multi-expert approach. Section 4 describes the data set and the market, section 5 reports the results and section 6 concludes.

2. FORECASTS AND DECISION RULES

2.1 Forecasts

A minimal use that can be made of neural networks in the context of financial applications is pure forecasting. If one denotes with x_t a (k,1) vector of inputs at time t, and with y_t a scalar variable at time t, and if the purpose is to find a neural network function that explains the last on the basis of the former, where y'_t is the fitted value, one may then look for the optimal weights contained in the matrices A and B of the function $y'_t=f(Af(Bx_t))$, where f(Z) is the logistic function applied to each element of the vector Z.

In a typical forecasting application in financial markets x_t is a vector of variables known at the beginning of period t, and y_t is the value of a return or excess return that is known by the end of period t. One is looking for a mapping between a set of available indicators and the value of the return. The very general learning abilities of neural networks may give some hope about explaining a larger part of the total variance of y than what may be done with linear models, a particular case of the above representation where A is equal to the identity matrix and f is such that Z=f(Z). Like in the linear case, the performance varies according to whether it is judged in the training set or in the generalization set. The learning ability of neural networks actually makes this problem even worse, since an excessive adaptability to the training set may turn out to be a disadvantage for forecasting out of sample.

The interpretation of such mapping is certainly not unique, and depends on the structure of the economic model that is proposed to explain reality. An efficient market populated with risk-neutral traders with expectations that are rationally formed with the true distribution should not show any mapping between known information and future realizations, since such mapping would be a sign of inefficiency. More general models taking account of risks or heterogeneous information are instead compatible with such mapping. A financial trader should autonomously decide whether the existence of such mapping is exploitable to devise profitable risk-adjusted trading rules. This amounts to answering the following question: are the predictable components of future returns large enough to generate positive wealth when exploited in the context of well-defined trading rules that specify the actions that should be taken for any given information set?

2.2 Exogenous and endogenous decision rules

Even though the discovery of such rules is the main part of the problem, one may well say that the representation of the problem that has just been given is not very helpful for this purpose. A point forecast of a +1% in future returns does not necessarily mean that the asset should be bought. Indeed, one may say that many important practical elements are left out of this representation. The position that should be taken as a function of the forecasts is not specified; the risk is not very clear, and transaction costs are ignored.

These elements may be specified when one establishes some rules, like for example:
buy 1 unit if the forecast is larger than 0
sell 1 unit if the forecast is lower than 0
or for example:
buy k unit if the forecast is larger than 0
sell k unit if the forecast is lower than 0
where k depends on the size of the forecast. Refenes et al. (1993) for example consider a rule that suggests buying a quantity that is proportional to the expected return. One may then check the results of the application of this rule to a time series of prices. Of course the financial results of the strategy depend strongly on the rule; it may certainly

be possible that the same forecasting mechanism would have produced much better results with a different trading rule.

Any trading rule implicitly reflects characteristics of the risk-aversion of the decision-maker and the structure of the market. For example the rule "buy if the price is expected to increase and sell if the price is expected to decrease" reflects risk-neutrality and no transaction costs, as we have ourselves discussed in the context of a theoretical paper on the use of neural networks in financial modelling (Beltratti and Margarita 1992a). However risk aversion and transaction costs give rise to more complicated trading strategies that have some intertemporal elements to them. As an example consider the combination of risk-neutrality and transactions costs, a case in which the rule to buy (sell) whenever the agent expects an increase (a decrease) in the price may generate excess trading. If the expected increase from today's price is lower than today's bid-ask spread for example it is not convenient to perform a buy-and-sell operation. But the issue is even more complicated than that: with transactions costs in some cases it may be worth to buy an asset when the strategy is considered from a multi-period point of view, even if the operation does not seem convenient from a one-period point of view. Consider the following example:

	time t	time t+1	time t+2
ask	102	105	108
bid	98	101	104

In this case it is neither worth buying at time t if one sells at t+1, nor buying at t+1 to sell at t+2, but it is worth buying at t if one sells at t+2. With transaction costs the horizon of the trader may affect the optimal decision. On the other hand, the optimal horizon is itself unknown, and may be an interesting subject of empirical research for a decision-maker who is not willing to specify a priori a very detailed trading strategy, but prefers to have such strategy endogenously emerging from the data.

By training networks to make decisions on the quantities that need to be bought and sold at each date one may devise a nonlinear system that does not force the decision-maker to invent an external trading rule for the evaluation of the forecasting algorithm. Beltratti and Margarita (1992b) discuss this problem in the context of a theoretical model where many agents interact in a decentralized stock market and exploit the opportunities to trade by forecasting the change in the price and by deciding the quantity. This model of decision-making is easily implementable in the context of a small decision maker in a large market, where the price is not affected by the decision to trade and can therefore be taken as exogenous. It is however possible to further simplify the structure of the model, as we will do in the empirical application that follows, and consider a trader who makes decisions about the direction of the trade of a specified number of shares of the asset. This still leaves an interesting problem of trading strategy formation by requiring the trader to decide at each time t whether she wants to a buyer, a seller, or stay out of the market.

2.3 Learning endogenous trading rules by genetic algorithms

The problem that arises in the context of the search for good trading rules is the appropriate learning methodology. For exogenous rules of the type "buy (sell) 1 unit before an increase (decrease) in the price" one can implement supervised learning. If instead one is after a rule of the type "buy (sell) k units before an increase (decrease) in the price", where k is to be learnt by the decision support system, supervised learning is not feasible. A viable alternative is given by unsupervised learning through Genetic Algorithms (GAs). In this setting the network may decide the action, and it is not necessary to give a target for each time t to calculate an error, since the error itself is hard to define. The impossibility of giving a target for each time t however does not imply the impossibility of evaluating the merits of the trading strategies. It is still possible to evaluate the fitness of the various strategies in terms of the wealth that is accumulated by their repeated applications to a time series of prices. This amounts to applying GAs to neural networks.

In our empirical work we follow Margarita (1991) and consider a single ANN-based agent acting in the bond market by selling and buying shares at a given exogenous price. She chooses directly the action to take without attempting to forecast future prices. The aim is to find the optimal strategy to follow in order to reach a target defined in terms of an economic outcome. The strategy is implicitly encoded in the network so we have to find the structure and the value of the weights. The decision system of the trader is based on a feed-forward, three-layer, fully connected ANN. A recurrent connection from output to input is added in order to take into account the previous action (Figure 1). As an input, the network receives the following information:

- The prices of the last 5 days: $P_{t-1}, P_{t-2}, P_{t-3}, P_{t-4}, P_{t-5}$;
- The average of the prices of the last 30 days: P_{mt-30};
- The action taken in the previous period: A_{t-1}.

As to output, there is only one neuron, A_t, that encode the three feasible actions: to buy, to sell, or to wait. These actions have the same probability to be taken, according to the activation value of the output neuron:

Sell If $0.00 \leq A_t \leq 0.33$
Wait If $0.33 < A_t \leq 0.66$
Buy If $0.66 < A_t \leq 1.00$.

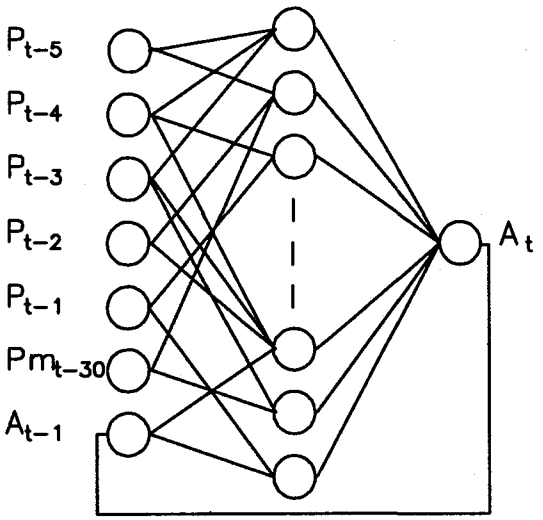


Figure 1. The structure of the trader's network

Wealth is the economic criterion chosen for valuation of trader's performance. At any time, trader owns a quantity of money and shares depending of the previous actions. Wealth is computed, in each time period, as the amount of money plus the number of owned bonds times their price. The learning technique we consider is Genetic Algorithms (GAs); although this model is a one-agent model, we have to consider what we call a virtual population, that is a fictitious population, made of non-interacting homogeneous agents, created only for the requirements of the GAs' selection process. The virtual population we create initially is made of 100 traders, all with the same structure but with different randomly generated weights. GAs use this population for finding the best performing trader.

The environment is based upon the bond price time series. In one run (called generation in analogy with GAs terminology) traders act every day by selling, buying or waiting along the whole time period. Performance of traders is measured in terms of accumulated wealth at the end of the run. Traders are subject to a set of market rules:

- Traders are independent and do not interact
- They receive every day the same public information about prices
- They trade only one share
- All the operations take place at the same price
- All the traders own initially the same number of shares and the same quantity of money
- Traders are subject to a budget constraint: they can act only if they own the money or the shares they want to trade
- Price is exogenous: actions of traders do not influence it
- Transaction costs are not considered
- Final performance of traders is evaluated adding the quantity of money and the number of owned bonds times the bond price
- Time is discrete.

For learning we use a modified form of the standard GA:

- Step 1: We generate randomly a population of 100 traders, all with the same structure
- Step 2: Traders act in the environment for a one-year long generation
- Step 3: Final wealth is computed for each network
- Step 4: The best 20 reproduce 5 times each in order to recreate initial population
- Step 5: Random mutation is applied to the new population
- Step 6: Next generation starts with the new population evolving in the same way over the same data
- Step 7: The different steps of this process (from Step 2) are repeated for 50 generations.

With respect to standard GA, we introduce some differences in our approach:

- We do not adopt a binary alphabet to represent the genotype (set of weights) of the networks and use instead floating-point real numbers. Standard GA act on binary representation
- We choose a deterministic selection technique by defining initially the percentage of networks to reproduce. Standard GA use stochastic mechanisms as roulette wheel selection
- We used only two out of three primary genetic operators: reproduction and mutation. The third, cross-over, very important in standard GA, is not implemented here
- We implement a form of real-valued mutation, rather than standard binary one. With a given probability, mutation change weights by an amount randomly distributed within a given range centred on the previous value of weight.

We are interested in following the performance of the best trader of each generation.

3. COMBINING FORECASTS AND ACTIONS: A MULTI-EXPERT APPROACH

It is conceivable that improvements in forecasting and in devising profitable trading rules increases when the outputs of many networks are combined. Before discussing such improvements we set up some notation. Suppose that J networks have been trained, and are available for forecasting on the basis of the time t information set:

$y'_{1t}=f_1(A_1 f_1(B_1 x_{1t}))$

$y'_{2t}=f_2(A_2 f_2(B_2 x_{2t}))$

...
$y'_{Jt}=f_J(A_J f_J(B_J x_{Jt}))$
where we allow for differences in the information sets, the parametrization and the optimal weights of the various networks. Such differences may be due to (combinations of) different architectures, different targets, different inputs, different training sets or training algorithms, and so on.

There is a literature on the combination of forecasts that explains why combining may be efficient; important references are Winkler (1981) and Granger (1989). In Winkler (1981) the problem is considered of combining the forecasts of an unknown variable on the part of many experts; the various forecasts can be correlated because of common information or common models. The decision maker combines the densities describing the forecasts of the various experts with a density f. Winkler shows that experts must have forecasts that are imperfectly correlated and with different means, otherwise combining is useless. How does this general idea translate in terms of neural networks?

Some possibilities are the following:

(1) one may use networks that have been trained with different parameters, like for example different learning rates and/or a different number of hidden units. Mani (1991) argues that by using portfolios of artificial neural networks it is possible to lower the variance of decisions, as long as the various forecasts are not excessively correlated.

(2) one may use networks that are trained on different data sets, maybe because the dimensionality of the complete data set would require an enormous network, and this is impossible for computational reasons or for the limited size of the data set.

(3) one may use networks that are trained with different samples, for example to distinguish between tranquil and chaotic periods, or to analyse separately periods that are separated by a structural break due for example to a change in economic policy. This possibility would seem to agree with the indications of LeBaron (1992) about the varying forecasting abilities conditional on turbulence of the time period.

(4) one may use networks that are trained to achieve different targets, for example the combination of some networks that try to forecast the short run changes in the price and others that try to forecast the longer run change in the price. This is justified by the forecast improvements that may be obtained by adaptive estimators (Tiao and Tsay 1991) in the context of misspecified models.

The problem then arises of how to combine these different forecasts. Winkler (1981) shows that, given an assumption about the density functions of the forecasts, the optimal weights depend on the covariance matrix of the forecasts, and this can be estimated and updated when more and more observations are available. In this way the networks that are more successful tend to obtain larger weights. There are at least a couple of possibilities for combining forecasts without a priori assumptions about the distributions and about the rules that should emerge from the combination:

(A) rule synthesis is the algorithm used by Trippi and DeSieno (1992); such algorithm derives from the application of Boolean operators and allows the formation of rules that result from the comparison of the forecasts of the networks at various time periods. For example the result is:

Rule:

if all the nets agree then do what is suggested by all of them

if 50% of the nets agree then do what is suggested by those that forecast an increase in price.

A variant of this technique has been used by Baxt (1992) in the quite different context of clinical patterns.

(B) combination of the forecasts of the various networks in a large network (a meta-net) that takes as inputs the outputs of the various nets, and is trained on a target that may be equal or different from the one that is used in the training of the single nets.

Both (A) and (B) may be used in a simple forecasting context or in the context of determination of trading rules, as explained in section 2. In a forecasting situation for example there may be two networks that predict the return on the basis of different sets of variables, and a meta-net that predicts the same return using as inputs the outputs of the two original networks plus some other information variable. The meta-net should be able to learn that in some cases one of the two nets is more appropriate than the other.

An example of trading rules consists of single nets that give suggestions about buying and selling and are the results of unsupervised learning through genetic algorithms. Also the meta-net can be trained with the same methodology to give suggestions about the best action.

4. THE ITALIAN BOND SPOT AND FUTURES MARKET

The Italian public bond market is the third largest in the world, after the U.S. and Japan. The two main bonds that are traded in such market are Buoni del Tesoro Poliennali (BTP) and Certificati di Credito del Tesoro (CCT). The former, roughly 40% of the market, is a fixed rate contract between 4 and 10 years, while the latter, roughly 50% of the market, is a variable rate contract between 5 and 10 years. The secondary market for the BTP is divided between a minor over-the-counter market and a more important continuous dealer-based market called

"telematico". Futures contract have recently been introduced on the BTP. Since its introduction in September 1991, the Italian Treasury Bond futures contract of the London International Financial Futures Exchange (LIFFE) has established itself as a very liquid contract. On average, almost 15,000 contracts are daily traded, with a maximum open interest equal to over 30,000 contracts. Futures contracts are written on a notional bond with a face value of 200 million lire (about 130,000 dollars), payable at maturity, and an annual interest rate equal to 12%, payable every 6-month. Any BTP with a maturity between 8 and 10 years can be delivered, if the outstanding amount of the issue is not lower than 4,000 billions lire. Therefore, the holder of a short position has a quality option which permits her to select any bundle of deliverable bonds from up to 6 deliverable issues (Tables 1-2).

Table 1

DELIVERABLE ISSUES

Bonds	Issue dates and re-openings				Amounts issued (blnLire)			
BTP 3/1/2001 12.5%	5-Mar-91	2-Apr-91	2-May-91		3,000	3,000	4,000	
BTP 6/1/2001 12%	3-Jun-91	1-Jul-91	2-Aug-91		2,500	2,000	2,000	
BTP 9/1/2001 12%	4-Sep-91	1-Oct-91	4-Nov-91	3-Dec-91	2,000	3,000	4,000	1,500
BTP 1/1/2002 12%	7-Jan-92	4-Feb-92	3-Mar-92	2-Apr-92	3,500	4,000	3,000	5,000
BTP 5/1/2002 12%	4-May-92	2-Jun-92	3-Jul-92		5,000	3,000	2,000	
BTP 9/1/2002 12%	1-Sep-92	7-Oct-92	5-Nov-92		1,500	2,962	1,500	
BTP 1/1/2003 12%	8-Jan-93	4-Feb-93			2,000	2,000		

Table 2

DELIVERABLE ISSUES PER CONTRACT

Delivery month	Codes					
Dec 91	12677	12679	12683	-	-	-
Mar 92	12677	12679	12683	12687	-	-
Jun 92	12677	12679	12683	12687	36605	-
Sep 92	12677	12679	12683	12687	36605	-
Dec 92	12677	12679	12683	12687	36605	36614
Mar 93	12679	12683	12687	36605	36614	36623
Jun 93	12683	12687	36605	36614	36623	-
Sep 93	12687	36605	36614	36623	-	-
Dec 93	12687	36605	36614	36623	-	-
Mar 94	36605	36614	36623	-	-	-
Jun 94	36614	36623	-	-	-	-
Sep 94	36623	-	-	-	-	-
Dec 94	36623	-	-	-	-	-

The price received by the party with the short position is adjusted according to the particular bond delivered on the basis of a conversion factor system. The conversion factor is equal to the value of the bond on the delivery date on the assumption that the interest rate for all maturities equal 12% per annum, with semiannual compounding (Table 3).

Futures are traded for four delivery months: March, June, September and December. The delivery date is on the tenth day of the delivery month. The last trading day is the fourth working day before the delivery date.

An initial margin, equal to a small percentage of

the notional value (0.008-0.03%), must be deposited at the time the contract is first entered into. At the end of each trading day, the margin account is adjusted to reflect the investor's gain or loss (marking to market). The futures contract is quoted in percentage terms, to two decimal places (i.e. to the nearest basis point). Therefore, the minimum price movement that can occur in trading is equal to 1 basis point (20,000 lire). The futures prices are fixed in an open outcry auction, conducted in trading pits where traders announce their proposals to buy or sell at bid and ask prices, respectively. Reporters, who are employees of the exchange, record the times and prices of bids, offers and actual trades. Trading starts at 8 a.m. (Greenwich Meridian Time - GMT) and ends at 4.05 p.m. For each contract month there is an opening call which establishes an opening range (initial bid and ask prices) and a closing call which establishes a closing range (closing bid and ask prices). The settlement price, which is a representative price from the closing range, is used to determine margin requirements.

For the purposes of this paper, the following information, in the period from September 26th, 1991 to November 30th, 1992 (297 working days), has been collected: the opening bid, ask and transaction prices; the spread, defined as the difference between the highest and the lowest price achieved during the day; the closing bid and ask prices; the settlement price and the trading volume.

Table 3

CONVERSION FACTORS

Delivery month	Codes like in table 2					
Dec 91	1,0267	0,99938	0,99928	-	-	-
Mar 92	1,0261	0,99895	0,99906	0,99911	-	-
Jun 92	1,0255	0,99938	0,99895	0,99958	0,99904	-
Sep 92	1,0262	0,99928	1	0,99911	0,99921	-
Dec 92	1,0253	0,9997	0,99961	0,99925	0,99968	0,99961
Mar 93	0,99895	0,99906	0,99911	0,99921	0,99906	0,99911
Jun 93	0,99895	0,99958	0,99904	0,99895	0,99958	-
Sep 93	0,99911	0,99921	1	0,99911	-	-
Dec 93	0,99925	0,99968	0,99961	0,99925	-	-
Mar 94	0,99921	0,99906	0,99911	-	-	-
Jun 94	0,99895	0,99958	-	-	-	-
Sep 94	0,99909	-	-	-	-	-
Dec 94	0,99925	-	-	-	-	-

5. Results

In our empirical work we have considered daily data on spot and futures markets. We have carried out the following tests:

(1) we have trained neural networks to forecast the change in futures prices on the basis of various information sets. Within this first experiment we define four different input-output information sets:

Experiment 1.A
Input: open at t
volume during t-1
CIR price at time t-1
spread at t-1
Output: change between the close price of day t and the open of day t

Experiment 1.B
Input: open at t
volume during t-1
CIR price at time t-1
spread at t-1
open at t-1
volume during t-2
CIR price at time t-2
spread at t-2
Output: same as 1.A

Experiment 1.C
Input: five lagged values of the change between open and close (from t-1 to t-5)
Output: same as 1.A

Experiment 1.D
Input: five lagged values of the change between open and close (from t-1 to t-5)
Output: change between the close price of day t+4 and the open of day t

In all the cases the results, measured in terms of variance of the change in futures prices that is explained by the network, are poor. The coefficient of determination ranges from 0.005 (experiment 1.D) to 0.012 (experiment 1.B).

(2) next we try to predict the change in bond prices by using a meta-network of three ANNs which process different lagged variables. The meta-net processes the three forecasts for outputing its own forecast. In this case there is some improvement in-sample but no improvement out-of-sample:

	Coefficient of determination	
	In-sample	Out-of-sample
Net 1	0.256	0.019
Net 2	0.272	0.020
Net 3	0.257	0.017
Meta-net	0.321	0.017

(3) we have trained on daily bond prices over one year a genetic algorithm as described in Section 2.3. First experiments show a 5% increase of trader's wealth in-sample, without considering transaction costs. These seemingly poor results must be explored more deeply because they are likely to be dependent on the structure of the network (number of hidden neurons). For achieving better results, a fine tuning of the genetic algorithm parameters and of the neural network structure is required.

6. Conclusions

In this paper we discuss the possibility of applying neural networks for analysing financial markets and specifically Italian bonds spot and futures markets. The experiments we performed show a high degree of explanation in-sample but only few signs of predictability out-of-sample. We think that the generalization ability of neural networks can be improved by using different approaches (non-forecasting neural networks) and different learning methodologies (genetic algorithms and meta-nets).
A multi-expert model is under development, which is based on deciding networks e.g. a set networks which do not attempt to forecast prices or change in prices but directly take a decision and a meta-net which processes these outputs to take the final decision.

Acknowledgements

We thank Daniele Chiarella, Marcello Esposito and an anonymous referee for helpful suggestions. Any errors are our own responsibility.

References

Baxt W.G. 1992, Improving the accuracy of an artificial neural network using multiple differently trained networks, Neural Computation, 4, 772-780

Beltratti A. and S. Margarita 1992a, An artificial adaptive speculative stock market, Working paper n. 22, LUISS University, Rome

Beltratti A. and S. Margarita 1992b, Stock prices and volume in an artificial adaptive stock market, to be presented at IWANN'93 International workshop on artificial neural networks, Sitges, Spain

Duffie D. 1989, Futures markets, Prentice-Hall, Englewood Cliffs, NJ

Fama E.F. 1970, Efficient capital markets: A review of theory and empirical work, Journal of Finance, 25, 383-417

Fama E.F. 1991, Efficient capital markets: II, Journal of Finance, 66, 1575-1617

Granger C.W.J. 1989, Combining forecasts-twenty years later, Journal of Forecasting, 8. 167-173

LeBaron B. 1992, Nonlinear forecasts for the S&P stock index, in Nonlinear modeling and forecasting, SFI Studies in the Sciences of Complexity, vol. 12, M. Casdagli and E. Eubank (Eds.), Addison-Wesley

Mani G. 1991, Lowering variance of decisions by using artificial neural network portfolios, Neural Computation, 3, 484-486

Margarita S. 1991, Neural networks, genetic algorithms and stock trading, in Kohonen T., Mäkisara K., Simula O., Kangas J. (Eds.), Artificial Neural Networks, Amsterdam, North-Holland.

Mills T.C. 1990, Nonlinear time series models in economics, Journal of Economic Surveys, 5, 215-241

Priestley M.B. 1988, Non-linear and non-stationary time series analysis, Academic Press, London

Refenes A.N., M. Azena-Barac, L. Chen and S.A. Karoussos 1993, Currency exchange rate prediction and neural network design strategies, Neural computing and applications, 1, 46-58

Scheinkman J.A. and B. LeBaron 1989, Nonlinear dynamics and stock returns, Journal of Business, 62, 311-337

Tiao G.C. and R.S. Tsay 1991, Some advances in nonlinear and adaptive modeling in time series analysis, Technical Report 118, Graduate School of Business, University of Chicago

Trippi R.R. and D. DeSieno 1992, Trading equity index futures with a neural network, Journal of Portfolio management, 27-33

Weigend A.S., B.A. Huberman and D.E. Rumelhart 1992, in Nonlinear modeling and forecasting, SFI Studies in the Sciences of Complexity, vol. 12, M. Casdagli and E. Eubank (Eds.), Addison-Wesley

White H. 1988, Economic prediction using neural networks: the case of IBM daily stock returns, Technical paper, University of California, San Diego

White H. 1992, Artificial neural networks, Blackwell, Cambridge, MA

Winkler R.L. 1981, Combining probability distributions from dependent information sources, management science, 27, 479-488

A "Neural" Decision Support System for Predicting Currency Exchange Rates

Diethelm Würtz [a], *Claas de Groot* [a],
Dieter Wenger [b], *Sigrid Unseld* [b], and *Bruno Schütterle* [c]

[a]Interdisziplinäres Projektzentrum für Supercomputing, ETH-Zentrum
and Institut für Theoretische Physik, ETH-Hönggerberg
CH-8092 Zürich, Switzerland

[b]Swiss Bank Corporation, Hochstrasse 16, CH-4002 Basel, Switzerland
[c]Swiss Bank Corporation, Bärengasse 16, CH-8010 Zürich, Switzerland

(E-mail: wuertz@ips.id.ethz.ch)

Abstract

We report on a research project for the analysis and forecasting of currency exchange rates. The heart of the system is based on concepts from the field of neural information processing. These sophisticated methods are able to analyse signals from economic systems, whose behavior is highly disturbed by noise or hidden by other signals. The research done in this field has extended the scope of system analysis, diagnosis and forecasts to include nonlinear phenomena which previously could not have been modeled. The research prototype shows on historical data during the last years (e.g. in the case of the IMM SFR/USD contract) up/down predictions with an accuracy of about 70% and allows to achieve annual returns of about 20%.

1. Introduction

New executive information systems for forecasting and optimization are mathematical and computational challenges. Numerically intensive computing is coming nowadays to the aid in many fields of financial services. New modeling, forecasting and decision supporting systems will become more and more invaluable tools in finding the best solutions to finance problems. The range and variety of the underlying information processing technologies goes far beyond traditional statistical techniques, such as regression and discrimination analysis, and decision support systems. The new aids to making better business decisions are based on techniques as diverse as artificial intelligence, connectionist networks and nature's algorithms like evolution strategies, Langevin equation methods and simulated annealing.

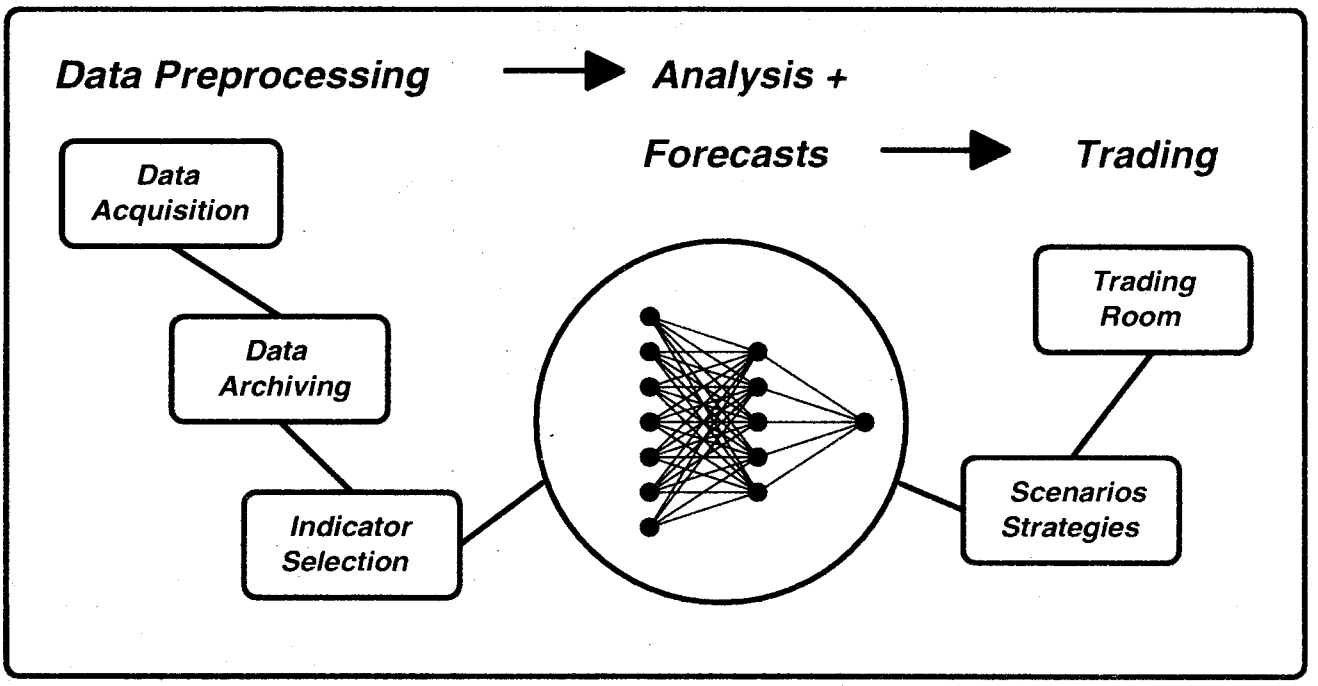
Recent developments in computer hardware and software as well as the evolution of the financial market have created a favourable synergy. Advanced computational technologies and techniques and on the other hand, with the globalization and de-regulation of the market, new opportunities and more competition go hand in hand. Thus interactively generated real time solutions are very important in today's volatile financial markets.

ZIP - Zurich Information Processing system is a collection of state of the art programs based on new information processing technologies. In this paper we present the connectionist network methodology implemented in *ZIP* and show as an application the forecasting of currency exchange rates.

2. Connectionist Network Methodology

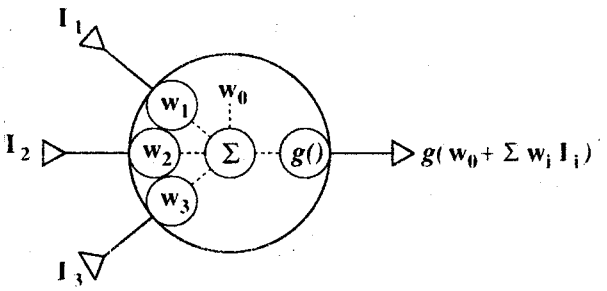
Connectionist networks of the feedforward type have recently been shown to be universal function approximators [1]. The simplest structures for which these theorems hold are networks with a single hidden layer and nonlinear (sigmoid) transfer functions in this hidden layer only. This theoretical result is accompanied by many numerical investigations which showed experimentally the capabilities of relatively simple connectionist networks to approximate nonlinear mappings. These findings encourage the application of connectionist networks in the field of (nonlinear) time series analysis.

We follow this relatively new and very promising concept of nonlinear function approximation using connectionist networks. Combining the techniques of statistical time series analysis [2] with the ideas inherent in the concepts of connectionist networks we build a feedforward connectionist net approach for modeling and forecasting time series [3]. The connectionist methodology implemented in *ZIP* includes all stages of the traditional



Block diagram showing the different modules of the research prototype system for the ananlysis and forecasting of currency exchange rates.

time series analysis like *(i)* data preprocessing, *(ii)* model identification and selection, *(iii)* model building and estimation and *(iv)* methods for diagnosis checks on its adequacy.



Furthermore the *ZIP* connectionist methodology includes a proper selection of indicators obtained from data preprocessing, a parsimonious network topology with an optimal selected starting solution and finally a monitoring scheme for the learning process giving us characteristic information on the internal structure and parameters of the network [4]. Since the approach is a nonlinear function mapping we are able to find different nonlinear network solutions with almost the same performance. From a directed search for those networks which are highly decorrelated with respect to their residuals we can construct in general better forecasts compared to the individual network solutions. Those networks which are "antithetic" in their construction [5] yield much more stable predictors and allow to specify error bounds which are an essential step forward towards a confident network design

How does a connectionist network work? A connectionist network describes a mapping from an input vector (e.g. different market indicators) to an output vector (the predictant, i.e. the market we want to predict). This mapping is determined by the network structure, usually including a hidden layer. The inputs are forced by a non-linear functional relationship to the nodes of the hidden layer and then transfered linearilly to the output. The connections from node to node are determined by a "learning" process, such that for a given historical data set the output matches the learning examples as good as possible. The same network is used with actual input information to predict future market situations.

Nonparametric randomness and dependencies tests					
Test method	significance levels for year				
	1985	1986	1987	1988	1989
Kendall's τ test	0.81	0.91	0.12	0.36	0.07
Spearman's rank test	0.78	0.93	0.15	0.41	0.07
Wald-Wolfowitz runs test	0.46	0.66	0.02	0.70	0.51
Mean squared differences test	0.50	0.28	0.55	0.79	0.02

Statistics of the IMM Swiss Franc exchange rate "tomorrow opening" - "today opening" for the years 1985 - 1989. Null hypothesis for all tests: The sequence is generated by a random sequence. If the "significance level" is below 0.05 then the hypothesis is rejected on a 5% level.

3. Software Design

ZIP is much more than a traditional software package *ZIP* contains on the lowest level a library of high quality FORTRAN and C programs implementing state of the art algorithms. On the next level we use a shell script oriented command language which allows to program simple scripts to perform special tasks calling the library routines. On a third level graphical displays are running under the X11 environment to visualize the obtained results. The whole system is running under the UNIX operating system and makes use of the client-server principle. *ZIP* uses the integrated environment of powerful SUN workstations connected to the Cray Y-MP supercomputer at ETHZ and to a locally used parallel computer parsytec GC-1. *ZIP* is also partially implemented on Convex and Silicon Graphics Computers.

#	Time Series:
1	Pound Sterling
2	German Mark
3	Swiss Franc
4	Japanese Yen
5	US Dollar Index
6	IMM Pound Sterling
7	IMM German Mark
8	IMM Swiss Franc
9	IMM Japanese Yen
10	FINEX US Dollar Index
11	IMM 3 Months Euro Dollar
12	SIMEX 3 Months Euro Dollar
13	LIFFE 3 Months Euro Dollar
14	LIFFE 3 Months Sterling
15	IMM 3 Months T-Bills
16	CBT T-Bond 8% FUT
17	LIFFE Long Gilt
18	COMEX Gold FUTURE
19	COMEX Silver FUTURE
20	NYMEX Crude Oil
21	Euro SFR 3 Months
22	London Gold 50

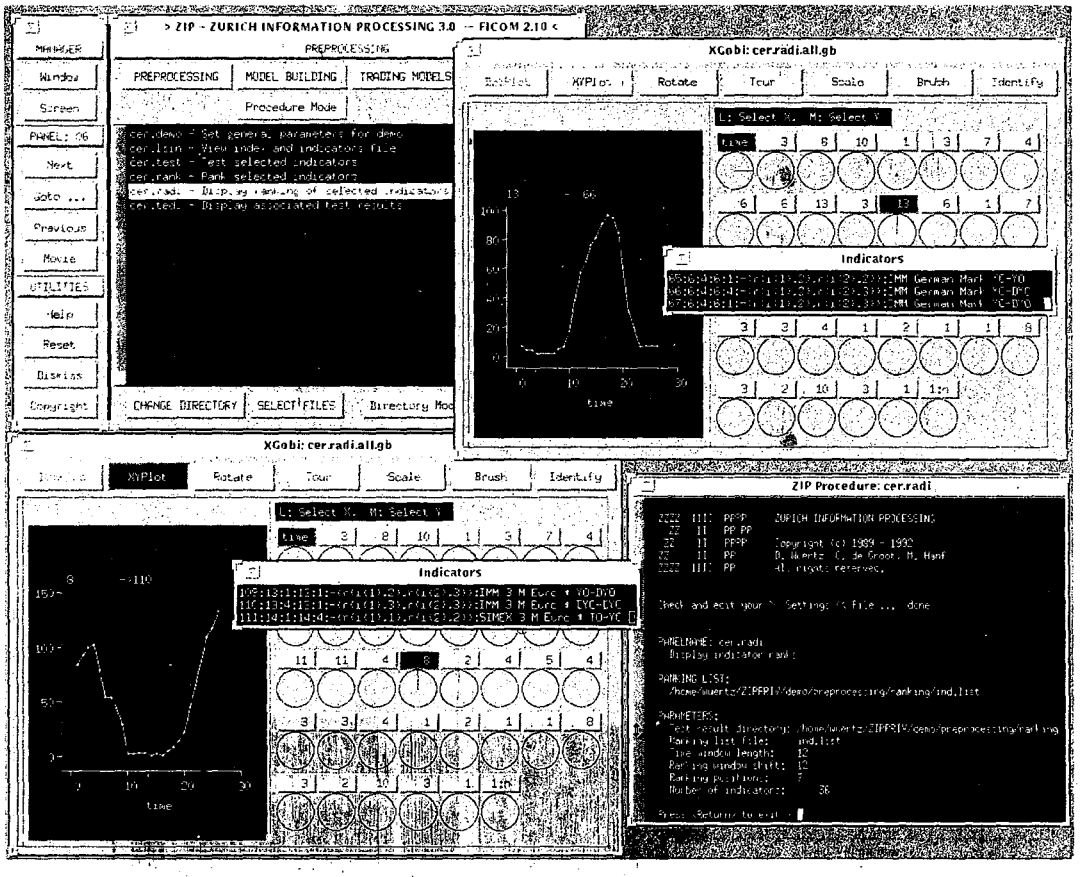
Since its first release a number of problems have been solved with *ZIP* ranging from statistical applications, via signal and image processing tasks to applications in banking and economics [5,6]. In this contribution we select one of the examples and present our approach to the forecasting problem of the *IMM Swiss Franc*: the currency exchange rate between the US Dollar and the Swiss Franc.

4. Search for Significant Timestable Indicators

In a first step we applied several (non-parametric) statistical tests to the time series in order to check for randomness. The vast majority of tests indicated that an univariate approach were useless, since the data represent *random walks*. Thus we reproduced this well-known result for our time series data

However, for an accurate analysis and prediction of changes in currency exchange rates the knowledge of market significant indicators plays an important role. But this is not enough, the indicators must also be timestable during the forecasting time horizon to be considered. This fact may be formulated as a statistical hypothesis in a multivariate context: There may exist dependencies between the time series such that some variables may be identified as indicators. We therefore call a value an "indicator", if there is a statistically significant correlation to its "predictant". The first question is, how to identify these indicators. We would like to comment that we do not share the opinion that the connectionist net itself is able to extract these dependencies

Overview of the time series data for exchange rate analysis. The first four series are interbank rates, the others give standard series from various sources. If there is no special tag, all time series are related to the US Dollar. In most cases daily opening closing, high and low were available.



Panel of the research prototype system which displays the ranking of timestable indicators. The plots in the figure depict the stability of two typical indicators over a period of 26 months. Whereas the indicator in the upper right hand plot is stable in the beginning and at the end of the time period considered, the indicator in the lower left hand plot shows the opposite behavior: a stable relationship between this indicator and the predictant is observed in the middle of the time period. Both graphics underline the importance of using stable indicators in predicting up and downward trends in currency exchange rates.

by some kind of built-in "intelligency". Instead, in our experience on the matter, connectionist nets, as used today, are quite unstable and sensitive to noise, some crucial issues in this highly noisy application. We therefore propose rigorous data preprocessing, leaving the net the "only" task of performing the time series analysis.

In order to find indicators we performed several statistical tests on dependencies between the time series. E.g. we checked the difference of interest rates against the *IMM Swiss Franc* time series. Among these tests were the Wilcoxon two sample test, Kendall's τ-test, Spearman's ρ-test. All these tests give siginificance levels for the dependencies. We devised a scheme to combine all these levels in a non-parametric way. Finally, we invented a method to adjust the indicators dynamically with the aim to select stable indicators for a given time horizon.

5. Analysis, Prediction and Trading

After the selection of indicators we first developed a *linear model* and performed some trading on that part of the data which we did not include in the search for indicators and parameter estimation process. We will briefly describe the trading scheme. The prediction procedure will tell, how much the exchange rate will raise or fall. If the prediction is "raise" we buy for the amount of one Swiss franc today and sell the dollars we get on the basis of the rate tomorrow. If the system predicts "fall" we sell the dollars today and calculate the return on the basis of the rate of tomorrow. This scheme is very simple, but it has its counterpart in reality. Note that it is possible to make money every single day. The results of our linear method were 13±4%. This figure is an average over three years of trading, and it is based on the maximal possible return. If we compute our return on the basis of the capital involved we get an averaged value of 20%.

