Neural Comput & Applic (1999)8:226–234

Neural
Computing
& Applications

Application of MLP Networks to Bond Rating and House Pricing

H. Daniels[1,2] and B. Kamp[1,3]

[1]Department of Economics, Tilburg University, Tilburg, The Netherlands; [2]Rotterdam School of Management, Erasmus University, Rotterdam, The Netherlands; [3]BDO CampsObers, Tilburg, The Netherlands

Feedforward neural networks are receiving growing attention as a data modelling tool in economic classification problems. It is well known that controlling the design of a neural network can be cumbersome. Inaccuracies may lead to numerous problems in the application, such as higher errors due to local optima, overfitting and ill-conditioning of the network, especially when the number of observations is small. In this paper we provide a method to overcome these difficulties by regulating the flexibility of the network, and by rendering measures for validating the final network. In particular, a method is proposed to equilibrate the number of hidden neurons based on 5-fold cross-validation. In the validation process, the performance of the neural network is compared with a linear model using 5-fold cross-validation. In both case studies, the degree of monotonicity of the output of the neural network, with respect to each input variable, is calculated by numerical differentiation. The outcomes of this analysis are compared to what is expected from economic theory. Furthermore, a special class of monotonic neural networks and a corresponding training algorithm are developed. It is shown in the second case study that networks in this class have less tendency to overfitting than ordinary neural networks. The methods are illustrated in two case studies: predicting the price of housing in the Dutch city of Den Bosch; and the classification of bond ratings.

Keywords: Classification; Error estimation; Finance; Monotonic neural networks

Correspondence and offprint requests to: H. Daniels, Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands.

1. Introduction

There is a growing interest in neural networks as a tool for data analysis. To a certain extent, the popularity of neural networks compared to other statistical methods may be caused by the failure of statisticians to communicate their methodologies and algorithms to non-statisticians. The vast amount of accumulated statistical knowledge erects a barrier for consumers of their methods. Neural networks, on the other hand, are in an embryonic phase, which means that the accumulated knowledge is relatively small. The language used within the neural network community is another factor which may explain the success of neural networks. However, the core problems of data analysis do not change when the techniques they are approached with are changed. Therefore, difficulties statisticians have run into will also affect neural network practitioners. The specification of a neural network involves not only a selection of the inputs, but also the selection of the various components of a network, such as the type of network to use, the squashing function, which error criterion to use, which learning algorithm, the number of hidden layers, and how many hidden units per layer there should be. Once these network components have been specified, the neural network is presented with the data.

In many classification and prediction problems in economics, data sets are small and special techniques are needed to reliably estimate the prediction error as well as to avoid overfitting. In particular, when neural networks are applied to time series prediction, where some of the input series are non-stationary, overfitting is very likely [1], but also in simpler classification tasks when no precautions are taken, overfitting may occur [2]. Another problem, much

ignored, is the landing in local optima of the error function during the training process [3]. These practical issues are important factors that determine the success of neural network applications, therefore they require careful investigation. The impact of particular choices of the network components is largest in small sample problems, where statistical theory is of little help.

The aim of this paper is to lay down the choices concerning the different aspects of neural network modelling mentioned above, and to establish a general network construction procedure. In particular, we discuss how techniques such as cross-validation and monotonicity analysis can be effectively combined to optimise the neural network. There are two main approaches to control the complexity and flexibility of the neural network: model selection and regularisation. Model selection for neural networks involves choosing the number of hidden units, the connections and the inputs. By regularisation the neural network solution is smoothed by stop training or (in this paper) by restricting the weights to represent monotonic relations only.

The simplest approach is to stop training after a predetermined number of 'epochs', which are complete presentations of the complete training set. It is obvious that this approach can only be suboptimal. A more realistic approach is to use a test set of data (set B) to indicate the error on 'unseen' cases; these data may not be used during training. When the error on the test set starts to increase, training is terminated. To measure the degree of generalisation, a third independent set (C, the validation set) is necessary to estimate the out-of-sample performance of the network. Sets B and C are not used in training the network, which incurs a loss of costly information for problems with limited data. In practice, C should be taken as large as possible to ensure a low variance prediction error estimator, but on the other hand, as many observations as possible should be used to reliably estimate the network weights.

