"DECISION SUPPORT IN NON-CONSERVATIVE DOMAINS: GENERALIZATION WITH NEURAL NETWORKS"

by

Soumitra DUTTA*
Shashi SHEKHAR**
and
W.Y. WONG***

N° 92/31/TM

* Assistant Professor of Information Systems, INSEAD, Boulevard de Constance, Fontainebleau 77305 Cedex, France.

** University of Minnesota, Minneapolis, MN 55455, U.S.A.

*** University of Minnesota, Minneapolis, MN 55455, U.S.A.

Printed at INSEAD,
Fontainebleau, France

Decision Support in Non-Conservative Domains: Generalization with Neural Networks

Soumitra Dutta
INSEAD
Fontainebleau
France 77305
dutta@freiba51.bitnet
Tel: 33-1-60724017

Shashi Shekhar
W.Y. Wong
University of Minnesota
Minneapolis, MN 55455
shekhar@cs.umn.edu
Tel: 612-624-8307

Abstract

Models in conventional decision support systems (DSSs) are best suited for problem solutions in domains with well defined/structured (mathematical) or partially defined/semi-structured (heuristic) domain models. Non-conservative/unstructured domains are those which either lack a known model or have a poorly defined domain model. Neural networks (NNs) represent an alternative modelling technique which can be useful in such domains. NNs autonomously learn the underlying domain model from examples and have the ability to generalize, i.e., use the learnt model to respond correctly to previously unseen inputs. This paper describes three different experiments to explore the use of NNs for providing decision support by generalization in non-conservative/ unstructured domains. Our results indicate that NNs have the potential to provide adequate decision support in non-conservative/unstructured domains.
Keywords: Generalization problem solving; Decision support with neural networks

1. Introduction

This section describes the recognition and generalization problem types solved by NNs and shows how NNs can add to the model base of a DSS to enhance its productivity. It next formulates the generalization problem in the context of the NN architectures considered in this paper. Finally, it outlines the focus and structure of the paper.

1.1 Problem types solved by neural networks

NN applications can be classified into two broad categories [18,37]: **recognition** problems and **generalization** problems. In the recognition problem, (input, output_label) pairs (I_1, O_1), (I_2, O_2),...,(I_n, O_n), are used for training the network and the trained network is tested with the input I_j ($1 \leq i \leq n$) corrupted by noise. The trained network is expected to reproduce the output O_j corresponding to I_j, in spite of the presence of noise. Applications of NNs to domains such as shape recognition [19], handwriting recognition [52], speech recognition [25] and pattern recognition [11,50] are examples of recognition problem solving.

A variant of the recognition problem is the *generalization* problem. The training phases of the recognition and generalization problems are similar. However, in generalization problems, the trained network is tested with input I_{n+1}, which is distinct from the inputs I_1, I_2,..., I_n used for training the network. The network is expected to correctly predict the output O_{n+1} for the (previously unseen) input I_{n+1} from the model of the domain it has learned from the training input-output pairs. Typical applications of NNs for generalization problems include the assignment of corporate bond ratings [9,13], economic prediction [49] and the treatment of hypertension [34]. The ability to generalize is an important advantage of NNs as compared to rule-based systems or other conventional solution techniques which are usually unable to adequately respond to "new, unseen" situations. NNs used for generalization problem solving have also been termed as *expert networks* [5] as they are able to respond to unseen inputs and thus can be considered as being able to display some "expertise" about the domain under consideration.

Generalization problems can be sub-classified on the basis of the underlying domain of application as shown in Figure 1. Some domains have well defined models (e.g., electrical circuit analysis) while others have partially defined domain models (e.g., the diagnosis of diseases from symptoms and laboratory tests). In this paper, we term the former domain as a *conservative* domain and the latter domain as a *partially conservative* domain. Conventional techniques (e.g., systems analysis) can be applied to conservative domains and many successful artificial intelligence applications have been devised for partially conservative domains (e.g., expert systems such as MYCIN [41]). Yet another important class of problem domains are those that lack a well defined domain model, e.g., the problem

of assigning ratings to corporate bonds (see section 3 for more details). Such domains are termed as *non-conservative* domains.

Figure 1 about here

It is difficult to apply conventional mathematical techniques in non-conservative domains as most mathematical models require a parametric formulation of the domain. Erroneous assumptions about the underlying distributions in formulating a parametric model distorts the results of the models. Rule-based systems also cannot be built as they require expert (deep) knowledge about the domain. It is interesting to explore the use of NNs in such domains because they are non-parametric and make weaker assumptions about the shapes of the underlying distributions than traditional statistical classifiers [26]. They are thus more robust in the face of non-linear processes, non-Gaussian behavior and noise. Many domain models change with time. The adaptive nature of NNs permits them to respond to and adapt with changes in the external world state (a feature lacking in most conventional techniques). NNs also do not require detailed rules about domain behaviors as they are trained by presenting examples of input-output pairs. More details about NNs can be found from references [17,18,26,29,36,42,45].

1.2 Decision support systems and neural networks

DSSs, broadly defined, are computer-based systems which aid decision processes. They come in a multitude of different architectures and are designed for both individual use and group work. Within the framework of the familiar D-D-M (dialog, data and model) architecture of a DSS [46], NNs can be applied to each of the three components: dialog, data and model [12]. This paper is primarily concerned with use of NNs for augmenting the model component of DSSs.

The model component of a DSS provides the analysis capabilities in the form of one or more stored models. There are many different types of models. In general, these models are usually either mathematical or heuristic in nature. Referring to Figure 1, mathematical models have traditionally been applied fruitfully to (conservative) domains having a well

defined domain model. Heuristic models (a result of research in knowledge-based systems) are more useful for (partially conservative) domains having partially defined domain models. DSS typically have ignored (non-conservative) domains lacking a domain model because it is difficult to build both mathematical and heuristic models for such a domain. NNs can play an important role in augmenting the range of applicable domains for DSS. As mentioned in section 1.1, the ability of NNs to learn the domain model autonomously from a set of training examples and then generalize from it makes them potentially useful for non-conservative domains. NNs thus represent a fundamentally different modelling technique for DSSs and have the potential to expand the range of domains to which DSSs can be applied. Figure 2 illustrates how NNs can augment the model base of a DSS.

Figure 2 about here

1.3 Network architecture and problem formulation

The experiments described in this paper only consider layered feed-forward networks [17,18,26,29,36,42,45]. Figure 3 depicts a simple feed-forward NN. There are three layers in the network. The first layer is the "input layer" of four neurons. Four (continuous) input signals, I_1-I_4, come into these four neurons. The output of each of the neurons in the input layer is fed as input to each of the neurons in the next layer, the "hidden layer". The outputs of each of the neurons in the hidden layer are in turn fed as input to the two neurons in the highest layer, the "output layer". There are two continuous output signals (O_1 and O_2) from the network. The neurons in the various layers are numbered for ease of reference. Though not shown in Figure 3, each connection between two neurons has an associated numerical number called the "weight" of the connection. The input to the network comes only via the lowest input layer and likewise the only output from the network comes from the highest output layer. Each node produces an output which is a function of the weighted sum of the inputs.

Figure 3 about here

Training the network consists of determining the appropriate set of connection weights given the training input-ouput sample pairs such that the trained network produces the correct ouput for each input pattern of the training sample. The learning algorithm used in our experiments is the back-propagation algorithm using the Generalized Delta rule [38]. In the back-propagation algorithm, learning is achieved by minimizing the total sum of square errors, TSS, which is defined below:

$$TSS = \sum_p E_p = \sum_p \sum_i (t_{pi} - a_{pi})^2 \qquad (1)$$

where the index pattern p ranges over the set of input patterns, i ranges over the set of output units, and E_p represents the error on pattern p. The variable t_{pi} is the desired output, or target, for the i^{th} output unit when the p^{th} pattern has been presented, and a_{pi} is the actual activation of the i^{th} output unit when the pattern p has been presented. More formal and complete descriptions of the back-propagation algorithm and feed-forward NNs are available in references [17,18,26,29,36,42,45].

The generalization problem can now be described as follows:

> Let I represent the possibly infinite domain of inputs (I_1,I_2,...), and O represent the possibly infinite range of outputs (O_1,O_2,...). Let F represent the k dimensional feature space, F_1,F_2,...,F_k, describing each of the input patterns. Each input pattern I_j can be considered as a k-tuple (F_{1j},F_{2j},..F_{kj}) in the Cartesian space $F_1XF_2X..XF_k$. Given a learning sample S = (S_I,S_O) and per sample error function E:IXO -> Real with S_I={I_1,I_2,...,I_n}, and S_O={O_1,O_2,...,O_n}, generalization involves finding the mapping function
>
> $$f: F_1 \times F_2 \times .. \times F_k \rightarrow O$$
>
> to minimize the error function E over the entire domain I. In particular, $f(I_j)=O_j$+<*small error*> for the learning sample.