Having performed the linear prediction we turned to the question whether a nonlinear approach allows even larger returns. For this task we chose optimal designed *feedforward networks* from *ZIP* based on various time horizons of the indicators and different degrees of nonlinearity of the starting solution. We select the best solutions and use them as possible scenarios or as independent "experts". After forecasting trading with different simple strategies is our next step. Within our approach we realize total returns beyond 20%. The conclusion from our investigation may be that the joint effort of nonlinear connectionist nets will lead to a higher return than a linear approach. The return is also more stable than in the linear case. Trading details are summarized in the table

6. References

[1] G. Cybenko, *Approximation by Superpositions of a sigmoidal function*, K.-I. Funahashi, Neural Networks 2, 183, 1989; *On the Approximate Realization of Continuous Mappings by Neural Networks*, Neural Networks 2, 189, 1989; K. Hornik, M. Stinchcombe and H. White, *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks 2, 359, 1989

[2] C.W.J. Granger and P. Newbold, *Forecasting Economic Time Series*, Academic Press, New-York 1986; M. B. Priestley, *Non-linear and Non-stationary Time Series Analysis*, Academic Press, London 1988; H. Tong, *Non-linear Time Series*, University Press, Oxford 1990

[3] C. de Groot, *Nonlinear Time Series Analysis with Connectionist Networks*, ETH Dissertation No. 10038, Zürich 1993

[4] C. de Groot and D. Würtz, *Non-linear time series analysis with connectionist nets: towards a robust methodology*, IPS Research Report 91-22, to appear in: Proc. of SPIE Conf. on "Applications of Artificial Neural Networks", Orlando 1992

[5] E. Pelikan, C. de Groot and D. Würtz, *Power Consumption in West-Bohemia: Improved Forecasts with Decorrelating Connectionist Networks*, Neural Network World 2, 701, 1992

[6] C. de Groot and D. Würtz, *Analysis of Univariate Time Series with Connectionist Nets: A Case Study of Two Classical Examples*, Neurocomputing 3, 177, 1991; C. de Groot and D. Würtz, *Signal Processing and Noise Reduction with Connectionist Networks*, Helvetica Physica Acta 64, 948, 1991; C. de Groot and D. Würtz, *Forecasting Time Series with Connectionist Nets: Applications in Statistics, Signal Processing and Economics*, Lecture Notes in Artificial Intelligence 604, F. Belli and J. Radermacher Eds., p. 461, Springer Verlag, Heidelberg 1992; D. Würtz and C. de Groot, *ZIP - Zurich Information Processing: Predicting Currency Exchange Rates Using Antithetic Connectionist Function Approximators*, Neural Network World 2, 847, 1992

Feedforward Neural Network Models as Universal Approximators: Some Methodological Considerations and an Application to One-Year Ahead GNP-Prediction

Pieter W. Otter and Arjen Jongma
Faculty of Management and Organization
and the Department of Econometrics
P.O. Box 800, 9700 AV Groningen
the Netherlands

March 17, 1993

Key-words:
Feedforward neural network models, stochastic approximation, prediction.

Abstract
Following White [9] feedforward neural network models will be considered as non-linear regression models for which optimal (non-linear) estimation methods exist. As special case of stochastic approximation, the Newton-Raphson method will be discussed and compared with backpropagation. A one- and two-layer neural model will be used to generate one-year ahead predictions of the GNP of 14 OECD countries. The prediction results will be compared with two more traditional forecasting methods.

1 Introduction

Economists face a complex, non-linear shifting world. Traditional regression models used by them can be seen as (non) linear approximations of modelling the economic world. A weak point of regression models is the specification (functional form) which has to be chosen a-priori based on theory and practice with the danger of misspecification. It is interesting to consider another class of input-output models, namely the class of feedforward neural network models, in the sequel denoted by neural models. According to Hornik et.al. [2] neural models can be seen as universal approximators of (non) linear functions. Therefore neural models might serve as a flexible tool to approximate complex economic systems avoiding the risk of misspecification. By considering neural models as a class of non-linear regression models with a flexible functional specification, the parameter estimation methodology of regression can be applied to train the neural model. The paper can be seen as an attempt to combine the functional flexibility of neural modelling with the estimation (training) methodology of statistics and econometrics.
In section two neural models are considered from a statistical perspective and discussed together with a learning method based on Newton-Raphson recursion. In section three a one-class and two-layer neural model with a Newton-Raphson learning algorithm is used to generate one-year ahead predictions of national growth rates of 14 countries with a training set of 33 yearly observations of a stock-price index, inflation index, money-supply and a stock indication. By the root-mean-squared-error (RMSE) criterion the prediction results are compared with two more traditional forecasting methods based on time-series modelling and statistics.

2 Feedforward neural models in statistical perspective

Consider an input random vector X with dimension $R+1$ and an output vector Y with dimension p. Let there be n observations or realizations of X and Y, $x_1 \dots x_n$ collected in the vector $z^n = (z_1 \dots z_n)$, the training sample where $z_t = (x_t' y_t')'$, $t = 1 \dots n$ can be seen as a realization of the random vector $Z = (X'Y')'$. Suppose an "environmental" probability law μ for the input vector, the probability measure with $\mu(\mathbb{R}^{R+1}) = 1$.

It assigns to the relevant subset of $\mathbb{R}^{R+1}$ a number between zero and one representing the relative frequency with which a realization x_t is observed to belong to that subset.

Let there be a probabilistic relation between the observations x_t and y_t expressed in the probability law ν for $Z = (X'Y')'$, that is, the relative occurrence of the realization z_t of Z. It is assumed that the probability law remains the same for all realizations, which is important when discussing time-series forecasting in section three. The probabilistic relation between Y and X is given by the conditional probability law $f(\cdot|x) \equiv f(A|x) = P(Y \in A|x = x)$ with $A \subset \mathbb{R}^p$, that is, the probability that the realization of Y belongs to A, given the realization x of X. The knowledge of $(\cdot|x)$ is fundamental. Let the conditional mean $\mathcal{E}\{Y|X = x\} = k(x)$ be the value of Y realized on the average for $X = x$. Let the approximation of the conditional mean be $f(x, w)$ where $w \in W \subset \mathbb{R}^s$ is the s-dimensional parameter vector of vector of weights. In the paper the following two neural models are used as approximations for the conditional mean namely the one-layer neural model $f_1(X, w) = F(X'\beta)$ with random vector X and parameter vector $w = \beta = (\beta_0 \ldots \beta_R)'$ and where $F(\cdot)$ is the sigmoid function, that is $F(\lambda) = 1/(1 + e^{-\lambda})$. The two-layer neural model is given by

$$f_2(X, w) = F(\sum_{j=1}^{m} g(X'\gamma_j)\beta_j)$$

where F and g are sigmoid functions with m hidden neurons and with parameter vector $w = (\beta'\gamma_1' \ldots \gamma_m')'$. Let the neural model performance be measured by the performance measure $\pi : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$ with $\pi(y, f(x, w))$. The neural model performs well on the average if the unconditional expectation of $\pi(Y, f(X, w))$, that is,

$$\begin{aligned} \lambda(w) &= \int \pi(y, f(x, w)\nu(dy, dx) = \\ &= \mathcal{E}\{\pi(Y, f(X, w))\} \end{aligned}$$

is small. Sensible threefore is to minimize $\lambda(w)$ with respect to w. Choosing $\pi(Y, f(X, w)) = (Y - f(X, w))^2/2$ with dimension of Y equal to $p = 1$ we have that $\lambda(w)$ is mimimum if $\mathcal{E}\{\nabla\pi(Y, f(X, w)) = 0\}$ where $\nabla\pi(y_t, f(x_t, w^*))$ is the gradient of $\pi(y_t, f(x_t, w))$ with respect to w evaluated at $w = w^*$ and realization $z_t = (x_t'y_t)'$. Replacing the expectation operator by the mean value for the sample we have to solve the equation

$$L(w) = \frac{1}{n}\sum_{t=1}^{n} \nabla\pi(y_t, f(x_t, w)) = 0$$

for w.

The general form of gradient recursion to solve the equation is $w_j = w_{j-1} - \eta_j P_j L(w_{j-1})$, $j = 1, 2, \ldots$ with starting value w_o and learning rate η_j. When $P_j = I$ for all j we have the steepest descent method and for P_j the inverse of the Hessian, that is

$$P_j = \left[\frac{\delta^2\pi}{\delta w \delta w'}\bigg|_{w=w_{j-1}}\right]^{-1}$$

the Newton-Raphson method. Backpropagation can be seen as a special case of steepest descent by estimating (learning) the vector of weights w one observation a time. An improvement of backpropagation would be to use *all* training observations in one-shot and minimize the *average* of the performance function π, which is done in section three. Because backpropagation, steepest descent and Newton-Raphson are special cases of stochastic approximation the statistic properties such as convergence applies to the methods discussed, see for instance Kushner and Clark [3].

3 Forecasting using one- and two-layer neural models

In Zellner and Hong [10] one-year ahead prediction are generated for the growth rates of Gross National Product (GNP) for 18 OECD-countries using linear regression models with different specifications, whereas in Otter [4] the same Zellner data were used to generate also one-year ahead predictions for 14 countries using a different statistical model based on canonical correlations. Aim is to generate one-year ahead predictions of growth rates for 14 countries using one- and two-layer neural networks with the same input variables and with a Newton-Raphson training algorithm and to compare the prediction results with Zellner and Otter by using the root-mean-squared-error (RMSE) criterion. The following variables are used: $O_{i,t}$ = Gross National Product (GNP), $SP_{i,t}$ = stock-index (1980 = 100), $P_{i,t}$ = general price index (1980 = 100), and $M_{i,t}$ = nominal money supply with $i = 1 \ldots 14$ the country index and $t = 1954 \ldots 1984$ the time index. The non-stationary time series were collected from the 1985 August issue of the International Financial Statistics. The non-stationary series were transformed into the following stationary series:

$g_{i,t} = \ln(O_{i,t}/O_{i,t-1})$, $GM_{it} = (1 - L)\ln(M_{i,t}/P_{i,t})$, $SR_{i,t} = (1 - L)\ln(SP_{i,t}/P_{i,t})$, $WR_t = ME(SR_{i,t})$, where ln is the natural logarithm, L the lag-operator

and $ME(x_i)$ the median of the series $\{x_i\}$.
The data were transformed as $x' = (x-a)|(b-a)$ with $x \in [a,b]$.
Zellner used neural model specifications under which the models D and G and three naive models A, B and C as bench marks for the predictions. One- and two-layer networks were used with the same input as in the models D and G where model D is $AR(3)$-model, that is

$$D. \quad g_{i,t} = \alpha_{1i} g_{i,t-1} + \alpha_{2i} g_{i,t-2} + \alpha_{3i} g_{i,t-3} + \epsilon_{i,t}$$

and model G is and $AR(3)$Li model

$$\begin{aligned} G. \quad g_{i,t} = \alpha_{0i} \quad &+ \quad \alpha_{1i} g_{i,t-1} + \alpha_{2i} g_{i,t-2} + \alpha_{3i} g_{i,t-3} + \\ &+ \quad \beta_{1i} SR_{i,t-1} + \beta_{2i} SR_{i,t-2} \\ &+ \quad \gamma_{1i} GM_{1,t-1} \\ &+ \quad \delta_{1i} WR_{t-1} + \epsilon_{i,t}, \end{aligned}$$

with $\epsilon_{i,t} - NID\ (0, \delta^2_{\epsilon i})$.
The three naive models are respectively

$$A : \hat{g}_t = 0\ ,\ B : \hat{g}_t = g_{t-1} \text{ and}$$
$$C : \hat{g}_t = \bar{g}_{t-1} = \frac{1}{t-1954} \sum_{j=1954}^{t-1} g_j$$

where $\hat{g}_t$ is the prediction for the growth rate. The data from 1954-1973 were used to train the neural model and the period 1974-1984 was used as prediction period. The Newton-Raphson training algorithm was

$$w_j = w_{j-1} - \eta_j (H^{-1}_{j-1} L(w_{j-1}) + \zeta_j)$$

where ζ_j is a Gaussian random variable with zero mean and as variance was chosen 0, 0.1 and 0.25. For $\eta_j = C/j$ we have chosen for C the values 0.5, 1, 2.5 and 5 with 1 as best result for convergence, and $P_{j-1} = H^{-1}_{j-1}$ is the inverse of the Hessian evaluated at w_{j-1}. As criterion we use the root-mean-squared-error (RMSE), that is

$$RMSE = \sqrt{\frac{1}{11} \sum_{t=1954}^{1984} (g_t - \hat{g}_t)^2}$$

The prediction results for the Netherlands using one-layer (with no hidden neurons) and two-layer networks for the $AR(3)$ model D with ($x_1 = 1$) and without extra input ($x_1 = 0$) are given in Table 1, whereas the results for the one-layer network for the $AR(3)$Li model G, are given in Table 2.

Table 1: ($AR(3)$-model)

extra input	hidden neurons (m)	objective function	RMSE
no	3	0.49	3.76
no	2	1.19	3.58
no	-	0.54	3.64
yes	2	0.66	4.80
yes	-	0.54	4.09
	Zellner		**3.35**
	Otter		3.42

Table 2: (AR(3)Li-model)

extra input	objective function	RMSE
no	0.32	3.87
yes	0.25	**2.20**
	Zellner	2.41
	Otter	2.97

In table 3 the RMSE results for 14 countries are summarized for the three naive models A, B and C, for the two-layer network with three hidden units ($N.N.(3)$) for the $AR(3)$ model D and for the one-layer network ($N.N.$) for the $AR(3)$Li-model G together with the results of Zellner and Otter.

Table 3

	Naive Models			$AR(3)$ model D			$AR(3)$Li model G		
	A	B	C	Zellner	Otter	$N.N.(3)$	Zellner	Otter	N.N
Belgium	2.59	3.53	3.01	2.98	**2.94**	3.07	1.73	1.97	**1.68**
Denmark	2.78	3.53	3.05	2.96	2.95	**2.94**	2.73	2.48	**2.45**
Germany	2.69	2.91	3.85	3.10	3.34	**3.03**	2.28	**2.24**	2.41
Finland	3.39	2.34	2.88	2.58	**2.29**	2.67	3.73	3.03	2.60
France	2.65	2.20	3.08	2.47	**2.36**	2.91	2.52	**2.36**	3.31
U.K.	2.30	3.69	2.62	3.21	3.18	**2.57**	**2.32**	2.47	2.57
Ireland	4.02	2.85	2.35	2.29	2.29	**2.13**	2.80	2.54	2.72
Italy	3.27	4.26	3.98	4.34	4.37	**3.71**	3.40	**3.10**	3.86
Netherlands	3.32	3.57	3.87	**3.35**	3.42	3.76	2.41	2.97	**2.20**
Norway	4.15	1.98	1.76	1.75	1.77	**1.60**	1.62	**1.56**	1.69
Austria	2.91	2.82	3.11	3.16	2.84	**2.70**	2.71	**2.40**	2.94
U.S.A.	3.79	3.98	3.09	**3.01**	3.09	3.10	2.14	**2.12**	2.14
Sweden	2.29	1.87	2.43	2.22	2.05	**2.03**	2.32	**2.14**	2.81
Switzerland	3.03	3.91	4.48	4.24	**4.14**	4.30	3.45	**2.87**	3.84
mean	3.08	3.10	3.04	2.98	2.93	**2.89**	2.58	**2.45**	2.66
median	2.78	3.56	3.08	3.01	2.94	**2.94**	**2.41**	2.44	2.57

The results from Table 3 seem to indicate that by using only the history of the output variable, neural network models might give better predictions on the average, whereas using indicator (input) variables the canonical correlation method of Otter gave slightly better results.

References

[1] Goodwin, G.C. and R.L. Payne (1977) *"Dynamic system identification; Experiment design and data analysis"*, Academic Press, New York.

[2] Hornik, K. M. Stinchcombe and H. White (1989) *"Multilayer feed-forward networks are universal approximators"*, in Neural Networks, Vol. 2, p.p. 359-366.

[3] Kushner, H. and D. Clark (1978) *"Stochastic approximation methods for constrained and unconstrained systems"*, Springer Verlag, Berlin.

[4] Otter, P.W. (1990) *"Canonical correlation in multivariate times series analysis with an application to one-year and multiyear ahead macroeconomic forecasting"*, Journal of business and economic statistics, Vol. 8, no. 4, p.p. 453-457.

[5] Robbins, M. and S. Morno (1951), *"A stochastic approximation method"*, The annals of mathematical statistics, Vol. 22, p.p. 400-407.

[6] White, H. (1988) *"Economic prediction using neural networks: the case of IBM daily stock returns"*, Int. Conference on Neural Network, IEEE-proceedings, Vol. II, p.p. 451-458.

[7] White, H. (1989a) *"Some asymptotic results for learning in single hidden-layer feed-forward network models"*, Journal of the American statistical association, Vol. 84, no. 408, p.p. 1003-1013.

[8] White, H. (1989b) *"Neural network learning and statistics"*, AI expert, Dec. 1989.

[9] White, H. (1989c) *"Learning in artificial neural networks: a statistical perspective"*, Neural computation 1, p.p. 425-464.

[10] Zellner, A. and C. Hong (1989) *"Forecasting international growth rates using Bayesian shrinkage and other procedures"*, Journal of econometrics, Vol. 40, p.p. 183-202.

Desperately Seeking Stability: Neural Networks with Insufficient Data

Susan Garavaglia
Dun & Bradstreet Information Services, N. A.
C. U. N. Y. Graduate Center

Abstract: *The bond rating simulation application from Garavaglia [2] is re-considered in the context of model stability. Using conventional statistical methods, a unique and stable solution is not possible without sufficient degrees of freedom. Although this is also a requirement for stable neural networks, the method of coefficient (weight) determination in neural networks is iterative in contrast to the solution of normal equations which is one of the paradigms in statistical regression. In this paper, we consider the behavior of the counterpropagation network under conditions of insufficient degrees of freedom.*

1. Background

A counterpropagation network for simulating the Standard & Poor's corporate bond rating systems was proposed by Garavaglia [2]. The initial results were that network performance measured in terms of overall accuracy ranged from 83.78% to 84.9% for classifying three general categories, AAA to BBB- as investment grade, BB+ to B- as speculative grade, and all lower ratings as poor quality, and ranged from 0% to 55.56% for classifying 17 specific rating grades. In the three category evaluation, about 98% of the misclassifications were "one off," i. e., investment grade companies were not misclassified as poor quality ("two off") and vice versa. These results raised a number of issues, including the feasibility of a usable model when the distribution of classes is uneven and the boundaries between the classes appears to be somewhat fuzzy.

This paper uses the same data and application but is concerned with the topic of neural network stability, i. e., the evaluation and maximization of network performance with out-of-sample data in terms of predictability, reliability, and usability. We consider the results of several new experiments that are evaluated in terms of the differences in performance between the training and testing data. The recommendations, given the non-optimal situation of an insufficient number of observations to provide the required degrees of freedom, are to: 1) take actions to reduce the degrees of freedom, and 2) analyze the rank ordering power of the network output values to determine if a subset of the network output is sufficiently stable and predictive. Dutta, Shekhar, and Wong [1] have addressed the stability problem from the standpoint of preventing network overfitting and failure to generalize as well as further discussions of the bond rating application.

The problem of insufficient data is not trivial. One may observe that no matter how much data is available, the complexity of the application almost always seems to require more data. Categorization type applications in financial services in which only a limited amount of data may be available include:

1. Corporate Bond Ratings: Standard and Poor's [7] has outstanding ratings on about 2,000 corporations, 1,600 issuers of commercial paper, and about 8,000 public sector entities (municipals and other government-related). The coverage of Moody's Investors Service is similar in scope. Depending on the number of rating classes, these quantities may not be sufficient. Other rating agencies, such as Fitch and Duff & Phelps, have even fewer outstanding ratings.

2. Senior Corporate Lending Risk Management: The top tier corporate lending function of a money center bank may have only between one and two thousand customers. Banks' internal rating systems usually have 9 or 10 classes. Depending on the number of data elements used, there may not be enough examples.

3. Transaction Risk Identification: Any categorization task in which some categories are represented by a very small (e. g. less than 1 %) of the sample present problems. This is typical in the areas of fraud detection and predicting default. When a large sample, which has an artificially high proportion of these categories, can be provided, these problems can be overcome.

2. Degrees of Freedom in Neural Networks

In conventional statistical techniques, the term degrees of freedom refers to "the number of free or linearly independent sample observations used in the calculation of a statistic."[1] In regression analysis, a requirement is that there are more observations than coefficients to be estimated, in order to guaranty a unique solution. This is because the solution process requires that the observation vectors be able to form a non-singular (invertable) square matrix when multiplied by themselves. With fewer observations than variables, the rows and columns of the square matrix are not linearly independent, and the rank of the matrix is less than the dimension of the matrix.

Another way of understanding this restriction is to consider the degrees of freedom as the number of observations minus the number of coefficients to be estimated. Zero or negative degrees of freedom means that a potentially infinite number of solutions exist to exactly fit the data, but these solutions will have no relationship to out-of-sample data. A simple example is a point in *n*-dimensional space. If this point were to represent a single observation, any number of lines of any slope can be drawn through that point. Therefore, if an arbitrary line is chosen, the predicted fit of this line to any population is indeterminate. If there are two points in 2-dimensional space, a line, defined by a intercept and slope, will fit these two points exactly, but have no predictable relationship to any other point. Therefore, if two observations are used to estimate a 2 parameter model, there will be overfitting.

The strict definition of degrees of freedom requires that the number be greater than zero. Therefore, this discussion, which refers to negative degrees of freedom, is stretching the definition for the purposes of exploring the behavior of neural networks when there are not enough data vectors, or observations, to produce a non-singular square matrix from the vectors. Kennedy [4] and Phillips [6] approach this subject from a viewpoint of practical implementations and Mac Lane's [5] chapter on Linear Algebra is an excellent mathematical treatment.

The analogy in neural networks is the problem of overfitting the data in the training sample. This occurs when the number of training examples, or observations, does not exceed the total number of connection weights in the network. If the training sample does not represent the characteristics of the population, results may also look like overfitting.

In a fully-connected counterpropagation network, the total number of connections C is:

$$C = (I + O) * H \qquad (1)$$

where I is the number of input layer units, O is the number of output layer units, and H is the number of hidden units. Therefore, the total number of connections in the bond-rating network #1 in [2] with 170 hidden PE's, is:

17,680 connections =
(87 input units + 17 output units)
* 170 hidden units

The number of degrees of freedom is therefore:
156 - 17,680 = -17,524.

3. The Revised Experiment

As a result of the initial research, a number of changes were made to the experiment:

1. The training, or development, data set was enlarged from 156 to 415 examples, and the testing, or holdout, data set was reduced to 381 from 640 examples. The objective of this realignment of the data was to provide the maximum degrees of freedom in development while still providing a full range of examples for testing. The fact that the sample did not contain equal proportions for each rating class made this more difficult.

2. The data elements, with the exception of the SIC code, and the two prior ratings, were transformed to their natural logarithms. Formerly, the transformation was to simply divide these data elements by powers of 10. Using natural logarithms is more consistent with common practice.

3. A separate test was performed using three output categories representing very general grades - investment, speculative, and poor quality, as described in the Introduction.

4. Evaluation Criteria

In the original research, the neural networks were evaluated based only on the results from the holdout data set. The main focus of this current research is to determine relative levels of model stability. Model stability will be evaluated on two

[1] Kennedy [], p. 214.

criteria: 1) the overall accuracy results on the training (development) versus the testing (holdout) samples, and 2) the overall accuracy results on the highest output unit value. In counterpropagation, the output values are in the range [0,1]; the value 1 affirms membership in the category represented by the output unit, and only one output unit contains a 1 for any given presentation of input vectors. However, in actual implementation, output values may exhibit a full range of values in the [0,1] interval, which may correspond to the degree of fit to the category, although not in a fuzzy-set membership sense. In Garavaglia [3], using this range to produce a rank-ordering is discussed in detail.

5. Results

Table 5-1 contains the results from the original network #1 (From Table 5-2 of [2]). Clearly, this is an example of overlearning or overfitting the training data. Overall accuracy falls to 22.46% from 97.44% and the accuracy of the highest output unit value falls to 11% from 100%. This network would be considered highly unstable as would be expected based on -17,510 degrees of freedom. Neither the overall accuracy nor the accuracy of any output unit value is stable.

Table 5-1
Stability Results of the Original 170-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	100.00%	44.44%
AA+	100.00	50.00
AA	100.00	20.25
AA-	100.00	7.14
A+	100.00	16.47
A	100.00	19.72
A-	100.00	31.15
BBB+	88.89	20.00
BBB	100.00	28.57
BBB-	88.89	30.43
BB+	100.00	18.75
BB	100.00	0.00
BB-	100.00	18.18
B+	88.89	11.11
B	88.89	4.00
B-	100.00	29.69
CD	100.00	34.38
Overall	97.44%	22.46%
Highest Output Unit Value:		0.997381
Accuracy in training:		100.00%
Accuracy in testing:		11.00%

Table 5-2 shows the results of a network that also contains 170 hidden units, but uses the new training data set of 415 examples. The number of degrees of freedom is -17,265. Overall accuracy falls from 60% to 32.02%. The highest output unit value is 100% accurate for both samples. As the difference in overall performance between the training and testing data sets is smaller than in the original 170-hidden unit network, and the accuracy of the highest output value does not degrade, this network is considered more stable.

Table 5-2
Stability Results of the New 170-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	70.00%	18.75%
AA+	60.00	0.00
AA	62.50	22.92
AA-	46.15	4.00
A+	90.00	70.37
A	62.50	47.50
A-	62.86	31.43
BBB+	43.33	33.33
BBB	60.00	16.67
BBB-	75.00	0.00
BB+	40.00	0.00
BB	30.00	0.00
BB-	60.00	20.00
B+	26.67	0.00
B	40.00	7.14
B-	93.33	73.68
CD	87.50	45.00
Overall	60.00%	32.02%
Highest Output Unit Value:		0.938848
Accuracy in training:		100.00%
Accuracy in testing:		100.00%

Table 5-3 shows the results when the number of hidden units is reduced to 51. The number of degrees of freedom is now "increased" to -4,889, (415 observations - (87 input units + 17 output units)*51 hidden units), based on formula (1). The difference in overall accuracy between the training and testing data sets is also reduced even further than the 170-hidden unit network, but the overall performance is poor. The highest output unit value is still reasonably accurate at 88%.

Table 5-3
Stability Results of the 51-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	35.00%	25.00%
AA+	20.00	44.44
AA	60.00	47.92
AA-	19.23	0.00
A+	17.50	5.56
A	57.50	47.50
A-	5.71	2.86
BBB+	20.00	10.34
BBB	6.67	0.00
BBB-	15.00	0.00
BB+	40.00	10.00
BB	40.00	0.00
BB-	20.00	0.00
B+	0.00	0.00
B	5.00	7.14
B-	73.33	73.68
CD	83.33	55.00
Overall	30.00%	22.10%
Highest Output Unit Value:		0.962427
Accuracy in training:		100.00%
Accuracy in testing:		88.00%

When the number of hidden units is again reduced to 17, the same number as the number of output units or classes, the overall difference in accuracy is minimal (see Table 5-4). The degrees of freedom has been increased to -1353. The stability of the output unit values is difficult to assess, however, because the five highest output unit values are assigned only one observation each in the training data set and the two highest output values are not assigned to any observations in the testing data set.

Table 5-4
Stability Results of the 17-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	70.00%	31.25%
AA+	10.00	0.00
AA	47.50	37.50
AA-	0.00	0.00
A+	22.50	27.78
A	17.50	17.50
A-	5.71	2.86
BBB+	0.00	0.00
BBB	3.33	0.00
BBB-	20.00	8.33
BB+	0.00	0.00
BB	0.00	0.00
BB-	0.00	0.00
B+	0.00	0.00
B	0.00	0.00
B-	73.33	68.42
CD	67.67	70.00
Overall	20.00%	19.42%
Highest Output Unit Value:		0.975326 (1 example)
Accuracy in training:		100.00%
Accuracy in testing:		(N/A)

The last test involved creating a network that had a positive number of degrees of freedom. In order to create such a network with the data, the number of output classes was reduced from 17 to 3. The three classes are investment grade (ratings AAA to BBB-), speculative grade (BB+ to B-), and poor (below B-). As there are 415 training set examples, a positive number of degrees of freedom required that a maximum of 4 hidden units is allowed:

degrees of freedom =
415 observations
- (87 input units + 3 output units)
* 4 hidden units
= 415 - 360 = 55

Table 5-5 shows the results of this network. Since there were only four hidden units, there are only four output unit values. The network effectively used two hidden units for classifying investment grade bond ratings, two for speculative bond ratings, and did not classify poor grade ratings at all. The accuracy percentages of all four output unit values are shown at the bottom of the table.

Table 5-5
A General Investment Class Categorization Network

Class	Training Pct. Correct	Testing Pct. Correct
Investment	91.8%	94.8%
Speculative	63.0%	58.3%
Poor	0.0%	0.0%
Overall :	79.52%	82.9%
Output unit values:		
0.991468	99.16%	96.77%
0.718827	79.26%	84.53%
0.620933	62.86%	58.33%
0.513653	50.00%	51.85%

6. Discussion

In reviewing the five tests discussed above, it appears that the data are most suitable for a classification task that is limited to three general investment grades. It appeared that by increasing the degrees of freedom, network stability improved, even though the degrees of freedom remained negative in the first four tests. The overall accuracy for the testing data set varied from 19.42% to 32.02%.

An attempt was made at a stable network that could classify the 17 ratings classes, by reducing the number of input data elements from 87 to 24, keeping only the SIC code, the past two period's bond ratings, and the data elements that represented financial ratios. The network could not discriminate at all, putting all of the examples in the same class in the hidden layer and having very low output values for every class. In addition, estimation of the model was attempted with logistic regression with no usable results.[2]

In the counterpropagation paradigm, each set of weight vectors connected from the inputs to a hidden unit adapts to become a form of linear discriminant.

[2] Using SAS software on an HP 750 processor, the problem required about 13 megabytes of memory and did not converge after 10,000 iterations.

Because the hidden layer activation function treats each hidden unit as an independent competitor for "claiming" an input vector, the trained model conceptually has as many discriminant functions as hidden units. Therefore, the total number of coefficients may not be the best term to use to compute the degrees of freedom, in that, by having stability improve within a negative range of degrees of freedom, the model behaves as if there are more degrees of freedom available. The backpropagation paradigm, in contrast to counterpropagation, should require degrees of freedom to be calculated on the total number of connections, because backpropagation learning adjusts all of the connections for each learning iteration. This is topic for further research and may suggest the application of certain network architectures when only a small amount of data is available.

7. Conclusion

The availability of a sufficient number of observations is critical to neural network stability and reliability. To assure a positive number of degrees of freedom, more observations are needed for a neural network model than for a conventional statistical regression model. The number of degrees of freedom is the number of observations minus the total number of connections, which, in the case of a fully connected three-layer network, can be calculated by using formula (1). Some improvement in stability may be gained, even if the number of degrees of freedom is negative, if the number of hidden units or input units can be reduced, but these results are not consistent with accepted practices in statistical modeling. In any case, the developer must be aware of the total number of connections generated by a neural network and anticipate stability problems if not enough observations, or training cases, are available.

8. References

[1] Dutta, Soumitra, Shashi Shekhar, and W. Y. Wong. Decision Support in Non-Conservative Domains: Generalization with Neural Networks. *Decision Support Systems*, forthcoming.

[2] Garavaglia, Susan. 1991. An Application of a Counterpropagation Neural Network: Simulating the Standard & Poor's Corporate Bond Rating System. Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street. New York. Oct. 9-11, 1991. Los Alamitos, CA: IEEE Computer Society Press.

[3] _____. 1993. Interpreting Neural Network Output. *Advanced Technology for Developers*. Vol. 2. February 1993. Sewickley, PA: High-Tech Communications, Inc.

[4] Kennedy, Peter. 1985. A Guide to Econometrics. 2d ed. Cambridge, MA: The MIT Press.

[5] Mac Lane, Saunders. 1986. Mathematics Form and Function. Berlin: Springer-Verlag.

[6] Phillips, John L., Jr. 1992. How to Think About Statistics. rev. ed. New York: Wm. H. Freeman & Co.

[7] Standard & Poor's Corp. 1991. S & P's Corporate Finance Criteria. Solomon Samson, ed. New York: Standard & Poor's Corp.

Paper Session: Quantitative Analysis

Chair: Ernest Racz, Polytechnic University

Combined Approximate Reasoning in the M&A Domain

Saad Ayub
GTE Laboratories Incorporated
40 Sylvan Road
Waltham, MA 02254
e-mail: ayub@gte.com

Piero P. Bonissone
Artificial Intelligence Laboratory
GE Research and Development Center
Schenectady, NY 12301
e-mail: bonissone@crd.ge.com

Abstract

In this paper, we describe the application of our system, named **CARS**, **C***ombining* **A***pproximate* **R***easoning* **S***ystems, in the domain of mergers and acquisitions (M&A). CARS uses both Rule-Based Reasoning (RBR) and Case-Based Reasoning (CBR) to develop a plan for an agent involved in an M&A deal. Partial domain knowledge of M&A (i.e., financial knowledge) is represented using rules and weak M&A domain theory is compensated by real M&A cases.*

The M&A cases are stored as a situation/solution pair, indexed by surface (observed) and abstract (derived) features. The mapping from surface to abstract features, provided a way for the situation descriptor to represent the case in a more robust feature space. The determination of similarity between the current case and the past cases is based on the values of abstract features. The similarity of each abstract feature in probe case to that in the retrieved cases is computed as the complement of the distance between the fuzzy numbers representing the feature values. The abstract features similarities are aggregated hierarchically, according to the needs of the reasoner. The solution of the most similar case is then adapted for the current situation.

We illustrate with examples the combined use of RBR and CBR within the domain of M&A by CARS. First we illustrate the use of RBR in enhancing case retrieval and better selecting the most appropriate case. Then we show the use of CBR in adapting a retrieved solution by selecting an appropriate action from other cases.

1 Introduction

A corporate Mergers and Acquisition (M&A) deal is enormously complex and requires sophisticated reasoning and planning on the part of several parties. The two parties that are always involved in an M&A deal are the raider and the target. The raider is the one that initiates the takeover attempt and the target is the corporation that is of interest to the raider. The way an M&A deal is structured and planned is influenced by the specialized knowledge of the agents about areas like finance, and their experiential knowledge of the past M&A deals.

Our system, named **CARS**, Combining **A**pproximate **R**easoning **S**ystems, utilizes both experiential and specialized knowledge for developing a plan(s) to achieve the goal(s) of an agent. The agent can be a raider or a target company. The experiential knowledge about other M&A transactions is stored as cases and the specialized knowledge is stored as rules. CARS uses Case-based Reasoning (CBR) as the dominant reasoning technique in order to leverage on the past experiences and to compensate for the lack of good theory of M&A domain. It uses Rule-Based Reasoning (RBR) on the available specialized knowledge to take advantage of partial domain knowledge.

We built our case base using real M&A cases. Each case in the case base was first defined in terms of its *overall structure* of events, goals, and interpretations and then augmented with more *detailed information* about the companies involved in the hostile battle.

The case base was built based on information from two types of sources: 1) cases used for teaching (developed at the Harvard Business School and the Darden Graduate Business School at the University of Virginia), 2) an on-line database service on M&A transactions (IDD Information Services). The developed cases were then augmented with other surface features, such as financial, industrial, and company related information. These features were obtained using company's annual reports on compact disclosure, Moody's industrial manual, and Dunn & Bradstreet Industrial financial ratio averages.

In this paper we illustrate how RBR is used in CARS to get additional information about the companies involved in a certain M&A case. This information is then used to better select the most appropriate case. We also show how solutions to various sub-problems in the past cases are identified and utilized for modifying the solution of the selected case.

1.1 Related Work

The need to integrate diverse reasoning techniques to effectively solve complex real world problems has recently been recognized. Within Case-Based reasoning, this trend is represented by the works of Carbonell and Veloso [14, 33] (integration of CBR and classical search problem solvers), Hammond and Hurwitz [22] (integrating CBR and explanation based reasoning), Braverman [13] (integrating CRB and explanation based learning), Koton [23] and Goel [20] (integrating CBR and model based reasoning), Rissland and Skalak [27, 26], Dutta and Bonissone [18, 19], Branting [12], Berg-Cross and Claudio [3], and Golding [21] (integrating RBR and CBR).

In particular, we believe that the integration of case-based (CBR) and rule-based (RBR) reasoning is a crucial factor for the development of reasoning systems solving realistic problems. RBR and CBR are largely complementary reasoning methodologies. RBR can better represent specialized domain knowledge in a modular, declarative fashion, while CBR can better represent past experiences and domain complexity [25]. There are numerous domains in which it is important to combine RBR and CBR, such as the legal [24, 28] and the financial domains [11] [3].

There are three possible ways of combining RBR and CBR: 1) a higher level controller (blackboard) decides whether to use the case-based or the rule-based reasoner; 2) the rule-based reasoner explicitly requests the use of the knowledge available from the cases; 3) the case-based reasoner uses the inference capabilities provided by the rules.

The first paradigm is exemplified by Rissland and Skalak's approach [27], and Berg-Cross and Claudio's approach [3]. While working in the legal domain of statutory interpretation, Rissland and Skalak have developed a hybrid reasoning system called CABARET. The system's architecture consists of two co-equal reasoners, an RBR and a CBR, with a separate agenda-based controller. The central controller contains heuristics to direct and interleave the two modes of reasoning and to post and prioritize tasks for each reasoner. In a more recent version of CABARET [26] this type of control is implemented by a blackboard system. Berg-Cross and Claudio have developed a system, MASTER, for aiding in the selection of an M&A candidate company. Their system also uses control rules to monitor case selection and rule firing. They use cases to provide precedence and arguments for or against selection of a company. They use rules for adaptation and evaluation of cases.

The second paradigm is illustrated by Dutta and Bonissone's approach [18, 19]. They have developed a system, MARS, that is controlled by the rule based system side. In MARS rules are used to represent the domain expertise required for structuring various aspects of the M&A deal or deciding upon the next best course of action. Cases provide the prior experiences of other companies involved in similar hostile takeovers. For example the raider may have certain rules to determine which company to select as a possible target. The system may have similar cases to determine whether the Justice Department will block the takeover to enforce anti-trust laws.

The third paradigm is exemplified by our work in the development of CARS [9, 8]. In the CARS system architecture, the CBR is the dominant reasoner. During case indexing, retrieval, and adaptation, the CBR activates the Plausible ReasonIng Module (PRIMO) [7, 1], a rule-based reasoner that contains plausible rules of abstraction, evaluation, and modification. This approach allows use of the general domain knowledge for solving subproblems which have good domain theory. Thus, we can exploit general theory for those parts of the domain that are well described even though we do not have a good theory for the whole domain (the reason for which we are using the past cases).

The first paradigm provides a nice modularity for control. However, it is predicated on finding the right heuristics for the controller (and on being able to extend or customize these heuristics to different domains.) The second approach uses more domain-dependent heuristics to determine the activation of the CBR.

The third approach is basically different from other integrated CBR and RBR reasoners in the sense that they require good domain heuristics (i.e., in first approach) and good general domain knowledge (i.e., in second approach). Our approach does not need good domain theory for solving the whole problem but uses the general theory for subproblems when there is one.

1.2 Paper Structure

In the next section we define the structure of a case as the pair situation/solution. We give a brief overview of the organization of the M&A case memory and the case representation language (CRL) used to represent the cases.

In Section 3 we show the mapping from surface (observed) to abstract (derived) features using rules. We describe this mapping process for the abstract feature representing the short term financial condition of a company. This mapping, based on fuzzy predicates and plausible rules is implemented in PRIMO.

In Section 4 we show how the cases are ranked by their similarity to the current situation. First, we compare the linguistic values assigned to each abstract feature and we compute the distance between their corresponding fuzzy sets. The similarity measure, obtained as the complement of this distance [34, 29], is translated into a label taken from a termset defining different degrees of matching. The similarity measures

of the abstract features are then aggregated according to the needs of the reasoner.

In Section 5 we show how a part of the solution from the most similar case is modified by using solutions from other cases.

2 Organization of M&A Case Memory

The M&A case memory was designed to represent cases consisting of the top-level goal(s) and information about the states and events. This information can be obtained from two basic sources: world observers and domain experts. *World observers* are capable of recording the *state* at any time, and of recognizing the execution of state changing *actions* in the world. Domain experts are capable of *interpreting/relating* these states and actions to the behaviors of an agent(s) attempting to achieve the top level goal(s) of the case.

The case memory is organized around two types of knowledge: conceptual knowledge and episodic knowledge. **Conceptual Knowledge** is the information about the objects, actions, goals. **Episodic Knowledge** is the collection of past M&A cases. In this section we will briefly describe the case memory. The detailed description is given in [8].

2.1 Conceptual Knowledge

The M&A domain knowledge needed for representing a case is organized into three hierarchies: **Object hierarchy**, **Action hierarchy**, and **Goal hierarchy**.

All objects used in representing the cases are part of the IS-A **Object hierarchy**, which describes the objects of the M&A domain and their relationships.

The actions represent the state changing operations on the objects in the domain. The IS-A **Action hierarchy** abstracts from special actions (with more restricted preconditions and effects) to more general actions. The instances of a particular action class are parts of executed plan actions in the stored cases.

The **Goal hierarchy** represents the partial knowledge of the M&A domain and provides an initial, albeit incomplete goal decomposition. Its incompleteness is the reason for resorting to case-based reasoning and mixed reasoning paradigms. It provides a mechanism for expansion of knowledge by indexing into each new case at various suitable levels.

2.2 Episodic Knowledge

Each case in our case base is represented as situation/solution pair. The situation consists of the top-level goals and starting states of the agents. The solution consists of the representation of the observable portions of the executions of actions by the agents and their effects on the states of world.