In many economic problems we expect a monotonic relation (but not necessarily linear) of the output with respect to some or all of the inputs [4–6]. In this paper, we introduce a class of monotonic neural networks that can be applied successfully to problems where the input/output relation is to a large extent monotonic. To compute the degree of monotonicity of the output with respect to each of the input variables, we derive an index between 0 and 1. The index is computed by numerical differentiation of the output of the original neural network. If the index is close to 1 for all (or the most important) input variables, monotonic neural networks will probably outperform ordinary neural networks. The restriction to a class of monotonic networks is similar to adding bias and suppresses (spurious) oscillations. If the problem is essentially monotonic, this will lead to better generalisation properties, as illustrated in the second case study below.

The remainder of this paper is organised as follows. The first part, Sections 2.1–2.5, deals with the Bond Rating case. In Sections 2.1–2.4 we describe a neural network model to classify companies into seven bond rating classes based on their financial characteristics. The performance of the neural network model is compared with a linear model using 5-fold cross-validation in Section 2.4. The monotonicity index is defined in Section 2.5. The second part of the paper, Sections 3.1–3.4, deals with the second case study: modelling house prices in a medium-sized Dutch town. The hedonic house price model is derived in Section 3.1. In Section 3.2–3.4 the results of the simulation studies of ordinary neural networks and monotonic neural networks are presented and evaluated. In the appendix, the training algorithm for monotonic neural networks is briefly outlined.

2. Bond Rating Classification

2.1. Description of the Case Study

Bond ratings are subjective opinions on the ability to service interest and debt by economic entities such as industrial and financial companies, or municipals, and public utilities. Bond ratings are published by two major bond rating agencies, Moody's and Standard & Poor's, in the form of a letter code, ranging from AAA–for excellent financial strength–to D for entities in default. Bond ratings are based on extensive financial analysis by the bond rating agencies. The exact determinants of a bond rating, however, are unknown, since the interpretation of financial information relies heavily on professional judgement.

During the last 30 years, several attempts have been made to model corporate (industrial) bond ratings. The methods employed include linear regression, multiple discriminant analysis–linear and quadratic–and neural networks. Linear regression models were proposed by Horrigan [7], Pogue and Soldofsky [8] and West [9]. Pinches and Mingo [10], Peavy [11] and Belkaoui [12] employed discriminant analysis. Moody [13], Dutta and Shekhar [14] and Kim et al. [15] recommended neural networks to model bond ratings. These studies were

directed to general corporate bond ratings. In other studies [16], models of bond rating classification within specific industries are described.

2.2. Formulation of the Empirical Model

The aim of the model as described here is to classify companies into the distinctive bond rating classes, based on their financial characteristics. Publications of bond rating agencies offer some insight into the relevant factors that determine bond ratings. Bond rating analysis recognises the following areas of attention [17]:

- Profitability;
- Liquidity;
- Asset protection;
- Indenture provisions;
- Quality of management.

Bond rating models use independent variables, often calculated as ratios, which are predominantly derived from public financial statements. However, not all of the above-mentioned areas can be covered by financial statement figures. Aspects like quality of management, market positions and asset protection can only be captured to a limited extent. Also for indenture provisions, like subordination status, financial ratios are not applicable. Most of the ratios that are used in bond rating models can also be found in the literature on general financial statement analysis [18].

2.3. Empirical Study Set Up

In the study set up, we follow the methodology described by Moody [13]. Two alternative approaches to model the classification of bond ratings are examined: a neural network with one hidden layer; and a linear model. The neural network weights are determined by error-backpropagation, the coefficients in the linear model are estimated by OLS. Comparing neural networks with multiple discriminant analysis would require a multiple output architecture of neural networks. Although neural networks are suitable for more than one output, we restrict our study to a one output architecture. Theoretically, these architectures are equivalent, since every function can be approximated arbitrarily well by a neural network with one hidden layer.