There are other approaches to formulating the generalization problem, such as time complexity formalisms [23,31] and as symbolic semantic networks [37],but the above formulation in terms of function learning and curve fitting [1,23,35] is most suited for the purposes of this paper. Pursuing the sole objective of minimizing the error function, E, can lead to overfitting [18,36] as shown in Figure 4 and thus result in poor generalization. To avoid overfitting, one may adopt one or more of the following strategies [35]: (a) use special

stopping criteria (b) add noise to the learning sample, and (c) use a simple network structure (i.e., with a small number of hidden units). Note that for the purposes of generalization, it is preferable to have the network learn a simple smooth function as opposed to a more complex function that fits all the training data precisely (the latter situation is more desirable for recognition problems). It is also obvious from the above problem formulation that the choice of the training sample has a significant effect on the generalization capabilities of the network. Choosing S_I such that it adequately represents I is important and useful.

Figure 4 about here

1.4 Focus and structure of paper

Traditional DSSs, with their emphasis on mathematical and heuristic models are limited in their applicability to non-conservative/unstructured domains. NNs present an opportunity for overcoming this limitation due to their capability to generalize from a domain model learnt autonomously from input-output examples. The focus of this paper is on exploring the use of NNs for decision support by generalization in non-conservative / unstructured domains.

Three different sets of experiments are reported in this paper, one in an artificially simulated environment and two with real world data. The performances of the NN models have been better compared to regression models for each of the three experiments. The first experiment is a study of the generalization capabilities of NNs for domains represented by simple, smooth functions. In this case, four different domains - linear, quadratic, logarithmic and trigonometric - were chosen and a series of experiments conducted to determine the generalization capabilities of NNs. As the true domain models are known (but not to the NNs), this provides a controlled setting for studying the generalization capabilities of NNs in the chosen domains.

For the remaining two experiments, the real world situations of assigning corporate bond ratings and predicting product purchase frequency have been chosen. Bond rating is the problem of assigning ratings (such as AAA,

AA, etc.) to corporate bonds. Such ratings are typically given by private agencies using proprietary analyses. The model used for determining these ratings is not known. These ratings reflect the default risk of the bond and are used by investors for selecting investments. It is thus an importance problem in finance. The last experiment is in the domain of market modelling and is concerned about modelling choice at the individual level by studying the effect of variables such as product purchase history, advertising, promotions, and store causal activity. Predicting short run individual product purchase frequency is an important problem in marketing science and has a significant influence on the marketing strategies of firms.

The structure of this paper is as follows. There are four additional sections. The next section describes the controlled experiment on the generalization capabilities of NNs. Sections 3 and 4 describe the experiments with real world data of bond ratings and product purchase frequency respectively. The last section concludes the paper.

2. Generalization in a Controlled Experiment

This section describes a controlled study of the generalization abilities of back-propagation networks in domains represented by smooth continuous functions. More details on these experiments are available in reference [51].

2.1 Generation of Learning and Testing Data

Four different domains were considered in this study: linear, quadratic, logarithmic and trigonometric. The data sets were derived from the four functions shown analytically in Table 1 and graphically in Figure 5. The following observations can be made regarding these functions: they are all continuous and of the form $R^2 \rightarrow R$, R representing the set of real numbers; each of them has two independent variables and while the first three functions are monotonic, the last function has maxima and minima.

Table 1 and Figure 5 about here

For use by the NN simulators, the outputs of these functions have to be mapped to values in the range (0,1). Similarly, the outputs of the NNs have

to be reverse-mapped to the valid output ranges for data analyses. To study the effects of noise, four noise levels of 0%, 2%, 5% and 10% are added to $f_i(x,y)$. The random noise is assumed to be a uniform distribution with zero mean and constant variance. If k is the level of noise, a noise generated uniformly in the range of $(-k*f_i(x,y), k*f_i(x,y))$ is added to $f_i(x,y)$. Each data set consists of 225 input pairs of (x,y) values which are generated from a rectangular grid with 15 distinct values each for x and y. Each data set of 225 points is grouped randomly into three sets: 125 points for the learning sample, and 50 each for the testing and overfit-testing samples.

2.2 Network Architectures

Three different feed-forward network architectures were considered: 2X2X1, 2X2X2X1 and 2X6X6X1, where the leftmost and rightmost numbers give the number of nodes in the input and output layers respectively and the intermediary numbers specify the number of nodes in the hidden layers. The choice of the network architectures represents a compromise between the necessity to include enough hidden units to learn simple nonlinear regularities and the necessity to avoid "memorization" of the entire data set [49] (i.e., overfitting [18]). Note that for achieving satisfactory generalization, it is desirable to fit a the NN to a simple curve (and tolerate some error in the output of the NN) as opposed to fitting the network very closely to the training data.

The learning algorithm used was the Back propagation learning algorithm using the Generalized Delta rule [38]. The network was trained in the epoch mode, i.e., weights on the connections of the network were changed only after each sweep through the patterns in the learning sample. This removes the sensitivity of the network to the particular permutation of training data samples (which would be lost if the weights were adjusted after each item of the learning set). The training was set to terminate if either the total sum of square errors, TSS, (equation (1)) fell below 10^{-6} or the number of epochs was 50,000. All the NNs (for all data sets) terminated upon reaching 50,000 epochs. The TSS for the linear and quadratic domains fell to 10^{-4}, but the TSS for the logarithmic and trigonometric domains only reached values of the magnitude 10^{-2} and 10^{-1} repectively. The trained networks (after 50,000 epochs) were applied to the testing samples to determine their generalization capabilities.

2.3 Performance evaluation by visual observation

It is possible to plot the functions learned by the trained networks (using the final connection weights) and compare them visually with the domain plots represented by the corresponding functions. Figure 6 presents a 3-D plot of the functions learnt by the 2X2X1 network. Comparing the plots of Figures 5 and 6, we observe that the network has been able to learn the form of the linear, quadratic and logarithmic domains quite well and that of the trigonometric domain to a fair degree.

Figure 6 about here

2.4 Performance evaluation by numeric metrics

The numeric metric chosen is the mean absolute percent error (MAPE) per pattern, which is defined for both the learning and testing samples as:

$$\text{MAPE} = \frac{1}{\text{no\_of\_patterns}} \sum_{\text{all\_patterns}} \frac{|\text{Actual\_output} - \text{Target\_output}|}{\text{Target\_output}} * 100\%$$

As the input patterns are corrupted by varying degrees of noise, a more accurate indicator of network performance is the noise adjusted MAPE (NAMAPE), defined as:

NAMAPE = MAPE - Average noise level

NAMAPE discounts the contribution of noise to the MAPE value. A value of NAMAPE close to 0 shows good generalization capabilities. A large positive value of NAMAPE points to poor generalization capabilities and a large negative NAMAPE value indicates the presence of overfitting.

Tables 2 & 3 summarize the results of the application of numeric metrics for the learning and testing phases respectively. Several runs of the various network configurations were made for different noise levels. For each of the four different domains, the best, the worst and average NAMAPE values are listed for the three different network configurations and for a regression

model based on reference. The apriori functional forms chosen for the regression models are based on the appropriate domains form (e.g., sin(x) and cos(y) are used for the trigonometric domain). It can be observed from Tables 2 and 3 that the values of NAMAPE (for the NN) are often higher for the learning sample as compared to the testing sample for the linear, quadratic and logarithmic domains. This is because, by chance, the random assignment of points resulted in a larger number of points with smaller output values in the training samples as compared to the testing samples for these domains. Reducing the total sum of square errors, TSS, in the back propagation algorithm only reduces the magnitude of error, and for small output values, this may still result in large absolute percent errors.

Tables 2 and 3 about here

2.5 The Presence of Overfitting

The presence of overfitting can be observed by drawing the learning curve and the overfit-test curve. The former is a plot of the MAPE values of the training data set versus the number of epochs of learning and the latter is a plot of the MAPE of the overfit-test data set versus the number of epochs of learning. To detect overfit, we have to look for the point on the overfit test curve where the curve reaches a minimum MAPE, but somehow moves away from the minimum with more training and never gets back to the same low MAPE value again. Figure 7 depicts the learning and overfit-test curves for the 2X6X6X1 network for the logarithmic domain.

Figure 7 about here

2.5 Interpretation of results

The following preliminary observations can be made from the results of this experiment:

- The NN model can learn the domain functions very well for the linear, quadratic and logarithmic domains, and reasonably well for the trigonometric domain. This can be observed by visually comparing

Figures 5 and 6, and by studying Tables 2 and 3 where the NAMAPE values are close to zero for the linear, quadratic and logarithmic domains. The NAMAPE values for the trigonometric domain is significantly higher than those for the other three domains.