Given the dynamic nature of M&A cases, we have chosen to divide each case into five phases: (1) Initial Condition, (2) Pre-tender, (3) Tender-negotiation, (4) Outcome and (5) Long-term results. Within each phase, the case is represented by states, events, goals, and interpretations. Each of the above five phases correspond to a top-level interpretation.

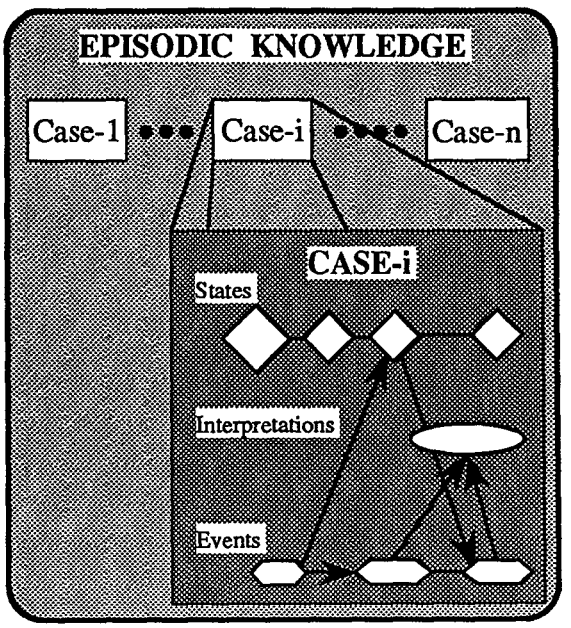


Figure 1: The representation of a case

The object descriptions at various stages of the cases are stored as **States**. The initial state of each object is represented by a set of state variables (surface and abstract features) with their associated values. Each state change contains only the slots whose values have changed.

Events are objects that associate action, goal, and state information. The actions and goals are defined in the conceptual knowledge. Each event has at least two slots: the (instantiated) *action* used to achieve the goal and the *context*, i.e., the state at the time this event occurred. Additional optional slots can be used to describe goal, cost, and time duration of the event. The event's contextual understanding is provided by various links like temporal links, causal links, and enable links.

Explanations for the occurrence of sets of events can be given using **Interpretations**. The interpretations are currently used to represent the case analysis of domain experts. Figure 1 shows an example of how events, states and interpretations can be linked for representing a case.

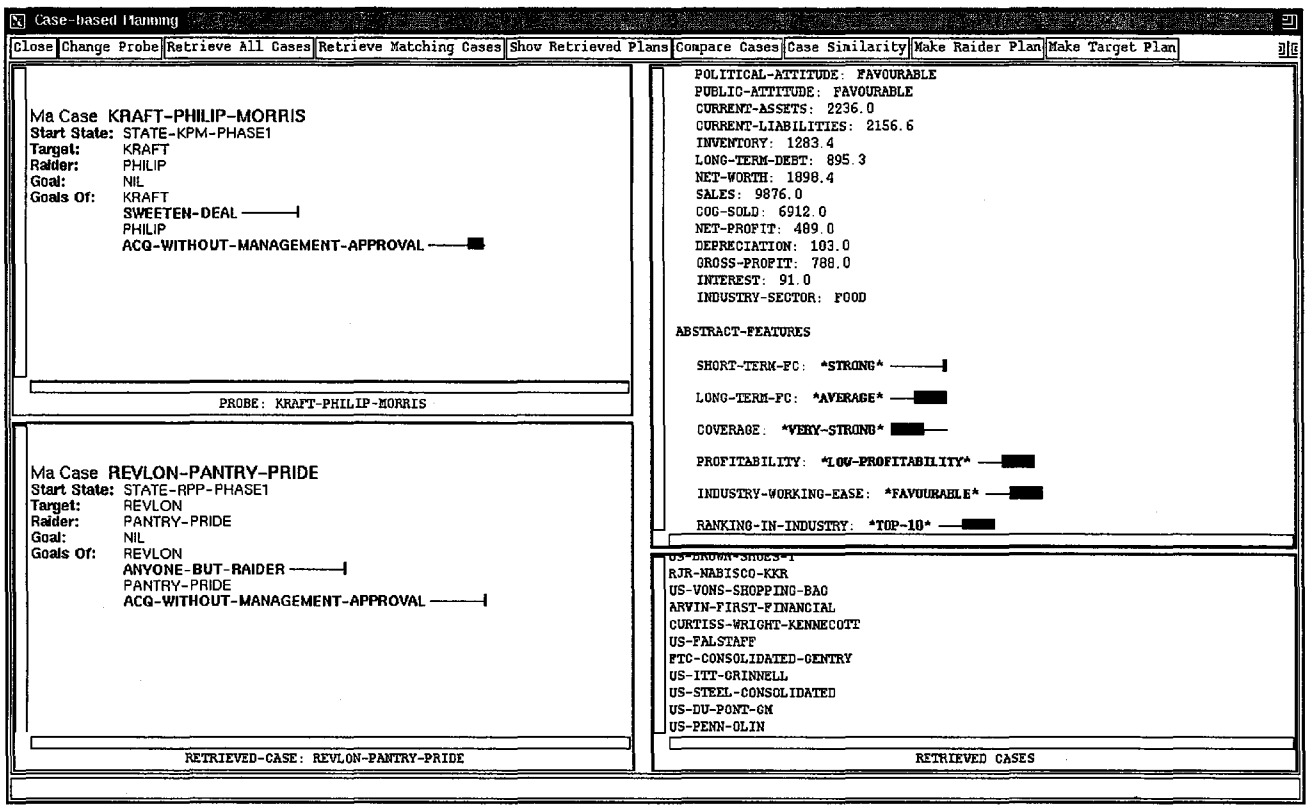


Figure 2: Surface and Abstract Features of Probe Target Company

The cases, interpretations and events are linked to the **goals** in the goal hierarchy. The uncertainty in the links represent the belief that the cases, interpretations and events are the executed plans/strategies/plan steps for achieving the goals/subgoals.

2.2.1 Surface and Abstract Features

The representation of cases is augmented by a mapping from surface to abstract features. The mapping is based on approximate deduction from the surface features using plausible rules of inference. The surface features, typically obtained from textual description of the case, provide factual information at a very low level of granularity. Figure 2 (right window) shows an example of surface and abstract features for the Probe Target company.

For each abstract feature we have defined a termset of linguistic values. Each value is defined by a label and its semantics. The semantics are represented by the membership distribution of a fuzzy set, defined on the unit interval, establishing a partial ordering among the labels.

We obtain abstract features by using the domain knowledge encoded as rules or by using the conceptual knowledge about the objects in the case. In the next section, we will illustrate how the domain knowledge encoded as rules is used to derive abstract features.

3 Deriving Abstract features using rules

The use of domain-specific rules is one of the two techniques used in CARS for analyzing cases to derive abstract features. These features are assigned a value and a degree of certainty. Values for features (abstract or surface), can be raw data or lexical terms (linguistic values representing fuzzy intervals [35]) chosen from feature value term-sets provided in CARS. The degree of certainty represents the extent to which the abstract features can be inferred from the surface features.

3.1 Determination of Short Term Financial Condition

One of the abstract features which is considered important in the M&A domain is `Short-Term--Financial-Condition`. It can be obtained from surface features such as `Receivable-Turnover`, `Cost--of-Goods-Sold`, `Inventory-Turnover`, `Current--Assets`, and `Current-Liabilities`, based on the outcome of various financial ratios which are calculated using the surface features. The value of this abstract feature indicates emergent information about

the company that was not available from the individual surface features used for deriving it.

Various PRIMO plausible rules are used for determining the value of `Short-Term-Fc` abstract feature. One such rule is `Acid-Ratio-St-Fc` which is given in Figure 3. We will now consider how the PRIMO plausible rules work and how the `Short-Term-Fc` abstract feature is determined by using `Acid-Ratio-St-Fc`.

```
(def-rule
 (acid-ratio-st-fc                ; RULE NAME
    case-based
    (resources)                   ; RULE CLASS
    (company*industry-ratios))    ; INSTANTIATION CLASS
 (?company ?industry-ratios)      ; OBJECT-VARIABLES
                                  ; DOCUMENTATION
"short term financial condition using acid ratio"
                                  ; CONTEXT
 (lb-pass-threshold
   (t3 (num-pred (current-assets ?company))
       (num-pred (current-liabilities ?company))
       (num-pred (inventory ?company))
       (num-pred (acid-ratio ?industry-ratios)))
   250)
                                  ; ANTECEDENT
 (acid-ratio-pred
       (current-assets ?company)
       (current-liabilities ?company)
       (inventory ?company)
       (acid-ratio ?industry-ratios))
                                  ; CONCLUSION
 (((short-term-fc ?company)
   ((acid-ratio-cons
       (current-assets ?company)
       (current-liabilities ?company)
       (inventory ?company)
       (acid-ratio ?industry-ratios))
                                  ; RULE STRENGTH
    (i::d3 *certain* *likely* :premise)
           :INTERSECT)))
```

Figure 3: Rule to determine the short term financial condition of a company

Rule Evaluation: In PRIMO, the rule *Antecedent* is a conjunction of (possibly) fuzzy predicates on object-level variables. The conjunction is implemented using T-norms [6]. The result of the antecedent is the degree of applicability of the rule. The output of the antecedent and the *Rule Strength* determine the truth value of the *Rule Conclusion.* In our example we have one predicate `acid-ratio-pred`, which computes the acid ratio of the company as:

$$AcidRatio = \frac{CurrentAssets - Inventory}{CurrentLiabilities}$$

and normalizes it with respect to the industry average acid ratio. The mapping, illustrated in Figure 4, is then used to select the term that best describes the short term financial condition of the company, given the acid ratio average of its industry sector.

Acid Ratio Percentage Interval	*Term Label*	*Term Semantics*
[0,60]	*VERY-WEAK*	(0 130 0 20)
[60,80]	*WEAK*	(170 270 20 30)
[80,90]	*BELOW-AVERAGE*	(310 410 30 30)
[90,115]	*AVERAGE*	(450 550 30 30)
[115,140]	*ABOVE-AVERAGE*	(590 690 30 30)
[140,170]	*STRONG*	(730 830 30 20)
[170, ∞]	*VERY-STRONG*	(870 1000 20 0)

Figure 4: Mapping of Percentage Acid Ratio to Terms Labels and Semantics

In our implementation the intervals used in the mapping are actually fuzzy intervals. Therefore, the membership value of the acid ratio percentage is computed for each term in the termset. The term with the highest membership value is selected. The corresponding membership value describes the degree of confidence of this linguistic value assignment.

Figure 2 shows that the value of `Short-Term-Fc` abstract feature of the probe case target company is *STRONG*. This value was derived using the `Acid--Ratio-St-Fc` rule along with four more rules which use other financial ratios that are also important for the short term financial condition of the company.

This is a typical example of integration between CBR and RBR. In this case the Case-Based Reasoner activates the Rule-Based Reasoner to augment the case indexing information. In the next section we will describe how we use this information for determining the most similar case for the current situation.

4 Ranking Cases by Similarity

In order to use the previous cases, we need to rank them according to how similar they are to the current situation (i.e., probe case). The needs of the reasoner (e.g., goal satisfaction or establishment of precedent) dictates the most appropriate similarity measures that should be chosen and applied in ranking the cases returned by the retrieval system.

The M&A cases have two distinct aspects of similarity that needs to be determined separately: situational similarity and dynamic similarity. The situational similarity determines similarity between the states of the worlds of the two cases. The dynamic similarity determines the similarity between the evolution of two cases. The detailed description of determination of dynamic similarity between the evolution of two case, represented by a sequence of events using CRL, is given in [2].

We will now briefly describe how the situational similarity is determined; more details on the technique can be found in [2], [9].

4.1 Abstract Feature Value Comparisons

By analyzing the probe and the retrieved case, we obtain a linguistic value for each of their abstract features. Each of these linguistic terms has a label and a meaning. This is illustrated in the second and third columns of Figure 4.

In the third column of Figure 4, a parametric representation is used to describe the membership distribution of each term, N_i. Using this representation we can describe a fuzzy set of a universe of discourse U as the four-tuple: (a, b, α, β). The universe U is unit interval (represented by an integer representation on the scale from 0 to 1000).

The computation of the similarity measure of each feature uses the meaning of the labels that define each abstract feature value. This similarity computation is done by executing a two-step procedure.

First we compute the distance between the fuzzy set representations of the corresponding values. For example, let us assume that the abstract feature `Short-Term-Fc` has value *STRONG* for the target company in the probe case and *VERY-STRONG* for the target company in the retrieved case. The distance between the two corresponding fuzzy sets is computed as the absolute value of their difference. This is done using fuzzy arithmetic operations that are closed under the four-tuple parametric representation [15, 5, 10]. Specifically, given two fuzzy numbers $X = (a, b, \alpha, \beta)$ and $Y = (c, d, \gamma, \delta)$, their difference is defined as: $X - Y = (a - d, b - c, \alpha + \delta, \beta + \gamma)$. This distance is then transformed into a degree of matching by taking the complement with respect to the unit interval. Using the same formula for the difference, by representing the unit as (1000, 1000, 0 , 0) we have the degree of matching $1 - |X - Y| = (730, 960, 30, 40)$.

The second step, referred to as linguistic approximation, consists of selecting a label (chosen from one of the similarity term-sets provided) whose meaning is the closest to that of the computed degree of matching. This semantic closeness is evaluated by a measure of set inclusion [30]: $\frac{|P \cap D|}{|D|}$ where P is the similarity term and D is the result of complementing the set-distance. This measure, representing the degree of matching between the reference (P) and the data (D), is used as an associated certainty value for the label. The interested reader is referred to reference [16] (page 23-24) for a detailed study of measures of inclusions. A simple example of a seven term similarity termset is given by Figure 5.

In the case of our example, the degree of matching was the fuzzy number $(730, 960, 30, 40)$. By using the termset described in Figure 5, we can see that the term with the closest meaning $(730, 830, 30, 20)$ is *ALMOST-COMPLETE-MATCH*.The degree of confidence in this label selection is

$$\frac{|\,(730, 830, 30, 20) \cap (730, 960, 30, 40)\,|}{|\,(730, 960, 30, 40)\,|} = \frac{125}{265} = 0.47.$$

Similarly, the degree of confidence in the term *COMPLETE-MATCH* would be 0.45.

Therefore, the term *ALMOST-COMPLETE-MATCH* is selected as the value for the similarity measure for the feature `Target-Short-Term-FC-SIM`.

Multiple similarity term sets are used in CARS to allow for different "views" of similarity (e.g., the lenient similarity term set has wide fuzzy intervals for the labels representing high similarity and narrower intervals for those representing low similarity. The opposite is true for the strict term set.)

Term Label	*Term Meaning*
NO-MATCH	(0 130 0 20)
ALMOST-NO-MATCH	(170 270 20 30)
LESS-THAN-PARTIAL-MATCH	(310 410 30 30)
PARTIAL-MATCH	(450 550 30 30)
MORE-THAN-PARTIAL-MATCH	(590 690 30 30)
ALMOST-COMPLETE-MATCH	(730 830 30 20)
COMPLETE-MATCH	(870 1000 20 0)

Figure 5: Termset For Partial Matching of Abstract Features

4.2 Combinations of Similarities

Once the abstract features similarities are determined, then these similarities are aggregated to get higher level or overall similarities. An example of a higher level similarity is financial condition similarity. This similarity is an aggregation of `Short-Term-FC`, `Long-Term-FC`, `Coverage` and `Profitability` similarity measures. The similarity measures are aggregated or chained (using the transitivity of similarity) according to well-defined operators called T-norms, Averaging Operators, and T-conorms [17], [31], [32], [6]. In CARS, five uncertainty calculi based on the five T-norms are used. These five calculi were selected because they could provide the ability to choose the desired uncertainty calculus starting from the most conservative to the most liberal.

The similarity mechanism aggregates similarities by taking as input a list of similarities to be combined, their associated uncertainties, and optional relevance weights indicating the importance of the feature in the aggregation. This mechanism is based on three aggregation operators: *T-norms*, *T-conorms*, and *Linear combinations*.

First we aggregate the low and high values of similar-

ity: low values are aggregated using the minimum operator (with the option of using other T-norms), while high values are aggregated using the maximum operator (with the option of using other T-conorms). The result of these partial aggregations (multiplied by the cardinality of the aggregated values) is averaged with the intermediate values of similarity.

The process normalizes the similarity values of various abstract features according to their relevance weights. Then the process penalizes bad matches and rewards good ones. Finally, the process considers tradeoffs by averaging the remaining intermediate values with the previous results.

The overall similarity between two cases is computed by aggregating the *five phase similarities*. This final aggregation can be customized to reflect different goals of the retriever, as we will show in the next paragraphs. Let us recall the definition of the five phases: (1) Initial Condition, (2) Pre-tender, (3) Tender-negotiation, (4) Outcome and (5) Long-term results.

For instance, by only aggregating phases 1, 2, and 4 we can stress the need to find successful cases with similar initial and pre-tender conditions. The result can give us the range of tender-negotiations (plans/counter-plans) which are applicable to the current situation.

Alternatively, by only aggregating phases 2, 3 and 4 we can observe the range of macro-economic conditions to ascertain raider/target financial assessments, for which a particular (pre-tender and post-tender) plan was successful.

5 Modification of a past solution

Once the most appropriate past case is selected, then it's solution is modified for the current situation. This modification can be done in various ways. MASTER [3] uses rules and domain knowledge for modification. In CARS, we use rules and also other cases in the case base. In this section we will give a simplified example of how sub-solutions in other cases are used in modifying the retrieved solution.

In our M&A case memory, we store the conceptual knowledge about objects, actions, and goals along with their connecting causal information. In the following example, we will illustrate the use of a simplified version of one of our adaptation techniques that uses this causal information for modifying the solution.

Suppose the goal of the raider is to takeover a company. This goal can be accomplished by acquiring the company with or without the management's approval. The goal of the retrieved case, with the highest similarity to the current case, was to acquire the company without management's approval. We will first try to adapt the plan of this retrieved case to get a plan for achieving the current goal. One of the actions in this plan is to make a `tender-offer`. However, the precondition of this action, which requires that `available-funds` of the agent should be greater than the total price of shares sought, fails during plan adaptation. One way of recovering from this failure is by checking how such failure was resolved in other cases. The simplified procedure for finding out how other cases resolved the failure of a precondition of some action a_i is given in Figure 6.

In our example, the action whose precondition failed is `tender-offer`. The description of how the procedure in Figure 6 is followed to determine which action to use for resolving the failure in the precondition of tender offer is given in Figure 7.

The action (`get-credit`) is chosen using this procedure. This action is then included in the plan. The execution of this action will increase the `available-funds` of the agent and thus the precondition for `tender-offer` action can be satisfied.

This example also illustrates how we can use case-based reasoning to support an expert decision maker considering various alternatives in a complex situation.

6 Summary

We have stored M&A cases as situation/solution pair. The situation consists of the top-level goals and starting states of the agents. The solution consists of the representation of the observable portions of the executions of actions by the agents and their effects on the states of world.

The M&A domain knowledge is represented as rules and conceptual knowledge in the system. The conceptual knowledge represents the information about the objects, events and goals in the domain. The rules capture the available general knowledge about the areas relevant to M&A, in this case financial knowledge. Using these two types of knowledge, the description of cases in the case base is augmented by abstract features derived from the cases' observed (surface) features.

To retrieve a solution for the current situation represented by the probe case, the probe and retrieved cases are first compared along their abstract features, generating a linguistic value representing their corresponding similarity measure. These similarity measures are then aggregated, according to the needs of the reasoner. This aggregation, based on T-norms, Averaging Operators, and T-conorms, generates a partial ordering on the retrieved cases.

The solution of the most similar case is then modified for the current situation. One of the modification techniques uses the sub-solutions of other cases in the case base. To overcome a problem in the retrieved solution, it considers the cases with similar sub-problem. The selection of the most appropriate solution is then based on the certainty of its success in the past case.

1. **Events Retrieval:** retrieve from the case library the events that have the same action a_i. The retrieval is done using the instantiation link between the action a_i (in the action hierarchy) and the events.
2. **Events Filtering:** select the events with enabled links.
3. **States Filtering:** obtain the enabling states of the selected events, and only maintain those states whose changed feature values affected the failed pre-condition.
4. **Actions Filtering:** select the actions of the events that caused the above states. These actions are the ones that solved the failing pre-condition problem in previous cases.
5. **Action Selection:** choose the action with the highest degree of certainty in its causal link.

Figure 6: Procedure for selecting an action to resolve a precondition failure

1. **Events Retrieval:** using the instantiation link of `tender-offer` from the action hierarchy, the events with tender offer action are retrieved. A subset of these events is:
   `kpm-e-tender-offer-pp-01` from Kraft-Philip-Morris case,
   `mmb-e-tender-offer-ut-01` from Martin-Marietta-Bendix case,
   `wdr-e-tender-offer-rel-01` from Walt-Disney-Reliance case,
   `pbc-e-tender-offer-kal-01` from Pabst-Brewing-Company case,
   `pbc-e-tender-offer-jmsl-01` from Pabst-Brewing-Company case.
2. **Events Filtering:** the events `wdr-e-tender-offer-rel-01`, `pbc-e-tender-offer-kal-01` and `pbc-e-tender-offer-jmsl-01` have enable links.
3: **States Filtering:** The three enabling states corresponding to the selected events are: `state-wdr-12`, `state-pbc-5`, and `state-pbc-7`. The changed feature values in these enabling states are:
   `(finance-committed Reliance)` $\leftarrow$ `20.5` for event `wdr-e-tender-offer-rel-01`,
   `(cash Kalman)` $\leftarrow$ `9.3` for event `pbc-e-tender-offer-kal-01`,
   `(with-in-law JMSL)` $\leftarrow$ `:YES` for event `pbc-e-tender-offer-jmsl-01`.
   Among these changed feature values, the feature `cash` and `finance-committed` affect the `available-funds`. Thus the states `state-wdr-12` and `state-pbc-5` are selected.
4. **Actions Filtering:** State `state-wdr-12` was caused by event `wdr-e-get-investors-rel-01` (generated by the `get-investor` action) with certainty `*most-likely*`. State `state-pbc-5` was caused by event `pbc-e-get-credit-kal-01` (generated by the `get-credit` action) with certainty `*certain*`.
5. **Action Selection:** Choose action `get-credit` (derived from event `pbc-e-get-credit-kal-01`).

Figure 7: Selecting an action to resolve a precondition failure in action `Tender-offer`

We have developed a hybrid system (CARS) that combines two reasoning methodologies for planning in the domain of Mergers and Acquisitions. We have used rules to model the available knowledge, and cases to supplement the sparse coverage of the rules. We have given an example of the use RBR in augmenting the description of cases using the domain knowledge of financial analysts. We have shown how this augmentation aided in determining the most appropriate case. We have also shown the use of CBR for adapting the retrieved solution.

References

[1] J. K. Aragones, P. P. Bonissone, and J. Stillman. Primo: A tool for reasoning with incomplete and uncertain information. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-90)*, July 1990.

[2] Saad Ayub. *Planning in an Uncertain and Dynamic Environment with Weak Domain Theory.* PhD thesis, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 1992.

[3] Gary Berg-Cross and Linh Claudio. Master: A case based-design to augment corporate mergers and acquisitions decisions. In *First International Conference on Artificial Intelligence Applications on Wall Street*, pages 188–193, April 1992.

[4] Piero Bonissone, Lauren Blau, and Saad Ayub. Leveraging the integration of approximate reasoning systems. In *Proceedings of the 1990 AAAI Spring Symposium Series, Symposium: Case-Based Reasoning*, pages 1–6. AAAI, March 1990.

[5] Piero P. Bonissone. A fuzzy sets based linguistic approach: Theory and applications. In M.M. Gupta and E. Sanchez, editors, *Approxi-*

mate Reasoning in Decision Analysis, pages 329–339. North Holland Publishing Co., New York, 1982.

[6] Piero P. Bonissone. Summarizing and propagating uncertain information with triangular norms. *International Journal of Approximate Reasoning*, 1(1):71–101, January 1987.

[7] Piero P. Bonissone. Now that I Have a Good Theory of Uncertainty, What Else Do I Need? In *Proceedings Fifth AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 22–33. AAAI, August 1989.

[8] Piero P. Bonissone and Saad Ayub. Representing cases and rules in plausible reasoning systems. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, November 1992.

[9] Piero P. Bonissone and Saad Ayub. Similarity measures for case-based reasoning systems. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-92)*, pages 483–487, July 1992.

[10] Piero P. Bonissone and Keith S. Decker. Selecting uncertainty calculi and granularity: An experiment in trading-off precision and complexity. In L. Kanal and J. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 217–247. North Holland, Amsterdam, 1986.

[11] Piero P. Bonissone and Soumitra Dutta. Mars: A mergers and acquisitions reasoning system. *Journal of Computer Science In Economics and Management*, 3:239–268, 1990.

[12] L. K. Branting. Exploiting the complimentarity of rules and precedents with reciprocity and fairness. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 39–50, San Mateo, CA, May 1991. Morgan Kaufmann Publishers, Inc.

[13] M.S. Braverman and R. Wilensky. Towards an unification of case-based reasoning and explanation-based learning. In *Proceedings of the 1990 AAAI Spring Symposium Series, Symposium: Case-Based Reasoning*, pages 80–84. AAAI, 1990.

[14] Jaime Carbonell and Manuela Veloso. Integrating derivational analogy into a general problem solving architecture. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 104–124, San Mateo, Ca, May 1988. Morgan Kaufmann Publishers, Inc.

[15] D. Dubois and H. Prade. Fuzzy real algebra. *Fuzzy Sets and Systems*, 2(4):327–348, 1979.

[16] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, New York, 1980.

[17] D. Dubois and H. Prade. Criteria aggregation and ranking of alternatives in the frame-work of fuzzy set theory. In H. Zimmermann, Lofti Zadeh, and B. Gaines, editors, *Fuzzy Sets and Decision Analysis*, pages 209–240. North-Holland, Amsterdam, Holland, 1984.

[18] S. Dutta and P.P. Bonissone. An approach to integrating diverse reasoning techniques. In *In Proc. Avignon-90, Conf. on 2nd Generation Expert Systems*, 1990.

[19] S. Dutta and P.P. Bonissone. Integrating case based and rule based reasoning: the possibilistic connection. In *Uncertainty In Artificial Intelligence - Vol. 6.* North Holland, Amsterdam, 1991.

[20] A. K. Goel. Grounding case modification in deep models. In *Proceedings of the 1990 AAAI Spring Symposium Series, Symposium: Case-Based Reasoning*, pages 41–44. AAAI, 1990.

[21] Andrew Golding and Paul Rosenbloom. Integrating rule-based and case-based reasoning for name pronunciation. In *Proceedings of the National Conference on Artificial Intelligence*, Menlo Park, August 1991. AAAI, AAAI Press.

[22] Kristian Hammond and Neil Hurwitz. Extracting diagnostic features from explanations. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 169–178, San Mateo, Ca, May 1988. Morgan Kaufmann Publishers, Inc.

[23] Phyllis Koton. Reasoning about evidence in causal explanations. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 260–270, San Mateo, CA, May 1988. DARPA, Morgan Kaufmann Publishers, Inc.

[24] A. Oskamp, R.F. Walker, J.A. Schrickx, and P.H. Vanden Berg. Prolexs, divide and rule: A legal application. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, 1989.

[25] Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[26] Edwina L. Rissland, Chumki Basu, Jody J. Daniels, Joseph Mccarthy, Zachary B. Rubinstein, and David B. Skalak. A blackboard architecture for cbr: An initial report. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 77–92, San Mateo, Ca, May 1991. Morgan Kaufmann Publishers, Inc.

[27] Edwina L. Rissland and David B. Skalak. Combining case-based and rule-based reasoning: A heuristic approach. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, San Mateo, Ca, August 1989. Morgan Kaufmann Publishers, Inc.

[28] Edwina L. Rissland and David B. Skalak. Interpreting statuatory predicates. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 46–53, 1989.

[29] Enrique H. Ruspin. On the semantics of fuzzy logic. *International Journal of Approximate Reasoning*, 5(1):45–88, January 1991.

[30] Elie Sanchez. Inverse of a fuzzy relations. applications to possibility distributions and medical diagnosis. *Fuzzy Sets and Systems*, 2(1):75–86, 1979.

[31] B. Schweizer and A. Sklar. Associative functions and abstract semi-groups. *Publicationes Mathematicae Debrecen*, 10:69–81, 1963.

[32] B. Schweizer and A. Sklar. *Probabilistic Metric Spaces.* North Holland, New York, 1983.

[33] Manuela Veloso and Jaime Carbonell. Variable-precision case retrieval in analogical problem solving. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 93–106, San Mateo, Ca, May 1991. Morgan Kaufmann Publishers, Inc.

[34] Lofti A. Zadeh. Similarity relations and fuzzy orderings. *Information Science*, 3:177–200, 1971.

[35] Lofti A. Zadeh. A computational theory of disposition. In *Proceedings of 1984 International Conference on Computational Linguistics*, pages 312–318, 1984.

A Knowledge-Based System for Financial Statement Analysis

Gerald J. Stuzin
St. John's University
Jamaica, NY 11439

Abstract

Financial statement analysis belongs to a class of decision-making activities that, in practice, requires the integration of analytical methods and expert judgment. This paper describes an approach to such an integration that combines several methodologies: knowledge-based tuning of analytical models; object-oriented logic programming; decision analysis.

1. Introduction

For the purpose of this paper, financial statement analysis is an exercise in utilizing historical financial data for one or more firms in order to predict the financial performance of a firm or to do asset valuation of a firm. As pointed out by H. Guggenheimer [10], such activities are "... extrapolation and as such (by the underlying mathematics) [are] ... unsafe and exponentially unstable." The activities are not without practical value, however, and it is common to exploit that value by combining methods of extrapolation and the subjective knowledge of experts. This of course is not a simple task. There are a number of competing analytical models for extrapolation and experts do not usually agree. In other words, financial statement analysis is a decision process involving several analytical models and the subjective input of several individuals. The author has designed and built a small prototype of a system intended to mitigate the resulting complexities.

2. Tuning Analytical Models

It has been pointed out by many economic and financial forecasters that mathematical models by themselves cannot consistently produce good forecasts. A blend of expert judgment and the output of analytical models is often suggested as the best forecasting method (see, e.g., ref. [9]). In macroeconometric forecasting, for example, it is common practice for a forecaster, after viewing a model's output, to alter the constant terms on the equations of the model in order to produce results he judges to be acceptable. Acceptability usually requires that the model's output agree with the analyst's a priori views on certain key points. As the analyst alters the model to reflect his own views and sees the results, he may alter his views. A process of adjusting the model and subsequent modification of views may go through a number of iterations before an analyst is satisfied with a forecast. The process is sometimes referred to as "tuning" the model. The author and a colleague have reported on a knowledge-based methodology (called TUNES) for tuning analytical models that expedites the tuning process in several important ways:

1) It allows the analyst to benefit from expert tuning rules stored in a knowledge-base. The rules offer suggestions for tuning the model and prevent inconsistent or nonsensical input.
2) It permits an analyst to enter subjective information linguistically (from menus), and it is converted into a quantitative adjustment to the model.
3) A history of alterations to the model is maintained and a forecast can "explain" itself by showing both the subjective input and the resulting quantitative adjustments that were used.

The TUNES combination of a knowledge-based system and an analytical model has similarities to model-based reasoning systems, which have been described by Alan J. Rowe [2] as "...expert systems that include models of domain objects...". There is a difference in priority, however. The TUNES system uses an expert system to

impact on a model; model-based reasoning uses a model to supply knowledge for an expert system.

2.1 The TUNES System

The purpose of TUNES is to facilitate and rationalize the tuning of models by utilizing knowledge-based methods. The knowledge base of TUNES contains the facts and rules that an "expert" would use for tuning a model. It also contains the rules for accepting new facts and subjective evaluation of a decision-maker and translating them into appropriate adjustments to the model. The TUNES system collaborates in the tuning process: it indicates to the decision-maker what information an expert would consider important; it obtains the decision-makers's input linguistically by means of menus; it compares user behavior to expert behavior and attempts to minimize the difference by "negotiating" with the user. The final output of the TUNES system is an adjusted model decision and the "reasoning" behind it (i.e., a review of the tuning rules, facts and subjective input that were employed).

The significant advantage of the TUNES approach are:

1) The utilization of the knowledge-base techniques makes it easier and faster to produce decisions (and to consider alternatives).

2) There is greater objectivity. The reasoning behind a decision is explicit, more open for critical inspection. It is easier to do a systematic evaluation of errors and improve performance over time. A discipline is exerted on the decision-making process which is otherwise lacking when judgment and qualitative analysis are combined. There is less chance of psychological bias, which can affect the judgment of even the best decision-maker.

3) TUNES allows the decision-maker to enter input using words (selected from menus) rather than numbers. This is convenient and, more important, it has been found that a higher degree of consistency in making subjective appraisals is obtained by using verbal input.

2.2 Knowledge Representation in Tuning

Two major issues must be resolved in representing the expert's knowledge (tuning knowledge) and the decision-maker's knowledge (subjective input) required by the tuning process:

1) How to represent input and output information.

2) How to represent facts and rules.

The underlying assumption in tuning a model is that the decision-maker has useful, albeit imprecise, knowledge that the model does not have. The imprecision of the knowledge is a deterrent to the use of numerical values to acquire the decision-makers's input, the fact that the model's output is numerical notwithstanding. Input using linguistic variables, i.e., variables whose values are words rather than numbers [4], is more natural for the analyst and, as indicated above, produces a higher degree of response consistency. Similarly, a decision-maker can understand expert knowledge (guidance) better when it is expressed linguistically. The TUNES system, therefore, uses linguistic variables for both input and expert knowledge, employing a specially devised methodology for computing with them [3].

The problem of representing the facts and rules of the knowledge base was solved by expressing them as statements in logic (using Prolog). This choice offered a number of advantages:

1) Knowledge consisting of a fact representing a relationship between variables can be easily represented by a unit clause in logic. Logic programming is convenient for deducing the logical consequences of such clauses and asking for conclusions in a straightforward manner.

2) It is easy to mix the linguistic values of the knowledge base and the numerical values of the models, since logic can express functions to transform from one to the other. This is done by utilizing a combination of the declarative and procedural interpretation of logic.

3) Prolog makes it easy to add or delete knowledge interactively.

2.3 Financial Statement Analysis and Tuning

Using the definition of financial statement analysis given above, the methods employed fall into three broad categories:
1) Trend analysis: time series analysis and econometric methods relating the firm's performance to the external economy.
2) Ratio analysis: the analysis of relationships within the firm.
3) Cross sectional analysis: the comparison of financial numbers for similar firms.

All three of the methods above employ mathematical relationships to which the TUNES methodology can be applied. In fact, the trend analysis models are exact analogues of the model around which TUNES was built.

3. Examining Alternative Decisions

The easiest (not the best) way to consider alternative decisions is sequentially, i.e., one at a time. That is the way tuning exercises have traditionally been done, and the method used in TUNES. In a series of experiments, a forecaster (decision-maker) examines forecasts, modifies his input and obtains a new forecast. There is only one forecast at any point in time. If the decision-maker wishes to resume work with a previous forecast he must "back-up", withdrawing input (if he remembers what it was) or start again from the beginning. It is much more efficient to develop alternatives in parallel and interact with them simultaneously. At any point in developing a solution, it should be possible to split the process into alternatives, each alternative inheriting the input of it's "parent", giving rise to a tree structure of alternative solutions. Furthermore, the decision-maker should be able to view the results of any alternative at any time and to switch to any alternative and continue development of the solution from that point. Such a capability is built into an extension of the TUNES system called Objective TUNES by using object-oriented design within a logic program.

Given the capability described above for a single decision-maker to interact with the analytical model, it is natural and straight-forward to extend it to a number of decision-makers. Each decision-maker is given access to the alternatives developed by the others by sharing the same knowledge base. Each decision-maker can view the alternatives developed by the others, see the reasoning behind them (the alternatives can "explain" themselves), and utilize any of them he wants to in his own work.

3.1 Objective TUNES

Although there are several variations of the object-oriented paradigm, the concepts of an "object" and of "inheritance", which are used by Objective TUNES, are common to all of them.

Associated with an object are data and procedures that define the behavior of the object. Thus, an object encapsulates both state and behavior (see [5]). In Objective TUNES, the objects are alternative model solutions. The state of the object is the set of values of the linguistic variables (input). The behavior of the object is determined by the facts and rules in the Objective TUNES knowledge base and in the equations of the analytical model.

Inheritance is a relationship between objects that allows for the definition of one object to be based on another object (or several objects, in some variations of the paradigm). In the Objective TUNES system, alternative model solutions inherit the knowledge of their "parent" solution. Additional knowledge, from the decision-maker, distinguishes them from their parent and from "siblings". In a trial and error exercise, rather than deleting information from the current solution, the decision-maker switches to a prior solution and proceeds from there. If he changes his mind, he can return to an abandoned solution. Most important, each solution has all the information for explaining itself at any time.

A method for implementing object-oriented programming in Prolog was shown by E. P. Stabler [6] and is used in Objective TUNES.

4. Forecast Synthesis

Given a tree of alternative solutions (forecasts) as described above, there are a number of possible ways of choosing the "best" solution. The decision-maker (or "chief" decision-maker if there are several) can simply select according to his own judgment. Early versions of the Objective TUNES system assumed that approach would be used. A more rational approach, and easily implemented with the system is to apply decision analysis. At each branch point in the tree, the subjective probability of each alternative solution is recorded, i.e., the probability becomes part of the object. A branch point could result from alternative analytical models, alternative analysts, or alternative subjective assessments. The ultimate solution is simply the expected value, using the usual decision analysis computations. The subjective probabilities can easily be altered in order to observe the effect on the ultimate solution.

5. Conclusion

A system has been described that allows an analyst or several cooperating analysts to collaborate with mathematical models for financial statement analysis. The system's knowledge base contains expert rules for guiding an analyst in interacting with the model, i.e., entering subjective information. Logic programming is used for knowledge representation. Object-oriented programming is employed to make it easy to perform experiments in parallel and to share information among analysts. A system feature is the ability to utilize decision analysis to synthesize an ultimate solution from a number of alternatives. Experience in building a small prototype system, employing cross sectional and trend analysis models suggests that the methodology is adequate to accommodate practical exercises in financial statement analysis.

References

[1] G. J. Stuzin, "An Expert System for Tuning a Macroeconometric Model", ACM CSC Proceedings, 1987.

[2] Alan J. Rowe, "Heuristics, Judgment and Model-Based Reasoning", Heuristics, Summer/Fall, 1990.

[3] R. S. Freedman and G. J. Stuzin, "A Knowledge-Based Methodology for Tuning Analytical Models", IEE Transactions on Systems, Man and Cybernetics, March/April, 1991.

[4] L. A. Zadeh, Foreword to Fuzzy Sets, Natural Language Computations and Risk Analysis, Schmucker, K. J., Computer Science Press, 1984.

[5] T. Korson and J. D. McGregor, "Understanding Object-Oriented Design", Communications of the ACM, vol. 33, no. 9, Sept. 1990.

[6] E. P. Stabler, Jr., "Object-Oriented Programming in Prolog", AI Expert, Oct. 1986.

[7] Baruch Lev, Financial Statement Analysis, Prentice-Hall, 1974.

[8] George Foster, Financial Statement Analysis, Prentice-Hall, 1978.

[9] Alan Greenspan, "Economic Forecasting in the Private and Public Sectors", Business Economics, Jan. 1991.

[10] H. Guggenheimer, "Predicting Short and Long Term Financial Trends", First International Conference on Artificial Intelligence on Wall Street, Oct. 1991.

[11] Howard Raiffa, Decision Analysis, Addison-Wesley, 1968.

An Expert System for Credit Evaluation

Oscar Castillo
UABC - University Tijuana *

Patricia Melin
CETYS Tijuana

P.O.Box 4207,Chula Vista,CA 91909

Abstract

We describe a program written in PROLOG that can be consider an expert system for a domain of economics. The program is a simulator of a human banking expert in the process of credit evaluation. The program contains the knowledge of the human experts expressed as rules and facts (knowledge base), and uses as a tool to evaluate the requests for credit the Inference Engine of PROLOG.

The system is a modification and extension of a prototype developed by Ben-David and Sterling (1985).

1. Introduction

We describe a program written in PROLOG that can be considered an expert system for a domain of economics. The domain is exactly the part of the financial world corresponding to the problem of credit evaluation. We all know that the financial world is too complicated to be expressed only with the numbers and ratios constantly being calculated. In order to make judgments, human experts in the financial domain tend to think in qualitative terms with which they are more confortable. To echo expert reasoning then, we have to make models that simulate this qualitative reasoning. In the system that we developed the qualitative reasoning of human banking experts of Mexico is modeled using rules and facts of PROLOG (knowledge base). The Inference Engine of PROLOG also does the evaluation of credit requests.

The system evaluates the data from the requests for credit to obtain a decision about giving the credit or not. The data that the system can use is the following:

- client's collateral
- client's financial records
- current gross profits on sales
- expected yield to the bank

We suggest that the expert system can be used as a tool to the human banker to make more efficient his credit evaluation work.