From the Standard & Poor's Bond Guide (April 1994), 256 companies were selected. The bond ratings of these companies range from AAA to D. The ratings are not homogeneously distributed. The largest classes are A, BBB and B. Only very few selected companies have ratings lower than CCC. Therefore, we decided to remove all ratings below CCC. As in other studies, the + and − signs were omitted (for example, AA+, AA and AA− are all considered as AA). The bond ratings are quantified by assigning from 1 to AAA to 7 to CCC.

From the S&P Bond Guide, several financial figures have been obtained. From Datastream additional financial figures and ratios relating to leverage, coverage, liquidity, profitability and size were downloaded. These figures have been restated to five year averages and trend indicators, resulting in 45 explanatory variables. For each variable the linear correlation with the quantified bond rating was calculated. Occasionally, a linear correlation test may not find possible non-linear correlations between input and output, although this will occur rarely in practice. The problem with measuring non-linear correlation is that it is highly arbitrary, since the nature of non-linearity between the variables is unknown.

It was found that neural networks trained on all 45 variables resulted in lower quality models than models based on only a carefully selected subset of these variables. Eight variables with the highest correlation are presented in Table 1. The variables represent the level and stability of profits and cash flow, liquidity and subordination status (indenture). Cross-correlations between all attributes are acceptable low (<0.60).

Several neural network architectures were evaluated, varying the number of hidden neurons (2, 4, 8, 12, 20), the learning rate (0.1, 0.01), the momentum term (0.8, 0.1, 0.01), the type of activation function in the output layer (sigmoid, linear) and batch or online weight update. The performance of the neural network is measured by two indicators, namely the Mean Squared Error (MSE) and Percentage of Correct Classification (PCC). The MSE is defined by

$$\mathrm{MSE}(C) = \frac{1}{|C|} \sum_{p \varepsilon C} (t^p - y^p)^2 \qquad (1)$$

Table 1. Definition of the model variables.

Symbol	Definition
D/C	Debt to capital ratio
CF/D	5 years average cash flow to debt ratio
CF	5 years average cash flows (in 100 millions)
Cov	3 years average interest coverage ratio
Vol/Cov	3 years volatility of interest coverage
Sr	Senior status [0,1]
SrSub	Limited senior status [0,1]
Sub	Subordinated status [0,1]

Here C denotes the testset, t^p is the target pattern and y^p the actual outcome of the neural network.

To compute the Percentage of Correct Classification (PCC), the prediction of the neural network was rounded to the nearest discrete value (4/5 rounding). This value was compared to the actual class value.

The overall performance of the neural network is measured as the average of MSE or PCC on test sets as used in the cross-validation procedure. They are denoted by MSE_{cv} and PCC_{cv}, respectively. For this purpose, the total set of patterns was divided into five mutually exclusive subsets, each containing (about) 50 patterns. The relative distribution of the subsequent classes are approximately equal for all subsets. For each neural network architecture, five different training runs were executed, each with another set serving as a test set. Training was accomplished on the remaining four subsets. During the training process, the performance was measured on both the training set and the test set. Training was stopped as soon as the lowest MSE on the test set was reached.

2.4. Results

The final results show that the lowest average error occurs in a neural network with eight hidden neurons. The MSE varies between 0.290 and 0.421, and PCC varies between 60% and 76% (see Table 2). Note that MSE and PCC will not correlate perfectly. This can be explained by the fact that the PCC only refers to errors smaller than 0.5. Errors larger than 0.5 receive equal weights, since both large and medium errors are considered to be false classifications, regardless of the magnitude of the error. Nevertheless, in former studies the PCC is regarded as the main performance indicator. However, the PCC is not suitable as an error function in backpropagation neural networks, since such a function does not have a continuous first derivative.

An important result is that the architecture can be kept fairly simple. A neural network with eight hidden neurons, sigmoid squashing functions, a learning rate of 0.1 and a momentum of 0.1 has an MSE_{cv} of 0.334 and a PCC_{cv} of 67%. The results on the training set do not differ from the results on the test set, indicating that overfitting in this case does not occur. When the number of instances in the training set was limited to 50, the MSE on the training set dropped to 0.111 and the PCC consequently rose to 90%. The performance on the holdout sample, however, was very bad (MSE: 1.455, PCC: 32%), clearly showing the effects of overfitting.