- There is a improvement in the performance of the NN with increasing network size. The enhanced performance of larger networks is especially marked for the trigonometric domain, which is the most complex of the four domains under consideration.

- The performance of the regression models is not significantly better than the best comparable network performance. Note that the regression models had an advantage over the NN models as they were trained using the known functional form (with unknown coefficients) of the corresponding domain.

- The presence of noise also helped the generalization abilities of networks. It was observed that the NAMAPE values generally decreased with increasing levels of noise. The presence of noise can be seen as helping the system search for a global minimum in the weight space.

- A relatively small amount of overfit was observed in our experiments. This is probably because the size of the networks were deliberately chosen to be small and the domains under consideration are relatively simple continuous functions. However, it can be observed from Tables 2 and 3 that some overfit occurs with the 2x6x6x1 network, specially for the linear and logarithmic domains.

This experiment can be considered as a first step in the design of experiments to study generalization by NNs in controlled settings. Further experiments is desirable using more complex domain functions (such as piece-wise smooth functions and step functions) and with larger network structures. However, even these results are useful for giving a sense of the generalization capabilities of NNs.

3. Corporate Bond Rating

This section describes our experiment in generalization for the problem of assigning corporate bond ratings. These results were first reported in reference [9].

3.1 Introduction

The issuance of corporate bonds is an important source of cash for companies. The default risks of the most actively traded bonds are rated by various independent organizations such as S&P and Moody's. The purpose of these ratings [27] "is to provide the investor with a simple system of gradation by which the relative investment qualities of bonds may be noted". These ratings are often used to define allowable bond purchases by banks and other investors.

To evaluate a bond's potential for default, rating agencies rely upon a committee analysis of the issuer's ability to repay, willingness to repay, and protective provisions for an issue. It is not known what model, if any, do these rating agencies use for rating the various bond issues. All aspects analyzed by the ratings committee are also not known completely, and some features such as *willingness to repay* are affected by a number of variables that are difficult to characterize precisely.

3.2 Prior Research

Many complex regression models have been proposed in the literature for the problem of corporate bond rating. The typical financial variables used include proxies for liquidity, debt capacity, debt coverage, size of issue, operating efficiency, depreciation policy, profit margin, long term debt, cash flow to debt [43] etc. Horrigan [21] regressed coded ratings with six ratios: total assets, working capital over sales, net worth over total debt, sales over net worth, profit over sales, and subordination. This model was correct for 58% of Moody's ratings during the period 1961-1964. West [48] has a similar approach, using logarithmic forms of nine variables including earning variability, solvency period, debt equity ratio and outstanding bonds. His model correctly predicted 62% of Moody's rating during 1953. Pogue and Soldofsky [33] used a regression model with a dichotomous (0-

1) dependent variable, which represents the probability of group membership in one group of pairs. This method predicted 8 bonds out of 10 in a hold out sample from the period 1961-1966. Pinches and Mingo [32] screened a set of initial 35 variables via factor analysis, and used multiple discriminant analysis to develop the final model. This model predicted roughly 65% and 56% of the Moody's ratings for hold out samples in the periods 1967-1969. Research on bond rating models has also been done by Ang and Patel [2], Belkaoui [3], Ederington [10], Gentry et. al. [14], and Kaplan and Urwitz [24].

More recently, some novel approaches have been used to the problem of corporate bond rating. Srinivasan and Bolster [43] have use the Analytical Hierarchy Process to model the process of assigning corporate bond ratings. They applied their model successfully to the examples of two large American motor companies. Dutta and Shekhar [9], Garavaglia [13] and Utans and Moody [44] have used NNs for the task of bond rating. The work of Dutta and Shekhar [9] was the first application of NNs to bond rating and is summarized below in this section. Garavaglia [13] has used the counter-propagation NN for bond rating. Her results indicate that while the NN gave poor results (maximum accuracy 55%) for predicting any one particular bond rating, the accuracy of NNs increased substantially (maximum of 92%) if the the problem was simplified to detecting three broad classes of bonds (investment, speculative and poor quality). Utans and Moody [44] conducted experiments in the domain of bond rating to select appropriate NN architectures. They used 196 companies and found a maximum accuracy of prediction of 28% for the prediction of any one rating. The success rate increased to 67% for errors of one or less rating.

3.3 Selection of Variables

Based on the results of prior researchers in bond rating, we selected ten financial variables for predicting bond ratings. A subjective estimate of the influence of a variable on bond rating and the ease of availability of data were the primary factors in the selection of the variables. The selected variables are listed in Table 4.

Table 4 about here

As the feature space of the input vector is not known accurately in the problem of bond rating, we ran two sets of experiments to verify whether choosing a subset of the initially selected feature space would significantly alter our results. Our first experiment used all ten variables in predicting the bond rating. Then we used only the first six variables (#1 - #6) to predict the bond ratings.

3.4 Data Collection

Values of the necessary input variables were collected from the April 1986 issues of the Valueline Index and the S&P Bond Guide. Bond issues of forty seven companies were selected at random. Thirty companies were used as the learning sample, i.e., to train the NN (learn the weights on the different connections) and obtain the regression coefficients. The rest seventeen companies were used to test the neural network and the regression performance. All the selected bonds had approximately the same maturity date (1998 - 2003). All variable values were scaled to lie in the range [0,1].

For the purposes of our experiment, the symbolic ratings of bonds were converted into numerical values as shown in Table 5. For enhancing discrimination in the output, we mapped similar ratings (e.g., AA+, AA and AA-) onto the same numerical value (e.g., 0.9). The varying difference in the numerical values corresponding to adjacent rating groups (e.g., AA and A) reflect the investment quality of bonds from one rating group to the next.

Table 5 about here

3.5 Neural Network Architecture

Five different feed-forward networks were used in this experiment: 6X1, 10X1, 6X3X1, 10X3X1 and 10X4X1. The back-propagation algorithm [38] was again employed for learning. The network was trained in the epoch mode and the termination condition was set to be 50,000 epochs or a value of TSS (equation (1)) of 10^{-4} whichever came first. In all cases, the training was terminated upon reaching 50,000 epochs.

3.6 Results

The models were expected to recognize if a given bond belongs to class of all bonds with AA rating. For an ideal model, our test results should only belong to the first and fourth rows of Table 6 where the ratings given by S&P and the model coincide. Rows 2 and 3 of Table 6 are the undesirable cases and represent false negatives and false positives respectively.

Table 6 about here

Tables 7 and 8 summarize the results of our experiments. Table 7 condenses the results of the learning phase and Table 8 summarizes the results of the testing phase. The % entries in Tables 7 and 8 represent the % of correctness of prediction of the two models (linear regression and NN) and are obtained by computing a weighted average for the response pairs (Accept, Accept) and (Reject, Reject). We also list the absolute error as tot_sq_err for each model to give an idea of goodness of fit by various models. The tot_sq_err gives the sum of the squares of the errors in prediction in all the cases. The Berkeley ISP [4] was used for running the regression models.

Tables 7 and 8 about here

Tables 9 and 10 specifies the deviations of the output of the NN and (linear) regression models from the actual ratings. It must be noted that the data of Tables 9 and 10 include all ratings and are not restricted to the problem of only recognizing the AA ratings.

Tables 9 and 10 about here

A logistic regression model was also run on the sample data using SAS [39]. An ordinal response function was used (corresponding to the different levels shown in Table 5). Three different link functions (available in SAS) were used:

- The logit function, $g(p) = \log((p/(1-p))$, which is the inverse of the cumulative logistic distribution function, $F(x)= 1/(1+\exp(-x))$
- The normit function, $g(p) = f^{-1}(p)$, which is the inverse of the standard normal distribution function, and
- The complementary log-log function, $g(p)=\log(-\log(1-p))$, which is the inverse of the Gompertz distribution, $F(x)=1-\exp(-\exp(x))$.

Table 11 summarizes the results for the training sample and for the combined training and testing data samples. The complementary log-log function could not be run for the training sample due to an inadequate number of data points.

Table 11 about here

3.7 Interpretation of Results

The following observations can be made from this experiment:

- Neural networks consistently outperform the linear regression model in predicting bond ratings from the given set of financial ratios. Both in the training and learning samples the total squared error for (linear) regression analysis is about an order of magnitude higher than that for neural networks . Also, the success rate of prediction for NNs is considerably higher than that for regression analysis, e.g., the success rate during the testing phase for the two layered NN (10 variables) is 88.3% as compared to 64.7% for the regression model.