The main modifications and extensions done in this system (with respect to the Ben-David and Sterling's system) are in the way of building the knowledge base to model the reasoning of banking experts of Mexico. Of course this was done for a particular bank of Mexico, because we all know that not all the banks have the sames policies, but we think that the model has some of the characteristics of Mexican bankers.

* Work partially supported by CONACYT under grant 0784-S9110

The development of the expert system was done in a personal computer 80386 using the programming language Arity Prolog 6.00.86 and the XShell tool of Covington, Nute, Vellino [2].

2. Description of The Expert System

A shell is a program that helps build the knowledge base for a given expert system. Other way of saying this, is that a shell is an expert system with no knowledge base.

For the development of this expert system we use the shell of Covington, Nute, Vellino [2]. This shell soports the following

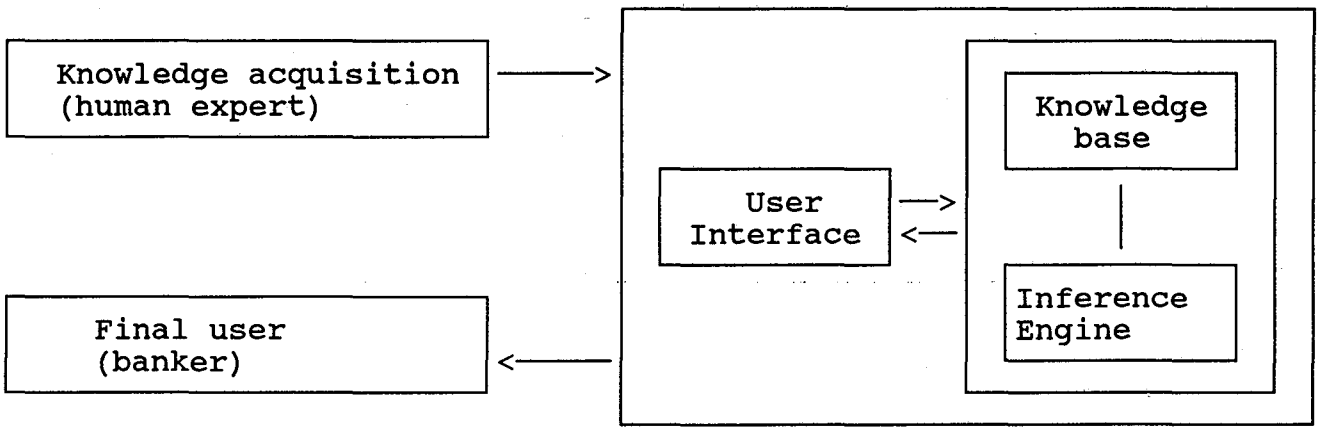


Figure 1.- General architecture of the expert system

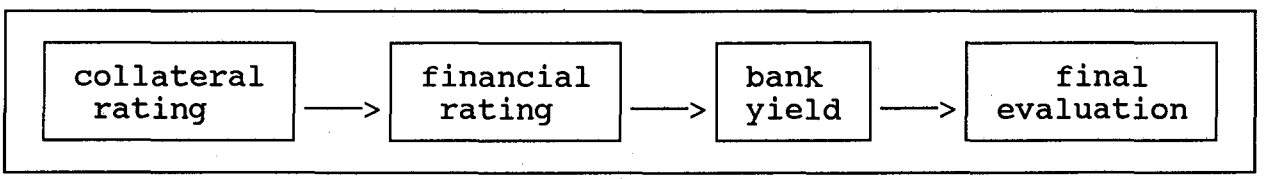


Figure 2.- Architecture of the knowledge base

2.1 Architecture of the Expert System

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

In figure 2 we describe the architecture of the knowledge base.

2.2 Shell for the Expert System

architecture:

XShell: user interface
Prolog: inference engine
Knowledge base: separate set of clauses for some special predicates the xshell routines use

The idea behind XSHELL is that it provides the auxiliary procedures we need to derive a consultation based on a

declaratively developed knowledge base. Diagnosis and prescription are really just two different kinds of identification. Diagnosis is the identification of a problem, and prescription is the identification of a solution to a problem. XSHELL makes one or more identifications during a consultation by using information in its knowledge base and information provided by the user. While identification is the central task for XSHELL, it must also do several other jobs during a consultation. Xshell does the following jobs:

1.- Tell the user what the expert system does and how it is used.
2.- Make an identification, remember it, report it, and (if necessary) explain it.
3.- If more than one identification is called for, keep making identifications until no more can be made.
4.- End the consultation smoothly and ask the user if another consultation is wanted. If so, begin it. If not, stop.

An XSHELL knowledge base will have seven things in it:

1.- An introductory statement (xkb_intro).
2.- A report phrase (xkb_report).
3.- xkb_unique (yes) or xkb_unique (no).
4.- xkb_explain (yes) or xkb_explain (no).
5.- A set of identification rules.
6.- A set of questions for the properties and parameters used in the identification rules.
7.- A starting query (:- xshell. or ?-xshell.) to begin the consultation when the knowledge base is loaded.

2.3 Description of the knowledge base

To evaluate requests for credit we consider the following factors:

a) Collateral rating
b) Financial rating
c) Expected bank yield

We will describe briefly each of these factors in the following lines.

a) Collateral rating

The various types of collateral are divided into categories. Currency deposits, whether local or foreign, are "first-class" collateral. Stocks are examples of "second-class" collateral, while the collateral provided by mortgages and the likes is regarded as "illiquid". The formulas we use are described in figure 3.

$$\text{first class} = \frac{\text{sum of deposits in banks X 100}}{\text{quantity requested}}$$

$$\text{second class} = \frac{\text{negotiate instruments X 100}}{\text{quantity requested}}$$

$$\text{Illiquid} = \frac{\text{sum of mortgages x 100}}{\text{quantity requested}}$$

Figure 3.- Formulas for collateral.

The rules that we apply to obtain the collateral rating are the following:

1) If first class >=100
   THEN the rating is "EXCELLENT"
2) If first class >70 AND first class + second class >=100
   THEN the rating is "EXCELLENT"
3) If first class + second class >60 AND first class + second class <70 AND first class + second class + illiquid >=100
   THEN the rating is "GOOD"

b) Financial rating

In the financial rating we consider the financial factors of the client and calculate a weighted sum of this factors. The weights are assigned according to the policies of the bank. So we have the formula:

client's score = factor1 x weight1 + factor2 x weight2 + ... + factorn x weightn

The rules that we apply to obtain the financial rating are in figure 4.

c) Bank Yield

In the expected bank yield we are considering that it is calculated by another program that gives as a result to our program one of three possibilities:

* excellent yield
* reasonable yield
* poor yield

Once we have the collateral rating, the financial rating and the bank yield, we can proceed to determine if we are going to give or not the credit. In making the decision about giving the credit we use the rules of the figure 5.

```
1) If score <=-500
   THEN financial rating "BAD"
2) If score >-500 AND score <150
   THEN financial rating "MEDIUM"
3) If score >=150 AND score <1000
   THEN financial rating "GOOD"
4) If score >=1000
   THEN financial rating "EXCELLENT"
```

Figure 4.- Rules of the financial rating.

```
1) If collateral rating "EXCELLENT" AND financial rating    "GOOD"
OR "EXCELLENT" AND Bank yield "REASONABLE" OR    "EXCELLENT"
   THEN the result is "GIVE CREDIT"
2) If collateral rating "GOOD" AND
   financial rating "GOOD" AND
   Bank yield "REASONABLE"
   THEN the result is "CONSULT SUPERIOR"
3) If collateral rating "GOOD" AND financial rating    "MEDIUM"
   THEN the result is "REFUSE CREDIT"
```

Figure 5.- Rules for the final decision.

3. Use and Validation of the System

We have been making tests of the performance of the system with real data and works well for the cases that we considered. We show a sample input-output to illustrate how it works in figure 6.

```
C> api
?- consult(credit.ari).
yes
CREDIT: AN EXPERT SYSTEM FOR CREDIT EVALUATION
Amount requested for credit?
> 50,000
Amount of local currency deposits?
> 30,000
Amount of foreign currency deposits?
> 20,000
Amount in negotiate instruments?
> 8,000
Amount in mortgages?
> 15,000
What is the net worth per assets?
> 40
What is last year sales growth?
> 20
What is the gross profit on sales?
> 45
What is the short term debt per annual sales?
> 10
What is the expected bank yield?
   (e) excellent
   (r) reasonable
   (p) poor
> e
RESULT: GIVE CREDIT
Do you want to conduct another consultation?
> n
yes
?- halt.
C>
```

Figure 6.- Sample Input-Output of the program.

4. Performance of the System

Space and time considerations were one factor that led to the choice of PROLOG as an implementation language. The PROLOG we chose has a compiler that produces reasonably fast code. The backtracking control structure of PROLOG provides a natural implementation of rule

based reasoning. There are several Prologs available for the IBM PC. After working with some of this, we finally selected ARITY PROLOG because it has a good compiler and has the ability to interface easily with C and Pascal. Another feature of this PROLOG is a nice interface to windows, which can be used to incorporate a menu-driven user interface into the program very easily.

The performance of the system at the present time is very good The credit evaluation done by the system on real data gave results that match pretty well the real decisions of the human bankers. The real time performance is also satisfactory the expert system takes almost no time in giving the decisions. Our intention is to use the expert system in the banking environment as an advisor, following its recommendations closely except in situations that can be considered outside the scope of the system by the human experts.

5. Conclusions

We have developed an expert system for credit evaluation in a banking environment. The system works well for the domain of the financial world corresponding to credit evaluation for a particular bank in Mexico. This system can be easily modified to serve the needs of other banks because of the modularity of the design.

The expert system can be improved in the following directions:

1.- Make a better interface with the user to make it easier to use.
2.- Provide the module to calculate the expected bank yield.
3.- Generalize the system so that it can make evaluations but not only for credits.

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

6. References

1.- Ben-David, A. and Sterling, L. "A Prototype Expert System for Credit Evaluation" Proc. International Workshop on Artificial Intelligence in Economics and Management, Zurich, Switzerland, 1985.
2.- Covington, M.A., Nute, D., Vellino, A., "Prolog Programming in Depth",Scott Foresman and Co. Computer Books, 1988.
3.- Sterling, L., Shapiro, E., "The Art of Prolog", MIT Press Cambridge Mass., 1987.

Using AI Technology for Technology Transfer

Patrick J. Lyons, Ph.D., Thomas Abraham, Ph.D.,
Larry W. Boone, Ph.D., and Brenda Massetti, Ph.D.
Department of Management
St. John's University
Jamaica, New York 11439 USA

Abstract

Organizations can facilitate the process of technology transfer by encouraging organizational members to suggest ideas that may not be practical today, but have a reasonable likelihood of implementation in the not too distant future. This paper describes a computerized system whose purpose is to help organizations become aware of the innovative ideas of their organizational members. Specifically, it describes a knowledge-based system to store, sort, evaluate, and make readily available to problem solvers both old and new ideas produced through employee suggestion programs and from other sources.

1. Introduction

The innovation process in financial services typically occurs in five stages: Awareness, Interest, Evaluation, Trial, Adoption [Goldsmith, 1992]. The first stage, Awareness, is costly and labor intensive. This paper describes a computerized system whose purpose is to help organizations become aware of the innovative ideas of their organizational members. Specifically, it describes a knowledge-based system to store, sort, evaluate, and make readily available to problem solvers both old and new ideas produced through employee suggestion programs and from other sources.

The distinguishing characteristic of this system is that it explicitly encourages people to suggest ideas that may not be practical today, but have a reasonable likelihood of implementation in the not too distant future. As will be described later, in addition to entering ideas, the submitter enters explicit limiting factors. These limiting factors facilitate technology transfer, because they help identify what emerging technologies would have the most impact on the organization's operations.

1.1 Traditional Suggestion Systems

One traditional avenue organizations have taken to enhance their creativity and innovation is the suggestion system. Generically, such systems entail some formal mechanism for organizational members to provide either tangible ideas which are intended to result in measurable profitability increases, or intangible ideas which are intended to produce improvements to such concerns as working conditions, employee safety or public relations [Moore, 1988; Burnette and Bousum, 1989]. Depending on the specifics of a given suggestion system, organizational members are rewarded with prizes, financial payments, or merit recognition awards when their ideas are accepted. Decisions concerning which ideas to implement are typically made by a committee which meets periodically to review all of the ideas submitted since its prior meeting [Lo Bosco, 1985]. Accepted ideas are turned into implemented projects, while rejected ideas are typically stored until a similar suggestion re-awakens interest.

The success of suggestion systems as tools for providing organizations with creative ideas has varied. Many find the ideas and suggestions employees provide to be very valuable [Stakel, 1988]. For example, Kodak [Allen, 1987], Hughes Aircraft [Stakel, 1989], Chase Manhattan [Scicutella, 1988], and Harleysville Insurance [Buckwalter, 1991], just to name a few, all claim substantial financial benefits in the millions of dollars annually from their suggestion programs. Moreover, these organizations also note significant non-monetary advantages including improved communication, employee safety, problem-solving ability, worker morale, productivity, and labor/management relations as a result of their

suggestion programs.

Other companies, however, have been disappointed by their experiences with suggestion programs. Rather than praise subsequent success, these organizations claim expensive administrative time consumed [Phillips and Wallace, 1988; NAPL], fragmented ideas, limited input from employees, [Crosby, 1986; Zakeski, 1988], and decreased morale when awards for ideas are not perceived as fair [Moore, 1988; NAPL].

In order to alleviate the difficulties found in these systems and provide a basis for organizations wishing to implement suggestion systems, a variety of researchers [Lo Bosco, 1985; Crosby, 1986; Allen, 1987; Moore, 1988; Scicutella, 1988; Phillips and Wallace, 1988; Stakel, 1989; Buckwalter, 1991] examining over 6,000 organizations have repeatedly uncovered the following critical success factors for suggestion programs:

- Management Support at all Organizational Levels,
- Timely and Responsible Idea Processing,
- Regular Program Awareness Promotion, and
- Clearly Established Award Systems.

Considering the significant benefits noted from the many organizations successfully using employee suggestion systems, and realizing the increased need for creativity and innovation by all organizations as they face the dynamic, turbulent, globally-based economic climate of the 1990s, devising means to improve a given organization's current success rate with employee suggestion systems should be an ongoing priority.

1.2 Computerized Suggestion Systems

One such effort involves the use of computer-based suggestion systems [Stackel, 1989]. These systems typically entail entering paper-based suggestions made by employees into a computerized database, which stores the ideas and then keeps track of awards given and successes incurred for implemented ideas [Johannes, 1992]. Such a database certainly increases the efficiency with which an idea can be processed and/or eventually tracked. It also provides a convenient means for storing and sorting suggestions. Moreover, it can provide a straightforward mechanism to perform analyses and produce reports concerning the general value of the overall suggestion system.

However, the domain of these suggestion databases typically rests with the personnel in charge of overseeing the suggestion system. Computerization is used to make more efficient the jobs of those who have the responsibility for storing, choosing, and tracking a large number of suggestions. Hence, only those individuals within the suggestion system function have general access to the suggestion database [Johannes, 1992]. Unfortunately, it is extremely difficult for any one group to be constantly aware of all of the challenges and opportunities an organization is, or may soon be, facing. Moreover, it is even more difficult for any one group to be constantly aware of how all the ideas might be applied to meet these ever-present and changing challenges and opportunities. So, while computerized suggestion systems have currently made the collection of ideas more efficient and convenient, there is still more that can be done to enhance their value. The knowledge-based system described in this paper represents an initial step in improving the effectiveness of suggestion systems.

1.3 Overview

In Section 2 of this paper, the design requirements for a computerized suggestion system are discussed. The two primary concerns are that limiting factors can be associated explicitly with ideas and that groups of related ideas can be easily retrieved. Section 3 describes how these requirements were implemented in an initial prototype, called the Idea Evaluator. Section 4 comments on the findings and conclusions of using the prototype.

2. Requirements for Idea Evaluator

In today's dynamic environment, suggestion systems should be designed so as to encourage the submission of ideas that may not be practical today, but have a reasonable likelihood of implementation in the not too distant future. For example, a employee may not submit an idea because the idea requires a technological advance, which according to the employee's limited knowledge, has not occurred. However, the company may be working on just such an advance and if the idea had been submitted, the company could capitalize more rapidly on it.

2.1 Explicit Limiting Factors

This type of situation is of primary concern for the Idea Evaluator. In fact, when an idea is submitted to the Idea Evaluator, the submitter can associate limiting factors with the idea. The submitter assigns a likelihood expressing the degree to which the limiting factor restricts the idea. The likelihood can vary from zero to one. A zero indicates that the limiting factor completely restricts the idea, and that the associated idea is not practical, because of the current state of the art corresponding to that limiting

factor. The nearer the likelihood is to one, the more likely it is that the limiting factor will **not** restrict the implementation of an idea. If an idea has several limiting factors, then the overall likelihood of the idea is equal to the lowest likelihood of all the associated limiting factors. Examples of this will be given in the discussion to follow. As a result of explicitly naming the limiting factors, employees will be more inclined to submit ideas, because the ideas will not be considered so outlandish since the ideas are qualified by limiting factors.

2.2 Selective Review

Another concern for the Idea Evaluator is that the ideabase of potentially fruitful ideas is easy to use to retrieve groups of related ideas. For example, to support a brainstorming session, a manager may wish to retrieve a group of similar ideas. For this reason the Idea Evaluator is designed so that when submitting an idea, the submitter associates the idea with one or more idea categories. As a result, when it is desired to consider a group of related ideas, the ideas can be retrieved by these categories. In addition, a group of retrieved ideas can be further restricted by the idea likelihood, such as all those ideas in a certain category with a likelihood of 0.8 or better. This type of retrieval is helpful in a dynamic environment, where the likelihoods of the limiting factors are changing. For example, suppose that the likelihood of a certain limiting factor is assessed initially at 0.1. Any idea associated with this limiting factor will not appear in any idea retrieval where the desired idea likelihood is greater than 0.1. However, if, because of some recent change in technology, the likelihood of this limiting factor changes to 0.9, then the overall likelihood of any idea associated with this limiting factor may also increase (it may not change because of other limiting factors). In any event, whenever a group of ideas is retrieved, it does reflect the latest assessment of the limiting factors.

3. Initial Prototype

To illustrate the capabilities of the initial Idea Evaluator Prototype, assume that the operations of a financial services organization are to be analyzed. The prototype can perform any of the following tasks:

- Add a new idea,
- Add a new limiting factor,
- Associate limiting factors with an idea, and
- Review existing ideas.

The expert system tool, *VP-Expert*, was used to write a few small knowledge bases. The IE_ctl.kbs knowledge base controls the various tasks to be performed and four other knowledge bases perform each of the above tasks.

3.1 Adding a New Idea

The first step is to enter the basic information about an idea. Assume that an idea offering computerized assistance to determine overdue accounts is under consideration. To enter the idea in the Idea Evaluator Prototype, *VP-Expert* is used to consult with the IE_ctl.kbs knowledge base. After a few introductory screens, the user is presented with the screen given in Figure 1. The option, Add_new_idea, is selected. Control automatically transfers to the IE_addi.kbs knowledge base, which guides the user in entering the basic information about an idea. All of the knowledge bases use dBase files to store the information concerning the ideas. Figure 2 shows the structure of these dBase files. The basic information concerning an idea is stored in the Ideas.dbf file and consists of an Idea_name (up to 20 characters) and an Idea_text (up to 250 characters). Figure 3 shows a partial listing of the Ideas.dbf file. For example, the complete text of the idea with Idea_name of Credit_Assistant is:

> Implement a rule-based system to help determine when overdue accounts should be turned over to collection agencies.

The user also enters the idea categories that the given idea should be associated with. The purpose of these categories is to facilitate retrieval groups of related ideas. For example, the Credit_Assistant idea is associated with the idea categories of Reduce_costs and Existing_products. Since each idea typically is associated with more than one category, this information is stored in the Icats.dbf file. See Figure 2 for the structure of the Icats.dbf file.

3.2 Adding a New Limiting Factor

The second step is to enter the basic information about any limiting factors. The process is similar to entering the information about an idea. The user consults with the IE_ctl.kbs knowledge base, but selects the Add_new_factor option as shown in Figure 1. Control automatically transfers to the IE_addf.kbs knowledge base, which guides the user in entering the information about a limiting factor. This consists of a name for the limiting factor, Fact_name (up to 20 characters), the text of the limiting factor, Fact_text (up to 250 characters), and the likelihood factor for the limiting factor. For example, in analyzing the Credit_Assistant idea, it is felt that it will not be a practical idea until the protocol converter is implemented. Since that is not currently

the case, a limiting factor with Fact_name of Protocol_converter and Fact_text of:

> The communications interface between the expert system and the cardholder databases must have acceptable response time.

is entered. It is assigned a likelihood of 0.1. The information about the limiting factors is stored in the Factors.dbf dBase file. See Figure 4 for a partial listing. Please note the Protocol_converter limiting factor.

3.3 Associating Limiting Factors with an Idea

The third step is to associate any required limiting factors with a given idea. The user consults with the IE_ctl.kbs knowledge base, but selects the Associate_factors option as shown in Figure 1. Control automatically transfers to the IE_assoc.kbs knowledge base. Figure 5 shows the typical dialog. The user is asked to select the name of the desired idea. The options are automatically retrieved from the Ideas.dbf dBase file. In this example, the user selects Credit_Assistant. Next, the user is asked what names of the limiting factors that are to be associated with this idea. The options are automatically retrieved from the Factors.dbf dBase file. Here the user selects Protocol_converter and Available_expert. These associations are stored in their own dBase file, Ifacts.dbf. See Figure 2 for a listing of the structure of this file.

3.4 Reviewing Existing Ideas

The fourth task that can be performed by the initial Idea Evaluator Prototype is to review ideas selectively. The user consults with the IE_ctl.kbs knowledge base, but selects the Review_ideas option as shown in Figure 1. Control automatically transfers to the IE_rview.kbs knowledge base. The user is asked to select the categories of the desired ideas. For example, the user may select Increase_revenue, Existing_products and Future_products. Next, the user is asked the names of the ideas to be reviewed. As shown in Figure 6, only the names of those ideas that are in at least one of the selected categories appear. Then the idea name, text and limiting factors are presented to the user, one screen at a time, for each selected idea name.

4. Finding and Conclusion

A finding of this study is that the initial Idea Evaluator Prototype has sufficient capabilities to demonstrate the feasibility of using database technology and expert system technology to assist organizations to better capitalize on the inherent creativity of their organizational members. This is based on the experience gained from the ideabase presented in this paper and a similar ideabase developed for the operations of a university.

As noted in [Abraham and Boone, 1991], the creative process involves two kinds of thinking: divergent and convergent. Divergent thinking expands and broadens the thought process, while convergent thinking is a process of narrowing and reducing. The process of reviewing ideas as currently implemented in the initial Idea Evaluator Prototype can be helpful in divergent thinking. On the other hand, a neural network may be helpful in convergent thinking. To test this hypothesis, a larger ideabase will be developed where the ideas are rated according to several properties. As noted in [Garavaglia, 1991] appropriate features include risk, out-of-pocket expense, return on investment, scalability and impact on existing operations. A neural network trained to recognize successful ideas from the past may be helpful in suggesting a subset of ideas to converge on a solution to a present day opportunity.

The primary conclusion of this study is that the initial Idea Evaluator Prototype should be developed further. Specifically, some requirements for the next iteration prototype are:

- to expand the Ideas.dbf file to include several features such as risk, out-of-pocket expense, return on investment, scalability and impact on existing operations,
- to add the capability of evaluating an idea with respect to these features and storing the result, and
- to develop a neural network to identify potentially successful ideas.

References

1. Abraham, Thomas and Larry Boone, "Potential Applications of Artificial Intelligence for Enhancing Creative Organizational Decision Making," Academy of Management Meeting, Organization Behavior Division, 1991.

2. Allen, John, "Suggestion Systems and Problem-Solving: One and the Same?," *Quality Circles Journal*, Vol. 10, No. 1, Mar. 1987, pp. 2-5.

3. Buckwalter, Randy, "Success at Harleysville Insurance: Using a Team Suggestion Program," *Journal for Quality & Participation*, Vol. 11, No. 2, Jun. 1988, pp. 8-9.

4. Burnette, Donna and Tim Bousum, "Calculate Suggestion Program Savings," *Personnel Journal*, Vol. 68, No. 2, Feb. 1989, pp. 33, 35.

5. Crosby, Bob, "Employee Involvement: Why It Fails, What It Takes To Succeed," *Personnel Administrator*, Feb. 1986, pp. 95-106.

6. Garavaglia, Susan, "Winning Management Support: Cost Justifying AI Projects on Wall Street", *Proceedings of the First International Conference of Artificial Intelligence Applications on Wall Street*, Oct. 10, 1991, IEEE Computer Society Press, Los Alamitos, CA, pp. 60-61.

7. Goldsmith, Neal M., "Technology Transfer Triangulation: Issues Influencing Innovation", presented at the AI in Business Workshop, American Association for Artificial Intelligence, San Jose, California, July 14, 1992.

8. Johannes, Gene, National Association of Suggestion Systems, Chicago, personal interview, Jan. 1992.

9. Lo Bosco, Maryellen, "Consensus on Suggestion Systems," *Personnel*, Oct. 1985, pp. 16-21.

10. Moore, P. Michael, "Employee Suggestion Systems Can Work," *CMA Magazine (Canada)*, Vol. 62, No. 9, Nov. 1988, pp. 40-42.

11. NAPL, National Association of Printers and Lithographers, Communications Dept., 780 Palisade Ave., Teaneck, NJ 07666.

12. Phillips, Dennis D. and Lee Wallace, "Initiate Employee Input," *Personnel Journal*, Vol. 67, No. 2, Feb. 1988, pp. 22-26.

13. Scicutella, John, "Using Employee Participation to Enhance Productivity," *Bankers Magazine*, Vol. 171, No. 6, Nov/Dec. 1988, pp. 71-73.

14. Stakel, Leslie, "Employment Relations Programs," *Employee Relations Today*, Vol. 16, No. 2, Summer 1989, pp. 167-169.

15. Zakeski, David M., "Reliable Assessments of Organizations," *Personnel Journal*, Vol. 67, No. 12, Dec. 1988, pp. 42-48.

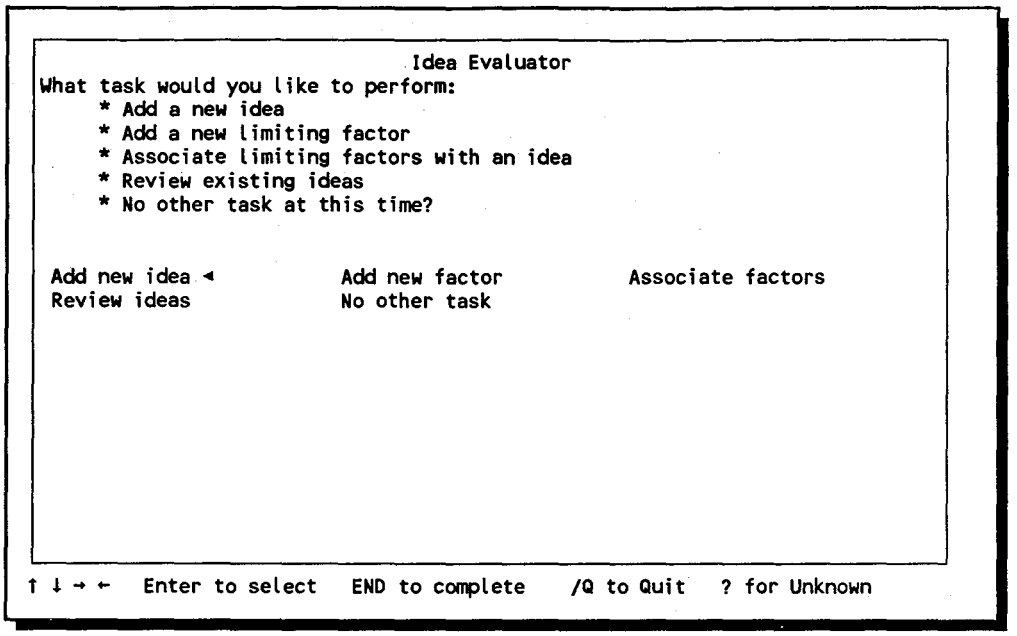


Figure 1 - Idea Evaluator Task Selection Screen

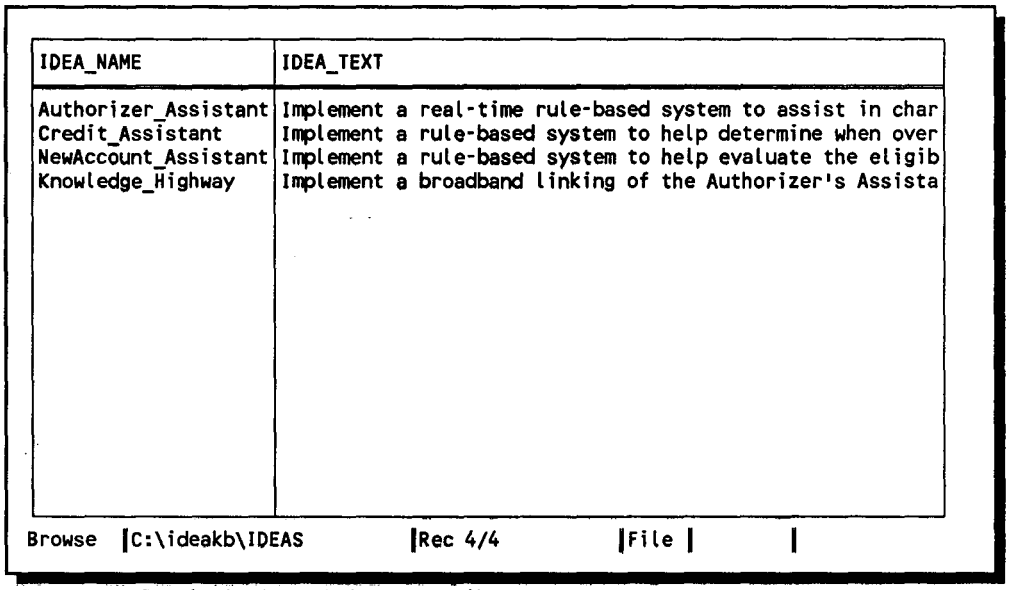
```
Structure for database: C:\IDEAKB\IDEAS.DBF
Field  Field Name  Type       Width    Dec    Index
    1  IDEA_NAME   Character     20             N
    2  IDEA_TEXT   Character    250             N
    3  IDEA_LHOOD  Numeric        6      3       N
** Total **                     277

Structure for database: C:\IDEAKB\FACTORS.DBF
Field  Field Name  Type       Width    Dec    Index
    1  FACT_NAME   Character     20             N
    2  FACT_TEXT   Character    250             N
    3  FACT_LHOOD  Numeric        6      3       N
** Total **                     277

Structure for database: C:\IDEAKB\ICATS.DBF
Field  Field Name  Type       Width    Dec    Index
    1  IDEA_NAME   Character     20             N
    2  IDEA_CAT    Character     20             N
** Total **                      41

Structure for database: C:\IDEAKB\IFACTS.DBF
Field  Field Name  Type       Width    Dec    Index
    1  IDEA_NAME   Character     20             N
    2  FACT_NAME   Character     20             N
** Total **                      41
```

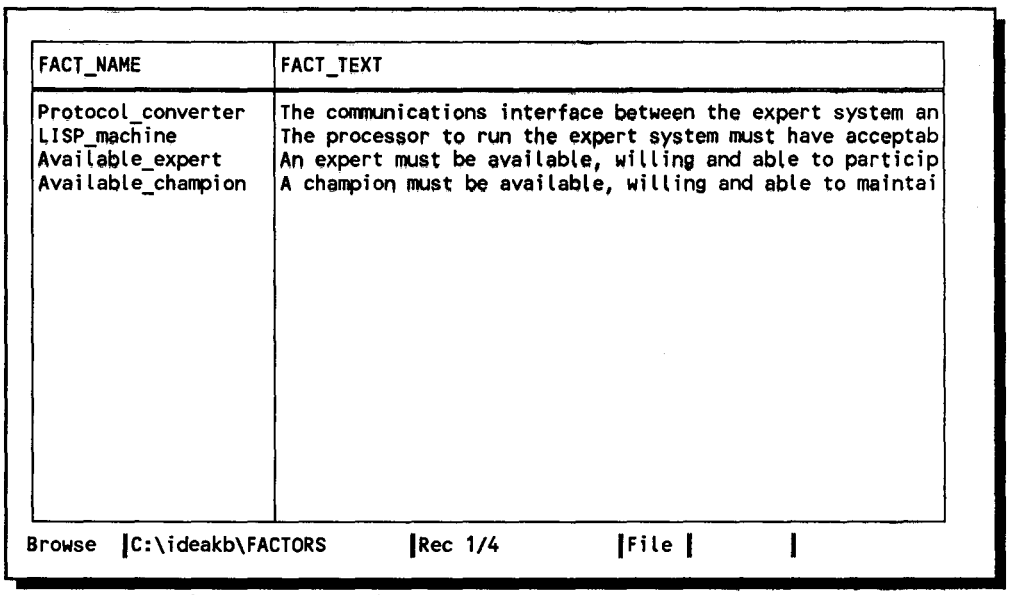
Figure 2 - Structure of dBase Files



IDEA_NAME	IDEA_TEXT
Authorizer_Assistant	Implement a real-time rule-based system to assist in char
Credit_Assistant	Implement a rule-based system to help determine when over
NewAccount_Assistant	Implement a rule-based system to help evaluate the eligib
Knowledge_Highway	Implement a broadband linking of the Authorizer's Assista

Browse |C:\ideakb\IDEAS |Rec 4/4 |File |

Figure 3 - Typical Ideas.dbf dBase File



FACT_NAME	FACT_TEXT
Protocol_converter	The communications interface between the expert system an
LISP_machine	The processor to run the expert system must have acceptab
Available_expert	An expert must be available, willing and able to particip
Available_champion	A champion must be available, willing and able to maintai

Browse |C:\ideakb\FACTORS |Rec 1/4 |File |

Figure 4 - Typical Factors.dbf dBase File

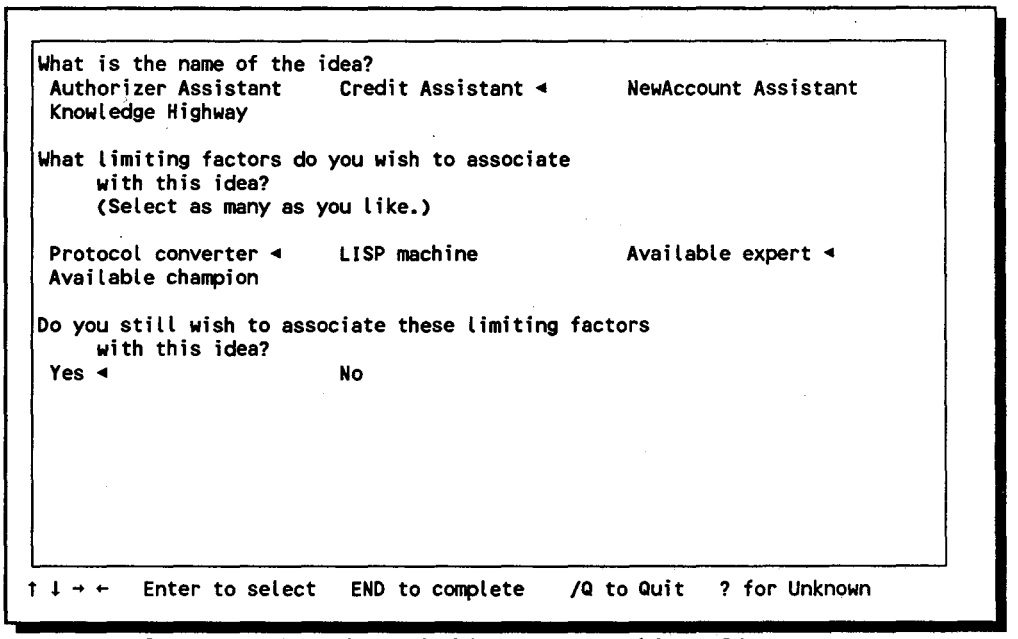


Figure 5 - Screen to Associate Limiting Factors with an Idea

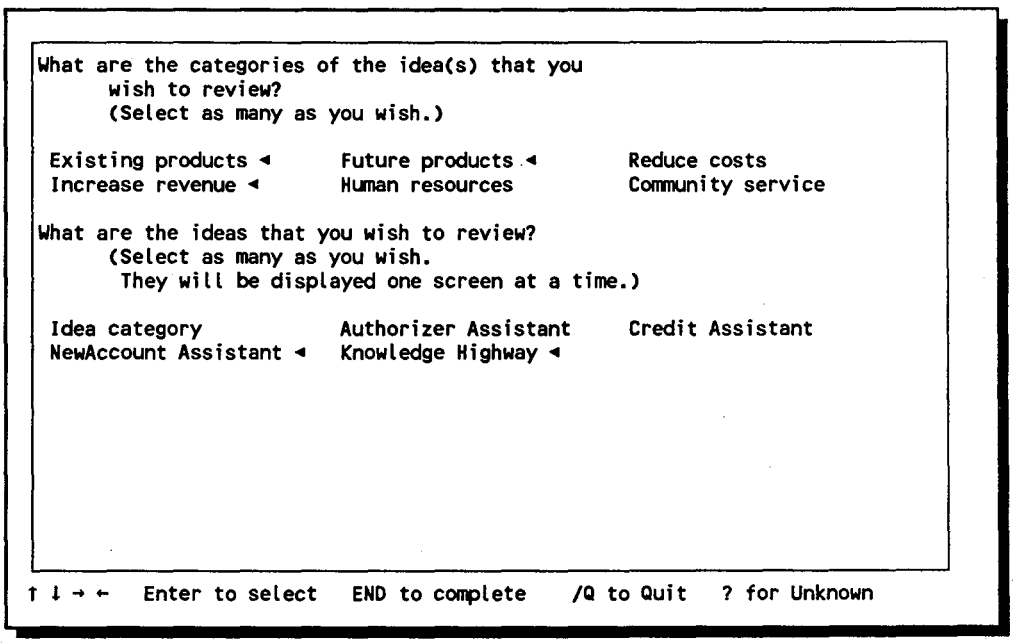


Figure 6 - Screen to Review Ideas

Paper Session: Discovering Stock Selection Rules

Chair: Gia-Shuh Jang, National Taiwan University

Trading Rules and Trading Cases in a Hybrid Architecture

David B. Skalak
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
skalak@cs.umass.edu

Abstract

We describe the CABARET system, which is a hybrid architecture that integrates case-based reasoning with rule-based reasoning. Case-based reasoning is an emerging subfield of artificial intelligence that is rooted in the observation that a problem often may be solved by retrieving and manipulating the solutions to previous similar problems. The CABARET shell provides several features that are potentially useful to trading systems, in particular (1) the ability to identify the situations in which a rule's requirements have been almost satisfied, and (2) the capacity to supply automatically examples of previous situations that are similar with respect to their behavior under a rule and other domain factors. We illustrate these and other features of the system with an example stock-screening application.

1. Introduction

Many traders and investors rely on rules to select a security or commodity for trading and to time a buy or sell decision. As a matter of discipline, system traders may rely on signals provided by a set of rules. Discretionary traders may place a trade when a majority of the criteria from a mental checklist have been satisfied[1]. While traditional rule-based expert systems can automate some aspects of rule-guided trading, they fail to provide two types of guidance that would aid both system and discretionary traders: (1) to indicate when the requirements of a rule have not been satisfied, but yet have been *nearly* satisfied; and (2) to supply automatically historical examples of similar situations in which a rule of interest was or was not satisfied. In this research we consider a timing rule or a security selection rule as "nearly" satisfied if all but one of the conjunctive requirements of the rule have been satisfied. We have termed this situation a "rule-based near miss," after Winston [21]. For example, if the requirements of a stock screening rule are satisfied except one condition requiring that the growth rate in sales revenues be greater than 20% per year, and the growth rate for Widget Co. is 19.7%, the rule is a near miss. Nevertheless, if Widget Co. turned out to be a successful stock pick, the knowledge that it almost satisfied the screening rule would be useful: for example, modification of the screening rule might be suggested.

In general, an indication of when a near miss occurs on a trading rule would be useful to the trader to indicate the need to re-optimize rule parameters, which some traders suggest be done frequently, especially by those trading in short time frames. A system that identifies situations when rule thresholds are almost met would permit traders to monitor trading such near miss signals. A trade that was not placed but that would have been successful could indicate that the trading rules ought to be re-optimized for the current market.

Classical expert systems also do not automatically supply examples of historical situations where a rule was or was not satisfied. One of the shortcomings of traditional production systems is that they fail to provide comprehensible explanations of their recommendations. The same is generally true of many

[1]See [20] for further elucidation of the distinction between system ("mechanical") and discretionary ("intuitive") traders.

advanced technologies being applied to market forecasting, such as artificial neural networks and genetic algorithms. A system that can provide actual, similar instances from a historical database (or "case" base) of previous situations in which a selection or timing rule fired gives the trader a partial explanation of how the market has performed in similar conditions, and therefore can provide a user with an improved sense of the risk of putting on a trade.