To evaluate the results of our model with respect to earlier studies, we also employed linear regression models. These models are based on the same data set as used for the neural network model. Also for this regression analysis, the 5-fold cross-validation method was implemented, resulting in five equations, that were tested on five holdout samples. The results are presented in Table 2. The signs of the coefficients correspond to what is expected on the basis of economic plausibility. The t-statistics range between 3.13 and 9.58.

The results clearly show the advantage of applying cross-validation. If training was performed using a single holdout sample, the values of the PCC would fluctuate between 60% and 76%, depending on whether subset 1, 2 or 3 was chosen as the holdout sample.

2.5. Monotonicity Index

In this section, we introduce a measure for the degree of monotonicity of the neural network with respect to each input variable. For every explanatory variable we compute the partial derivative df/dx_i at each data point x_p. Here f denotes the neural network solution. The *degree of monotonicity* in x_i is defined as

Table 2. PCC calculated using 5-fold cross validation.

CV set	Data set		Holdout sample	
	neural network (%)	linear (%)	neural network (%)	linear (%)
1	47	57	60	48
2	51	55	60	52
3	51	54	76	57
4	48	53	69	61
5	50	56	72	48
	67	55	67	53

$$\text{mon}(x_i) = \frac{1}{n}\left|\sum_{p=1}^{n} I^{+}\left(\frac{\partial f}{\partial_{xi}}(x_p)\right) - I^{-}\left(\frac{\partial f}{\partial_{xi}}(x_p)\right)\right| \qquad (2)$$

where $I^{+}(z) = 1$ if $z > 0$ and $I^{+}(z) = 0$ if $z \leq 0$ and $I^{-}(z) = 1$ if $z \leq 0$ and $I^{-}(z) = 0$ if $z > 0$. n is the number of observations, and x_p is the pth observation (vector). Note that $0 \leq \text{mon}(x) \leq 1$. A value of this index close to zero indicates a non-monotonic relationship, a value close to 1 indicates a montonic relationship. The value of sign indicates whether the relation of f with respect to x is increasing or decreasing. The results for the bond rating model are presented in Table 3.

Some interesting conclusions can be drawn from Table 3. The sign of the relationship between D/C, CF/D and the rating is as expected. However, the relative non-montone relationship of the rating with respect to D/C and CF/D cannot easily be explained by theories on financial statement analysis. The monotonicity indices of Cov and Vol-Cov correspond to what is expected from general theories on financial statement analysis. The more interest that is covered by excess earnings (Cov), the lower is the risk of insufficient available cash to service debt. Secondly, bond rating analysts favour stable coverage ratios (Vol-Cov), so high volatility results in a higher rating class (e.g. a higher estimated default risk). The monotonicity of the rating with respect to CF is more difficult to explain. There are two counteracting effects that occur as a consequence of variations in CF. In general, more cash flows results in more collateral value to support the repayment of debt. On the other hand, large cash flows are more likely to be found for large firms, which often also have large amounts of debt. Firm size is not considered to be an important rating determinant, therefore the sign of CF cannot be predicted *a priori*.

The observed non-monotonicity of the problem does not support the use of monotonic networks. By using monotonic networks, structural non-monotonic relations cannot be modelled, which will result in lower performance of the model. Therefore, no monotonic neural network model was developed. In the Den Bosch case, we will see that *a priori* assumed monotonicity and observed monotonicity of the model does support the use of monotonic neural networks.

Table 3. Monotonicity indices of the bond rating model.

Variable	Mon(x_j)	Sign
D/C	0.51	+
CF/D	0.52	–
CF	1.00	–
Cov	0.96	–
Vol/Cov	0.77	+
Sr	0.79	+
SrSub	0.73	+
Sub	0.71	+

3. Den Bosch House Price Estimation

3.1. Description of the Case Study

In this section we want to compare the performance of an ordinary neural network model and a monotonic neural network to estimate house prices in Den Bosch, a city with approximately 110,000 inhabitants in the Netherlands.