- Comparing the performance of 2 and 3 layered networks, we observe that the performance of 3 layered networks is better during the learning phase (the accuracy of prediction is marginally better but the total error is much lower). But surprisingly, the performance of 2 layered networks is superior during the testing phase. This is possibly due to "over-fitting" to data while using 3 layered networks.

- The testing accuracy is generally higher while considering 10 variables as opposed to 6 variables. This indicates the value of choosing a as complete (as possible) input feature space.

- Tables 9 and 10 indicate that the NN models usually generated errors of a single rating category (i.e., an AA bond was rated as an A bond or a BB bond was rated as a BBB bond and so on). In contrast, regression models usually generated errors of two or more rating categories.

- There is also no appreciable change in the prediction accuracy of the regression model from the learning to the testing phases. No significant differences were also observed between the linear and logistic regression models.

It is useful to compare the results of our experiments with those obtained by other researchers (even though this comparison is not fully fair because all three experiments used different input feature spaces and data). The success rates obtained by Utans and Moody [44] and Garavaglia [13] for the prediction of any one bond rating is fairly low (28%-55%). The problem posed in our experiment was simpler: just to recognize one rating (AA). Garavaglia found that prediction accuracy increased substantially (to 92%) when three broad categories (investment grade, speculative and poor quality) of bonds were chosen to be recognized. A look at Table 5, which specifies our conversion of ratings to numbers in the range [0,1], shows that our experimental design (in retrospect) also had three broad categories of bonds (0.95-0.9, 0.7-0.5, 0.35-0.25). Thus our high prediction accuracy is not inconsistent with her observations. It must also be noted that, being one of the first experiments in this domain, our experimental design was considerably simpler than that of Utans and Moody and Garavaglia. The choice of a small sample and the simplest holdout method for estimating prediction accuracy can lead to an overly optimistic prediction success rates [28,47].

4. Prediction of product purchase frequency

This section describes our experimental results in using NNs for predicting product purchase frequency.

4.1 Introduction and prior research

Market modelling is an extremely important issue in marketing. At the aggregate level, market share models are commonly used in marketing for a number of different purposes. These include the estimation of price and advertising elasticities as well as more generally predicting the effects of changes in marketing variables. A number of different model formulations have been proposed with a major issue being the specific form of the functional relationship assumed. Previous researchers have used linear, multiplicative and attraction-type models which are typically estimated using OLS and GLS procedures [7,15]. However, no single model or estimation procedure has consistently outperformed the others. An excellent comparison of a number of market share models is reported by Ghosh, Neslin and Shoemaker [15] while a recent review of the area is provided by Cooper and Nakanishi [7].

This section is more concerned with the use of NNs for studying response at the individual level. In this area, individual choice behaviour can be studied on three dimensions: purchase timing, brand choice and purchase quantity. Two broad approaches have been used to study these areas in the past: stochastic models and more formal choice models (see [16]) using random utility theory. Stochastic models of buyer behaviour have tended to focus on the timing aspect of the choice problem, though they have also been used with less success in modelling brand choice. A major weakness of these models is the intuitively unappealing assumptions regarding the distribution of individual purchase frequencies. Further, these models do not include any marketing variables and are thus useful for estimating baselines but not for directly assessing the effect of marketing activities. Formal models of choice have been shown to have good face validity in a number of marketing analyses but still have some disturbing properties such as the Independence of Irrelevant Alternatives (IIA) problem. Further, all such models must necessarily make assumptions about the error structure of the model and often these assumptions are difficult to justify.

Our experiment explores the use of NNs to model choice at the individual level by studying the effect of variables such as product buying history, advertising, promotions and store causal activity. Our particular focus is on

the use of NNs to assess the short-term response to marketing activity in a limited domain to illustrate the use of NNs for the decision support of a user.

4.2 Data Collection

The source of data for this experiment is the MSI Library of Single-Source data. The data are supplied by Nielsen Co. using the ERIM marketing testing service. The data used in these experiments cover purchases of all 6 ozs packs of yogurt in a typical small town in central USA, and participating stores account for 93% of total yogurt sales in that town. The time span covered by the data is from the first week of 1986 till the 34th week of 1988.

As the data are aggregated weekly (i.e., all product purchases during the week are aggregated into one value), there are 138 different data vectors in the data set. Each data vector contained the following elements:

- ERWK: the week number coded to a value between 198601 to 198834
- NOCS: the number of purchase occasions per week
- DISP: weighted display activity in the market
- ADC: weighted advertising code activity
- IACC: weighted in-ad coupon code
- PPC: weighted point-of-purchase code
- SPC: weighted special price code

Our aim is to be able to predict the variable NOCS given the data about the last five variables mentioned above. For the purposes of experimentation, the data set was divided into two samples: the data for the years of 1986 and 1987 were included in the learning sample and the data for the 34 weeks of 1988 were included in the testing sample.

4.3 Neural Network Architectures

Similar to the experiments described in sections 2 and 3, the back-propagation network was used here also. Three different configurations of network architectures were tried: 5X3X1, 5X6X1 and 5X9X1. Most of the networks were trained for 10,400 cycles through the data. This number was chosen as a compromise between the time for training (each training session typically occupied about 45 minutes on a 20Mhz PC) and the

general observed improvement in the convergence of the network beyond a certain number of iterations. None of the networks converged during the specified limit of the training cycles.

4.4 Results

The results obtained from the testing phases of the various networks are shown in Table 12. The network characteristics specifies the number of hidden nodes, the network alpha value (which affects the learning rates of the networks), a variable "I/O Connection" which indicates whether the input nodes have a direction to the output nodes in the feed-forward network and the number of cycles for which the network was trained. A multiple linear regression was also run on the learning data and later used for predicting the test data. The results obtained with the regression model are also included in Table 12.

Table 12 about here

4.5 Interpretation of results

The results of Table 12 should be considered as preliminary since our experiments in this domain are at an initial stage. The performance of only one NN outperformed the multiple linear regression model. This particular neural network's (N6A4NT1) prediction of the average NOCS (440.78) is very close to that the actual value (447.64). The performance of networks with 3 to 6 nodes in the hidden layer was generally superior to networks with a larger number of hidden layer nodes (the best network has 6 nodes in the hidden layer). This can be observed from Table 12 by comparing the performance of the network (N9A1YT1) to the other networks. This gives some idea about the desirable number of nodes in the hidden layer.

It is not evident from these first results that NNs can lead to effective generalization in this domain. An important problem is the determination of the fact whether the input feature space is complete or adequate for determining the output variable. Given the poor performance of the NNs in these first experimental runs, it is likely that the input feature space has to be modified to consider other factors (such as historically aggregated values of

variables such as advertizing). This indicates the importance of being able to determine a reasonably complete input feature space - a feat not easily accomplished in non-conservative domains. Extensive experimentation with different combinations of input/output feature spaces and network structures/types is necessary to overcome this bottleneck.

5 Conclusion

Mathematical and heuristic models are commonly used for problem solving. The construction of mathematical models demands a good understanding of the underlying domain model. Heuristic rule-based systems also require a fair knowledge about the relevant domain model (which is usually acquired incrementally during knowledge engineering). Both these modelling techniques are not very applicable for non-conservative domains where knowledge about the underlying domain model is poor and knowledge engineering is difficult. NNs represent an alternative modelling technique which is applicable to problem solving and modelling in non-conservative domains. NNs autonomously learn the domain model from examples and do not require the a priori specification of the model.

Most successful NN applications have been reported for recognition problems [11,19,25,40,50,52]. Recently there has been an increased emphasis on the application of NNs for generalization problems [1,5,6,8,9,13,20,22,28,30,34,47,49] and the findings in the literature are mixed. This paper has described the use of NNs for generalization in three different non-conservative domains, an artificially simulated controlled setting and the real world domains of corporate bond rating and market modelling. Our results from the controlled experiment indicates that NNs have the potential to generalize effectively in domains with simple continuous models. Our observations from the real world domains of corporate bond rating and market modelling are mixed. The performance of NNs for corporate bond rating is impressive, but one must keep in mind the limitations of the experimental design. It is quite likely that the performance accuracy of NN for bond ratings in our experiments is optimistic. However, it is useful to note that both our results and those of Garavaglia [13] indicate that NNs can play an important role in predicting bond ratings if the problem is suitably posed (in this case, as the determination of three broad bond categories as opposed to distinguishing between individual categories). Our

first results in the use of NNs for market modelling are not very encouraging, with only one network being able to outperform regression models. Our experiments in this domain are still continuing and it is possible that further research and experimentation will yield better results.