Newsletter writers and market analysts are often students of market history and cite previous cases to explain present market action. For example, *The Zweig Forecast* [22], a respected market newsletter, frequently contains references to historical situations where similar monetary, sentiment, technical, or general economic conditions prevailed.

Identifying and reasoning with situations that are near misses on rules and providing historical examples of instances when rules fired or failed to fire are two facilities provided by CABARET, developed in the Case-Based Reasoning Group at the University of Massachusetts. CABARET (CAse-BAsed REasoning Tool) is a hybrid shell that integrates traditional rule-based reasoning, as found in most expert systems, with case-based reasoning (CBR). CBR is an emerging subfield of artificial intelligence that deals in part with the problem of storing previous experiences ("cases") in a computer memory so that they may be retrieved when similar circumstances arise. The field is founded on the idea that many problem-solving tasks are really memory tasks: humans often remember previous times when they have solved a *similar* problem and adapt the remembered solution to the current situation, rather than working "from scratch". Retrieving previous cases that are similar to the current situation can show how to deal with the current problem. One of the principal focuses of CBR research is how to define "similar" in a useful way for a given application. For a recent overview of CBR, see [18].

Cases used in CBR systems may take a variety of forms. In a stock-screening application, for example, a case may consist of a collection of fundamental data for a stock. One could retrieve such cases from a library of fundamental data to determine whether previous stocks with particular characteristics increased in value over a time period of interest. In order to perform this retrieval, the screening application would have to contain the knowledge to determine when two sets of fundamental data are similar. A variety of domain-dependent and domain-independent techniques may be used to this end.

This paper argues that tasks such as security selection and trading may benefit from hybrid systems that intelligently apply case-based methods in conjunction with traditional rule-based methods. Case-based methods ground trading rules and market analysis in actual experience. Trading rules can provide inductive generalizations of the trading cases as well as provide discipline for a trader. In order that rule-based and case-based methods may work synergistically, the CABARET project heuristically combines these two styles of reasoning. CABARET also has a variety of (1) features for maintaining and browsing case databases and rule bases, (2) analytical capabilities, and (3) facilities for explanation and argument. Only a few of these features are mentioned here. For additional background on this system, see [14].

In the remainder of this paper we describe the skeleton of the CABARET architecture and give an example of how the system may be used to screen stocks. A discussion of related research and a summary close the paper.

2. The CABARET Shell

CABARET is a domain-independent computer program that consists of a number of integrated modules. The primary modules are a case-based reasoner, a rule-based reasoner and a heuristic control module. See Figure 1.

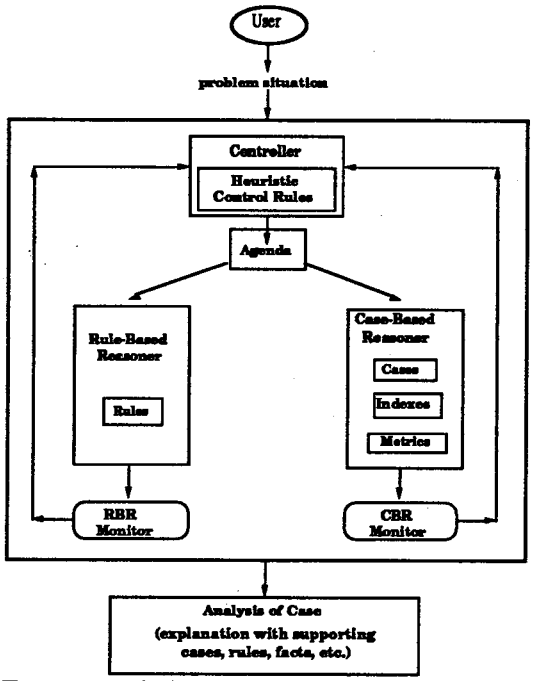


Figure 1: CABARET architecture and data flow.

The CBR module is modeled upon HYPO, a CBR system that created legal arguments in an area of the law dealing with trade secrets misappropriation [1]. CABARET's domain-independent CBR reasoner relies in part upon indexing and retrieving cases in memory using "dimensions," (also called "factors") [15] [1], which are important domain factors on which cases may be compared as weaker or stronger. Example of possible dimensions in a stock screening application are the annual earnings per share of a firm or a corporation's bond rating.

CABARET's rule-based reasoner is a production system that performs both forward and backward chaining on rules. This reasoning module was designed as a "glass box" reasoner so that its internal processing would be available for inspection by the system's control module and by the user. The control module can therefore determine when a rule has almost fired — when all but one of the rule's antecedents have been satisfied — and take appropriate action. Appropriate action includes retrieving cases from memory where the rule also failed to fire, retrieving in particular those cases that failed to satisfy the specific antecedent that is unsatisfied in the current situation. Cases are indexed in memory by rule failures and rule successes so that they can be retrieved for comparison with the current case by the system along the domain dimensions.

CABARET relies for its control structure — how the program decides what to do and when — on a collection of 30 heuristic rules that mediate between the rule-based reasoner and the case-based reasoner. These heuristic rules propose tasks for the system to work on and assign the rule-based reasoner or the case-based reasoner to perform the task [14].

This design enables CABARET to perform several functions that are potentially useful to the trader. Among others, the system can

- supply examples that support the application of a rule,
- find counter-examples to spot failures of a rule, and
- retrieve examples of "near-misses" where a rule nearly succeeded.

Furthermore, CABARET outputs comparisons of retrieved cases with the current case — informing the user of both similarities and differences between the current problem and each retrieved case. These comparisons are presented using natural language explanation templates, to explain, for example, why a security should or should not be purchased. The system thus can give explanations that might be useful to a trader, including both examples and comparisons of cases along dimensions that are deemed by the application domain experts to be important. CABARET's explanation facility improves upon the classical rule-based expert system, which often merely provides a trace of rule-firings as a proxy for an explanation of its conclusions.

3. Example: Using CABARET to Screen Stocks

3.1. Rule and Case Base Construction

To function in a specific domain, the CABARET shell requires a rule set, a collection of cases to which the rules apply, and a set of dimensions that are used to index and compare the cases. In general, the rules are supplied by the developer of a trading or screening system, but they may also be developed inductively from the cases themselves using a machine learning algorithm such as ID3 [13]. See also [16].

The case base may be constructed in several ways. The trader may use all available previous instances of cases in the case base, possibly via an on-line data source that has been bridged to CABARET. Alternatively, the trader may select either automatically or by-hand particular cases that he wishes to have available. For instance, having been burnt by a long position in a stock that declined in value after having passed a rule-based filter, a trader may wish to include that counter-example in the case base so that future stocks that pass the filter may nonetheless be compared to the counter-example for clues as to the possible repetition of that loss.

Cases in CABARET must be tagged as "positive" or "negative" examples. A stock that subsequently increased in price over some time period would be considered a positive example (of a long position); a stock whose price declined over the given period would be considered a negative example. CABARET can provide better pro and con arguments when supplied with a case base that contains both positive and negative instances of cases that can be compared and contrasted by the system.

3.2. Example

For illustration, we have compiled a case base consisting of a subset of the "Past Great Winners" identified in [11] and [12]. This set of equities that have increased substantially in price provides a useful case base[2]. This collection permits the comparison of current stocks to these prototypical "great winners" to assess the potential for significant appreciation by current stock prospects. See Figure 2 for a portion of the case representation.

```
(make-past-great-winner <xyz-corp>
        :number-shares-outstanding 19.0
        :group-relative-strength 30
        :most-recent-annual-earnings-
estimate-increase 41
        :group-stocks-confirming-strength
'(90 77 74 67 52)
        :percent-shares-owned-by-funds 7
        :percent-shares-owned-by-banks 1
        :up-down-volume 1.5
        :relative-strength 94
        :relative-strength-trend :up
        :average-daily-volume 39800
        :base :cup-with-handle
        :length-of-base 4 ;months
        :years-of-increasing-eps 5
        :last-4-quarters-earnings-%-
increase '(58 138 107 82)
        :five-year-growth-rate 128
        :price-within-12-month-high -4%
        :date 800520
        :subsequent-percent-gain 236)
```

Figure 2: Partial frame representation of a "Past Great Winner" common stock case [11][12].

In this representation, the group-stocks-confirming-strength slot is filled by a list of the top five relative strengths of stocks in the same industry group as the represented case [11].

For the sake of argument, suppose one had created a screening rule designed to capture features of the well-known "CAN-SLIM" system for equity selection [12]. In particular, let us suppose that an investor wanted to trade stocks on the basis of some of the characteristics of the average stock selected by David Ryan to implement aspects of the CAN-SLIM system [12, p.135]. See Figure 3 for an

[2]A case base should also include some past great losers, however.

example rule that supplies four antecedents in conjunction to capture some features of this approach to security selection.

```
If <the-equity> has
  (and
    (annual-earnings-growth-rate>20%)
    (eps-percentage-increase-for-current-
quarter > 34%)
    ( p-e < 15)
    ( relative-strength > 85))
then (buy <the-equity>).
```

Figure 3: Example stock selection rule reflecting aspects of the "CAN-SLIM" approach.

Using the rule in Figure 3 to filter the case base returns two cases: *computer-associates* and *prime-computer*.

However, CABARET also informs the user that *computer-factory* is a near-miss on the selection rule, which satisfies the final three criteria as to earnings per share percentage increase for the current quarter, price-earnings ratio and relative strength, but whose annual earnings growth rate is 19%, just missing the 20% threshold. Recognizing this near-miss, CABARET then automatically compares *computer-factory* to the other stocks that have passed the rule screen. For example, CABARET reports to the user that even though *computer-factory* failed to pass the screen, it had superior values to *computer associates,* a security that did pass the filter, along the *most-recent-annual-percent-earnings-increase* and *group-stocks-confirming-strength* dimensions. These two favorable comparisons would argue in favor of purchasing *computer-factory*, even though it did not pass the rule screen. *Computer-factory* did indeed go on to record impressive gains.

4. Related Research

Case-based reasoning is an active research area in artificial intelligence. Recently, applications of case-based reasoning to a variety of problems on Wall Street have been made (see, e.g. [2] (merger and acquisition analysis), [5] (market surveillance), [6] (debt analysis), and [17] (financial decision-making). The design of the standalone features of the case-based reasoning component of CABARET is modeled after the HYPO system [1], whose domain is trade secret misappropriation.

Hybrid systems that combine rule-based and case-based reasoning have been built in several "real-world" domains (see, e.g., [3] (merger and acquisition analysis), [4] (legal argument), and [8] (word pronunciation). For example, Golding and Rosenbloom [8] demonstrated that a hybrid case-based and rule-based approach to word pronunciation was more accurate than either a case-based or a rule-based approach used alone. Researchers have also investigated the integration of case-based reasoning and reasoning paradigms other than rule-based reasoning (e.g., model-based reasoning [7][9], utility-based decision theory [19], and genetic algorithms [10]).

5. Conclusion

CABARET is a hybrid architecture that can provide assistance to traders and investors who rely on rules for security screening or market timing. In particular, the system has the capacity to extend strictly rule-based approaches by identifying situations in which a rule is almost satisfied. In addition, CABARET retrieves and compares previous cases to the current situation in order to provide concrete examples that may be used to extend or limit a rule's scope.

6. Acknowledgments

The Principal Investigator of the Case-Based Reasoning Group at the University of Massachusetts is Professor Edwina L. Rissland. The development of the CABARET shell was supported in part by the National Science Foundation, contract IRI-890841, the Air Force Office of Sponsored Research under contract 90-0359, the Office of Naval Research under a University Research Initiative Grant, contract N00014-87-K-0238, and a grant from GTE Laboratories, Waltham, MA. Daniel Suthers programmed the rule-based

reasoner used by CABARET and several utility modules invoked by the system.

7. References

[1] Ashley, K. D. (1990). *Modelling Legal Argument: Reasoning with Cases and Hypotheticals*. Cambridge, MA: M.I.T. Press.

[2] Berg-Cross, G. & Claudio, L. (1992). *MASTER: A Case Based-Design to Augment Corporate Mergers and Acquisitions Decisions*. The First International Conference on Artificial Intelligence Applications on Wall Street. New York: IEEE Computer Society Press, 188-193.

[3] Bonissone, P., Blau, L. & Ayub, S. (1990). Leveraging the Integration of Approximate Reasoning Systems. *Proceedings of the AAAI Symposium on Case-Based Reasoning 1990*. Palo Alto, CA.

[4] Branting, L. K. (1991). *Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis*. Ph.D. Thesis, Technical Report AI90-146, Artificial Intelligence Laboratory, University of Texas, Austin, TX.

[5] Buta, P. and Barletta, R. (1991). *Case-Based Reasoning for Market Surveillance*. The First International Conference on Artificial Intelligence Applications on Wall Street. New York: IEEE Computer Society Press, 116-121.

[6] Freedman, R. S., Frail, R. P., Schneider, F. T. and Schnitta, B. (1991). *Expert Systems in Spreadsheets: Modeling the Wall Street User Domain*. The First International Conference on Artificial Intelligence Applications on Wall Street. IEEE Computer Science Press, 296-301.

[7] Goel, A. (1989). *Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving*. Ph.D. Thesis, Dept. of Computer and Information Science, The Ohio State University.

[8] Golding, A. R. and Rosenbloom, P. S. (1991). *Improving Rule-Based Systems through Case-Based Reasoning*. Proceedings, Ninth National Conference on Artificial Intelligence, Anaheim, CA: American Association for Artificial Intelligence.

[9] Koton, P. A. (1988). *Using Experience on Learning and Problem Solving*. Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA.

[10] Louis, S., McGraw, G. & Wyckoff, R. O. (1992). *Case-Based Reasoning Assisted Explanation of Genetic Algorithm Results*. Journal of Experimental & Theoretical Artificial Intelligence, 5(1):21-38.

[11] New York Stock Exchange Daily Graphs (1992). Los Angeles, CA: William O'Neil & Co., Inc.

[12] O'Neil, W. J. (1988). *How to Make Money in Stocks* (Revised 1st ed.). New York: McGraw-Hill.

[13] Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1(1), 81-106.

[14] Rissland, E.L. & Skalak, D.B. (1991). CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 34, 839-887.

[15] Rissland, E. L., Valcarce, E. M. and Ashley, K. D. (1984). *Explaining and Arguing with Examples*. AAAI-84, Proceedings of the National Conference on Artificial Intelligence, Austin, TX: American Association for Artificial Intelligence.

[16] Skalak, D. B. & Rissland, E. L. (1990). Inductive Learning in a Mixed Paradigm Setting. *Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence*, 840-847. Boston, MA. American Association for Artificial Intelligence.

[17] Slade, S.B. (1991). *Case-based Reasoning for Financial Decision Making*. The First International Conference on Artificial Intelligence Applications on Wall Street. New York: IEEE Computer Society Press, 232-237.

[18] Slade, S.B. (1991). Case-based Reasoning: A research paradigm. *AI Magazine*, 12(1):42-55.

[19] Sycara, K. P. (1987). *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods*. Ph.D. Thesis, School of

Information and Computer Science, Georgia Institute of Technology, Atlanta, GA.

[20] Tharp, V. K. (1992). *Investment Psychology Course*. Cary, NC: Investment Psychology Consulting.

[21] Winston, P. H. (1982). Learning new principles from precedents and exercises. *Artificial Intelligence*, 19, 321-350.

[22] Zweig, M. E. (1993). *The Zweig Forecast*, Zweig Securities Advisory Service, Inc., Wantagh, NY.

A Framework for Rule Base Refinement in a Stock Market Technical Analysis — Towards Discovering Anomaly from Granville's Law —

Takahira YAMAGUCHI and Yoshiaki TACHIBANA

Department of Computer Science
Faculty of Engineering
Shizuoka University
3-5-1, Johoku, Hamamatsu, Shizuoka 432, Japan e-mail: yamaguti@cs.shizuoka.ac.jp

Abstract

We have developed a framework for refining an initial object-level rule base with a rule induction to learn meta-level rules which find out a data set applicable to it. A rule induction process like ID3 tries to learn meta-level rules and classify training data sets into positive data sets and negative ones. A rule refinement process tries to refine an initial object-level rule base on classified data sets, using four refinement strategies. Unifying these two processes, we can get a refined object-level rule base with high performance, a meta-level rule selecting a data set applicable to it. In order to evaluate the framework, an experiment on real Japanese stock price data shows that a refined object-level rule base, which comes from the initial object-level rule base for representing Granville's Laws, has the performance beyond that of the average stock price. The performance is difficult for human technical analysts in a stock market to achieve. The result implies that the framework could invent anomaly from Granville's Law in a stock market technical analysis.

1 Introduction

In building expert systems, an initial object-level rule base can be developed easily, because human experts can tell their expertise to knowledge engineers roughly. The object-level rules from textbooks may also be initial object-level rules. The initial object-level rule base, however, usually has just lower performance than that of human experts. It takes so long time to polish up the initial object-level rule base with low performance into an excellent object-level rule base with high one. So it is a key issue to develop useful automatic rule refinement systems.

In this paper, the initial object-level rule base comes from a textbook. The framework for refining an initial object-level rule base has two processes: a rule induction process and a rule refinement one. The former tries to learn a meta-level rule which finds out a data set applicable to an object-level rule. This process is almost similar to ID3 proposed by J.R.Quinlan(1986). The latter tries to refine an initial object-level rule based on training data sets, using four refinement strategies. The framework unifies these two processes and tries to get a refined object-level rule base with high performance, a meta-level rule selecting a data set applicable to it.

Maybe SEEK2 developed by A.Ginsberg et al.(1988) is the most representative and practical rule refinement system. It has two strategies to refine a rule base: generalization and specialization . SEEK2 applies the refinement strategies to an initial object-level rule base within a given rule space and tries to find the most adequate point in the rule space. The refinement strategies within the rule space, however, have the limitation to get the refinement gain from the initial object-level rule base. In the case that the representation of the rule space would be insufficient, unless it could change into another powerful rule space with new effective descriptors (attributes), SEEK2 could not refine the initial object-level rule base adequately. Inductive learning (Similarity Based Learning) approaches also have the same limitation as that of SEEK2. It is a big issue called new term problem all over the field of inductive learning.

Considering the above-mentioned background, our framework is an approach to manage the issue indirectly, learning meta-level rules which find out a data set applicable to an object-level rule. When the acquired meta-level rules will go well, the rule space could get higher performance than before learning.

As a matter of fact, an experiment on real Japanese stock price data shows that a refined object-level rule base together with a meta-level rule base, which comes from the initial object-level rule base for representing Granville's Laws from J.E.Granville(1960), has the performance beyond that of the average stock price. The performance is difficult for human technical analysts in a stock market to achieve. The result implies that the framework could make an initial rule space more powerful than before learning, together with acquired meta-level rules, and invent anomaly from Granville's Law in a stock market technical analysis.

2 A Technical Analysis Expert System

Before this paper comes into our rule refinement system, it reviews the expertise and the expert system which our rule refinement system is applied to.

2.1 Granville's Law

Many charts, which mean figures for drawing stock price fluctuation, have been used among Japanese chartists. The expert system has expertise of Granville's Law from Granville (1960) and Gouhou (1986) as the initial rule base. The expertise uses the chart of "Moving Average Lines (MAL)".

In the expert system, A MAL on an investing point is computed, averaging all end prices from the investing point through thirteen weeks before (13MAL). A 13MAL shows fluctuation direction for the stock price at middle range.

There is an important feature in MAL: "Division (between a stock price and a MAL)". Division is computed by means of the following formula: $(stockprice/MAL - 1) \times 100$.

Division implies popularity of a stock name in a stock market. If a stock name gets more than thirty percent (less than minus twenty percent) as a value of division, it means over(under)- popularity of the stock name and the stock name should be bought (sold).

The position relation between a MAL and a stock price is also an important feature. Using two kinds of feature, heuristics called Granville's Laws shown is known.

2.2 Representation of Granville's Law

Table 1 shows all rule primitives to represent Granville's Law. Three kinds of level exists in increasing/decreasing transition of MAL and "Moving End price Line(MEL)". The larger suffix number of increasing/decreasing is, the steeper their transition inclination is. Granville's Law is represented using these rule primitives. Using them, the rule representation of Granville's Law is as shown in Figure 1.

Table 1: Rule Primitives for Representing Granville's Law

MAL	i(1),i(2),i(3):	increasing
	d(1),d(2),d(3):	decreasing
	c:	constant
	o:	a MAL is over a MEL
	b:	a MAL is below a MEL
Division	Division(−%)	
MEL	i(1),i(2),i(3):	increasing
	d(1),d(2),d(3):	decreasing
	c:	constant

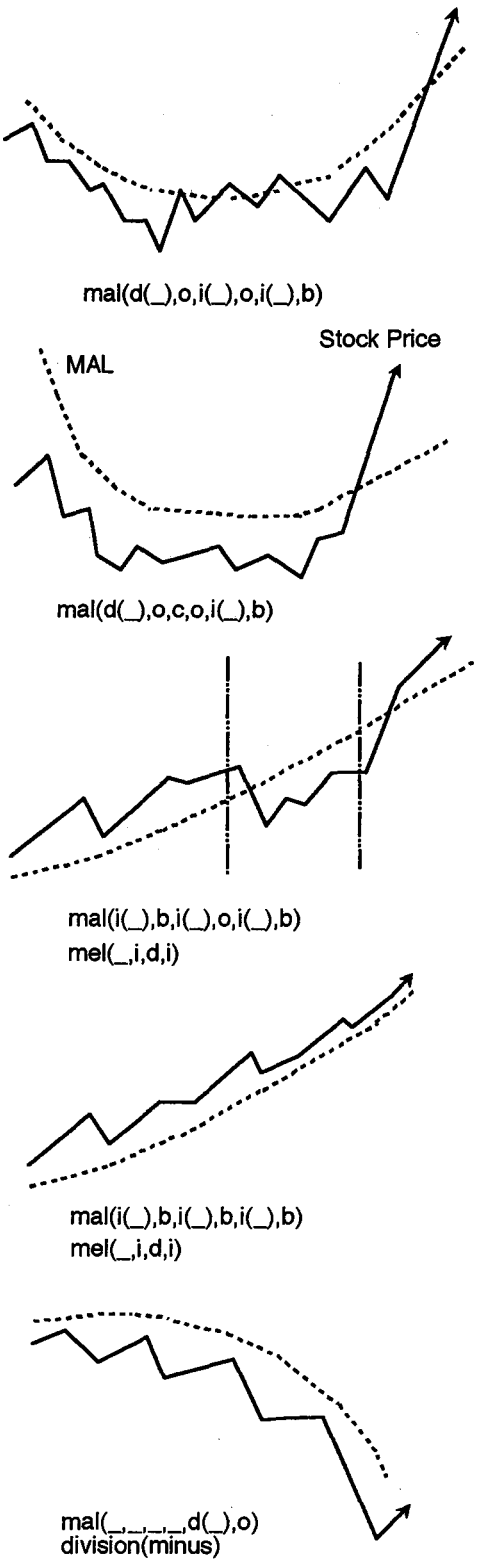


Figure 1. Granville's Law and Rule Representation

2.3 Expert System Organization

The expert system is organized as shown in Figure 2. Part "A" in Figure 2 is implemented by means of C language and "B" by means of Prolog language.

Chart display module draws stock price fluctuation, a 13MAL (using full line) and a 26MAL (using dotted line).

Converter translates numerical data into symbolic data equivalent to rule primitives shown in Table 1. In this translation, transition inclination between two sequential data of MAL (or MEL) was firstly computed and symbolic data (i(1)–i(3),c,d(1)–d(3)) is assigned to each transition inclination. Then the average value is computed using all symbolic values of three transition inclination. For example, from i(2), i(2), and i(1), the average value is computed to i(2). However, when the following situation appears, the average value is computed before the situation.

1. the situation where i(n) (d(n)) changes to d(n) (i(n)), n=1,2,3
2. the situation where i(3) or d(3) (c) changes to c(i(3) or d(3))

List 1 shows symbolic data which is stored in Working Memory. In MAL predicate, a pair of arguments represents inclination of MAL and MAL's position to the stock end price. The state for three points just before the investing point is described by means of three pairs in MAL predicate. In MEL predicate, arguments describe the level of four MEL's inclinations at four points just before the investing point.

An inference engine tries to match symbolic data in WM with production rules to represent Granville's Law. If the matching would succeed, the stock name would be bought.

```
mal(d2,o,d3,b,i3,b)
mel(i(3),i(3),d(3),d(3))
division(3.71)
```

List 1. An EXample of Symbolic Data from Numerical Data

2.4 Experiments and Evaluation

The expert system's performance for predicting the buying has been evaluated by means of 1097 pieces of stock price data on Tokyo security market at the following investing points:

1. the third week of August, 1985 (when the whole average stock price was stable)
2. the third week of April, 1986 (when the whole average stock price was increasing)
3. the third week of September, 1986 (when the whole average stock price was decreasing)

The hit-ratio for the buying has been evaluated by confirming that selected stock names have increased more than five per cent through two months after the investing point. The average hit-ratio for the buying has been 40 per cent and the result implies that this initial rule base must be refined.

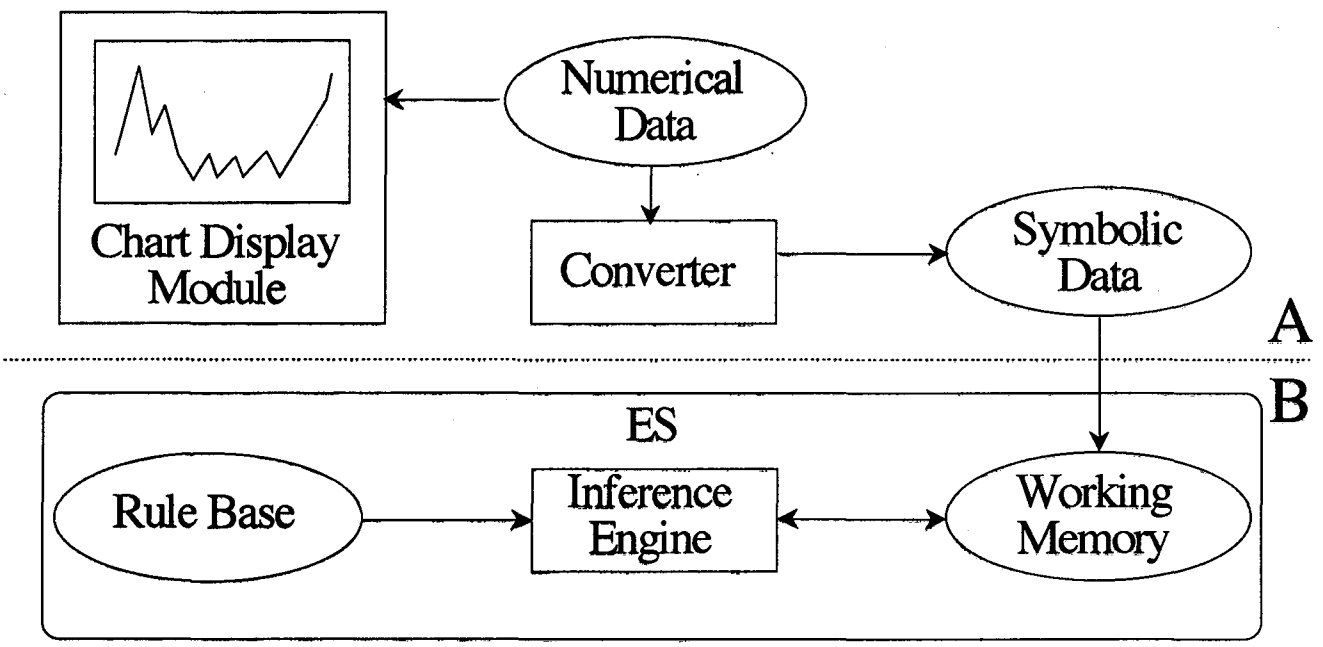


Figure 2.. An Overview for A Technical Analysis Expert System

Table 2: Primitives to Compose Meta-level Rules

0	Names	
1	Tokyo Stcok Market Average Price (First Section, T.S.E. 225 Selected Stocks)	trend_TSMA[1-3],length_TSMA[1-3]
2	Call Rates (Tokyo, Collateral & Uncoditional, Central Rate)	value_CR,trend_CR,length_CR
3	Yields of Government Bonds (Longest-term(10 Years) Bonds)	value_YGB,trend_YGB,length_YGB
4	Tokyo Foreign Exchange Rates (Average of Central Rates, Yen/$)	value_TFER,trend_TFER,length_TFER
5	Consumer Price Indices (All Japan, P.Y.)	value_CPI,trend_CPI,length_CPI
6	Overall Wholesale Price Indices (All Commodities, P.Y.)	value_OWPI,trend_OWPI,length_OWPI
7	Money Supply (Cash Currency in Circulation + Deposit Money + Quasi Money + Certificates Deposits, P.Y.)	value_MS,trend_MS,length_MS

3 Unifying Rule Induction and Rule Refinement

Figure 3 shows a system overview for unifying rule induction and rule refinement to get a refined object-level rule base with high performance, a meta-level rule selecting a data set applicable to it.

3.1 Rule Induction

Firstly an inference engine tries to match training data sets with LHSs in an initial object-level rule base and then classifies them into positive data sets (where RHS of an applied rule was adequate based on some criterion) and negative data sets, based on the evaluation of the result. A rule induction subsystem from ID3 takes these two kinds of data sets and attributes to compose a decision tree (which are equal to primitives to compose meta-level rules as shown in Table 2) and then generates a meta-level rule base, finding useful paths from the decision tree based on some criterion in order to avoid over-fitting to training data sets and low robustness.

Secondly, the meta-level rule base tries to select data sets applicable to an object-level rule base from training data sets. The inference engine tries to match selected data sets with LHSs in an initial object-level rule base and then classifies them into a success data set (where RHS of an applied rule was adequate beyond some criterion) and a failure data set (where RHS of an applied rule was not adequate), and an unmatched data set (where it has not been matched with LHSs of all rules but should have been).

3.2 Rule Refinement

As shown in Figure 4, using all the above-mentioned data sets, the rule refinement subsystem tries to apply four refinement strategies to the initial object-level rule base and generates refined object-level rule candidates. It evaluates them based on the following criterion and selects refined object-level rules from them : If the hit-ratio of a refined object-level rule candidate would be improved beyond that of an original rule or the match-ratio would be improved, it would be accepted as a refined object-level rule. The refinement process is done repeatedly until refined object-level rules have no problems, judging from final criterion which an user gives to the rule refinement subsystem. Four refinement strategies are as follows:

(1) GENERALIZATION STRATEGY

IF The difference between LHS of some rule and unmatched data is just one symbol (constant)
THEN Change the constant in LHS of the rule into variable (represented by _ in a rule)

For example, when unmatched data to Granville's first law is mal(d(2),o,c,b,i(1),b), the rule changes into the following:

IF mal(d(_),o,c,_,i(_),b)
THEN buy.

(2) SPECIALIZATION STRATEGY

IF The difference between positive data and failure data is just one symbol (constant)
THEN Change the variable in LHS of the rule which corresponds to the symbol into constant

For example, when positive data and failure data to Granville's first law are as follows:

positive data: mal(d(2),o,c,o,i(3),b)
failure data: mal(d(2),o,c,o,i(1),b)

the rule changes into the following:

IF mal(d(_),o,c,o,i(3),b)
THEN buy.

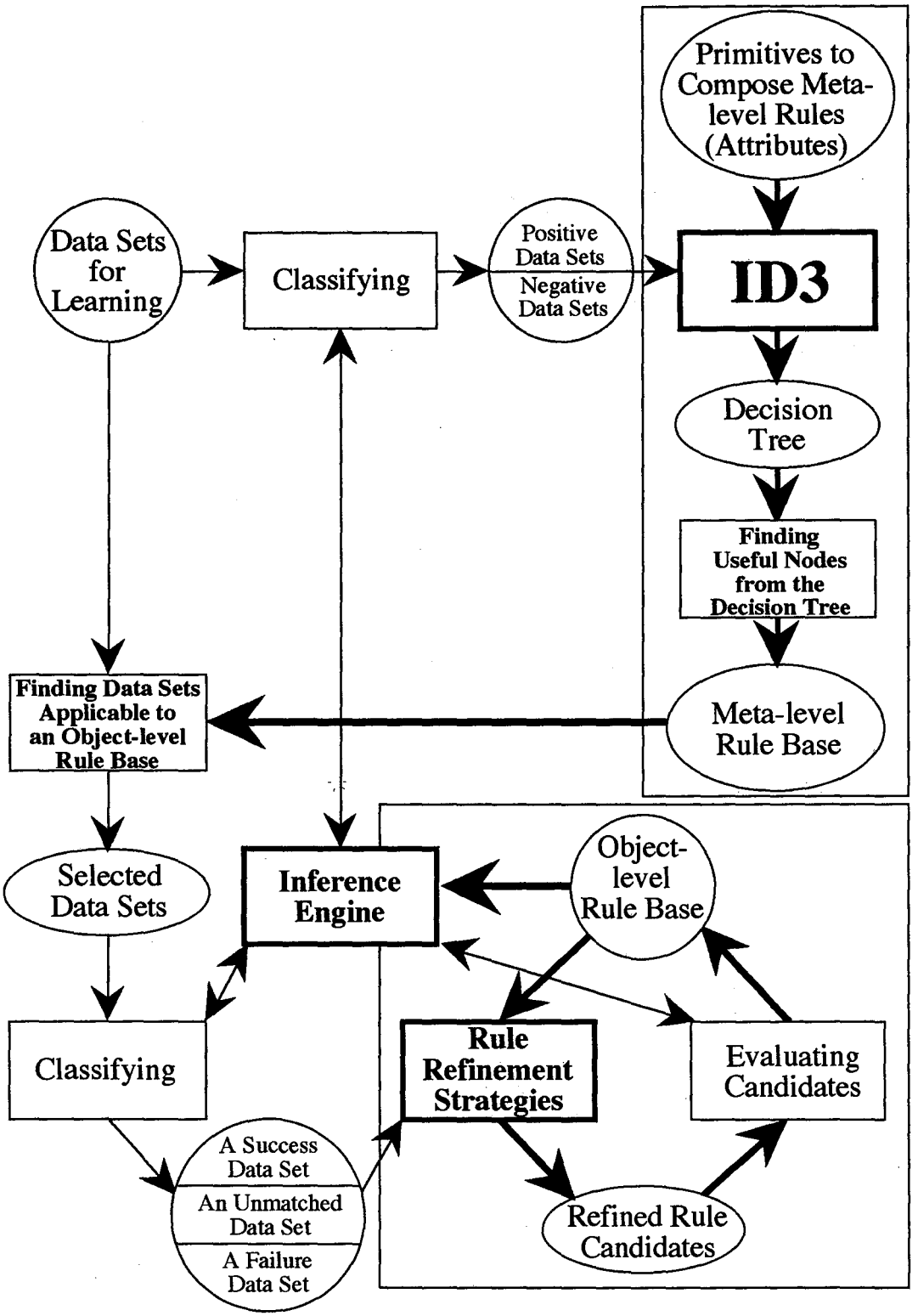


Figure 3. Unifying Rule Induction and Rule Refinement

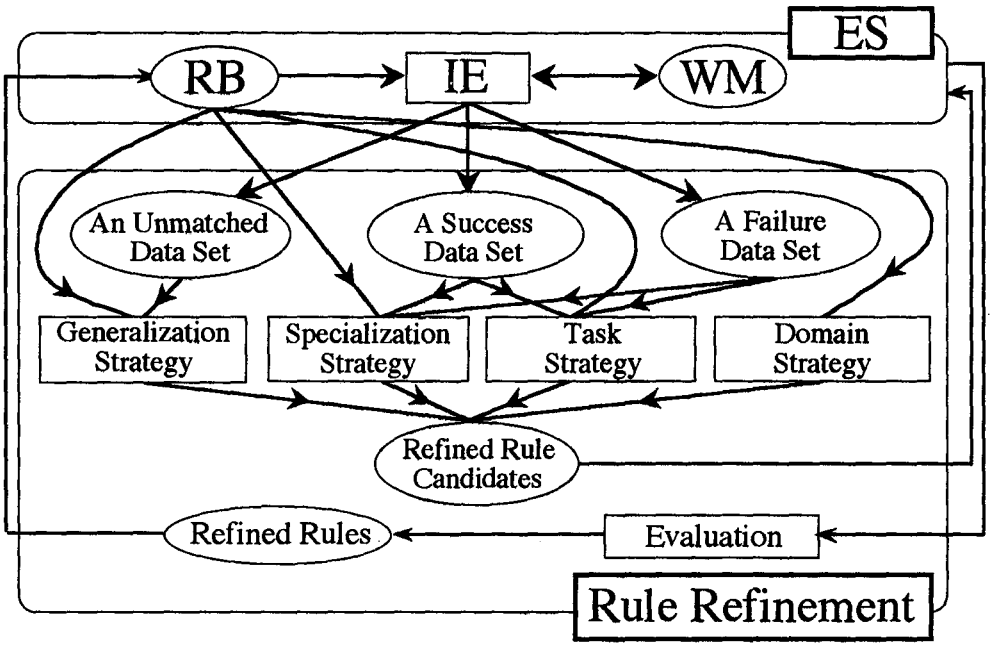


Figure 4. A Rule Refinement Subsystem

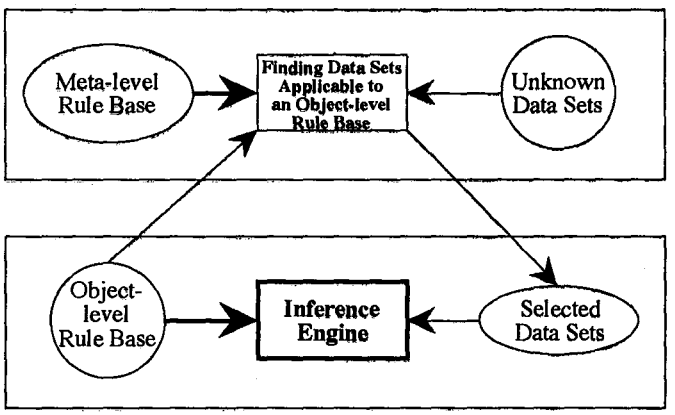


Figure 5.
System Organization at the Time of Execution

Table 3: An Experimental Result (1)

		The Number of Selected Names		After 1 Month	After 2 Months	After 3 Months
A)	Initial Object-level Rule Base	1	517	101	104	108
		2	385	102	105	112
		3	385	102	105	112
B)	Refined Object-level Rule Base	1	677	101	104	108
		2	543	104	105	108
		3	543	104	105	108
C)	Meta-level Rule Base + Initial Object-level Rule Base	1	96	102	106	109
		2	17	116	105	108
		3	73	99	97	110
D)	Meta-level Rule Base + Refined Object-level Rule Base	1	16	112	121	134
		2	16	141	183	176
		3	53	100	97	115
E)	Nikkei Average		—	102	104	106

	Data Sets for Learning in A–D 1984 Jan.–1987 Sep.	Unknown Dats Sets in A–D	Meta-level Rule in C–D
Upper Row	1st week	2nd–4th week	α
Middle Row	3rd week	1st,2nd,4th week	β
Lower Row	3rd week	1st,2nd,4th week	γ

Table 4: Rule Base(1)

D-1
(α) Commercial Names,length_TSMA3>2,value_TFER<=89,value_MS<=45
mal(_,o,3-6,c,o,_,i(_),b,3-6) mal(_,_,_,d(_),o,16,i(_),b,2-9) mal(_,_,_,_,_,16,i(_),b,2-9),mel(_,i,d,i) mal(_,_,3,_,_,_,d(_),o,_),division(minus)
D-2
(β) Financial Names,trend_TSMA2<=5,value_OWPI<=75,value_CPI<=72
mal(d(3),_,2,c,o,_,i(_),b,_) mal(d(_),o,_,i(_),_,_,i(_),b,6-10) mal(_,o,i(_),b,i(_),_),mel(_,i,d,i) mal(_,_,_,i(_),_,1-3,_,o,_),division(minus)
D-3
(γ) Financial Names, length_TSMA2<=2, length_TSMA3<=1, value_OWPI<=75, value_CPI>72, length_YGB<=4,value_YGB<=60
mal(d(_),o,_,c,o,1-2,i(_),b,11-15) mal(d(_),o,_,_,o,_,i(_),b,11-12) mal(_,_,_,_,_,1-2,i(_),b,14-15),mel(_,i,d,i) mal(_,b,_,d(_),b,_,i(_),b,13-17),mel(_,i,d,i) mal(_,_,_,_,_,o),division(minus)

Table 5: An Experimental Result (2)

		The Number of Selected Names		After 1 Month	After 2 Months	After 3 Months
A)	Initial Object-level Rule Base	1	273	102	106	110
		2	188	103	107	116
		3	161	99	102	104
C)	Meta-level Rule Base + Initial Object-level Rule Base	1	146	103	107	113
		2	172	103	108	116
		3	17	111	111	109
D)	Meta-level Rule Base + Refined Object-level Rule Base	1	98	111	112	116
		2	226	105	110	117
		3	11	105	101	113

	Data Sets for Learning in A,C,D	Unknown Dats Sets in A,C,D	Meta-level Rule in C–D
Upper Row	1985 Jan.1st –1985 Dec.4th	1986 Jan.1st –1987 Sep.1st	δ
Middle Row	1985 Jan.1st –1985 Dec.4th	1986 Jan.1st –1987 Sep.1st	ε
Lower Row	1986 Jan.1st –1986 Jun.4th	1986 Jul.1st –1987 Sep.1st	θ