The basic principle of the hedonic approach to economics is that each consumer good is regarded as a bundle of characteristics for which an implicit valuation exists [19]. Harrison and Rubinfeld [20] regard each house as a bundle of characteristics, and the price of each house as reflective of the value of its characteristics. So the price of a house is estimated by the equation

$$H^p = g(x_1,...,x_q) \qquad (3)$$

where each x_i denotes a characteristic of the house. In this case, there is no theoretical knowledge of what the function g should look like. We therefore employ a data driven approach to specify Eq. (3). In a similar study of Harrison and Rubinfeld [20], their interest was to estimate the impact of air pollution on the price of houses. However, the model may serve many different purposes. For example, in many countries the local tax authorities require house values to calculate the amount of property tax due. The data are of cross-sectional type, i.e. the attributes are measured across various suburbs of Den Bosch at a particular time point.

3.2. Empirical Study Set Up

The explanatory variables were selected on the basis of interviews with experts of local house broker offices, and advertisements offering real estate in local magazines. The most important variables that came out of this study are listed in Table 4.

The data set consists of 118 instances, and was collected from several local house broker offices. Cross-correlations between all attributes are acceptably low. The correlation matrix suggests, for example, that the volume of the house (VOL) and the surface (SURF) are the most important determi-

Table 4. Definition of model variables.

Symbol	Definition
DISTR	type of district, four categories ranked from bad to good
SURF	total area including garden
RM	number of bedrooms
TYPE	1. apartment 2. row house 3. corner house 4. semidetached house 5. detached house 6. villa
VOL	volume of the house
GARD	type of garden, four categories ranked from bad to good
GARG	1. no garage 2. normal garage 3. large garage

nants of the housing value. The direction of influence corresponds to common sense: more volume and surface will in general result in a higher housing value.

3.3. Results

In the simulation study, we applied ordinary neural networks with 5, 10, 15 and 20 neurons in the hidden layer. The lowest average error occurs in a neural network with five hidden neurons. R^2 varies between 0.8089 and 0.9288 on the holdout sample (see Table 5).

The performance on the test set is comparable to the performance on the training set, indicating that the models have reasonable generalisation power. However, this does not imply that the internal representation of the neural network model is consistent with the presupposed relationships within the hedonic problem. The degree of monotonicity of each of the explanatory variables is presented in Table 6.

Table 6 shows that the network output behaves monotonically with respect to almost all variables.

Table 5. R^2 calculated using 5-fold cross-validation.

CV set	Data set	Holdout sample
1	0.9010	0.9031
2	0.7894	0.8089
3	0.9267	0.9288
4	0.8404	0.8410
5	0.7607	0.8057
	0.8430	0.8575

Table 6. Monotonicity indices of the Den Bosch house price model.

Variable	mon(x_i)	Sign
DISTR	1.00	+
SURF	1.00	+
RM	1.00	+
TYPE	1.00	+
VOL	1.00	+
GARD	1.00	+
GARG	0.56	+

With respect to CARG (garage), the observed non-monotonicity cannot be explained by plausible economic arguments. Sensitivity analysis with respect to individual variables suggests that the importance of CARG is relatively low. The non-monotonicity in the model may be the result of noise. To take advantage of the monotonic nature of the problem, a monotonic neural network was trained using the algorithm described in Section 1. Using the same number of hidden neurons (five), the monotonic neural network reached a performance of $R^2 = 0.8679$ on the training set and $R^2 = 0.8815$ on the test set. Thus, the monotonic neural network slightly outperforms the ordinary network.

To study the effect of monotonic neural networks on overfitting, we compared the divergence in performance on the training set and the test for ordinary neural networks and monotonic neural networks when varying the number of hidden neurons. For this purpose, we trained both types of neural networks using 5, 10, 15 and 20 hidden neurons. The

standard deviations between R^2 on the training set and R^2 on the test set are presented in Table 7.

Table 7 shows that monotonic neural networks are less sensitive to overfitting when the number of hidden neurons is increased than ordinary (non-monotonic) neural networks. For monotonic neural networks, the performance on the test set remains close to the performance on the training set, even if the number of hidden neurons is increased to 20. For non-monotonic neural networks there is a clear effect of overfitting.