Though NNs have the potential to provide effective decision support in non-conservative domains, one must not forget that "success is not guaranteed" by simply using NNs. There are no formal guidelines for developing NN applications. Designing an application with NNs is an intricate merger of art and science. Important factors are:

- the design of appropriate input and output feature spaces for the domain
- the choice of the appropriate training/testing samples
- selection of the proper network type and structure
- running different experiments with various combinations of the above

DSSs have traditionally relied only on mathematical and heuristic models. Thus they have limited applicability in non-conservative/ unstructured domains. NNs present an opportunity to enhance the productivity through model base of DSSs and extend the range of domains to which DSSs can be applied. We hope that our experiments are useful in this regard.

References

[1] S. Ahmad, and G. Tesauro, Scaling and Generalization in Neural Networks: A case study, in the Proceedings of the International Conference on Neural Information Processing Systems (1988) 160-170.

[2] J.S. Ang, and K.A. Patel, Bond Rating Methods: Comparison and Validation", Journal of Finance, Vol. 30, No.2 (May 1975) 631-640.

[3] A. Belkaoui, Industrial Bond Ratings: A New Look, Financial Management, Autumn (1980) 44-51.

[4] Berkeley Interactive Statistical Package, Department of Statistics, UC Berkeley (1988).

[5] M. Caudill, Using Neural networks: Making an Expert Network, AI Expert (July 1990) 41-45.

[6] E. Collins , S. Ghosh and C.L. Scofield, An Application of Multiple Neural networks to Emulation of Mortgage Underwriting Judgement, in Proceedings of IEEE International Conference on Neural networks, San Diego, Vol II (1988) 459-466.

[7] L.J. Cooper and M. Nakanishi, Market Share Analysis:Evaluating Competitive Market Effectiveness (Kluwer, 1988).

[8] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel and J. Hopfield, Large Automatic Learning, Rule Extraction and Generalization, Complex Systems, 1 (1987) 877-922.

[9] S. Dutta and S. Shekhar, Bond-Rating: A Non-conservative application of neural networks, in the Proceedings of the IEEE International Conference on Neural Networks, San Diego, Vol. II (July 1988) 443-450.

[10] L. Ederington, Classification Models and Bond Ratings, The Financial Review, November (1985) 237-262.

[11] K. Fukushima, A neural network for visual pattern recognition, IEEE Computer (March 1988) 65-75.

[12] S.I. Gallant, Connectionist Expert Systems, in the Communications of the ACM, Vol. 31, No. 2 (Feb. 1988) 152-168.

[13] S. Garavaglia, An application of a Counter-Propagation neural network: Simulating the Standard & Poor's corporate bond rating system, in the Proceedings of the 1st International Conference on Artificial Intelligence Applications on Wall Street (IEEE Computer Society Press, 1991) 278-287.

[14] J.A. Gentry, D.T. Whitford, and P. Newbold, Predicting Industrial Bond Ratings with a Probit Model and Funds Flow Component, BEBR

Working Paper No. 1198, University of Illinois, Urbana-Champaign (1985).

[15] A. Ghosh, S. Neslin and R. Shoemaker, A Comparison of Market Share Models and Estimation Procedures, Journal of Marketing Research (May 1984).

[16] P. Guadagni and J.D.C. Little, Logit Model of Brand Choice Calibrated on Scanner Data, Marketing Science (Summer 1983).

[17] R. Hecht-Nielsen, Neurocomputing (Addison Wesley, 1989).

[18] J. Hertz, A. Krogh and R.G. Palmer, Introduction to the theory of neural computing (Addison Wesley, 1991).

[19] G.E. Hinton, Learning Translation Invariant Recognition in a Massively Parallel Network, Lecture notes in computer science #258, Springer-Verlag (1987) 1-13.

[20] K. Hornik, M. Stinchcombe and H. White, Multi-layer Feedforward Networks are Universal Approximators, Neural networks 2 (1989) 359-366.

[21] J.O. Horrigan, The Determination of Long Term Credit Standing with Financial Ratios, Empirical Research in Accounting: Selected Studies, Journal of Accounting Research (1966) 44-62.

[22] A. Irani, J.P. Matts, J.M. Long and J.R. Slagle, Using artificial neural nets for statistical discovery: Observations after using back-propagation, expert systems, and multiple-linear regression on clinical trial data, University of Minnesota Technical Report (1989).

[23] J.S. Judd, Neural network design and the complexity of learning (MIT Press, 1990).

[24] R. Kaplan, and G. Urwitz, Statistical Models of Bond Ratings: A Methodological Inquiry, Journal of Business, 52 (1979) 231-262.

[25] T. Kohonen, The "neural" phonetic typewriter, IEEE Computer (March 1988) 11-22.

[26] R.P. Lippman, An Introduction to Computing with Neural networks, IEEE ASSP Magazine (April 1987) 4-22.

[27] Moody's Bond Record, Moody's Corporation, NY, (1988).

[28] R. Mooney, J. Shavlik, G. Towell, and A. Gove, An experimental comparison of symbolic and connectionist learning algorithms, in the Proceedings of the International Joint Conference on Artificial Intelligence (1989) 775-780.

[29] M.M. Nelson, and W.T. Illingworth, A Practical Guide to Neural Nets, (Addison Wesley, 1991).

[30] H.V. Parunak, Material Handling: A Conservative Domain for Neural CoNeural networksectivity and Propagation, in the Proceedings of the AAAI Conference (1987) 307-311.

[31] J. Patarnello and P. Carnevali, Meaning of generalization, in Neural Computing Architectures, I. Aleksander (Ed.) (MIT Press, 1989).

[32] G.E. Pinches and K.A. Mingo, A Multivariate Analysis of Industrial Bond Ratings", Journal of Finance (March 1977) 1-18.

[33] T.E. Pogue and R.M. Soldofsky, What is in a Bond Rating?, Journal of Financial and Quantitative Analysis (June 1969) 201-228.

[34] R. Poli, S. Cagnoni, R. Livi, G. Coppini, and G. Valli, A neural network expert system for diagnosing and treating hypertension, IEEE Computer (March 1991) 64-71.

[35] C.V. Ramamoorthy and S. Shekhar, A Stochastic learning algorithm for generalization problems, in the proceedings of the IEEE International Conference on Neural Networks, (1989).

[36] D.E. Rumelhart and J. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vols. 1-2, (MIT Press, 1986).

[37] D.E. Rumelhart, Brain Style Computation: Learning and Generalization (Academic Press, 1990).

[38] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning Representations by Back-Propagation Errors, Nature, Vol. 323 (1986) 533-536.

[39] SAS Reference Manual, SAS Institute Inc. (1989).

[40] T.J. Sejnowski and C.R. Rosenberg, NETtalk: A Parallel Network that Learns to Read Aloud", Johns Hopkins Univ. Tech. Report JHU/EECS-86/01 (1986).

[41] E.H. Shortliffe, Computer Based Medical Consultations: MYCIN, (Elseveir, 1976).

[42] P.K. Simpson, Artificial neural systems: Foundations, Paradigms, Applications and Implementations, (Pergamon Press, 1990).

[43] V. Srinivasan and P.J. Bolster, An industrial bond rating model based on the Analytic Hierarchy Process, European Journal of Operational Research, 48, (1990) 105-119.

[44] J. Utans and J. Moody, Selecting neural network architectures via the Prediction Risk: Application to Corporate Bond Rating Prediction, in the Proceedings of the 1st International Conference on Artificial Intelligence Applications on Wall Street (1991) 35-41.

[45] P.D. Wasserman, Neural computing: theory and practice, (Van Nostrand Reinhold, 1989).

[46] H.J. Watson and R.H. Sprague, Jr., The Components of an Architecture for DSS, in Decision Support Systems, R.H. Sprague and H.J. Watson, (Eds.), (Prentice Hall, 1989) 107-117.

[47] S.M. Weiss and I. Kapouleas, An empirical comparison of pattern recognition, neural nets, and machine learning classification methods, in the Proceedings of the International Joint Conference on Artificial Intelligence (1989) 781-787.

[48] R.R. West, An Alternative Approach For Predicting Corporate Bond Ratings, Journal of Accounting Research, (Spring 1970) 118-127.

[49] H. White, Economic prediction using neural networks: The case of IBM Daily stock returns, in the Proceedings of the IEEE International Conference on Neural Networks, Vol II (July 1988).

[50] B. Widrow and R. Winter, Neural nets for adaptive filtering and adaptive pattern recognition, IEEE Computer (March 1988) 25-39.

[51] W.Y. Wong, A controlled study of the generalization ability of a backward propagation neural network, M.S. Thesis, Computer Science Dept., University of Minnesota (Dec. 1990).

[52] K. Yamada, H. Kami, J. Tsukumo, and T. Temma, Handwritten numerical recognition by multi-layered neural network with improved learning algorithm, in the Proceedings of the International Conference on Neural networks, IEEE and ICNN, (June 1989).