Table 6: Rule Base(2)

D-1
(δ) Commercial Names,value_OWPI<=69,trend_YGB<=3,value_MS>33
mal(d(_),o,2-6,c,o,1,i(_),b,_) mal(d(_),o,3-6,_,o,_,i(_),b,_) mal(c,_,_,i(_),b,_,i(_),b,2-6),mel(_,i,d,i) mal(_,b,c,o,d(3),o),division(minus)
D-2
(ε) Financial Names,value_OWPI<=69,value_CPI<=84, length_CR<=12,value_TFER<=74
mal(_,_,_,_,o,_,_,b,12-14) mal(_,_,_,_,o,_,i(_),_,10-14) mal(_,_,17,_,_,_,i(_),b,_),mel(_,i,d,i) mal(_,_,_,_,_,_,d(_),o,4-6),division(minus)
D-3
(θ) Commercial Names,trend_TSMA2<=1,value_OWPI<=16, length_YGB<=4,value_MS>32
mal(d(3),_,_,c,o,_,i(3),b,13-19) mal(_,_,17,_,_,_,_,b,_) mal(_,_,16-17,_,b,_,i(_),b,_),mel(_,i,d,i) mal(_,_,17,_,_,_,i(_),b,_),mel(_,i,d,i)

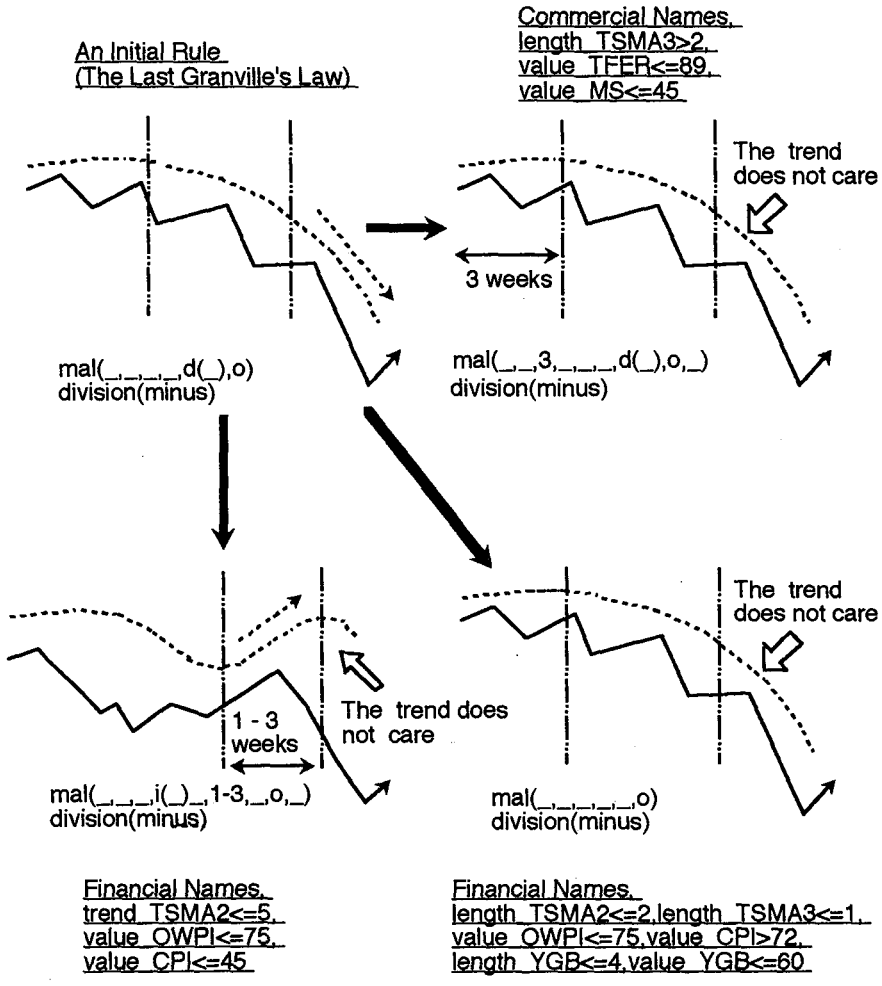


Figure 6. Refined Granvilles's Laws with Different Meta-level Rules

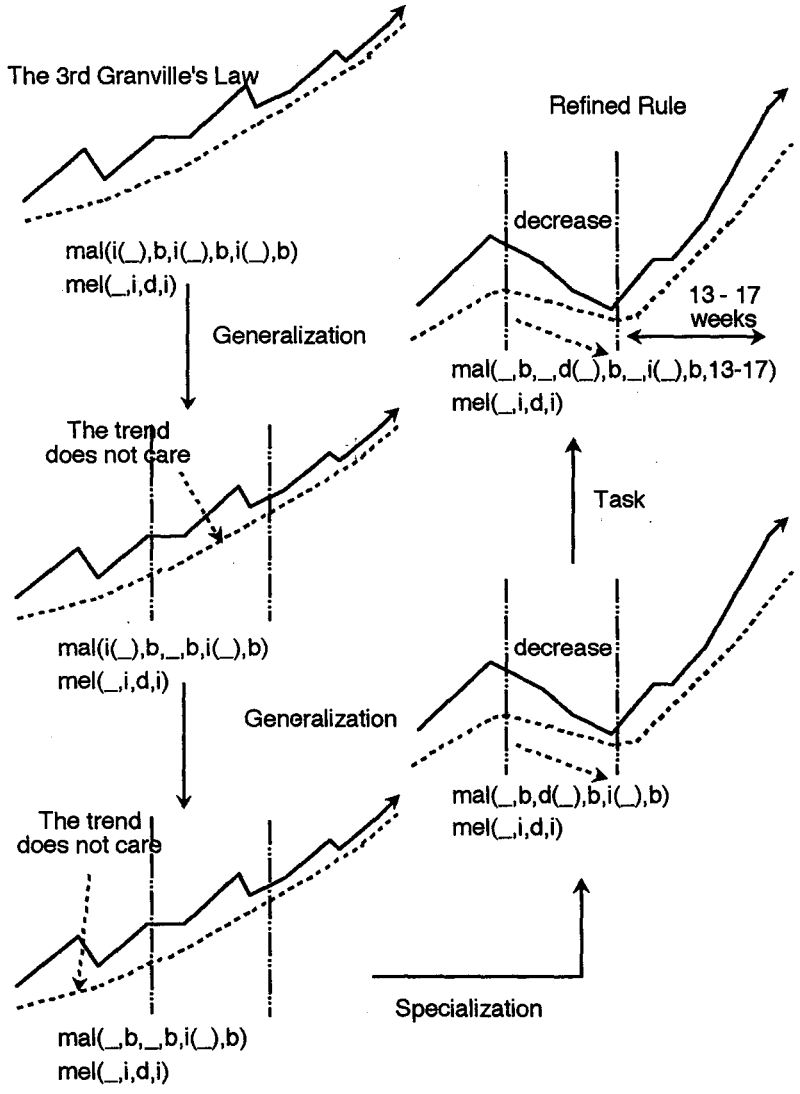


Figure 7. An Example of Refinement Process

(3) DOMAIN STRATEGY

IF LHS of the rule = mal(s1,s2,s3,s4,s5,s6)
THEN Change LHS into mal(_,_,s1,s2,s3,s4)

If we would apply the above domain strategy to Granville's first law, we could get the following new rule:

IF mal(_,_,d(_),o,c,o)
THEN buy

(4) TASK STRATEGY

IF LHS of the rule = mal(s1,s2,s3,s4,s5,s6)
THEN Change LHS into
mal(s1,s2,news1,s3,s4,news2,s5,s6,news3)
newsi = segment length of MAL or MEL
or distance between MAL and MEL
or area enclosed by both MAL and MEL

If we would apply the above task strategy with newsi = segment length of MAL to Granville's first law, we could get the following new rule:

IF mal(d(_),o,_,c,o,_,i(_),b,>8)
THEN buy.

(>8 means that the last segment of MAL is longer than 8 weeks)

3.3 Rule Execution

Finally, as shown in Figure 5, at the time of execution of a refined object-level rule base, the meta-level rule base tries to select data sets applicable to the refined object-level rule base from given test data sets and the inference engine tries to match the selected data sets with LHSs in the refined object-level rule base.

4 Experiment and Comments

Figure 6–7 are execution examples of rule induction and rule refinement. In order to evaluate our framework for unifying them, we have applied three kinds of training data sets(1–3) and test data sets to four kinds of rule base(A–D), as shown in Table 3. We let the capital index for buying stock names selected by each rule base at the investing point just one hundred. Table 3 shows how the index would change after one, two, and three months. Compared A with B in Table 3, it turns out that a refined object-level rule base does not get high capital gain just by itself without a meta-level rule base. Then compared A and C, we find that a meta-level rule base does not give an initial object-level rule base high capital gain and so the number of selected stock names decreases, in spite of that it finds a data set applicable to an initial object-level rule base. It implies that a meta-level rule base has some incompleteness about the rule space as well as an object-level rule base and so does not work well just together with the initial object-level rule base. Furthermore compared C with D, it shows that a refined object-level rule base gets high capital gain all the time, together with a meta-level rule base.

Finally compared A with D, we understand that we can change an initial object-level rule base with low performance into a refined object-level rule base with high performance by learning a meta-level rule base. It is said that even human experts, such as technical analysts, have the difficulty to get the same capital gain as Nikkei Average (which is similar to Dow Jones Average). So compared D with E, it tells us that a learned refined object-level rule base, together with a meta-level rule base, has the performance beyond that of human experts. Table 4 show us three kinds of refined object-level rule bases together with a meta-level rule base to three kinds of data sets. They might be anomaly from Granville's Law in a stock market technical analysis.

In this experiment, however, some test data sets are before a training data set and so this experimental environment looks strange, considering from real investment. Table 5 is the evaluation of our framework in the environment near to real investment and shows that refined object-level rule bases together with a meta-level rule base (in Table 6) have the best performance.

5 Conclusion

Although technical analysts in a stock market is so ill-defined task, our framework has learned a refined object-level rule base together with a meta-level rule base which has good performance, comparing to human technical analysis in a stock market. This experimental result implies that the framework could make an initial rule space more powerful than before learning, together with learned mete-level rules. Now we are going to confirm the basic nature of our framework for unifying rule induction and rule refinement, using a test data set which is so different from a training data set at the point, such as a data set from 1990 to 1992 when Nikkei Average is going down and down.

References

[1] A.Ginsberg, S.Weiss, and P.Politekis, "Automatic Knowledge Base Refinement for Classification systems", *Artificial Intelligence*, 35, pp.197–226, 1988.

[2] I.Gouhou, *Technical Analysis for Stock Market*, Japanese Economic Press, 1986.

[3] J.E.Granville, *A strategy of Daily Stock Market Timing for Maximum Profit*, Prentice-Hall, 1960.

[4] J.R.Quinlan, "Induction of Decision Tree", *Machine Learning*, 1, pp.81–106, 1986.

[5] T.Yamaguchi, "A Technical Analysis Expert System in a Stock Market", *Future Generation Computer Systems*, Vol.5, No.1, pp.21–27, 1989.

[6] T.Yamaguchi, "An Incremental Knowledge Refinement System Based on Search, Task, and Domain Strategies", *Proceedings of Pacific Rim International Conference on Artificial Intelligence*, 1, pp.601–606, 1990.

[7] T.Yamaguchi, and Y.Tachibana, "A Technical Analysis Expert System with a Knowledge Refinement Mechanism", *Proceedings of The first International Conference on AI Applications on Wall Street*, pp.86–91, 1991.

Experiments with Optimal Stock Screens

Everett J. Rutan, III, CFA
Dolcyna Research
54 White Cedar Drive
Madison, CT 06443-1652

Abstract

Screening--creating portfolios of stocks on the basis of summary statistics--is the most commonly used quantitative equity tool. It includes simple stock selection, proprietary indicators, ranking methods and multivariate regression. There is, however, no theory justifying the technique, or even accepted guidelines for building and testing a screen.

Optimal screening is an empirical approach which builds stock selection rules from portfolio objectives. It includes or excludes variables based on their contribution to explicitly desired portfolio characteristics like return, variation of return, "batting average" and size. Optimization structures the process of developing, testing and validating a screen.

Screening has been greatly facilitated by, but does not require, inexpensive computing power and standardized databases. Optimal screens are computer-intensive, and rely on genetic search optimization for their implementation.

1. Introduction

1.1 What is Screening?

Screening is the selection or grouping of stocks because their summary statistics fall in a certain range. Graham and Dodd [1] cite--without necessarily recommending--many rules of thumb using one or two data items to pick stocks. Looking for high dividend yield, low price to earnings (P/E) ratio and low asset valuation (price/book value) dates from the 19th century. Measures of price and earnings momentum may be as old. With computers and easy availability of data, it's not unreasonable to suggest that almost every data item has been part of someone's selection criteria.

Screening is probably the most commonly used quantitative tool for stock selection and equity portfolio construction. All of the commercial providers of financial data and software provide a screening facility: Barra, CompuServe, COMPUSTAT, DAIS Group, Disclosure, DRI, FactSet, Ford, IBES, Lotus OneSource, Moody's, Frank Russell, Value Line, Vestek, Wright Investors' Service, Zacks. (This is only a cursory sample, my apologies to those left out.) The research departments of most brokerage firms publish lists of stocks which meet certain quantitative criteria, and report the performance of these standard screens over time. Any number of firms have proprietary models which rank stocks, and which are available to clients for hard or soft dollars.

Screening is much more common than normally supposed. It embraces many other techniques. When Prudential Securities or Abel-Noser present a weighted factor model, that summary indicator is used to rank and ultimately group stocks into portfolios. When Value Line applies its formula to come up with "timeliness," and "safety" rankings, these become the basis for further screening. The use of these derived indicators can become quite involved. Some portfolio managers like to buy, not the top Value Line ranked stocks, but those which have just moved up a category, say from 3 to 2 or from 2 to 1. Others will combine the results of different models with more primary data and build multi-factor screens on top of multi-factor indicators.

Most academic work on valuation anomalies implicitly rely on screening. The typical study divides a universe of stocks into deciles based on the statistic of interest, and analyzes the divergent performance of these portfolios over time. Similarly, statistical work using multivariate regression sets up another indicator--the predicted return from the estimated equation--and ranks stocks accordingly. Jacobs and Levy [2] in an impressive summary of the field, attempt to bring sense and consistency to and measure the importance of valuation anomalies using a 25-term regression (which could, of course, be used to select stocks).

1.2 Why Screening?

There is an accepted but little used theory for portfolio construction. Given expected returns, expected risk, and risk aversion, the Capital Asset Pricing Model (CAPM) suggests quadratic optimization as the way to compute an optimal portfolio. This is the basis for services like Barra, which provide risk measures, correlations and optimization algorithms. But the lists of stocks fed into such algorithms are often the result of some screen. And many more

portfolios are created by equal dollar or capitalization weighting of such lists.

There are several reasons for the prevalence of screening. It is a very simple strategy to implement. While greatly enhanced by computers and electronic databases, screening can be done by hand. Screening is intuitive: a screen can be devised which reflects any theory, story or fad. It seems reasonable that low P/E stocks are undervalued, or that earnings momentum is a good thing, or price momentum, or high price to book value. And if they work individually, then some combination of them all should be even better.

Other than the dividend discount model and its extensions, there aren't any theories relating the wealth of stock market data to expected risk and return. This isn't to say there is a lack of evidence. Almost every indicator used is backed by a good story, the belief of the user and an empirical study. Every method that sees the light of day seems to work with a modest degree of accuracy on average over time. And, as certainly, there are periods when it doesn't work at all.

1.3 Problems with Screening

In addition to a lack of a theoretical basis, there are a number of empirical difficulties which are not recognized by standard screening methodologies and which adversely affect their results. First, all large databases will have a certain error rate. They will be there due to mistakes made copying the data if nothing else. The extensive nature of quantitative work means that it is not practical to verify and correct all of the figures used.

Second, accounting differences mean that data is not absolutely comparable. Statistics from two different companies, or from the same company at two different times, will not measure the same things. Again, there is no way to correct for this.

Third, timing differences alter the information content of the data. The "latest" numbers at any point in time reflect different accounting periods, different reporting lags, and even different frequencies. An earnings report announced yesterday seems likely to have more impact than one which is three weeks old, even if they refer to the same calendar period. Financial data is released quarterly or annually, while prices change every day. Earnings are reported quarterly, but earnings estimates may also change daily.

These difficulties imply that any relationship between ex ante data and ex post returns will be fuzzy at best. It is reflected in the fact that most screens are not particularly consistent. But "advanced" screening methods do not properly correct for them either. Scoring and multivariate regression are inherently monotonic: higher values of a statistic make a stock more desirable or less desirable, but not both, and not differently over different value ranges or over different sets of values for other data items used.

Econometric analysis appeals to theory to develop a correct functional form for the relationship among variables and error processes. Concepts like substitution (more of one factor can replace less of another) and marginal utility (the more of one factor the lower its additional impact) might eventually find a place in stock selection models. Understanding of the source of error in the return relationship may lead to appropriate estimation procedures. But for now, lack of theory is what screening is all about.

1.4 Not by Return Alone

An examination of the literature--academic, financial press, brokerage reports--would lead one to believe that the purpose of screening is to generate portfolios with the highest total return. But practice tells a different story. Even the way the results are presented demonstrates this.

The result of a screen is always a list of stocks. Quantitative analysts never want to stand behind an opinion on a single firm, perhaps a defining characteristic. Returns are presented as averages, preferably over time. In some cases absolute return is hidden behind measures like differential return between top and bottom ranking groups, or the number of periods a group out-performs a benchmark like the universe average return.

A portfolio manager's response to a quantitative screen is often to use it as one more input into the construction of a portfolio. There is a tendency to believe that the best stocks can be culled from a given list. There is no evidence as to whether this "cherry picking" actually adds value. Portfolio managers insist that it does; analysts typically insist that it does not.

From a portfolio manager's or portfolio construction point of view, superior average return is only one requirement. There must be a reasonable number of stocks in the portfolio to insure diversification. The selection must be investable, that is have market capitalization appropriate to the funds to be invested. The portfolio's return shouldn't be the result of one stock's return: most of the firms selected should contribute by showing above average performance. Over time, neither the variability of the return nor the turnover should be too high.

2. The Best E/P Ratio

2.1 E/P Performance by Decile

In order to clarify the concept of an optimal screen, this section builds up to such a result starting with a traditional screening problem and methodology. The earnings

multiple, or price/earnings ratio, is a familiar component of many stock selection algorithms. Its inverse, earnings yield--defined as the trailing twelve-months earnings divided by current price--is used here in order to avoid excluding companies with negative earnings. The data comes from Value Line, and covers a slowly changing universe of just under 1600 firms. [3]

The optimal screen to be developed is loosely described as finding the best E/P ratio. More precisely, determine an upper and lower bound on the E/P ratio, which, when used to select stocks, yields the most desirable portfolio. At this point the problem description is deliberately left vague.

One common way to approach the problem is to sort the firms by E/P ratio and compute the performance of each decile. These portfolios are a bit large--about 150 firms each--and the cutoff points are arbitrary and chosen without respect to return. But such a strategy does give an idea as to whether high or low earnings yield is preferable, and whether it changes over time. Deciles are measured relative to the universe of stocks considered, and therefore they are robust with respect to changes in the general level of prices or earnings.

Table 1
3 Month Returns to E/P Decile

E/P Decile	Total Return 1/1-3/31/86	Total Return 4/1-6/30/86
1	7.69	-9.91
2	10.79	-0.13
3	10.56	5.32
4	11.92	6.82
5	12.44	3.36
6	14.60	3.01
7	12.61	2.51
8	14.46	5.71
9	17.00	8.16
10	19.22	2.26
All Firms	13.21	2.74

Treating the first quarter of 1986 as "history," and the second quarter as "forecast," Table 1 shows how looking at deciles would have fared. Higher E/P ratios are generally favored, and, in fact, the tenth decile has both the highest earnings yield and the highest total return in the first quarter. However, a tenth decile strategy would have been the third worst portfolio in forecast. The ninth decile is only the second best in the first quarter, but turns in the best return one quarter later. Perhaps a better strategy would be to use both deciles, or perhaps some stocks from each, or maybe if the very highest earnings yields were excluded, or ....

Of course, one would want to do this analysis for many successive quarters, and look for some stability in the results. Screens are typically build by "backtesting" or simulating the performance of one set of rules after another until the result seems satisfactory.

2.2 The "Best" E/P Range

Because decile boundaries are arbitrary, it does seems likely that some variation in the ranges will improve the portfolios. There are a number of ways in which this could be done. The universe could be divided into smaller parcels, say percentiles, and examined the same way as deciles. Alternatively, one could compute the return of all fixed-sized portfolios of firms with contiguous earnings yields. These are exhaustive search methods.

Both approaches--using fixed cutoff points or fixed portfolio size--are limited by our lack of knowledge of the shape of the solution. If the E/P ratios used for selection are moved off of percentile boundaries, if the portfolios are a bit smaller or a bit larger, the return might be higher. The number of possibilities makes it difficult to test them all in a reasonable amount of time.

In this case, however, the range of earnings yields which brackets the portfolio with the highest historic return is easy to determine. Simply find the firm with the highest return for the period and use earnings yield values marginally above and below. A search for the best E/P range in the first quarter of 1986--conducted by more scientific means--has exactly this result. It returns a very narrow range of earnings yields which includes one firm with a total return of 31.8% in the quarter. Of course, this solution has all the characteristics of an over-fitted statistical model: it perfectly explains the past and has no forecasting power at all. The same screen also selects one firm in the second quarter of 1986, with a negative return of 47.8%!

2.3 Best E/P with Restrictions

The "best firm implies best range" solution simply

emphasizes that the purpose of screening is to build portfolios not to pick stocks, and portfolios are judged by factors in addition to return. The search for the best E/P range was also conducted with a minimum firm size of 30. For the first quarter of 1986 it selected a range which spanned most of the 9th and 10th deciles (but not all of either) containing 217 firms and having a portfolio return of 18.2%. In the next quarter the same screen selected 314 stocks with a return of 5.2%. These are comfortably above the universe average return, but not as good as the best decile in either quarter (the search on first quarter data was terminated after a fixed period of time).

Both of these portfolios are still probably too large, indicating the need for an upper limit on the number of firms. And results from one quarter are anecdotal, not conclusive (though the same calculations performed over several years yield similar results each period).

2.4 Best E/P and Growth Rate and ...

Any attempt to extend the use of fixed groupings to multiple indicators soon gets out of hand. With one variable there are ten deciles. With two data items there are 100 cross-deciles, with an uneven distribution of firms. With three indicators there are 1000 cross-deciles, and so on. A similar combinatorial explosion occurs if fixed sized portfolios are used. The problem becomes tractable only when placed in a search optimization framework.

3. Optimal Screens

3.1 Definition

An optimal screen is the solution to a portfolio selection problem which maximizes an objective function subject to a set of constraints. The objective function may consist of more than one term, weighted to reflect relative importance. The constraints restrict the solution to acceptable portfolios based on management style.

An optimal screen consists of a list of the names of various summary data items about a stock, such as P/E ratio, expected growth rate, return on assets, etc., and for each a high or low value, or both. Stocks whose summary statistics fall in the specified ranges form portfolios which, historically, have met the stated constraints and yield a higher objective function value than any other portfolio which also meets the constraints. If the relation between factors and returns is stable, the optimal screen will show good forecasting accuracy as well. An optimal screen is like an optimal control problem, in that the solution is not a portfolio, but a rule for building a portfolio.

3.2 Objectives

Screening is a portfolio selection process, not a stock selection process. Therefore it should be judged by criteria relevant to the portfolio, not individual stocks. These criteria can be used to build an objective function to either measure the results or drive the development process. The following measures are offered:

Return which may be measured as total return or capital gain, equal or dollar weighted.

Risk dispersion of return.

"Batting Average" the percentage of firms in a portfolio which outperform a benchmark, such as the universe average or zero return.

Turnover often measured as a charge against returns for trading costs.

These measures may all be stated as percentages and tend to have reasonably similar natural ranges. This simplifies weighting them in a single objective function.

3.3 Constraints

Some aspects of the portfolio may be best stated as constraints. Again, the following items are offered:

Size the size of the portfolio may be measured in terms of number of firms selected, total capitalization, or both.

Style a restriction of the universe based on the manager's portfolio commitment, such as small cap, growth, income, etc.

3.4 Rebalancing, Horizon and Averaging

Three further aspects of the problem must be specified in order to measure the results. First, the interval at which portfolios are built by re-selecting them with the screen must be set. This process is traditionally called "rebalancing." Both the firms selected and their weights in the portfolio are adjusted.

The timespan for optimization may cover one or more rebalancing intervals. It is this horizon over which the objective function is evaluated. The elements of the objective function from each interval in the horizon can be combined in a number of ways. Total return may for optimization may be calculated as a compound or a simple average of the returns of the rebalanced portfolios. Dispersion of return may be measured intra or inter-period. Batting average could be measured as the average over each rebalanced portfolio, or the minimum over each.

3.5 Implementation

Screening has developed haphazardly because the technology to treat it as an optimization problem hasn't been available. Screening is used because the relationship between financial data and returns is not known. The description of the problem shows that the solution space is a combinatoric nightmare. In short, it is a problem made for genetic search optimization.

A **genetic algorithm** [4, 5] is an advanced optimization or search procedure that is based on the concept of natural selection. Solutions to a problem are treated as organisms whose fitness is judged by some measure of performance on the problem to be solved. Better solutions are more likely to be chosen for further modification, generating new solutions to be tested. Less fit solutions are discarded. Rather than examine all possible portfolios or all possible cutoff values for screening variables, a genetic algorithm evolves a set of values which yield a good portfolio.

Genetic search optimization is an automated form of the generate-and-test method used to develop most screens. Automating the procedure certainly reduces the effort required. But by focusing the developer on questions of portfolio design--what is a good portfolio--it puts the problem in a new context.

Both the "one-firm" and the constrained screens discussed in sections 2.2 and 2.3 were derived using a genetic search technique.

3.6 Comments

The suggested objectives and constraints tend to enter screen construction informally. Deciles and other arbitrary cut-offs are used to insure enough stocks are selected. Parameters are adjusted so that turnover is limited and accuracy improved. "Cherry picking" and combining data from diverse screens are probably aimed at further improving portfolio characteristics.

There are several advantages in reformulating the problem as an optimization. First, implicit concerns become explicit. Their importance is clearly stated and balanced against other objectives. They are measured, and success or failure easily noted.

Second, the problem may have more than one answer. The multiplicity of successful screens offered commercially suggests more than one good portfolio, or more than one way to describe it. Optimization methods recognize that more than one solution may exist, and provide ways to find them.

Third, different investment strategies may have different answers. Screening and ranking tend to reduce the portfolio management problem to one dimension. The top stocks are buys and the bottom stocks are sells. In fact there is no reason why purchase and sale criteria should be opposites; the only requirement is that they should not include the same stocks. Traditionally, sell candidates have been listed as an afterthought to screens developed for building portfolios, that is, finding stocks to recommend. By setting the objective function to reward low return, low batting average, and low dispersion of return, an optimal screen will return a portfolio of stocks best shorted.

As with any form of analysis, optimal screens can only be calculated using historic data. There is no assurance that past returns will be any guide to future performance, as the disclaimer goes. Implicit in the concept of an optimal screen is an assumption of stable relationships necessary for forecasting. There is an estimation problem similar to that of a statistical problem. Data problems do not go away. Some data transformations may be more robust than others. Too many terms in the screen my "over-fit" the sample and lead to poor forecasting performance.

4. Conclusions

We have only started calculating optimal screens and measuring their performance. The initial results are promising. Framing a common quantitative practice as a portfolio optimization problem brings a degree of discipline to this field.

5. References

[1] Graham, Benjamin, David L. Dodd and Sidney Cottle, *Security Analysis: Principles and Techniques*, 4th Ed., McGraw-Hill, New York, 1962.

[2] Jacobs, Bruce I., and Kenneth N. Levy, "Disentangling Equity Return Regularities: New Insights and Investments Opportunities," *Financial Analysts Journal*, May-June 1988.

[3] Value Line, Inc. *Value Screen* and *Value Screen II* , 1986-1993. This is an electronically delivered version of some of the data in the familiar *Value Line Survey*.

[4] Davis, Lawrence, ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[5] Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

A Case-Based Reasoning Paradigm for Mining Financial Databases

Peter N. Johnson
Cognitive Systems, Inc
220-230 Commercial St.
Boston, MA 02109

Abstract

With the advent of more powerful desktop computers, analysts can now begin to exploit computer learning technologies to help identify key opportunities in financial databases. Like human experts, these approaches "learn" from prior experience, as opposed to "being told" (as with rule-based systems). This paper introduces a learning technology called Case-Based Reasoning (CBR) and discusses practical applications for using CBR to research financial databases.

1. Introduction

Quantitative analysts have long used statistical methods to wade through the reams of information that flow on Wall Street everyday. In the last decade, advances in personal/workstation computers coupled with the electronic distribution of financial databases such as Value Line and Compustat [1] have enabled more sophisticated analysis to occur at the analyst desktop. More recently, technologies in machine learning, including induction, statistical and conceptual clustering, neural networks, and genetic algorithms have been added to the arsenal of weapons for data analysis. Successful applications have been constructed around these technologies for tasks ranging from bankruptcy prediction and securities fraud detection to mutual fund selection and stock picking. These applications mine financial databases to discover relevant generalities for purposes of classification and/or prediction. Piatetsky-Shapiro and Frawley [2] provides an extensive survey of these applications as well as technical detail.

An important barrier to the application of these new technologies has been the black-box nature of the underlying mechanisms. For example, while neural networks have yielded successful trading strategies [3], their predictions are, at best, difficult and at worst, impossible to explain. This compromises their utility in financial forecasting in two respects: 1) analysts are reluctant to rely on methods they don't understand, and 2) it is difficult to assess unexplained conclusions in the context of additional (outside) information sources such as news and/or rumor.

Case-Based Reasoning (CBR) is an alternative machine learning paradigm that addresses these obstacles by providing example-oriented explanations. As with all machine learning technologies, CBR is founded on the premise that experts (human or otherwise) reason from prior experience. However, unlike other techniques, CBR links its generalized conclusions to the specific motivating examples that were provided during the trading process.

For example, suppose that CBR selected IBM today in a stock-picking application. It would back up this conclusion by comparing IBM today to specific historical examples (such as MCI three years ago) along various relevant features. This is the fundamental assumption behind using machine learning for prediction. The disposition of historical examples (called *cases*) can be applied toward predicting the disposition of current cases. In other words, if I can find stocks today that look like winners did in the past (just before they went up), then perhaps these stocks will become winners themselves. Of course, the heart of this issue is "What constitutes similarity?" when comparing two stocks. In CBR parlance, this is addressed by establishing a robust *case model*, which will be discussed in Section 3.

2. The Case-Based Reasoning Paradigm

The CBR paradigm for financial forecasting is a three-step process. First, the system is trained over a target historical period to discriminate cases along a desired dimension such as performance. For example, the system is trained with historical sample data from December-1991 (or prior) to discriminate three-month performance by supplying the performance outcomes for March-1992 (or prior). Through induction (and other machine learning variations), CBR produces a decision tree that indexes the December-1991 cases in various outcome categories (*winners, so-so, losers*, etc.). Induction automatically finds the most discriminating features for categorizing cases along the target outcome category.

Figure 1 shows a small portion of a decision tree that was generated to predict bond ratings. This

application was produced using ReMind [4], a CBR development shell from Cognitive Systems. The underlying data was an abstraction from the Compustat database. (The system was trained with various computer companies, using 13 quarters of Compustat data, 100 fields for each quarter.)

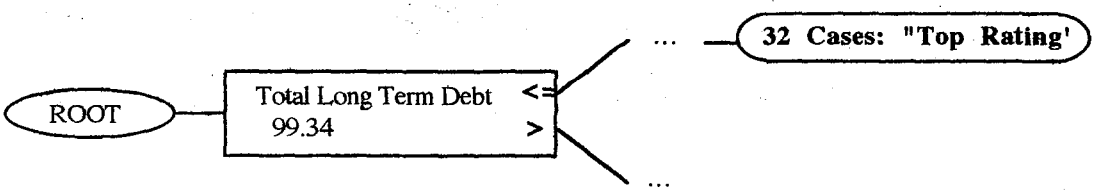


***Figure 1*Decision Tree for Assigning Bond Rating**

In the example, ReMind's inductive algorithm identified "long term debt" as the premier (root) discriminating factor when assigning bond ratings. At the leaves of the top path, are 32 cases with the top S & P bond rating.

The second step in the paradigm is to evaluate new cases (whose outcomes are not necessarily known) by traversing the decision tree to retrieve relevant historical cases. This retrieval process is rapid (logarithmic) since it merely involves following the root of the tree down to individual leaves. When values are unknown, more than one path is considered simultaneously.

It is important when back-testing the tree to evaluate cases from a "future" time period that extends beyond the predictive period (in the database) that was used to train the system. For example, if the system was trained with cases (using data) prior to March-1992, then it should be tested on cases from April-1992 or later.

In the third step, the historical cases are compared to the situation at hand and a predicted outcome is assigned through a process known as *adaptation*. In the bond rating example, a bond rating is assigned by looking at the bond ratings (and various other features) of the retrieved cases. The top path (Figure 1) retrieves 32 cases with the top bond rating. If this path is traversed when evaluating a new case, the analyst might assign it the top bond rating since there is a preponderance of evidence to support this conclusion.

Generic techniques in adaptation are presently being researched in the academic community. However, from a practical perspective, the decision-support (or interactive) approach should not be overlooked for financial applications. Before drawing any final conclusions, many analysts prefer to examine the new case with each retrieved case to "eyeball" the relevant similarities and differences.

In the bond rating application, Apple Computer was evaluated on September-1992. The top path, with 32 highly ranked companies was traversed. Figure 2 shows the first sheet in which analysts compared Apple to one of these 32 retrieved cases: E-Systems on December-1988. This comparison screen (shown sideways) was provided by ReMind.

The three-step (train, retrieve, evaluate) paradigm for CBR is repeated many times by the analyst through various historical periods. This back-testing procedure is critical for model refinement and for gaining confidence in the application's performance.

The bond rating application discussed earlier was tested with ReMind's "Test Clusters" facility. It was able to exactly predict the bond rating for over 85% of the test cases and was off by one for an additional 14% of the test cases. However, these raw figures are only meaningful in the context of considering the general distribution of the population. Weiss and Kulikowski [5] provide an excellent treatment on the subject of evaluating model performance.

3. The Case Model

As with other machine learning approaches, the key to success lies in case representation. In its most basic form, a case may be represented by the raw financial data embodied in a time-oriented database record (eg. IBM on 4/1/92). However to yield true predictive power in a learning application, a more sophisticated case model may be required. At least two types of extensions to this basic model should be considered: 1) *time series* representations, and 2) *knowledge base* extensions.

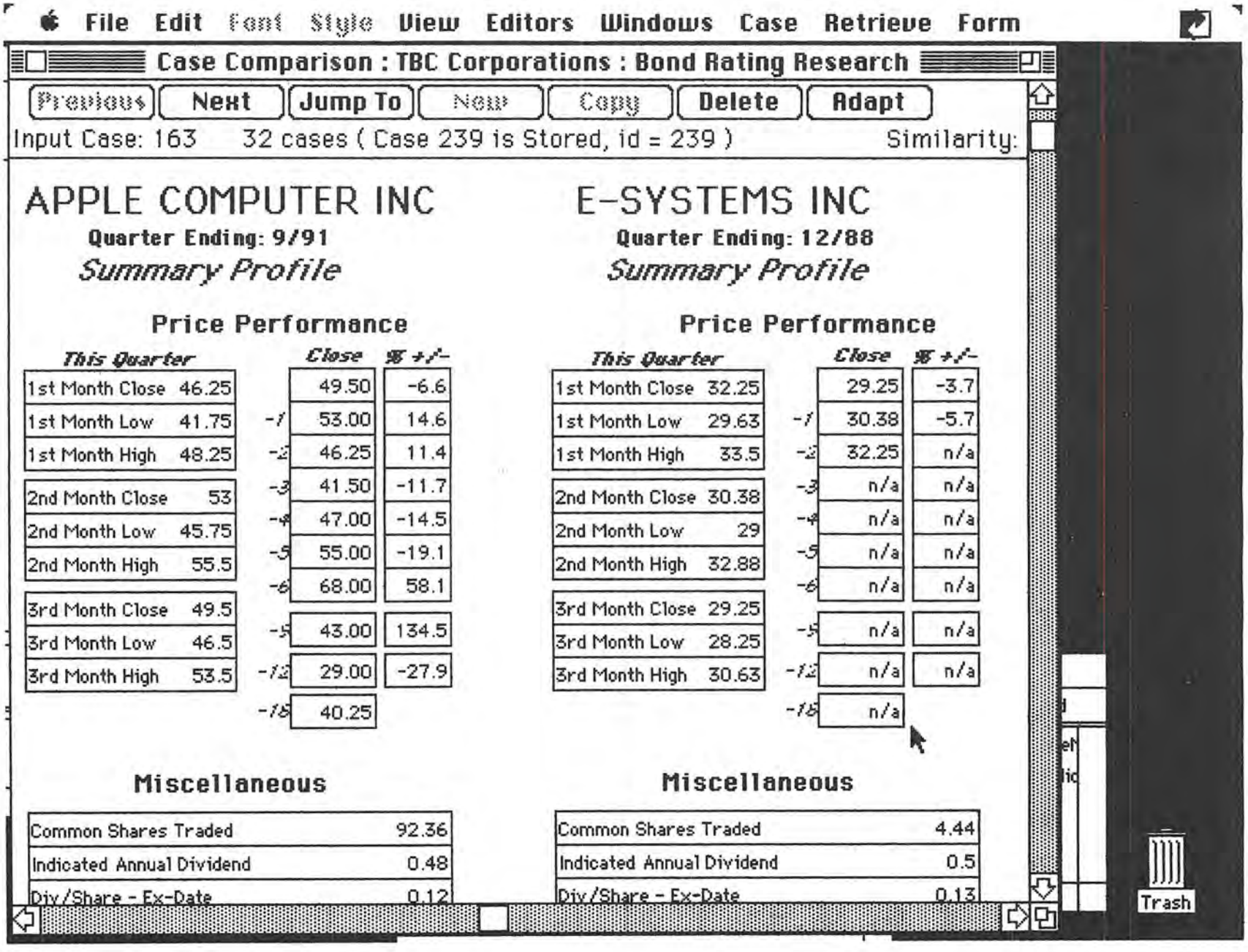


Figure 2: Case Comparison

Time series calculations can be used to extend the case representation from a single point in time to a time range, thus taking into account changes in critical features such as earnings. As an example, a case might still represent IBM on 4/1/92, but also take into account critical features over the past six months. Thus, the case would not only include IBM's P/E on 4/1/92, but it might also incorporate IBM's earnings growth (or reduction) from 11/1/91 to 4/1/92.

A variety of statistical techniques are available for reducing time series data. These include moving averages, linear (or higher order) regression, and Fast Fourier analysis to capture cyclical changes. Deboeck [6] itemizes several techniques for pre-processing time series data.

In the bond rating application, the case representation incorporated slopes (and difference in slopes) over eighteen and six month historical time periods. For example, the cases looked at a current value for long term debt. They also incorporated figures for increasing/decreasing debt rates over the past eighteen and six month intervals.

Knowledge Base extensions are somewhat controversial in the academic community (refer again to Piatetsky-Shapiro and Frawley [2]) because they rely on the addition of domain (human) expertise to the case model. However, from a practical perspective, analysts welcome the opportunity to selectively meld their own expertise with the generic indexing mechanisms that provided by CBR engines. In the bond rating application, analysts use the conceptual hierarchy mechanism (available within ReMind) to represent industry hierarchies.

For example, *computer* companies are broken down into *hardware* and *software*; *software* is broken down into *systems* and *applications*, and so forth. When retrieving cases, two software companies are considered to be more similar than a hardware company and a software company (along the industry feature). Other AI knowledge representation techniques (such as frames) have also been used to enhance the case representation.

In general, the richness of the case model greatly effects the number of cases that will be needed to successfully train the system. While it may be possible (given enough cases) for an application to induce these higher level features, it may be impractical (with respect to computing power and/or development time) to work with raw data alone.

4. Conclusions and Future Directions

Preliminary results in applying this paradigm to a popular commercial database (Value Line) have been promising. In conjunction with a major Boston-based financial institution, we have consistently outperformed the Standard and Poor's 500 Index in average total return over a one-month and three-month investment horizon. These results were achieved when back-testing performance on a stock-picking application over a two year time period. Similar tests (with successful, but less impressive margins) have been achieved for predicting changes in Value Line's timeliness ranking.

As we move this application into production, we are currently exploring the following issues:

- Hybridization: providing statistics, graphics, and other learning techniques such as neural networks on a common software platform; Graphics are particularly important when evaluating the quality of a model. Plotting *actual* versus *predicted* values reveals which portion of a model is most predictive. Search techniques (such as genetic algorithms) may also be useful in supporting adaptation by finding the most effect cutoff parameters when interpreting a set of retrieved cases.