4. Software

In our experiments we used the neural network simulation software Neuroshell2, Brainmaker and the Matlab neural network toolbox. The implementation of monotonic neural networks and the training algorithm was done in Matlab. Matlab and Excel provided an interactive computing environment for graphical data analysis, statistics and simple calculations. All neural networks used are feedforward neural networks with one hidden layer, and one output unit; the activation functions of the hidden units sigmoid, the activation function of the output sigmoid or linear. Estimation of the weights is done by backpropagation. All software runs on a Windows PC-platform.

5. Conclusions

We applied standard statistical techniques and neural networks to two economic classification problems. In particular, we addressed different aspects of neural network modelling that are essential to obtain reliable predictions. The flexibility and the degree of non-linearity of the network are optimised by calibrating the number of hidden neurons.

In the first case, a model was developed to classify companies in different bond rating classes, using a linear and neural network approach. It was found that the neural network clearly outperformed the linear model on the percentage of correct classifications (67% versus 53%). These results are in line with previous observations of other authors. Sensitivity analysis revealed that the neural network classifier clearly shows non-linear behaviour in the data set. It was shown that derivatives of the classifier with respect to firm's leverage and cash flow to debt ratio have different signs in the domain of the data set. This non-monotonic behaviour prohibits a meaningful application of monotonic neural networks.

In the second case study, house prices in a medium sized Dutch town are estimated by neural networks and monotonic neural networks. The output variable behaves monotonically with respect to all the important input variables. There is strong evidence that the non-monotonic behaviour with respect to less significant inputs is due to noise in the data. Experiments with increasing numbers of hidden neurons show that monotonic neural networks are far less sensitive to overfitting compared to ordinary non-monotonic neural networks. It was shown that the goodness of fit of monotonic networks is better compared to non-monotonic networks. It was also shown that overfitting does not occur using monotonic networks, even with a larger number of neurons.

In both cases it was shown that the response function behaves monotonically with respect to several input variables. This is not surprising, since one expects that most classification problems in economics and accounting possess monotonicity properties. It is therefore natural to impose monotonicity constraints on neural network architecture, because it reflects the generic properties of the underlying domain. This approach is particularly useful if the neural network tends to overfit the data by 'spurious' oscillations near the classification boundary. This behaviour typically occurs if the number of hidden neurons is larger. These spurious

Table 7. Variance of R^2 of ordinary and monotonic neural networks.

Number of hidden neurons	Variance of R^2 between training and test set	
	Ordinary NN	Monotonic NN
5	0.01	0.01
10	0.06	0.03
15	0.12	0.04
20	0.15	0.03

oscillations can be suppressed by the addition of monotonicity constraints, using a special class of inherently monotonic neural networks. In cases where the monotonicity index of the most important explanatory variables is close to one, it is likely that monotonic neural networks have better out-of-sample performance. Currently, we are studying mixtures of neural networks that are monotonic with respect to a subset of the explanatory variables. These can be applied to problems that are partially monotonic, as in the first case study.

Acknowledgements. Part of the research presented in this paper was done within an EC-funded network of the SPES programme, contract number 0065, which the authors gratefully acknowledge. The SPES (Stimulation Plan for Economic Science) project, entitled 'Artificial Intelligence approaches to modelling in Economics', is a joint research project with participants from Heriot-Watt University (United Kingdom), Tilburg University (The Netherlands), Politecnico Milano (Italy), ABN/AMRO Bank (The Netherlands), and Digital Equipment Europe (France).