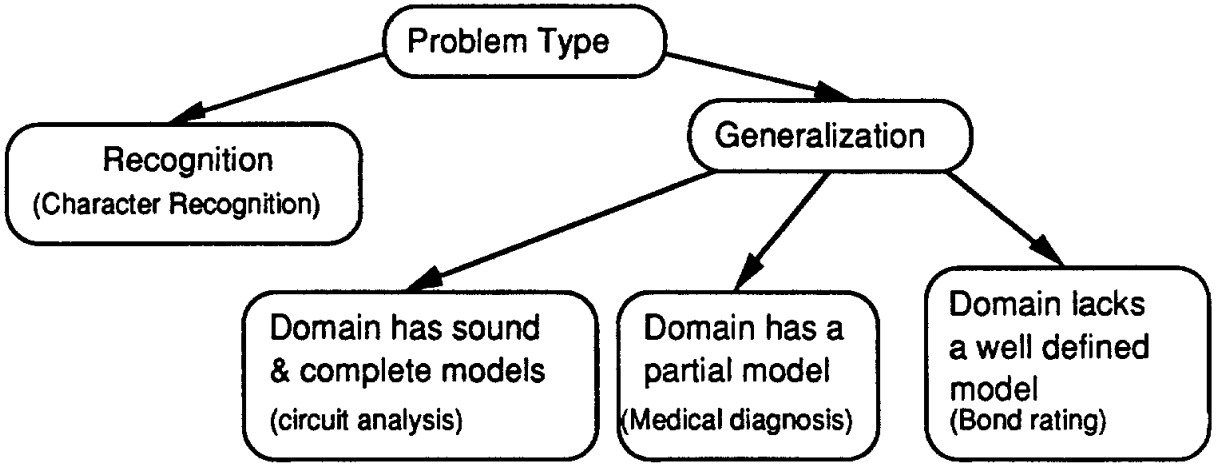


Figure 1: Recognition and Generalization Problems

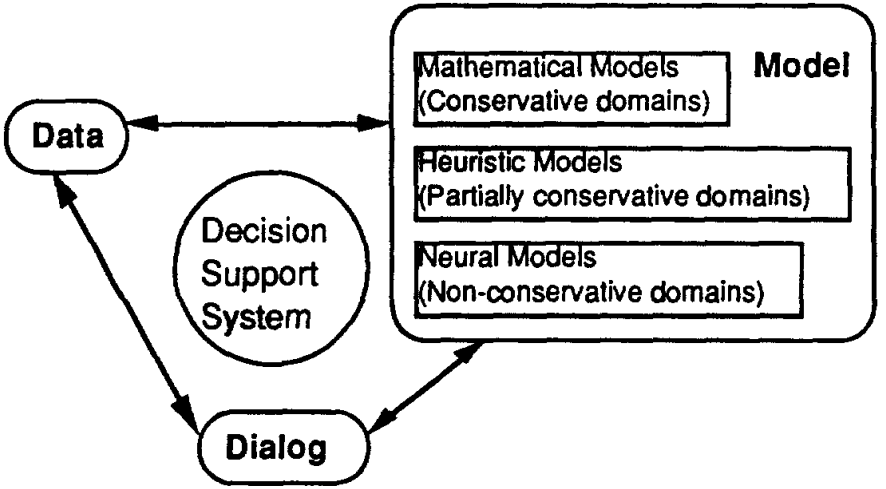


Figure 2: NN augmenting the model base of a DSS

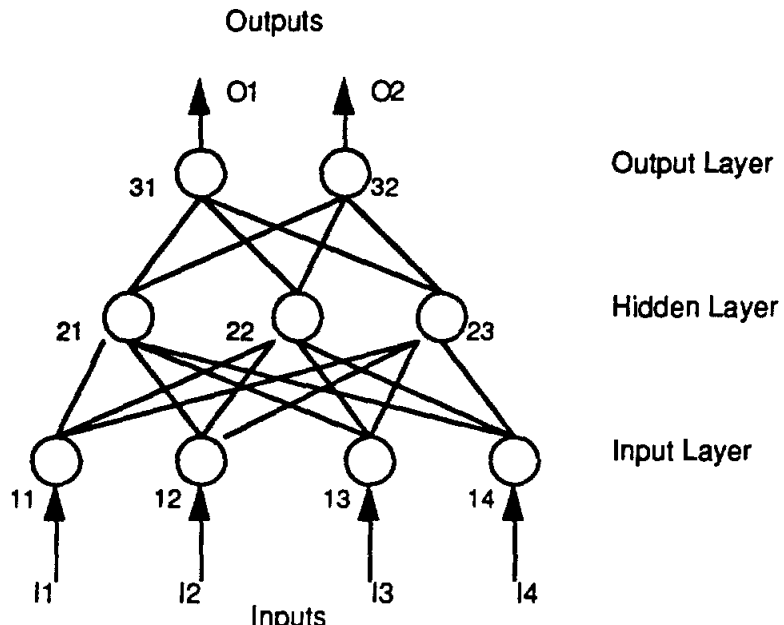


Figure 3: A simple three layered feed-forward network

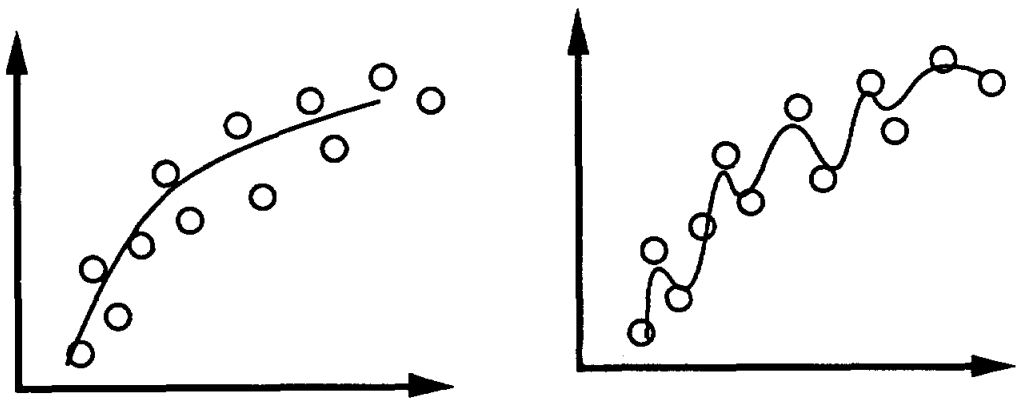
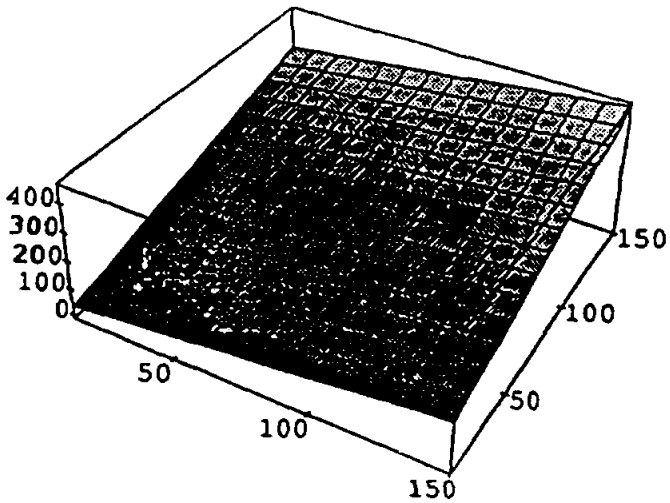
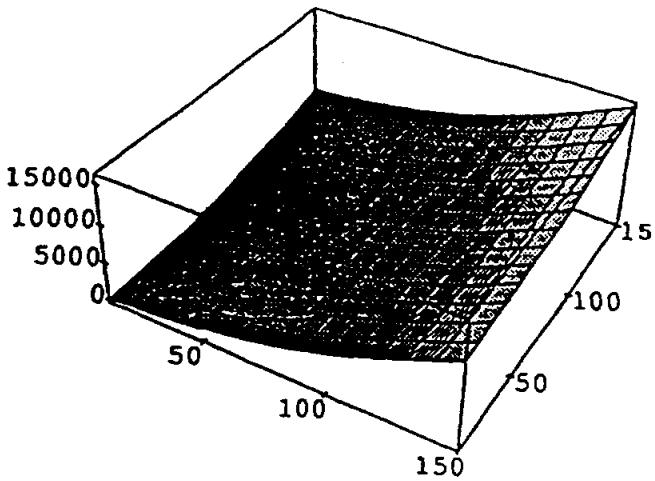