- Decision Tree Stability: From an analyst's perspective, a stock-picking model is only valuable if it is stable over a wide range of training sets.

- Decision Tree Pruning: There are far fewer learning parameters in building a CBR model than there are for building a neural network. However, some of the most important parameters relate to decision tree depth. If the tree isn't pruned at all, the system tends to respond to noise and is therefore less predictive.

In conclusion, we must not lose sight of our primary motive for choosing CBR among alternative machine learning technologies. Applications must be well-understood if they are to be widely accepted by financial analysts. We will continue to focus on solutions that are both flexible and easily explained.

4. References

[1] *Value Line* and *Compustat* belong to Value Line Publishing, Inc. and Standard and Poor's - McGraw Hill respectively, New York.

[2] Piatetsky-Shapiro, Gregory and Frawley, William J., *Knowledge Discovery in Databases*, AAAI Press/The MIT Press, Cambridge, MA, 1991.

[3] Klimasauskas, Kasimir C., *Applying Neural Networks (Part I-IV)*, PC-AI Magazine, beginning Jan/Feb, 1991.

[4] ReMind belongs to Cognitive Systems, Inc, Boston, MA.

[5] Weiss, Sholom M. and Kulikowski, Casimir, *Computer Systems That Learn*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991

[6] Deboeck, Guido T., *Pre-processing and Evaluation of Neural Nets for Trading Stocks* in *Advanced Technology for Developers*, NeuralWare Corporation, Pittsburg, PA 1992

Paper Session: Fuzzy Financial Time Series

Chair: Dan Schutzer, Citibank, N.A.

A Model-Free Trainable Fuzzy System for the Analysis of Financial Time-Series Data

with fuzzy set morphology and rule association optimization through a Genetic Optimizer

Earl Cox
CEO/Chief Technical Officer
The Metus Systems Group
1 Griggs Lane, Chappaqua, New York 10514
(914) 238-0647

Abstract

The Wang-Mendal algorithm provides a simple yet robust method of generating a fuzzy logic model from input and output relationships. These relationships can be used to model both parametric as well as chaotic-delay time series data. This paper explores the use of such techniques in interpolating financial time series data. We also examine how the algorithm can be improved through a genetic algorithm that optimizes the fuzzy set and rule components.

1. Overview

This paper describes a new technique for automatically generating a fuzzy expert system from trainable data tuples describing the input and output properties of a financial time-series system. This is not a neural network approach, but rather is a heuristic-based technique that is both tunable by the user and the underlying algorithm[1].This technique is divided into two parts: the generation of a fuzzy logic time series analysis system and the optimization of the analysis structure. This structure consists of a fuzzy logic inference system produced from the data, and a set of rules in the form,

$$\textit{if } x_i \textit{ is } Y \textit{ and } z_i \textit{ is } W \textit{ then } s \textit{ is } T$$

Which are also automatically generated from the fit of data elements into the associated fuzzy sets. From this initial pool of fuzzy sets and rules, a genetic optimization routine attempts to find the best set of rules and the optimal organization of fuzzy sets in terms of dispersion, overlap, and shape. This system is then used to make predictions across parametric and chaotic-delay time series. We will make a comparison between the way fuzzy time series analysis is performed and both conventional autoregressive and rather unconventional feedforward neural networks. That accurate time series analysis can be formulated and performed by fuzzy systems is a consequent of the fact that fuzzy systems are universal approximators. The problem of time series estimation is a mapping and transformation problem in the classical approximator sense. A time series (Z) is represented as monotonically subscripted data state,

$$Z(t) for \{k = 1,2,3,\ldots n\}$$

The general problem state for time series prediction is stated as an estimation problem. Given a time series,

$$Z(t-n+1), Z(t-n+2), Z(t-n+s), \ldots Z(t)$$

determine the value for,

$$Z(t+l)$$

That is find a transformational mapping function in the problem state R^w such that it obeys the transfer function,

$$Z(t_n) \in R^w \rightarrow Z(t_{n+l}) \in R$$

In a method analogous to the neural network approach, but with greater flexibility, the model-free fuzzy system generator assumes that we have

[1]This paper is based on an original concept developed by Li-Xin Wang and Jerry M. Mendel of the Signal and Image Processing Institute of the University of Southern California in Los Angeles. The technique was published in USC-SIPI Report #169 —— *Generating Fuzzy Rules from Numeric Data, with Applications.* The current paper is based on my implementation of the technique with some important extensions for rule and fuzzy set discovery and optimization.

sufficient predicate training sets to develop a causal relationship between the input and output model states.

2. Rationale and Approach

A model designer is often faced with a problem of quantifying model control and solution parameters in terms of their linguistic representation. If the model is well understood and mathematical descriptions are both robust and deterministic, then we can describe the system behavior with confidence. In many instances, however, the underlying phenomenology of the model—the underlying casual relationships—are not well understood, highly non-linear, and subject to multiple interpretations by a family of experts. These problems are very difficult to model. In many instances, however, the case data for a model exists, that is, we have a fundamental knowledge about the level of output when we have a series of inputs. Figure 1 illustrates this situation.

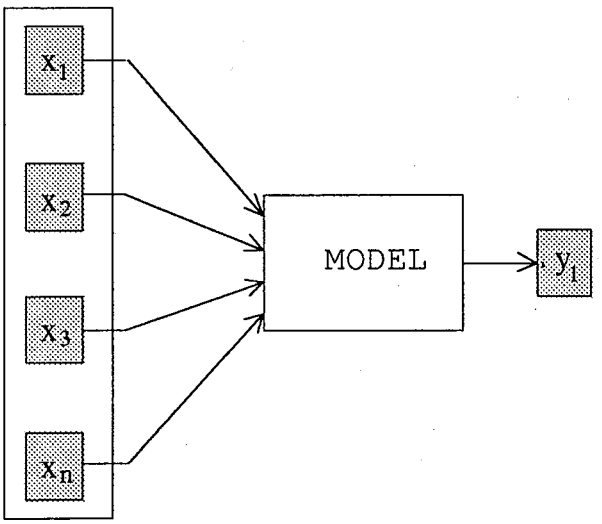


Figure 1 Input-Process-Output Schema of a Fuzzy Model

In this situation we have a mapping between an initial model state represented by the quantizied data vector $\{x_i 1...x_{in}\}$ and the problem solution state represented by the decomposed state variable $\{y_i\}$ for the i-th case of the model. This means that there is probably some transfer function[2] (g) between these states.

$$y_1 \leftarrow g(x_1,...,x_n)$$

.

.

.

$$y_k \leftarrow g(x_{k1},...,x_{kn})$$

using this relationship, we would like to derive a set of conditional fuzzy associations or rules in the form,

$$\textit{if } x_1 \textit{ is } P_{i1} \bullet x_2 \textit{ is } P_{i2} \bullet ... \; x_j \textit{ is } P_{ij} \textit{ then } y \textit{ is } C_i$$

where "P" is a predicate fuzzy region and "C" is a consequent fuzzy region. These fuzzy regions are generated directly from the data as approximations of the underlying domains. Figure 2, illustrates the schematic properties of the first step in the process—the generation of the underlying fuzzy analysis system.

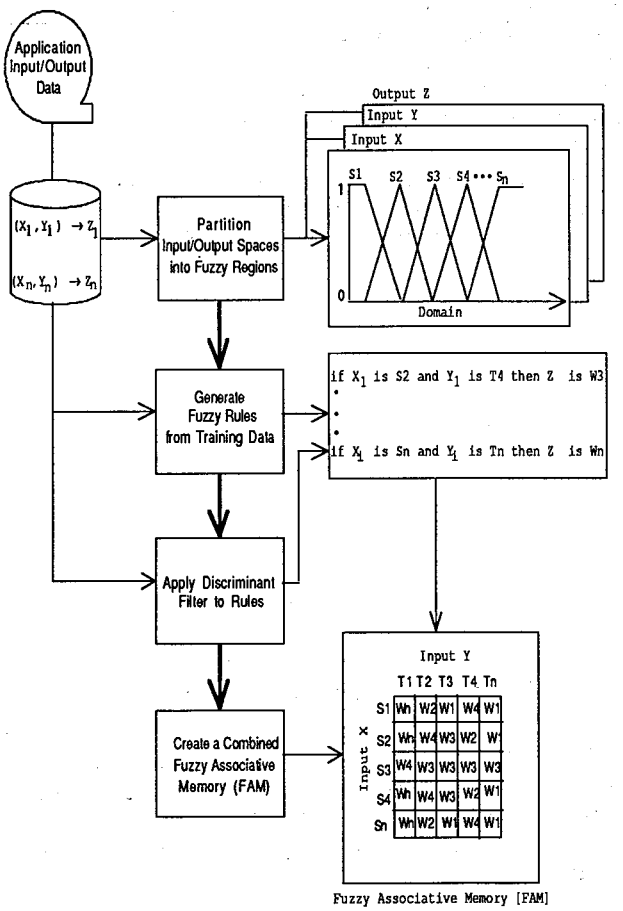


Figure 2. The Initial system configuration form Wang-Mendel

[2]With the use of the term "function" here in the broadest possible way.

3. Generating the Fuzzy System

In general then we could derive this function (g) by approximating the fuzzy system. This means that we want to treat the control and solution parameters as fuzzy sets or linguistic variables across the domains of the individual variables. A general algorithm for discovering this mapping function and generating fuzzy rules from data proceeds as follows[3],

Step 1 Organize the input and output variables into Fuzzy Regions. For reasons not discussed in this paper, the number of partitions of a variable should be an odd number. Select an appropriate number of fuzzy regions as 2N+1. Note that the number of partition regions ion each variable can be different. We are going to partition the variable across it domain. This domain is the probable minimum and maximum values associated with the variable state.

The division of the control and solution spaces into fuzzy regions is accomplished by approximating the value that lies at the center of the domain. The Wang Mendel algorithm specifies triangular functions, with the center of the domain having a value of [1] and the edges linearly sloped to zero [0]. However, you can also use standard fuzzy hedge approximators to produce PI or BETA distributions. Figure 3 illustrates how the fuzzy partition is formed.

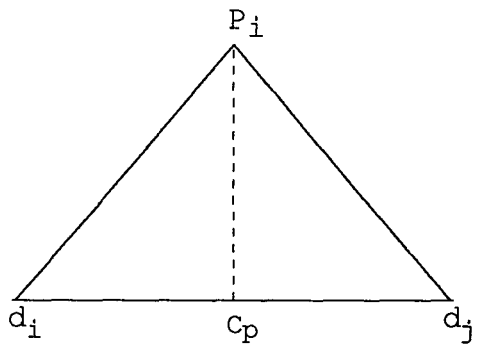


Figure 3 The topology of a partition for variable x

The partitioning process results in a set of fuzzy regions for the variable as shown in Figure 4. We are following the Wang-Mendal algorithm and using triangular fuzzy sets to represent the variable partitions.

[3]This is a brief outline. Consult the source document for a complete discussion of the algorithm.

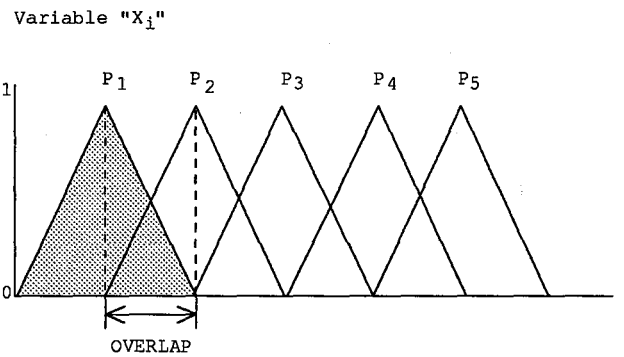


Figure 4 The partitioning of variable x into 5 fuzzy sets (N=2)

Notice that we are defining complete fuzzy regions at this point so that shouldered or trapezoidal fuzzy sets are not used since the open domain of these curves implies at extension to the end of the universe of discourse. We can view the partitioning process simply as a problem in creating an array of fuzzy sets associated with the universe of discourse for the overlying variable. Since each variable has an associated domain we can use this to partition the variable.

```
in N,Partitions;
double
lo,hi,DomRange,MidPoint,PartDomain[2];
N=2;
lo=VDBptr->VDBdomain[0];
hi=VDBptr->VDBdomain[1];
DomRange=hi-lo;
MidPoint=lo+(DomRange/2);
Partitions=(2*N)+1;
PartDomain[0]=Lo;
PartWidth=DomRange/Partitions;
for(i=0;i<Partitions;i++)
   {
      PartDomain[1]=PartLo+PartWidth;
      FDBptr[i]=FzyCreate(
         Setid[i],TRIANGLE,
         PartDomain,Parms,0,&status)
      PartDomain[0]=PartDomain[1];
   }
```

This procedure will create a set of evenly spaced triangular fuzzy sets across the domain of the variable. We can modify this algorithm to create different types of fuzzy regions along the variable space (such as PI or BETA curves.)

The vertices of each fuzzy set lie the center of the adjacent fuzzy region with a truth value of zero [0]. This is one possible division of a variable into

fuzzy regions, others are also possible. One possible modification is to increase the number of fuzzy regions in the solution variable space and apply space elongation and compression to enforce a better control space. As an example, Figure 5 shows how a variable can be partitioned to reflect the higher level of control associated with the center of the variable domain.

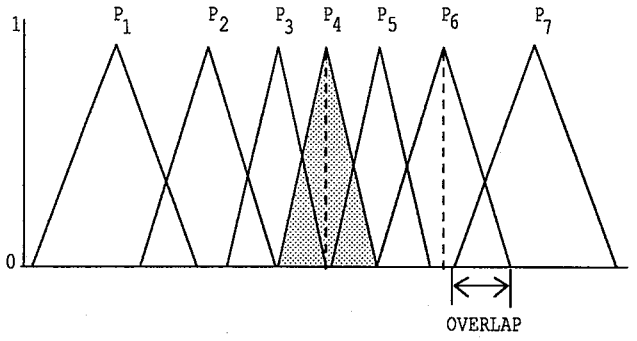


Figure 5 Changing the partitioning density of the variable with disjoint overlaps

When we cluster fuzzy regions around the center of the variable's value, the overlap of adjacent fuzzy regions will not meet at the 50% (or center) point of the neighboring partition.

Step 2 Generate Fuzzy Rules from the Data Pair Space. We use the individual data points to find their truth values in the various fuzzy partition states. When we find a relationship,

$$t_{max} = g(x_i, P_i)$$

(that is, the sample point x has its maximum truth in partition Pi) then we make an association by saying that, as an example,

$$x_1 \text{ is } P_3$$

from this selection process we will find a maximum degree of compatibility between each data point and some fuzzy partition in its over-all domain. Using this information we generate a rule,

$$\text{if } x_1 \text{ is } P_3 \text{ and } x_2 \text{ is } P_7 \text{ and ... then } y \text{ is } P_9$$

This technique produces a rule set for an NxM fuzzy model. The rule population generated can be quite extensive and we need some method to reduce the total rule set without loss of functional completeness.

Step 3 Assign a conflict resolution weight to each rule. This process exploits both the maximum truth of the various rule components but also a measure of the rule's over-all utility. In lieu of explicit weight assignment, we can re-process the training set to find the rules that exhibit the best behavior under the restrictions imposed by the training set. This process will collapse the complete rule set and generate the final Fuzzy Associative Memory.

4. Genetic Optimization

Once we have generated an initial system, it is necessary to find the combination of fuzzy sets and rules that produces the best truth membership correspondence between the fuzzy sets and the training set. This is illustrated in Figure 6. The optimization is necessary due to the way the initial fuzzy set partitioning is derived. For each variable, we select a partitioning ratio (N) that sub-divides the domain of the variable into 2N+1 fuzzy regions. A slight modification is made for solution variables (where the ratio N is Nk, and k is selected from the number of control states that contribute to the solution state.) The genetic optimization function attempts to change both the degree of overlap between adjacent fuzzy regions and the number of regions. A goodness of fit metric measures the dispersal of data elements (x1,x2,...xi) within the variable to find the point where elements in adjacent regions are mapped to maximally constrained membership values. This maximal constraint means that a point occurring in multiple fuzzy regions falls on a vertical line through the regions. The truth of these points obeys the general restriction,

$$\sum_{i=0}^{n} \mu_s(x) \leq 1$$

while the optimization engine obeys a more critical goodness of fit expressed as,

$$\sum_{i=0}^{n} \mu_s(x) \equiv 1$$

meaning that, on the vertical line, the sum of the points will always equal one. Changing the degree of overlap and the morphology of the variable will be reflected in changes in the final fuzzy associative memory (including the rule base.)

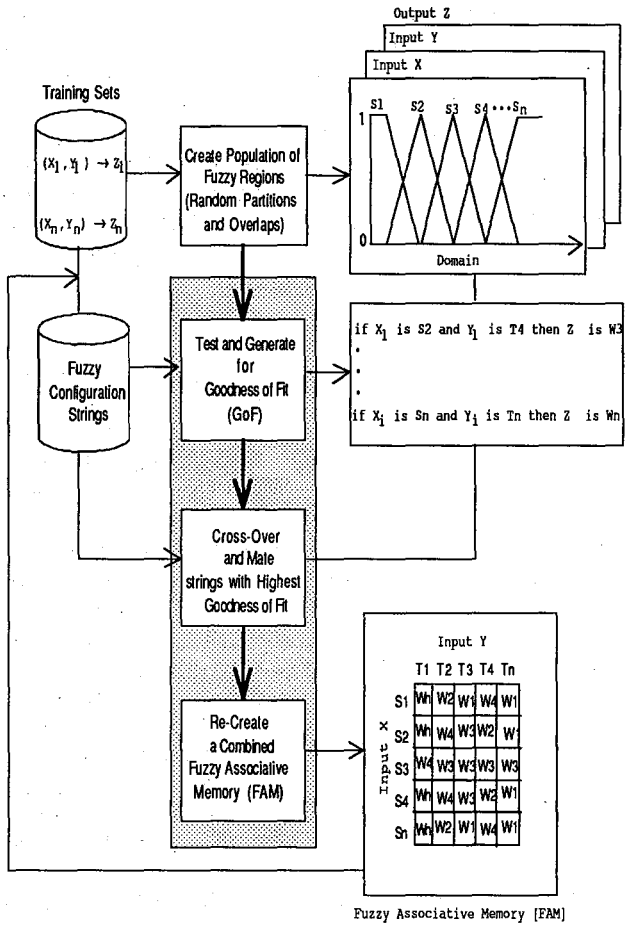


Figure 6. The Genetic Optimization Phase (Simplified)

5. Time Series Forecasts

In our prototype system the daily time series data represented rather noisy and perturbed financial transactions. An initial training set of 9078 tuples consisting of three system inputs and the current [t] value of the solution variable. In our tests we followed the analysis examples recommended by Wang-Mendal—an autoregressive [AR] model, a least-squares regression, and the fuzzy-numeric analysis model. We also followed the standard autoregressive model for time series Z(t),

$$Z(t+1)=\sum_{i=0}^{m} p_i Z(t-i+1)+W(t^k)$$

where pi is a parameter estimated from the known values of the series by substituting the known Z(t)'s into this equation and ignoring the white noise W(t) to form a set of linear equations. When these equations are solved for pi using least-squares, the solutions are the prediction-error estimates of pi. Then, ignoring the white noise W(t) and substituting the estimated pi, we obtain a forecasting model for Z(i).

6. Conclusions

Figure 7 shows the results of mapping the forecasts for the time series against the fuzzy-numeric method, the simple least squares, and the auto-regression model. As you can see, the fuzzy model is consistently better than least squares for noisy data, and, for a large sample size, gives a better approximation than the AR model.

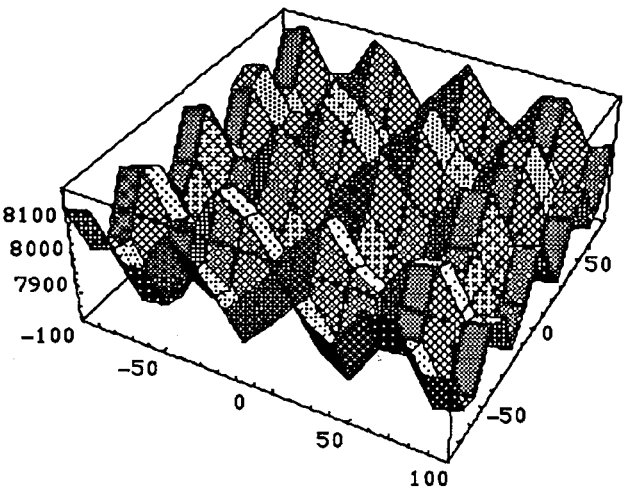


Figure 7. Least Squares versus Fuzzy Approximation

7. References

[1] Cox.E.. Adaptive Fuzzy Systems, IEEE Spectrum Magazine, Vol. 30, No. 2, February, 1993, pages 67-70.

[2] ——. Applications of Fuzzy System Models, AI Expert Magazine, Vol 7, No. 10, October, 1992, pages 33-45.

[3] ——. Fuzzy Fundamentals, IEEE Spectrum Magazine, Vol. 29, No. 10, October, 1992, pages 58-61.

[4] ——. Integrating Fuzzy Logic into Neural Nets, AI Expert Magazine, Vol. 7, No. 6, June 1992, pages 42-47

[5] ——. Solving Problems with Fuzzy Logic, AI Expert Magazine, Vol 7, No. 3, March 1992, pages 28-37

[6]——, Myths of Fuzzy Logic, AI Expert Magazine, Vol 7, No. 1, January, 1992.

[7] G.E.P.Box and G.M.Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day Inc., 1976

[8] Goldberg,D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, 1989

[9] Koza, J.R., *Genetic Programming*, The MIT Press, Cambridge, Mass.,1992

Financial Data Modeling with Genetically Optimized Fuzzy Systems

Stephen Welstead
COLSA Corporation
6726 Odyssey Drive
Huntsville, AL 35806

Abstract

Fuzzy logic systems can be used to model nonlinear deterministic processes. This paper exploits this idea by using such a system to model changes in treasury bill interest rates. A fuzzy associative memory (FAM) matrix forms the heart of the fuzzy system. A genetic algorithm adapts the FAM matrix to fit observed data. The performance of the fuzzy system is compared with a neural network approach and a nonlinear difference equation model. The fuzzy system is shown to be an effective means of modeling nonlinear data. Furthermore, the FAM matrix can be used to analyze the data and provide an indication of the reliability of forecasts generated by the model.

1. Introduction

Recent advances in nonlinear science have provided new tools for the analysis of complex data relationships. Neural networks, recognized for their universal approximation property [2], have been applied to a diverse range of modeling and classification problems. Recent results have also shown that systems based on fuzzy logic are also capable of approximating general nonlinear functions [3]. This paper exploits this idea by using a fuzzy logic system to model changes in treasury bill interest rates. A genetic algorithm is used to optimally select the parameters which define the fuzzy system. An advantage of the fuzzy system approach is that, unlike a neural network, the model itself can provide some insight into the workings of the observed system.

While the ability of nonlinear models to fit data is undisputed, some modeling practitioners have been reluctant to use nonlinear techniques for fear of "overfitting" the data. Overfitting refers to the phenomenon of producing a model that fits its training data particularly well, but does a poor job of forecasting on data from outside of the training set. Neural networks, in particular, do extremely well in fitting data, but the structure of these models gives no clue as to what they will do with data not seen during training.

One recourse to try to improve the forecasting ability of data models is to increase the volume of data on which the model is trained. This may help, but if the test data is unlike anything in the training set, the size of the training set will not make any difference in the accuracy of the forecast.

The fuzzy system model that we develop here will do no better than any other model if it must operate on data significantly different from anything seen in training. However, the fuzzy system approach has one clear advantage over other methods. The fuzzy system has built into it the capability of analyzing the distribution of training data versus the distribution of the test data. If these are radically different, then one knows in advance that the model is not going to produce a reliable forecast. If the two distributions are similar, there is a higher probability of an accurate forecast.

In the sections that follow, we will examine in more detail the application problem and the fuzzy system structure with its genetic optimization, and present experimental results, using real data, which compare the fuzzy system with a neural network and a nonlinear difference model.

2. The Application: Interest Rate Modeling

The application considered here is modeling of changes in U.S. Treasury bill (T-bill) interest rates. A recent article by Larrain [5] discusses a nonlinear model for T-bill rates that includes both a technical basis, considering prior interest rates, and a fundamental basis that takes into account external factors that may affect interest rates. These external variables include gross national product (GNP), consumer price index (CPI), M2 money supply (M2), and personal wealth (W) (expressed as the accumulation of the difference between personal income and personal consumption). Larrain presents a statistical analysis of the validity of this choice of variables. For the period under consideration (1962 - 1989), these variables statistically account for 89 - 93 % of the variation in interest rates.

In the present paper we will use Larrain's work as a starting point. We will use the four external variables, together with prior interest rates, as input to a fuzzy system model that will output the next quarter's interest rates. Our purpose here is only to investigate the viability of a fuzzy system approach to financial data modeling, not to completely explain all variation in interest rates. Thus, we will not analyze any further the

appropriateness or completeness of our set of input variables.

3. Fuzzy Systems

The particular type of fuzzy system that we use here is a fuzzy associative memory (FAM) system. Such systems are described in [4]. They are characterized by a FAM matrix, together with fuzzy sets and membership functions defined for each input variable and the output variable. An example of a FAM matrix is shown in Figure 3.1.

Y \ X	N	ZE	P
N	NL	NS	NS
ZE	NS	ZE	PS
P	PS	PS	PL

Figure 3.1 An example of a two-dimensional FAM matrix.

In this example, there are two inputs, X and Y. Each input variable has three fuzzy sets associated with it, which we have labeled "N", "ZE" and "P", for "Negative", "Zero", and "Positive" (these need not be the same for each input variable). The output variable, which we'll call Z, has five fuzzy sets associated with it: "NL" (Negative Large), "NS" (Negative Small), "ZE" (Zero), "PS" (Positive Small), and "PL" (Positive Large).

The FAM provides a means of storing fuzzy rules, such as "If X is Positive and Y is Negative, then Z is Negative Small", or "If X is Positive and Y is Zero then Z is Positive Small". Note that "Zero" here represents a fuzzy set, not necessarily the precise value of 0. The fuzzy rules consist of combining the inputs with the conjunctive "and", and obtaining the output fuzzy set from the corresponding FAM matrix entry.

The appeal of fuzzy systems is that they deal with information in a way in which we are used to thinking. Most observers of interest rate trends would agree that when inflation (CPI) goes up, interest rates go up, while an increase in money supply generally indicates that the price of money, as indicated by interest rates, should go down. Yet even expert observers would be hard pressed to quantify the relationship any further than this. In a fuzzy system, the rules need only be expressed in terms of fuzzy sets. However, there is nothing fuzzy about how the model itself operates. Precise values are input, and precise values are obtained as output.

3.1 Fuzzy Membership Functions

Fuzzy membership functions are the mechanism through which the fuzzy system interfaces with the outside world. Given a precise data value from the outside world, the membership function associated with a fuzzy set indicates the degree to which that value is a member of the set. The membership function takes on values between 0 and 1. Its domain is a subset of the set of possible values for the input variable. In the fuzzy model system discussed here, the input variables are normalized so that their values are between -2.5 and +2.5. Figure 3.2 shows the fuzzy membership functions that are used for the normalized input variables in our fuzzy model.

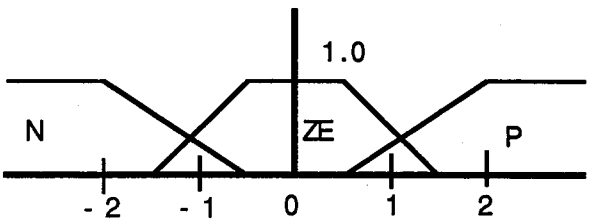


Figure 3.2 Fuzzy Membership Functions

These membership functions are piecewise linear trapezoidal functions, although other functions, such as gaussian curves, can be used. The three functions used here correspond to fuzzy sets for "Large Negative" (N), "Zero" (ZE), and "Large Positive" (P). Points in the interval (-0.5,0.5) receive the value 1 from the ZE membership function, and 0 from the N and P functions. Points in the interval (-1.5,-0.5), however, receive positive values from both N and ZE, and points in the interval (0.5, 1.5) receive positive values from both ZE and P.

3.2 Fuzzy System Operation

Our fuzzy system has five input variables X_1, X_2, X_3, X_4, X_5 and one output Y. Suppose we present the system with the input vector (x_1,x_2,x_3,x_4,x_5) (these are real normalized values, for example (0.2,0.5,-1.3, 0.8,2.1)). There are three fuzzy sets associated with each input variable, and three corresponding membership functions, as shown in Fig. 3.2. When an input vector is presented to the system, each of the three membership functions is evaluated on each component of the input vector. Thus, we must cycle through 3^5, or 243, possible combinations of membership functions.

One such combination is, for example, (N,ZE,N,P,ZE). This represents a fuzzy rule antecedent corresponding to one entry in the five dimensional FAM matrix. The corresponding FAM matrix entry itself is the output, or consequent, fuzzy set. Denote the fuzzy output sets O_1,

O_2, ..., O_N. For reasons explained below, we use N = 8 in our system. For ease of implementation, we assume that only a single real value y_j belongs to the output set O_j. If the corresponding consequent set is O_j, then the fuzzy rule rule can be translated as: If "X_1 is N" and "X_2 is ZE" and "X_3 is N" and "X_4 is P" and "X_5 is ZE", then "Y is O_j".

The first step in obtaining a "defuzzified" or "crisp" output value for the system is to evaluate each fuzzy rule antecedent at the input vector. This is done by taking the minimum of the membership function values. In our example, this is computed as:

$$w = \min (N(x_1),ZE(x_2),N(x_3),P(x_4),ZE(x_5)).$$

This value is then multiplied by the value y_j representing the output set O_j that corresponds to the FAM matrix entry at (N,ZE,N,P,ZE). This process is done for each FAM matrix entry. The fuzzy system output is then computed as

$$y = \left(\sum_{n=1}^{M} w_n \, y_{\alpha(n)} \right) / \left(\sum_{n=1}^{M} w_n \right)$$

Here, M is the number (3^5) of FAM matrix entries, w_n is computed for each entry as described above, and $y_{\alpha(n)}$ is the value representing the output set that corresponds to the FAM matrix entry.

This fuzzy system is thus a well defined process that maps a real 5-dimensional input vector to a single real output. The action of this system is determined by the entries in the FAM matrix. We will adapt the FAM matrix entry values by matching system behavior to observed data. The entries can be considered to comprise a (large) parameter set, much like the weights in a neural network, whose values determine the behavior of the system. However, unlike the neural network case, we cannot use a method analogous to backpropagation which would depend on the existence of a parameter gradient. Instead, we must use an optimization method that does not rely on any analytic properties of the parameter set. This is why we use a genetic optimization method to select the FAM matrix entries.

4. Genetic Optimization

Genetic algorithms are a biologically inspired class of algorithms that can be applied to, among other things, the optimization of nonlinear multi-modal (many local maxima and minima) functions. These algorithms do not rely on any analytical properties of the function to be optimized (such as the existence of a derivative). This makes them well suited to optimization over parameter sets such as the FAM matrix in our fuzzy logic model.

Central to any genetic algorithm is the concept of *chromosome*, and the operations of *crossover* and *mutation*. Chromosomes represent an encoding of the information upon which the algorithm operates. Each chromosome consists of a string of bits. Each bit may consist, for example, of a binary 0 or 1, or it may consist of a symbol from a set of more than two elements. Mutation randomly changes the value of a single bit. Crossover is a way of combining two chromosomes to produce two new chromosomes. A crossover site is randomly selected along the length of the chromosome, and the two chromosomes are then split into two pieces by breaking them at the crossover site. The new chromosomes are then formed by matching the top piece of one chromosome with the bottom piece of the other. This process is illustrated in Figure 4.1.

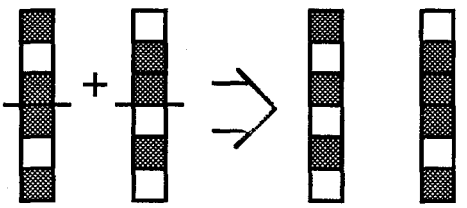


Figure 4.1 Crossover

A population of chromosomes is maintained during the operation of the algorithm. Each chromosome is assigned a fitness value, based on a problem-specific evaluation function. Genetic algorithms reward maximum fitness, so the evaluation function must be chosen so that its maximum corresponds to the desired value of the function to be optimized. At each step, called a *generation*, of the algorithm, pairs of individual chromosomes are chosen for the crossover operation. The fitter the chromosome, the more likely it is to be chosen. Mutation is randomly applied during this operation. A new population results from these operations. Over the course of a number of generations, the average fitness of the population increases, and the fittest members of the population approach acceptable solutions to the application problem.

5. The Fuzzy-Genetic Interest Rate Model

In our fuzzy-genetic system, each chromosome corresponds to a FAM matrix. The evaluation function consists of operating the fuzzy model with this FAM matrix, using the set of training data. The cumulative error is normalized to a value between 0 and 1. Since genetic algorithms are maximum-seeking algorithms (as opposed to algorithms which search for a minimum), this value is subtracted from 2 in order to obtain a positive fitness value that will be maximized at 2 when the error is zero.

We use binary values for each bit of the chromosome. Three bits are used for each entry in the FAM matrix, resulting in 8 possible output values for each FAM matrix entry.

We use five inputs for the interest rate model. We work with differences in the data, rather than actual data values, since it was felt that it would be easier to model the change in interest rate, rather than actual interest rate values. Thus, the input variables are: change in CPI, change in GNP, change in M2, change in personal wealth (the difference between personal income and personal consumption), and change in interest rate. Quarterly data is used. Following [5], we use a two quarter moving average for interest rates. The output is the next quarter's change in two quarter moving average interest rate. Also, following [5], the input CPI is delayed by one quarter, and the input M2 and personal wealth are delayed by two quarters.

As mentioned earlier, input data is normalized to values between -2.5 and 2.5. Fuzzy sets N, ZE, and P are assigned to each input variable, as shown in Figure 3.2. Note that with 5 input variables, we have a 5 dimensional FAM matrix, with 3^5 entries. Three bits for each FAM matrix entry results in a chromosome of length 729. If we had chosen, for example, to use five fuzzy sets for each input variable, this would have resulted in a chromosome length of $3*5^5$, a much larger number that would significantly slow the processing and increase the storage requirements of the genetic algorithm.

Two variations on standard genetic optimization are incorporated into our system. The first of these is fitness scaling ([1] contains an extensive discussion of fitness scaling). Fitness scaling is desirable when the fitness levels of all the individual members of a population cluster in a relatively small range. Scaling assigns a new fitness level of 1.0 to the most fit member of the population, and a small positive number (eg., 0.1) to the least fit member of the population. In this way, fitness levels are spread out so that the algorithm can distinguish the most fit from the least fit.

The second modification is the use of a "selection of the fittest" procedure. In the normal operation of a genetic algorithm, the operations of crossover and mutation are applied to the old population, with candidates for these operations being selected on the basis of fitness, to produce a new population. The new population then carries on, regardless of whether fitter individuals may have been left behind in the old population. Selection of the fittest picks the fittest members of the combined old and new populations to create the next generation. This process significantly speeds up the convergence time of the algorithm. There may be some danger that this modification speeds the algorithm to a non-optimal local fitness maximum. However, with the significant processing times required by our system, it was felt that this tradeoff was acceptable.

6. Alternative Models

The purpose of this article is to point out the feasibility of using a fuzzy logic system to model changes in interest rates, and to discuss some of the advantages of such an approach. To provide a performance baseline, we will compare this method to two other approaches for modeling interest rate changes. The first of these is a nonlinear model based on the work in [5], and the second is a feedforward neural network model.

6.1 Larrain Model

The interest rate model discussed in [5] is of the form:

$$r(t+1) = a + br^2(t) + cr^3(t) + dy(t) + eP(t-1) + fM(t-2) + gW(t-2). \quad (6.1)$$

Here, r(t) is a two quarter moving average of interest rate, y(t) is GNP, P(t) is CPI, M(t) is M2 money supply, and W(t) is personal wealth. The constants a,b,c,d,e,f,g are parameters which may take on positive or negative values. From (6.1) we can obtain an expression for $\delta r(t) = r(t+1) - r(t)$ as

$$\delta r(t) = b(r^2(t) - r^2(t-1)) + c(r^3(t) - r^3(t-1)) + d\delta y(t) + e\delta P(t-1) + f\delta M(t-2) + g\delta W(t-2), \quad (6.2)$$

where δ indicates the difference operator. Note that $\delta W(t)$ is equal to Y(t) - C(t), where Y(t) is real personal income, and C(t) is real personal consumption. Equation (6.2) can be viewed as a nonlinear nonhomogeneous difference equation.

Equation (6.2) is the model used to obtain the experimental results presented below. Note that equation (6.2) has six parameters, b,c,d,e,f,g. These are fitted to the data using a steepest descent method [6] which minimizes the sum of the squared errors. A genetic algorithm could have been used here as well. However, when the parameters appear in a simple way, as they do here, steepest descent provides the optimal solution much more quickly.

6.2 Neural Network Model

Three layer feedforward neural networks are universal approximators [2], and their application to data modeling has generated recent widespread interest. The backpropagation training method [4] for these networks (which is really just a from of steepest descent) provides a straightforward means of fitting the model to the data.

In the experimental results presented below, we use a three layer feedforward network with 5 input nodes, 5 hidden layer nodes, and a single output node. Training was accomplished via the backpropagation method.

7. Experimental Results

In applying a model to a data set, two concerns arise. First, we would like the model to at least show some semblance of fit to the training data. Second, we would like to see how well the model does in forecasting with data outside the training set, and, if possible, give us some reason for confidence in its forecasting abilities.

7.1 Fitting the Data

To test the ability of the fuzzy system model to fit a variety of data, the model was trained over the 20 year period from 1966-85. The results are shown in Figure 7.1. The actual output represents a two quarter moving average of change in T-bill rates. Quarterly data was used, for a total of 80 data points. The FAM matrix entries were adapted using genetic optimization, as discussed above. The fuzzy model is able to follow the volatile interest rate movements of the early 1980's, when rates rose from their mid-1970's level of around 5% to over 15%.

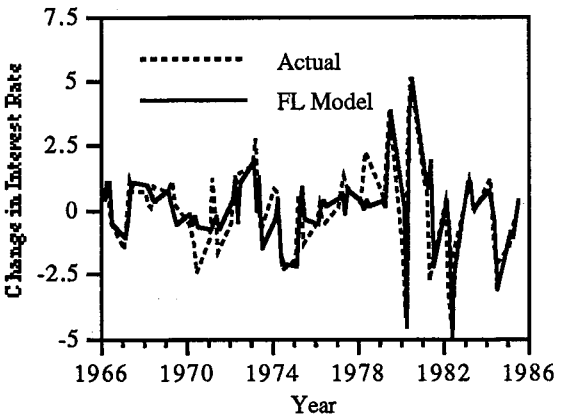


Figure 7.1 Comparison of fuzzy system model with actual data from 1966-1985. The output variable is two-quarter moving average change in T-bill rate.

To compare the data fitting capabilities of the different modeling approaches, we use quarterly data from 1967 to 1986, divided into 5-year moving window increments. Each of the three models is adapted, or "trained", on each of the 15 data windows 1967-71, 1968-72,...,1981-85. The first two quarters from the year immediately following the training window are used for measuring forecasting performance.

The neural network models, as mentioned above, consisted of 5 input nodes, 5 hidden layer nodes, and one output node. For some of the data windows, fewer hidden layer nodes could have been used, while for others, a better fit could have been accomplished with more hidden layer nodes. However, for comparison purposes, the same size network was used for each window. Backpropagation, with momentum term, was used for training. Typical training times ranged from 300 -2000 iterations through the training sets to produce the results reported here.

The Larrain model parameters were adapted using the steepest descent algorithm applied to minimizing the sum of the squares of the data point errors. The algorithm converged after a few hundred iterations in each case.

Figure 7.2 compares the data fitting capabilities of each of the three models over the 15 data windows. In each case, the output data has been normalized to the range 0.0 - 1.0. The number that is plotted here is cumulative average normalized error:

$$E = \frac{1}{N}\left(\sum_{n=1}^{N} | M_n - A_n |\right)$$

In the above expression, N is the number of data points in the data. In this case, N = 20 for our quarterly data in each 5-year data window. M_n is the normalized model output, A_n is the actual normalized output, and E is the cumulative average normalized error that appears in the plots shown in Figure 7.2

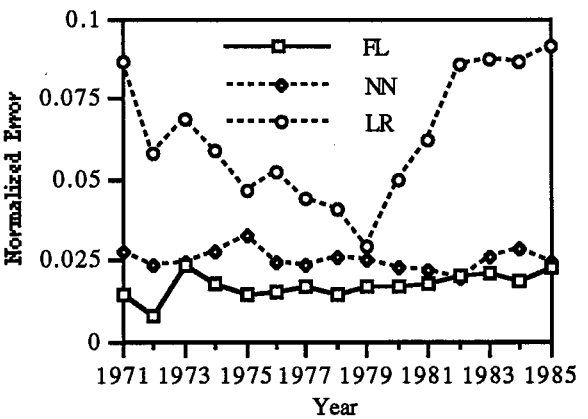


Figure 7.2 Cumulative normalized error for each model: fuzzy system (FL); neural network (NN); Larrain (LR). The year is the ending year of the five-year data windows 1967-71, 1968-72,..., 1981-85.