References

1. Verkooijen WJH. Neural Networks in Economic Modelling, An Empirical Study. CentER dissertation, 1996
2. Geman S, Bienenstock E, Doursat R. Neural networks and the bias/variance dilemma. Neural Computation 1981; 4: 817–823
3. Ripley BD. Flexible non-linear approaches to classification. In From Statistics to Neural Networks; Theory and Pattern Recognition Applications. Springer-Verlag, 1993
4. Farley AM, Lin KP. Qualitative reasoning in economics. Journal of Economic Dynamics and Control 1990; 14: 465–490
5. Berndsen R, Daniels H. Causal reasoning in economic systems. Journal of Economics and Control 1994; 18: 251–271
6. Daniels HAM, Kamp B, Verkooijen WJ. Modelling non-linearity in economic classification with neural networks. International Journal of Intelligent Systems in Accounting, Finance and Management 1997; 3: 287–308
7. Horrigan JO. The determination of long-term credit standing with financial ratios. Journal of Accounting Research 1966 4 (supplement): 44–62
8. Pogue TF, Soldofsky RM. What's in a bond rating? Journal of Financial and Quantitative Analysis 1969; 4: 201–228
9. West RR. An alternative approach to predicting corporate bond ratings. Journal of Accounting Research 1970; 7: 118–125
10. Pinches GE, Mingo KA. A multivariate analysis of industrial bond ratings. The Journal of Finance 1973; 28: 1–17
11. Peavy JW. Long run implications of industrial bond ratings as risk surrogates. Journal of Bank Research 1982; 34: 331–341
12. Belkaoui A. Industrial bond ratings: A New Look. Financial Management 1980; 9: 44–50
13. Moody J. Architecture selection strategies for neural networks: application to corporate bond rating prediction. Neural Networks in the Capital Market. Wiley, New York, 1994
14. Dutta S, Shekhar S. Bond rating: A non-conservative application of neural networks. Proceedings of the IEEE Conference of San Diego, 1988
15. Kim JW, Weistroffer HR, Redmond RT. Expert systems for bond rating: a comparative analysis of statistical, rule-based and neural network systems. Expert Systems 1993; 10: 167–188
16. Altman E, Katz S. Statistical bond rating classification using financial and accounting data. Proceedings of the Conference on Topical Research in Accounting, 1976
17. Hawkins DF. Rating industrial bonds. Financial Executives Research Foundation, Morristown, NJ, 1983
18. Lev B. Financial Statement Analysis: A new approach. Prentice Hall, Englewood Cliffs, 1974
19. Janssen J. De prijsvorming van bestaande koopwoningen. PhD thesis, Catholic University Nijmegen, 1992
20. Harrison O, Rubinfeld D. Hedonic prices and the demand for clean air. Journal of Environmental Economics and Management 1978; 53: 81–102
21. Archer NP, Wang S. Application of the back propagation neural network algorithm with monotonicity conditions for two-group classification problems. Decision Sciences 1993; 24: 60–75
22. Wang S. A neural network method of density estimation for univariate unimodal data. Neural Computation & Applications 1994; 2: 160–167

Appendix on Monotonic Neural Networks

The implementation of certain monotonicity constraints in neural networks and learning algorithms has been studied in Archer and Wang [21] and Wang [22]. In this paper, we apply a different class of monotonic neural networks. This class is obtained by considering multilayer neural networks with positive (non-negative) weights. It can be shown that the elements of this class can approximate any monotonic increasing function f of n-variables on compact subsets of R^n.

Note that without loss of generality, we may assume that f is increasing with respect to all variables. This can be achieved by simple linear transformations of the inputs.

The maximum number of layers needed is equal to the number of inputs. The proof of this result is beyond the scope of this paper, and will be given elsewhere.

In the many simulation studies performed on artificially generated data-sets, it turned out that in practice less than n-layers were needed to get good fit. In all cases studies, 1 or 2 hidden layers were sufficient.

The training algorithm for monotonic neural networks that we have developed is a modification of the standard backpropagation algorithm. We have studied two ways of enforcing positive weights. The first one is to set all negative weights equal to zero in each training step. This algorithm is sketched below. In the second method, we add a bias term to the error function of the neural network such that negative weights are penalised. During the training process the weight of the penalty term is increased gradually, until in the final network all weights are non-negative.

Sketch of Backprop Algorithm with Non-negative Weights

1. *Initialisation*: start with a reasonable network configuration and set all weights and threshold (bias) levels to small, uniformly distributed random numbers.
2. *Presentation of training patterns*: compute for each data pattern the output value using the current weight structure.
3. *Weights update*: using the backprop algorithm, determine the weight adjustments.
4. *Monotonicity correction*: change all negative weight values to zero.
5. Repeat steps 3 and 4 for all hidden layers.
6. Repeat steps 1 to 5 until the error is acceptable.