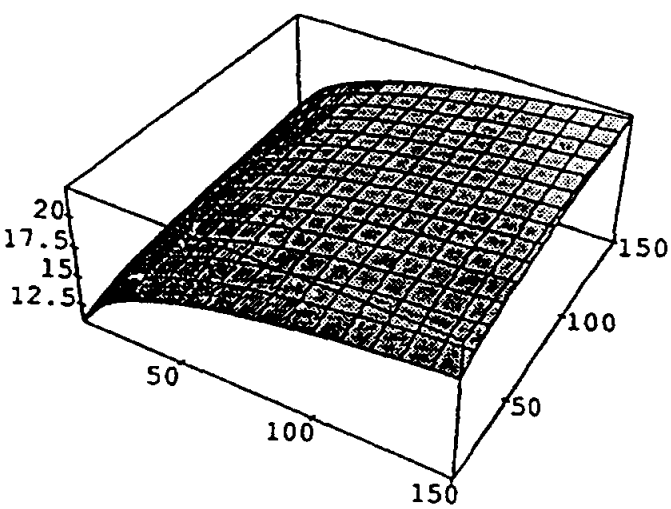
Figure 4: Over-fitting to training data



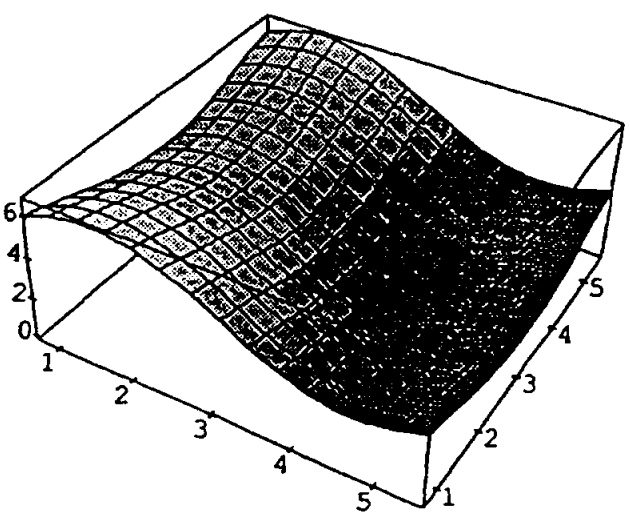
Linear Domain



Quadratic Domain

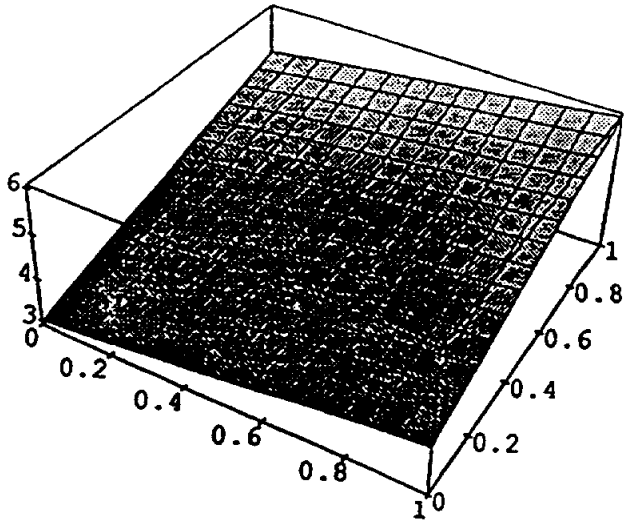


Logarithmic Domain

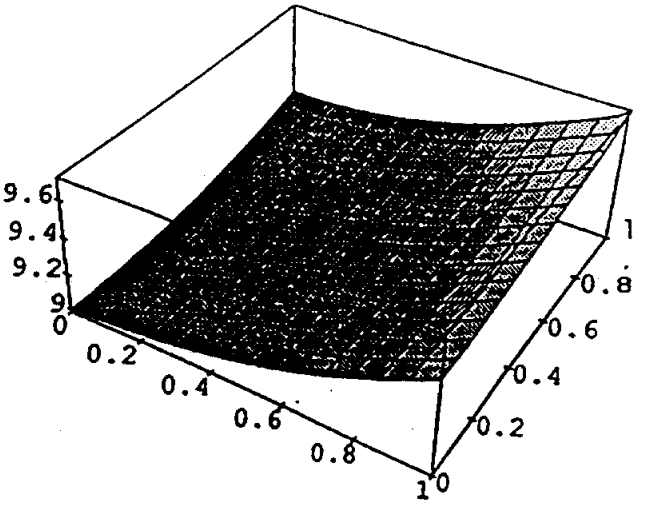


Trigonometric Domain

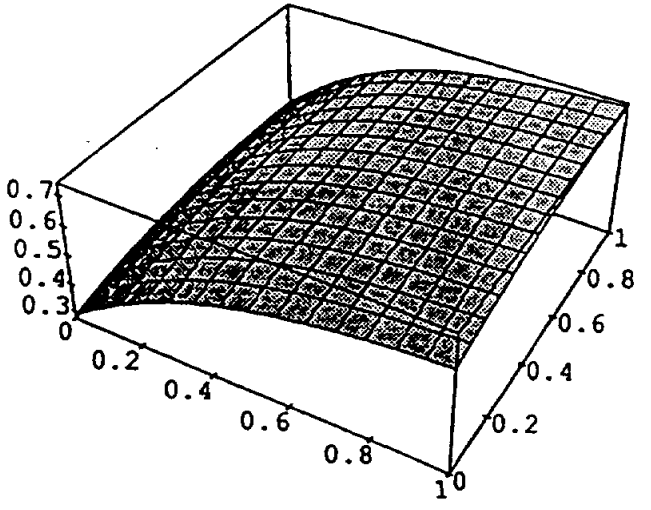
Figure 5 : Surface plots of the four controlled domains under consideration