The fuzzy system (FL) model does at least as well as the other two here. We shouldn't read too much into the

differences between the models, however. In particular, the neural network models could have produced better fits with more hidden layer nodes and a tighter training tolerance. The significance of Figures 7.1 and 7.2 is that they show that the fuzzy system is a viable candidate for modeling this type of data.

7.2 Forecasting

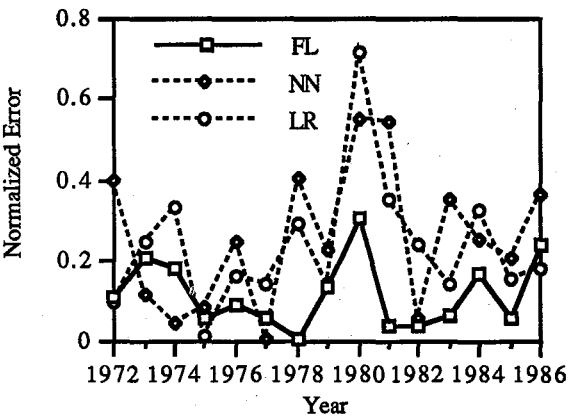
He who knows does not predict.
He who predicts does not know.

Lao Tzu

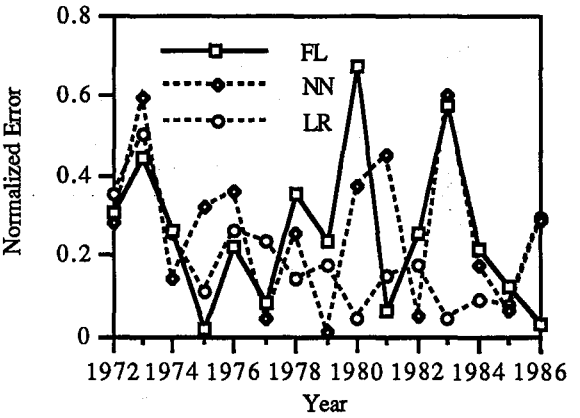
Forecasting refers to the ability of a model to deal with data from outside of the training set. In attempting to find a deterministic model that matches the behavior implied by the data, we are tacitly assuming the existence of a deterministic process that produces the data. If we in fact have correctly modeled that deterministic process, then the model should behave correctly on all data produced by the process, whether from the training set or not. In practice, however, this is rarely the case. We will see that our fuzzy system model, while not providing significantly better prediction performance than the other models, does have a property that may enable one to at least predict its prediction performance.

Figure 7.3 shows the prediction performance of each of the models on the two test quarters for each of the data windows. Figure 7.3 (a) shows the results for the first quarter immediately following the data window. Thus, the first data point refers to the first quarter following the 1967-71 data window, that is, the first quarter of 1972. The troublesome 9th data point represents the first quarter of 1980, when interest rates jumped 1.66%, following an increase of 2.17% the previous quarter. This left interest rates at 13.46%, unlike anything seen during the 1975-79 training data period. Figure 7.3 (b) shows similar results for the second quarter following each data window.

The fuzzy system does relatively well compared with the other two models over the first quarter data, as shown in Figure 7.3 (a). However, Figure 7.3 (b) casts doubt on the significance of this, since the fuzzy system often is the worst of the three models in predicting second quarter data. It is interesting to note that the Larrain model holds its own through the turbulent years of the early 1980's. This is probably due to the fact that this model is not limited in the magnitude of the output values that it can produce. Both the fuzzy system and the neural network model can produce values no larger than the predetermined maximum of their output range.



(a) Prediction errors for the first quarter of each year



(b) Prediction errors for the second quarter of each year.

Figure 7.3 Errors in predicting change in interest rate, for the fuzzy system (FL), neural network (NN), and Larrain (LR) models. Each data point represents a different model trained on the five years of data preceding the indicated year.

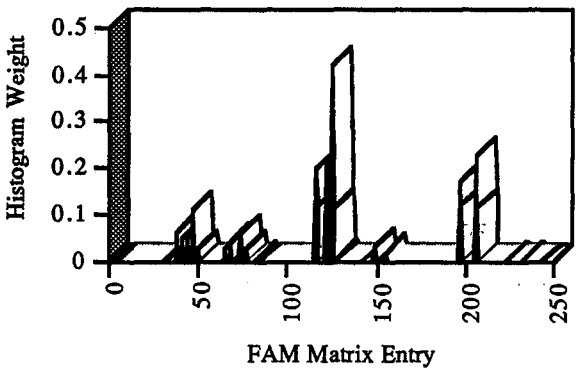
7.3 Predicting Prediction

What advantage does the fuzzy system offer over the other two modeling approaches discussed here? The processing demands of optimizing the fuzzy system are far greater than those of either the neural network or Larrain models, and there is not a significant difference in the resulting modeling capability.

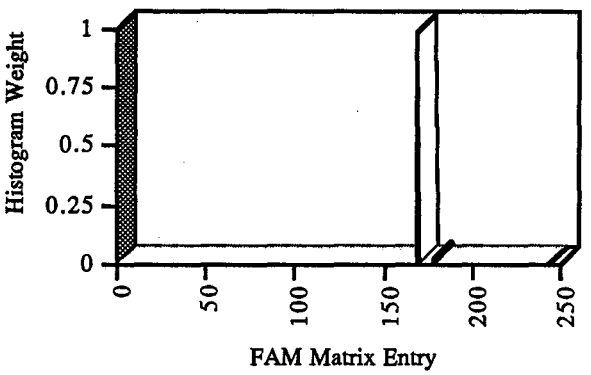
The advantage may be that the fuzzy system can reveal to us through the FAM matrix how it uses data. As discussed in Section 3.2, each entry in the FAM matrix generates a nonnegative weight w for each data point that passes through the system. The magnitude of this weight affects the output response of the system. By summing these weights over all the training data points, we can generate a "histogram" corresponding to the FAM matrix

entries. This histogram tells us the relative importance of the FAM matrix entries to one another over this data set. A large histogram level at a particular FAM matrix entry tells us that entry was strongly activated by the data set. Note that the histogram does not depend on the entry values in the FAM matrix. It depends only on the FAM matrix structure and the membership functions associated with the input fuzzy sets.

Figure 7.4 (a) shows a histogram for the data period 1975-79. The weights are normalized so that the area under the histogram curve is 1. Since we can't easily depict the five dimensional FAM matrix, the matrix entries are simply represented linearly along the horizontal axis.



(a) Histogram for training data from 1975-79.

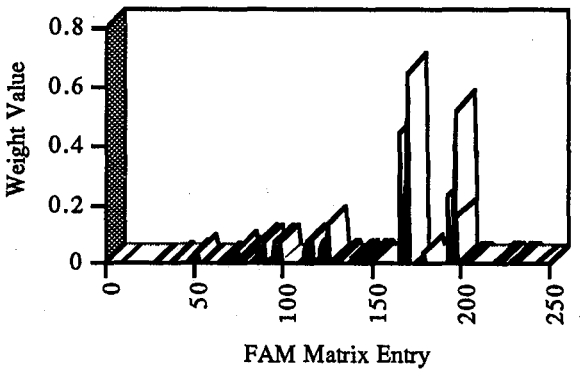


(b) Histogram for test data from second quarter of 1980.

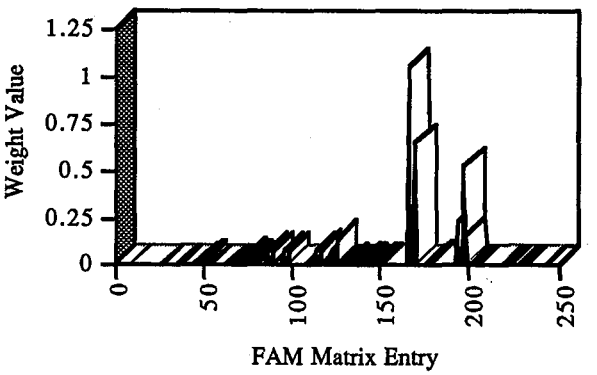
Figure 7.4 Histograms for training data (a) from 1975-79 and test data (b) from second quarter 1980. There is no overlap between the two histograms. The corresponding prediction error is large in this case.

Now compare the histogram of (a) with that of Figure 7.4 (b). This histogram was generated by operating the fuzzy system, trained on 1975-79 data, on the single data point from the second quarter of 1980. Note that there is essentially no overlap between the two histograms. This means that the trained fuzzy system has not seen anything like the data that generated the histogram of Figure 7.4 (b). It should not be surprising, then, to see the large normalized error (0.67147) associated with this prediction in Figure 7.3 (b).

Contrast this with the situation depicted in Figure 7.5 (a) - (b). Here, the test data from the first quarter of 1975 generates a histogram (Figure 7.5 (b)) whose profile shows a significant amount of overlap with the histogram generated by the training data (Figure 7.5 (a)). The normalized prediction error for the test quarter in this case was relatively small, at 0.0591.



(a) Histogram for training data from 1970-74.



(b) Histogram for test data from first quarter of 1975.

Figure 7.5 Histograms for training data (a) from 1970-74 and test data (b) from first quarter 1975. Note the similar profiles. The corresponding prediction error is small in this case.

Figures 7.4 and 7.5 indicate that there may be information in the data histograms which can tell us whether to expect a reliable forecast from our model. Further research is required to quantify this relationship. Note that a poorly trained model may produce large prediction errors, even

though there is a great deal of similarity between training and test histograms. This is because the histograms are independent of the FAM matrix entry values, and hence do not depend on the quality of the training. Similarly, a model may randomly produce small prediction errors in spite of no similarity between training and test histograms, due to unadapted FAM matrix entries randomly receiving nearly correct values.

One could argue that non-fuzzy models, such as the Larrain model or the neural network model, could also make use of a FAM-matrix approach to data analysis. This may be the case. However, only the fuzzy system makes use of this information in the exact format in which it is presented. The fuzzy system is positioned to receive the greatest benefit from this type of analysis.

8. Conclusions

We have shown that a fuzzy logic system can effectively model financial time series data, such as changes in interest rates. The FAM matrix entries in the fuzzy system can be adapted using genetic optimization. Furthermore, the FAM matrix structure provides a data analysis tool that offers insight into the capability of the model to forecast over certain combinations of test data.

References

[5] Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[2] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, Vol. 2, 1989, pp. 359-366.

[3] Kosko, B., "Fuzzy Systems as Universal Approximators", FUZZ-IEEE, 1992, pp. 1153-1162.

[4] Kosko, B., *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[5] Larrain, M., "Testing Chaos and Nonlinearities in T-Bill Rates", *Financial Analysts Journal*, Sep-Oct, 1991, pp. 51-62.

[6] Mikhlin, S.G., *The Problem of the Minimum of a Quadratic Functional*, Holden-Day, Inc., 1965.

The Second Annual International Conference on ARTIFICIAL INTELLIGENCE APPLICATIONS ON WALL STREET, April 19-22, 1993, New York, NY USA.

Hybrid Neural Network For Stock Selection

Francis Wong
NIBS Pte Ltd , 62 Fowlie Rd, Singapore 1542
Tel (+65) 344 2357, Fax (+65) 344 2130

David Lee
Smith Barney, Harris Upham Securities Pte Ltd
9A Recreation Rd, Singapore 1954
Tel (+65) 728 3161, Fax (+65) 7335128

Abstract

When it comes to stock selection, it is not enough to just determine the dollar weightings and measure the risk vs. returns. In bear markets, your portfolios need extra protection. A proposed approach is to use a neural network with various hybridized technologies to capture a wider variety of fish in both long and short positions. Our study showed that such an approach can achieve a greater than 3% net monthly return, and a greater than 90% accuracy in determining the positive and negative directions in two stock baskets with opposing risk levels. Moreover, the entire portfolio risk can be reduced by 80% or more. We will describe some useful techiques and strategies to analyze the input data, to predetermine the predictivity of a given time series, to unveil the hidden cycles, to determine the appropriate input window size the time cycles, to speed up the training and to achieve a better accuracy.

1. Introduction

This paper addresses the application of hybrid technologies to investment analysis and trading techniques. Specifically, can hybrid technologies differentiate between stocks that perform well and those that perform poorly? Can they reveal important opportunities in trading and give you better risk management? If so, how are these technologies applied to such problems?

Artificial neural network (ANN) for classiciation problems -- such as bankruptcy prediction, credit scoring of loan applicants and bond rating analysis -- and time-series forecasting have already been studied extensively. Stock selection isa much more sophisticated process because it involes a 3-dimensional problem -- it involves time, different stocks, and the various indicator signals associated with the stocks. Cross-sectional analysis is needed to do a comparative study among the stocks, while time-series analysis is needed to keep track of the momentum patterns of the individual stocks and the overall market direction.

2. Hybrid Technologies

Recently there has been considerable interest in creating hybrid systems, especially of neural networks (NN), fuzzy logic (FL), genetic algorithms (GA), expert systems (ES) and chaotic systems (CS). For instance, NN can be used to learn the patterns of trading in a commodity trading system, while ES can be used to manage the investments according to some rules [Bergerson 91]. GA can be used to construct or train NNs to achieve a better performance [Montana 89] [Harp 91] [Wong 92]. The fractal dimension of chaotic time-series can be used to obtain the optimal size of the input and output layers of NN [Matsuba 92]. Some of these possible ways of hybridization is shown in Table 1.

	NN	FL	ES
NN	--	NN to generate membership functions and fuzzy rules	NN for Patterns and ES for rules eg. Commodity Trading Model
GA	GA Creates Better NNs eg. Market Timing Adviser	GA to optimize fuzzy rule base	--
CS	Fractal Dimension to optimize NN	--	--

Table 1 - Hybridization of technologies.

Neural networks are known to be very good at recognizing patterns from noisy, complex data; Fuzzy systems are useful for decision-making where there is a great deal of uncertainty as well as vague phenomena; Dynamical systems theory provides a clearer understanding of the dynamics and evolution of financial systems, and can help find periods of chaotic

behaviour in financial markets. However, each technique is known to have its own limitations. Through hybridization of these techniques, it has been demonstrated that they can be applied to solve complex financial applications in a number of ways. More general discussions on how these technologies can be combined are described in [Treleaven 92] and [Klimasauskas 92]. In this paper, we proposed a possible approach to integrate several technologies for investment analysis.

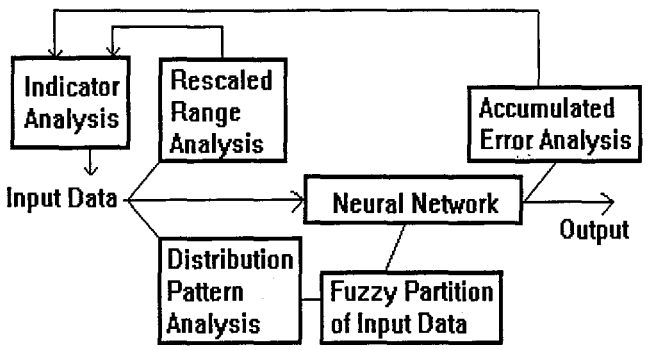


Fig.1 - The proposed architecture.

The objective of the architecture depicted in Fig. 1 is to apply these various technologies to analyze the input data, to predetermine the predictivity of a given time series, to generate the fractal dimension, to unveil the hidden cycles, to determine the appropriate input window size and forecast horizon, to determin the sizes of input and output layers of the neural network, to decompose the problem to speed up the training, to achieve a better accuracy and etc. The next few sections briefly outline these techniques. For more detailed discussions, please refer to the respective references.

(The various components of Fig. 1 are implemented in NeuroForecaster® version 2.10.)

3. Distribution Pattern Analysis

The phases of data preparation for modeling can be broadly classified into three areas [Stein 93]: Data specification and collection, data inspection and data preprocessing. After the data is identified and collected, it must be examined for those characteristics that may reveal important relationships between the target and the indicators. The distribution pattern of a variable gives valuable information about existence of outliers, correlation of that variable with the target to be forecast, and whether enough data points have been gathered for training. A useful way to inspect the distribution pattern is described below.

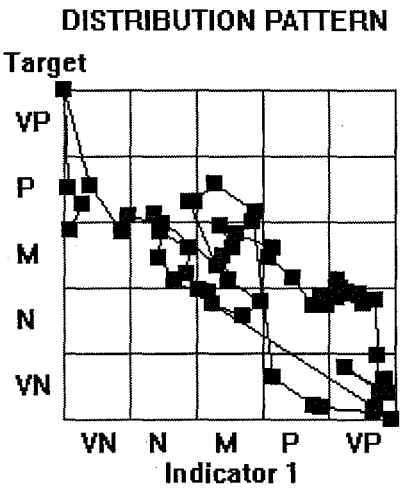


Fig. 2 - The distribution pattern of an indicator vs the target.

The Distribution Pattern function often reveals the quality of the input data (eg. the existence of outlying data or insufficiency of training samples). The amount of training data is not very important for training, but the distribution pattern is. This function provides an easy way to visually inspect the distribution of the indicators versus the target in the input space. Each data point, or a cluster of neighboring data points, represents a rule which describes the relationship between the indicator and the target. The input space is divided into 5 regions: VP (very positive), P (positive), M (moderate), N (negative) and VN (very negative).

A number of fuzzy if-then rules can be derived from this distribution pattern easily. For instance,

If Indicator 1 is VP Then Target is VN
If Indicator 1 is N Then Target is M
etc.

as illustrated in Fig. 2. The next section will describe how these fuzzy If-Then rules that characterize the input data can be used to decompose a given problem to achieve a faster training and more accurate results.

4. Fuzzy Partitioning of Input Data

Fuzzy Logic is designed to handle imprecise concepts by moving away from a 'yes or no' binary logic to a logic scheme which can handle 'shade of gray'. Systems based on fuzzy logic have proven to be very useful in a variety of industrial control operations and in pattern recognition tasks. There is a growing interest in using fuzzy logic in combination with neural networks to make them more flexible.

We proposed using a neural net structure as shown in Fig. 3. Each of the hidden layers is partitioned to represent 5 fuzzy rules. The Fuzzy Selection Module receives the current set of input vector, determines the fuzzy partition, and select one group of the neurons to activate. The mathematical description of the selection strategy can be found in [Wong 92]. The basic idea is decompose a given problem into sub tasks via divide-and-conquer strategy, without having to train the entire network for a given set of input vector.

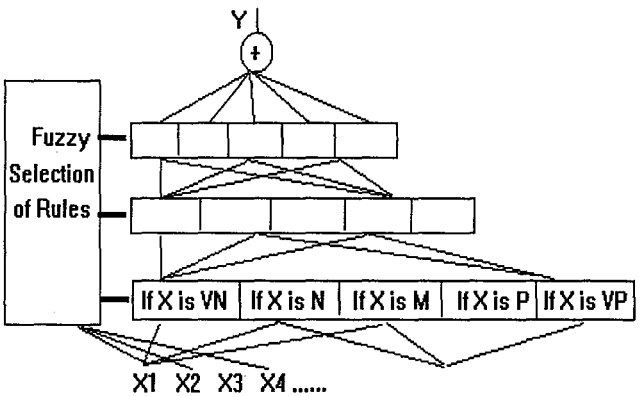


Fig. 3 - The selection of neurons for activation.

The Fuzzy Selection Module partitions the input space into the 5 regions: VP (very positive), P (positive), M (moderate), N (negative) and VN (very negative) (see Fig. for illustration). Because the output of the neurons have a sigmoidal nature, the input space will not be divided crisply for either binary or continuous input values. This is similar to the membership function of fuzzy set. The target value corresponding to the current input partition is used as the supervised data to train the network to generate the THEN part of the current partition. The output will be a value in one of the 5 regions.

If any of the regions is not well represented by the input data points, as illustrated in the figure below using a different problem, the network will not be able to learn the rule(s) describing that region (Region P below) and will fail if the test or future input data lies in that region.

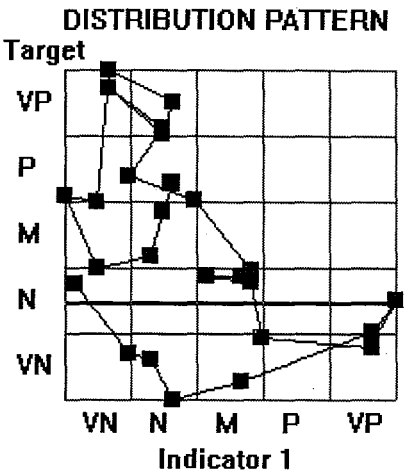


Fig. 4 - Absence of training data in the P (Positive) region.

However, since this architecture is able to self-organize the input-output mapping relationship by learning, it is applicable to problems where unknown logical relationships exist. Moreover, it is capable of expressing any nonlinear relationships by virture of its nonlinear structure, and there is no need to design membership functions which are difficult to implement in a multidimensional space using experience and intuition. It is also possible to extract "human comprehensible" rules from such a trained network using methods described in [Towell 91].

5. Rescaled Range Analysis

We developed a method for anaylzing the trend and to reveal hidden cycles. It is a simple extension of Mandelbort's "range over standard deviation" or "rescaled range" analysis which was proposed by Hurst (1951) in his studies of river discharges. The R/S statistic is the range of partial sums of deviations of a time series from its mean, rescaled by its standard deviation. In several papers, Mandelbrot demonstrated the superiority of R/S to more conventional methods of determining long-run dependence such as autocorrelation analysis and spectral analysis.

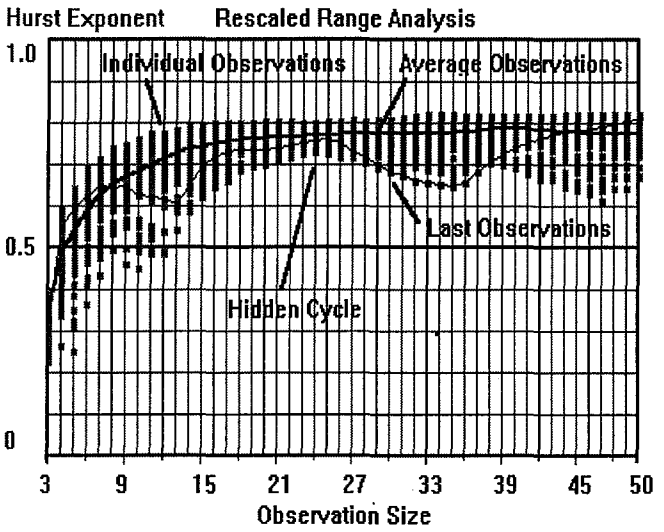


Fig. 5 - Rescaled Analysis of US$/DM monthly exchange rate from Feb85 to Aug92.

Fig. 5 illustrates the display of the Hurst exponent for individual observations, average observations and the last observations. The **Hurst exponent** is useful for estimating the predictability and the fractal dimension of the time series, to unveil any hidden cycle and the cycle length. The illustration shows that the time series has a maximum average Hurst exponent of around 0.78, and the narrowest band at the center suggests that there is a hidden cycle of length around 24. If a time series exhibits a strong cyclical behavior, the individual observation points at the corresponding periods will have very small spread. This features can be tested easily using signals mixed with known cycles.

The Hurst exponent is also useful as a measure of risk for individual stocks, bonds and currencies [Peters 91]. A Hurst exponent different from 0.5 is interpreted as a bias or memory effect. A Hurst exponent greater than 0.5 indicates a memory effect that is biased towards reinforcement of the trend. A Hurst exponent less than 0.5 indicates a negative bias or a tendency to reverse the ongoing trend. This is called anti-persistence. For instance, a Hurst exponent of 0.3 indicates that there is 70% chance that the current trend will be reversed in the time horizon. In our study, we have successfully extended the technique to compute the Hurst exponent of the target together with its indicators, and we found that more cyclical behaviors can be revealed. In summary, the R/S analysis is useful for:

1. Checking for predictability [Peters 91]
2. Revealing hidden cycles [Peters 93]
3. Optimizing NN architectures [Matsuba 92] using the fractal dimension

It was also suggested in [Peters 91] that Hurst exponent to be used as a more accurate estimate of risk for the capital market instead of volitality.

6. Indicator Analysis Using Accumulated Error

To explain how indicator analysis can be carried out in a neural network, we use the example of US Dolloar to Yen daily exchange rate forecast as illustrated in Fig. 6. In this example, the daily closing rate was used as the target to train the network, and 5 indicators as input data. We used a sliding window of variable size to capture the dynamics of the data. The entire window contents were presented to the network to perform a one-day-ahead forecast. The target captured in the window can also be used as input, and the exchange rate of the next day is used as the training value. In this example, a window size of 3 was used.

Title:US$/Yen_Daily_Closing_Forecast

day	US$ YEN	Gold	US$ DM	US$ Pound	DJIA	NYSE Volume
1	145	365	4.12	.623	2650	75.62
2	147	368	4.10	.584	2620	56.34
3	150	370	3.92	.614	2635	98.22
4	150	365	3.89	.605	2634	43.76
5	147	372	3.78	.595	2590	52.33
8	146	366	4.32	.583	2588	61.08
9	148	373	4.08	.573	2576	63.54
10	150	375	3.88	.566	2630	48.22

Fig. 6 - Applying a sliding window of size 3 to the input data to capture the dynamics of the indicators.

During the training process, we accumulated the error back-propagated to the input nodes and the relative amounts of error were shown in Fig. 7. It has been shown [Wong 92] the accumulated errors represent the relative significance of the indicators captured in the window.

Window Size	Gold Price	US$ DM	US$ Pound	DJIA	NYSE Volume
at t	2	0	5	2	3
at t-1	0	3	2	5	3
at t-2	4	5	7	3	2

Fig. 7 - The relative signicifance of the various indicators captured by the window with size 3.

As can be seen from Fig. 7, if a window size of 1 or 2 was used instead of 3, the significance of US$/DM and gold price with a 2-day delay would be ignored. One should experiment with a number of window sizes in order to find the most appropriate size. The importance of the appropriate window size and how it affects the forecast accuracy will be further elaborated in Section 8 below.

8. Application: A Stock Selection Example

To recap, we have explained the various techniques and now we shall go on to tackle the problem of building defensive portfolios in bear markets. When equity markets enter a bearish phase, fund managers face the pressure of keeping book losses to a minimum, by shifting to so-called "defensive" stocks. But this is usually easier said than done because there are not many defensive stocks around that can weather the winds of bear. In this study, we have experimented a new form of defense that can achieve far better results. Risk reduction is defined as maintaining the stock returns on an even keel even on a monthly basis, i.e., to reduce the volatility of returns. It also means to reduce the quantum of unrealized loss monthly and keep it small as compared with the respective market index. Return is defined, in our context, as getting in excess of 30% per year -- which is far superior to that expected in most equity investments, and is especially impressive when secured within a bear market environment.

8.1. Cross-Sectional and Time-Series Analysis

The main idea is to lay a short layer of stocks against the longs, so that our net exposure to the market is reduced. We do this by performing a cross-sectional analysis of 17 stocks that have a 90% correlation with the Nikkei 225. The exclusion of low-correlated stocks is to weed out inactively-traded stocks. The next step is to construct an efficiently diversified portfolios using the 17 correlated stocks selected earlier on.

The neural net loads up two years of data from 1990 to 1991. Each row of data contains a spectrum of 24 indicators to start with. This initial sets of indicators included liquidity and profitability ratios; particularly sales growth ratios. In Japan, market players favor companies with increasing market shares (sales growth ratios) compared with earnings growth and abnormal earnings which can be generated by abnormal asset sales. By using distribution pattern analysis and fuzzy partitioning of data sets, we were able to reduce the 24 indicators to 8, and still able to capture the important relationships between the target and the remaining indicators.

After 50,000 iterations of training cycles, the neural net was tested with 1992 data which it has never seen before. These stocks were then grouped into two separate baskets based on whether their forecast returns were positive or negative. The results showed that an over 90% accuracy was achieved in correct grouping of the stocks (see Fig. 8). With a long-short equitized strategy, a greater than 2% net monthly return for the entire period of 1992 can be achieved, despite the fact that the NIKKEI 225 plunged from 24,000 to 17,000 Yen in that year -- a whooping 29% annual loss or 2.4% monthly decline. With the help of the hybrid neural net, the fund managers would be able to go long on the positives and short on the negatives, and construct a well balanced, neutral portfolio easily.

The cross-sectional analysis was useful in choosing which stocks to go long or short, but not when and by how much. We then performed a time-series analysis on the Nikkei 255 to further boost the net monthly performance. We loaded up 4 years of Nikkei data and used R/S analysis to identify the hidden cycles to derive the right combination of window size and forecast horizon, which are 6 and 4 steps ahead respectively. We were able to eventually boost the net monthly returns to more than 3%. The significance of the window size will be discussed in below.

8.2. The Importance of Window Size

We built several identical networks with different window sizes to learn the Nikkei data. Our study showed that the network with a window size of 6 gave the best performance. As shown in Figure 9, it has an average error of 1.63%, while that of window size 1 is 2.34%. This comparative study showed that an

appropriate window size is important in capturing the dynamics of the indicators, and learning the input-output mapping function.

Another interesting observation made from this study was that, there was no "untrained" neurons in the network with a window size of 6. The accumulated error histogram of the neurons of this network is shown in Fig. 10, in which 9 neurons have at least 76.4 which is the amount of error accumulated during the training process. Wherease in the case of other window size 1, there were some untrained or lightly trained neurons. The problem of untrained neurons present in a network is that they might generate spurious outputs when the network is put to test on unseen data.

Other than the stock selection problem, we had also applied the same strategy to other financial instruments using various window sizes and forecast horizons. The results are presented in Table 2.

9. Conclusion

The various techniques outlined in this paper were able to discriminate between stocks that outperformed and those that under performed, and secure a far better return than most traditional long-only investment strategies. The results should be persuasive enough to encourage long-only fund managers to explore using both arms to swim in troubled financial waters.

In conclusion, we believe that these new technologies will have a major impact in many areas of financial information processing. It may be possible that these technologies will lead to a situtation where traders can use these extremely adaptive intelligent tools to maximize profit with reduced portfolio risks.

References:

[Bergerson 91] K. Bergerson & D. Wunsch, "A Commodity Trading Model basd on a Neural Network-Expert System Hybrid," Proceedings of International Joint Conference for Neural Networks, Seattle, 1991.

[Harp 91] S. harp & T. Samad, "Genetic Synthesis of Neural Network Architecture," in the Handbook of Genetic Algorithms, ed. L. Davis, Van Nostrand Reinhold, 1991.

[Klimasauskas 92] C. Klimasauskas, "Hybrid Technologies: More Power for the Future," in Advanced Technologies for Developers, No. 1, 1992, pp 17-19.

[Matsuba 92] I. Matsuba, H. Masui & S. Hebishima, "Prediction of Chaotic Time-Series Data using Optimized Neural Networks," Proceedings of International Joint Conference for Neural Networks, Beijing, 1992.

[Montana 89] D. Montana & L. Davis, "Training Feedforward Neural Networks Using Genetic Algorithms," Proceedings of the 11th International Joint Conference on Aritificial Intelligence, 1989.

[Peters 91] E. Peters, "Chaos and Order in the Capital Markets," published by John Wiley & Son, 1992.

[Peters 93] E. Peters, "Fractal Statistics: Applying Chaos Theory to Capital Markets," to appear in Proceedings of the 4th International Conference on Advanced Technologies for Trading and Asset Management, The New York Marriott Financial Center, July 1993.

[Stein 93] R. Stein, "Selecting Data for Neural Networks," AI Expert, Feb 1993, pp 42-47.

[Towell 91] G. G. Towell, J. W. Shavlik, "The Extraction of Refined Rules from Knowledge-Based Neural Networks," Machine Learning, August 1991.

[Treleaven 92] P. Treleaven & S. Goonatilake, "Intelligent Financial Technologies," Proceedings of Parallel Problem Solving from Nature: Applications in Statistics and Economics, by EUROSTAT, 1992.

[Wong 91] F.S. Wong, "A Stock Selection Strategy Using Fuzzy Neural Networks," NeuroComputing, No.2, 1990/91, Elsevier.

[Wong 92] F. Wong, P. Tan & X. Zhang, "Neural Networks, Genetic Algorithms and Fuzzy Logic for Forecasting," Proceedings of the 3rd International Conference on Advanced Trading Applications on Wall Street & Worldwide, New York Marriott Financial Center, 1992.

[Wong 92a] F.S. Wong, P.Z. Wang, T.H. Goh, B.K. Quek, "Fuzzy Neural Systems for Stock Selection," Financial Analysts Journal, Jan/Feb 1992.

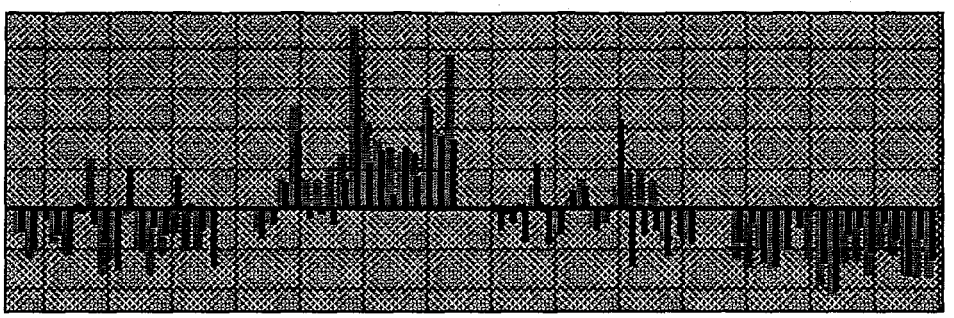


Fig. 8. The gray bars are the returns of 17 stocks while the bars in darker color are the monthly forecasts.

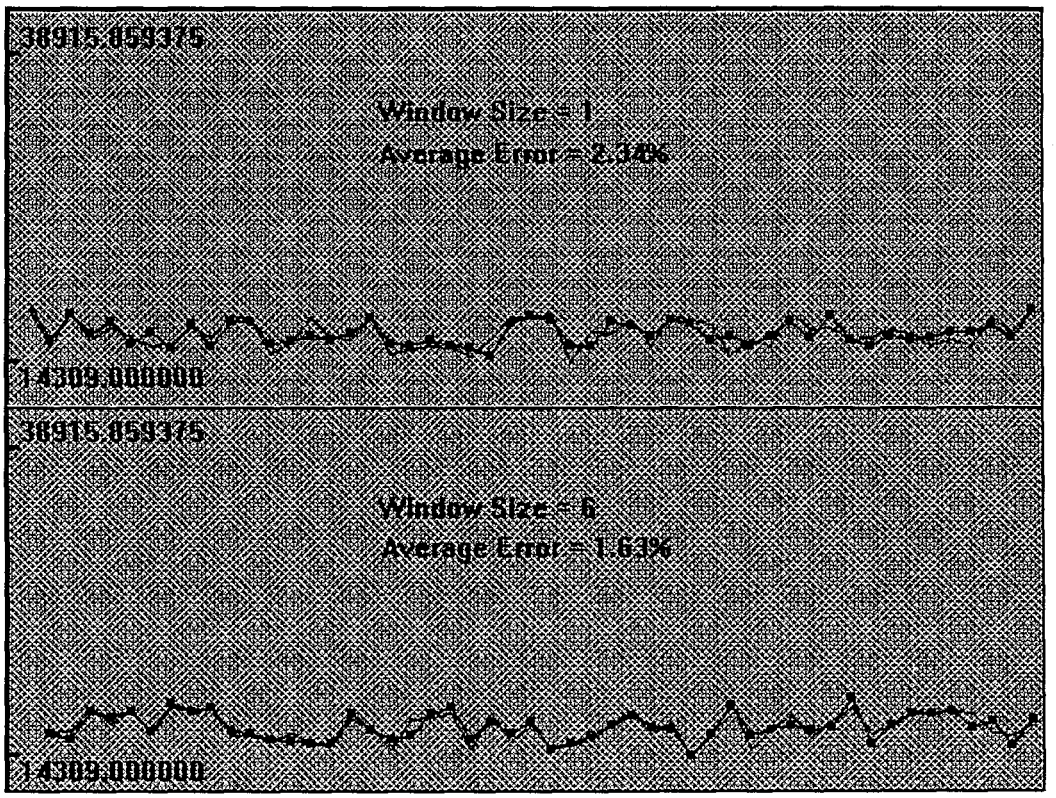


Fig. 9 . The Nikkei 225 index vs the network output (darker lines with square boxes) using a window size of 1 (upper graph) and 6 (lower graph). The network with a window size of 6 can learn the training data better.

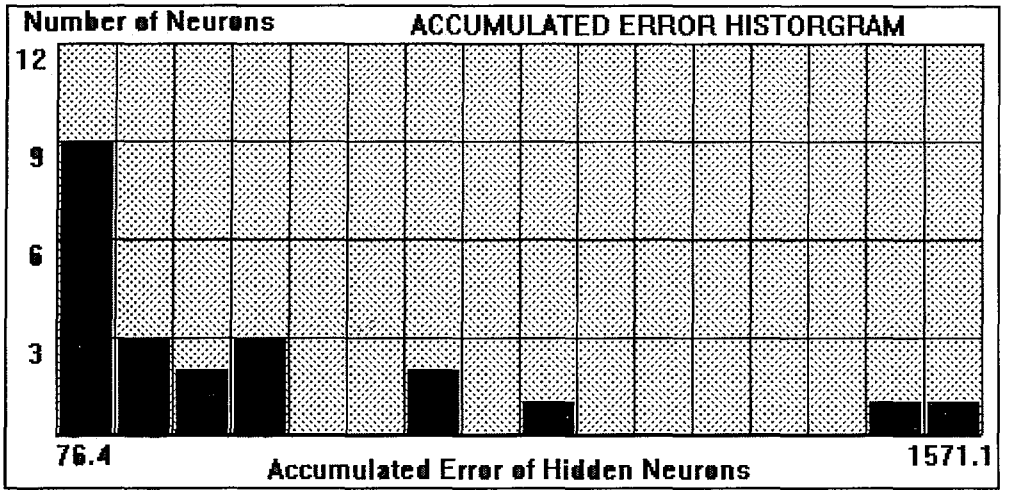


Fig. 10. The errors accumulated by the neurons in the networks indicate the amounts of training they received.

FINANCIAL INSTRUMENTS :	5 DAYS AHEAD	10 DAYS AHEAD	20 DAYS AHEAD
1. US TREASURY BOND	1.2%	1.5%	1.6%
2. EURODOLLAR INTEREST RATE	0.3%	0.4%	0.4%
3. S&P STOCK INDEX	1.8%	2.5%	4.0%
4. US$/YEN CURRENCY RATE	1.1%	1.5%	1.8%
5. US$/SFR CURRENCY RATE	0.8%	1.2%	1.6%
6. NIKKEI 225 STOCK INDEX	1.6%	2.5%	3.9%
7. HANG SENG STOCK INDEX	2.1%	2.3%	4.1%
8. SINGAPORE STI INDEX	1.4%	1.8%	3.6%
9. MALAYSIA KLCI INDEX	1.3%	1.7%	3.3%
10. SING$/M.RINGGIT	0.3%	0.4%	1.0%

Table 2. The same technique applied to other financial instruments.

Index by Author

Titles of Related Interest...

Standards and Review Manual for Certification in Knowledge Engineering: Handbook of Theory and Practice
Edited by Milton White, Ph.D., Joe Goldsmith, Ph.D.
More than 600 pages, glossary, index, bibliography, sample questions and answers. A study guide for Knowledge Engineers and valuable to those sitting for the Certification in Knowledge Engineering Examination (CKE™). Contributing writers are professionals in government, academia and major U.S. industries
$75.00 ISBN 0-938801-04-X

'New Generation' Knowledge Engineering
PROCEEDINGS of the Third Annual Symposium of the International Association of Knowledge Engineers November 16-19, 1992
More than 800 pages; Papers from Government, Academia, and Industry covering topics such as V&V, Knowledge-Based System Management, AI Applications for Energy, Medicine, Military, etc.; Management of Knowledge Assets; Directions in KADS; Knowledge Engineering Education Worldwide; Knowledge Management; Intelligent Multi-media and Virtual Reality; Intelligent Database Technology; Knowledge Engineering Careers and the Profession.
$65.00 ISBN 0-938801-06-6

Knowledge Engineering Today's Marketplace II
PROCEEDINGS of the Second Annual Symposium of the International Association of Knowledge Engineers
More than 455 pages. Papers from Knowledge Engineers worldwide. Topics include knowledge engineering in medicine, business, and space; state of the profession; knowledge representation and knowledge acquisition.
$50.00 ISBN 0-938801-05-8

To place an order, please contact:
The Software Engineering Press
973C Russell Avenue
Gaithersburg, MD 20879
(301) 926-7303

International Association of
Knowledge Engineers
973D Russell Avenue
Gaithersburg, MD 20879
(301) 948-5390

$65.00
ISBN 0-9-38801-07-4

Software Engineering Press
an imprint of the
Systemsware Corporation
P.O. Box 2635
Kensington, MD 20891-2635