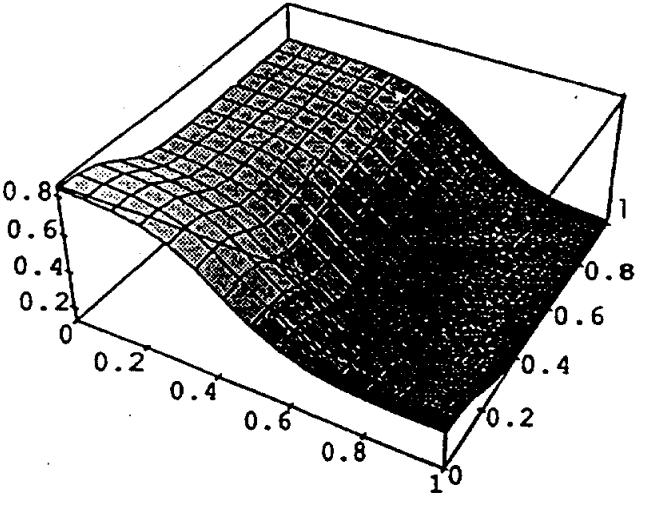
Linear Domain



Quadratic Domain



Logarithmic Domain



Trigonometric Domain

Figure 6 : Surface plots learnt by neural network in the controlled experiment

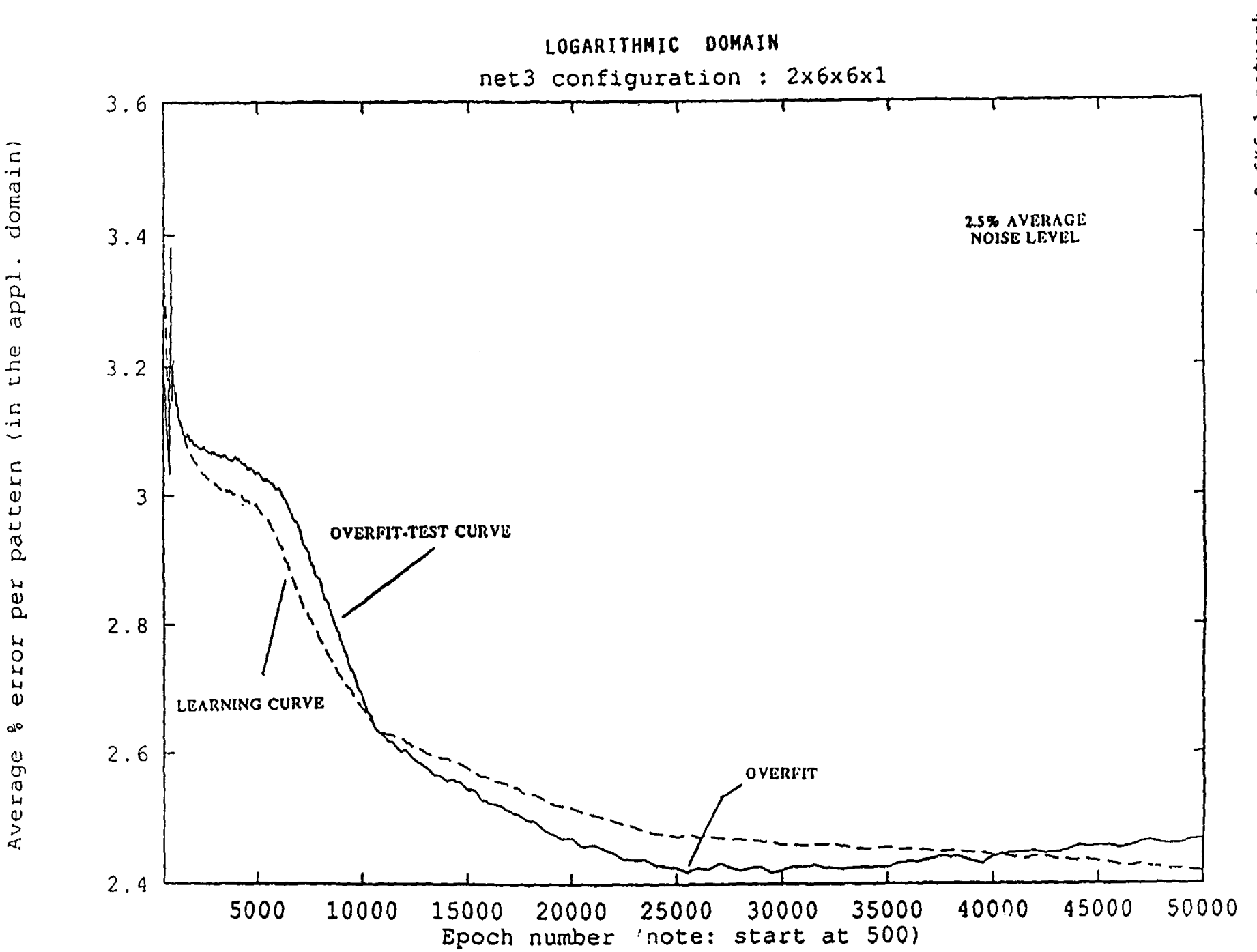


Figure 7: Learning and overfit test curves for the 2x6X6x1 network

Function	Valid Input Ranges	Type of Application Domain
$f_1(x,y) = x + 2y + 3$	$10 \leq x,y \leq 150$	linear
$f_2(x,y) = 0.5x^2 + 0.2y^2 + 9.0$	$10 \leq x,y \leq 150$	quadratic
$f_3(x,y) = 3\log(x) + \log(y) + 2.0$	$10 \leq x,y \leq 150$	logarithmic
$f_4(x,y) = 2\sin(x) + \cos(y) + 4.0$	$40 \leq x,y \leq 320$	trigonometric

Table 1: Functions used for data synthesis

Domain Form (Learning)	2X2X1 Network	2X2X2X1 Network	2X6X6X1 Network	Regression Oracle
Linear NAMAPE values				
Best	0.16	0.05	0.11	0.00
Worst	1.41	1.49	0.82	0.12
Average	0.60	0.63	0.37	0.04
Quadratic NAMAPE values				
Best	0.75	1.39	0.64	0.00
Worst	12.27	3.89	4.09	0.48
Average	5.87	2.22	1.97	0.19
Logarithmic NAMAPE values				
Best	0.19	-0.01	-0.01	0.00
Worst	1.48	1.14	0.30	0.06
Average	0.63	0.36	-0.01	0.02
Trigonometric NAMAPE values				
Best	8.55	3.67	0.12	0.00
Worst	13.81	10.67	2.15	0.02
Average	11.26	7.33	0.94	-0.01

Table 2: NAMAPE values for the learning phase

Domain Form (Testing)	2X2X1 Network	2X2X2X1 Network	2X6X6X1 Network	Regression Oracle
Linear NAMAPE values				
Best	-0.03	0.11	0.00	0.00
Worst	0.84	0.91	-0.55	-0.52
Average	0.12	0.12	-0.06	-0.22
Quadratic NAMAPE values				
Best	0.00	0.08	-0.21	0.00
Worst	5.32	2.15	2.05	-0.63
Average	2.25	0.84	0.75	-0.26
Logarithmic NAMAPE values				
Best	-0.01	-0.07	0.00	0.00
Worst	1.34	1.05	-0.41	-0.47
Average	0.53	0.25	-0.06	-0.20
Trigonometric NAMAPE values				
Best	9.62	4.10	0.26	0.00
Worst	14.75	11.41	1.78	-0.52
Average	12.12	7.32	1.01	-0.22

Table 3: NAMAPE values for the testing phase

Var #	Definition	Var #	Definition
1	Liability/(Cash + Assets)	2	(Total funded debt)/(net property)
3	Sales/Net worth	4	Profit/Sales
5	Financial strength of company as given in the Valueline Index	6	Number of times available earnings cover fixed charges
7	Past 5 year revenue growth rate	8	Next 5 year projected revenue growth rate
9	Working capital/Sales	10	Subjective prospects of company (from Valueline)

Table 4: Financial Variables Used to Predict Bond Ratings

Bond Rating	Numerical Value	Bond Rating	Numerical Value
AAA	0.95	BBB+, BBB, BBB-	0.5
AA+, AA, AA-	0.9	BB+, BB, BB-	0.35
A+, A, A-	0.7	B+, B, B-	0.25

Table 5: Conversion of Bond Ratings to Numerical Values

Actual	Prediction	Description
Accept	Accept	S&P rating is AA and rating of model is also AA
Accept	Reject	S&P rating is AA and rating of model is not AA
Reject	Accept	S&P rating is not AA and rating of model is AA
Reject	Reject	S&P rating is not AA and rating of model is also not AA

Table 6: Possible Classification of Responses

Number of Variables	Neural Network		Regression
	2-layered	3-layered	
6	80% tot_sq_err = 0.236	80% tot_sq_err = 0.175	63.33% tot_sq_err = 1.107
10	80% tot_sq_err = 0.224	92.4% tot_sq_err = 0.054	66.7% tot_sq_err = 0.924

Table 7: Results from the learning Phase

Number of Variables	Neural Network		Regression
	2-layered	3-layered	
6	82.4% tot_sq_err = 0.198	76.5% tot_sq_err = 0.194	64.7% tot_sq_err = 1.528
10	88.3% tot_sq_err = 0.164	82.4% tot_sq_err = 0.228	64.7% tot_sq_err = 1.643

Table 8: Results from the testing Phase

Testing Phase 10 Variables	Number of ratings by which the output is in error 1 rating	2 ratings	3 ratings	4 ratings
2 Layer Neural Network	5	0	0	0
3 Layer Neural Network	6	1	0	0
Regression	3	8	1	1

Table 9: Analysis of errors in the output ratings (10 variables)

Testing Phase 6 Variables	Number of ratings by which the output is in error 1 rating	2 ratings	3 ratings	4 ratings
2 Layer Neural Network	5	1	0	0
3 Layer Neural Network	5	1	0	0
Regression	3	9	0	1

Table 10: Analysis of errors in the output ratings (6 variables)

Data Sample	% Accuracy Logit	% Accuracy Normit	% Accuracy Log-Log
Training	63.3	50%	-
Training + Testing	61.7	57.4	51

Table 11: Logit regression model results

Network Characteristics					Average NOCS	Average Absolute	Average Squared
Network name	# of hidden nodes	Alpha value	I/O connection	Cycles run	Predicted	Difference	Difference
N3A5NT1	3	0.5	No	10,400	407.91	87.53	13806
N3A1NT1	3	1	No	10,400	436.31	91.97	13623
N3A2NT1	3	2	No	10,400	359.08	104.41	20432
N3A4NT1	3	4	No	10,400	331.73	122.44	25528
N6A5NT1	6	0.5	No	10,400	431.31	90.00	13556
N6A1NT1	6	1	No	10,400	440.93	90.13	14113
N6A2NT1	6	2	No	10,400	394.40	97.49	17218
N6A4NT1	6	4	No	10,400	440.78	87.79	12661
N3A5YT1	3	0.5	Yes	10,400	432.20	87.76	13149
N3A1YT1	3	1	Yes	10,400	435.97	91.34	13532
N3A2YT1	3	2	Yes	10,400	435.03	90.05	13520
N3A4YT1	3	4	Yes	10,400	433.36	89.15	12999
N3A5YT5	3	0.5	Yes	52,000	435.27	89.60	13022
N9A1YT1	9	1	Yes	10,400	438.09	93.76	14069
Regression estimate					424.66	85.09	12748
Actual NOCS '88					447.64		

Table 12: Results of prediction of product purchase behavior

List of Table and Figure Captions