Artificial Neural Networks in Accounting and Finance: Modeling Issues

James R. Coakley* and Carol E. Brown
Oregon State University, USA

ABSTRACT **This article reviews the literature on artificial neural networks (ANNs) applied to accounting and finance problems and summarizes the 'suggestions' from this literature. The first section reviews the basic foundation of ANNs to provide a common basis for further elaboration and suggests criteria that should be used to determine whether the use of an ANN is appropriate. The second section of the paper discusses development of ANN models including: selection of the learning algorithm, choice of the error and transfer functions, specification of the architecture, preparation of the data to match the architecture, and training of the network The final section presents some general guidelines and a brief summary of research progress and open research questions. Copyright © 2000 John Wiley & Sons, Ltd.**

INTRODUCTION

Artificial neural networks (ANNs) have emerged as a powerful statistical modeling technique. ANNs detect the underlying functional relationships within a set of data and perform such tasks as pattern recognition, classification, evaluation, modeling, prediction and control (Lawrence and Andriola, 1992). Several systems based on ANNs and in widespread commercial use for accounting and financial tasks are described by Brown *et al.* (1995). These include:

- *FALCON*, used by six of the ten largest credit card companies to screen transactions for potential fraud.
- *Inspector*, used by Chemical Bank to screen foreign currency transactions.
- Several ANNs used to assist in managing investments by making predictions about debt and equity securities as well as derivative instruments.
- Several ANNs used for credit granting, including GMAC's *Credit Advisor* that grants instant credit for automobile loans.
- *AREAS*, used for residential property valuation.

Developers of commercially used ANNs generally consider the inner workings of their systems to be proprietary information that gives them a competitive advantage. Thus, they are reluctant to disclose information about those systems. The fact that a system is in commercial use provides some insight into the kind of tasks and domains for which ANNs are likely to be useful. Unfortunately, the lack of details on internal system structure precludes using the published information about those systems to help answer open research questions about the best structure for ANNs in accounting and finance.

* Correspondence to: James R. Coakley Business, Department of Accounting, Finance and Information Management, Boxell Hull 200, Corvallis, OR 97331-2603, USA. E-mail: coakley@bns.orst.edu

Accepted 7 July 1999

In the past few years, many researchers have used ANNs to analyze traditional classification and prediction problems in accounting and finance. The research that has compared the ANNs with traditional statistical methods produced mixed results. In spite of all the research, we still do not have any general guidelines to tell us when to use ANNs or which ANN architecture to employ.

Numerous articles have appeared recently that surveyed journal articles on ANNs applied to business situations. Wong *et al*. (1997) surveyed 203 articles from 1988 through 1995. They classified the articles by year of publication, application area (accounting, finance, etc.), journal, various decision characteristics (problem domain, decision process phase, level of management, level of task interdependence), means of development, integration with other technologies, comparative technique (discriminant analysis, regression analysis, logit and ID3 were the most common techniques), and major contribution. The survey included five articles in accounting and auditing, and 54 articles in finance. Given the focus toward decision process, and the limited coverage of accounting and auditing, the paper does not provide much insight into accounting and finance issues.

O'Leary (1998) analyzed 15 articles that applied ANNs to predict corporate failure or bankruptcy. For each study, he provided information about the data, the ANN model and software (means of development), the structure of the ANN (input, hidden and output layers) training and testing, and the alternative parametric methods used as a benchmark. He then analyzed the overall ability of the ANN models to perform the prediction task.

Zhang *et al*. (1998) surveyed 21 articles that addressed modeling issues when ANNs are applied for forecasting, and an additional 11 studies that compared the relative performance of ANNs with traditional statistical methods. For the modeling issues, they addressed the type of data, size of the training and test samples, architecture of the model (number of nodes in each layer and transfer function), training algorithm used, and the method of data normalization. The paper provides insights into these modeling issues by summarizing the numerous, sometimes conflicting, suggestions derived from the articles. Only one of the articles pertained to an accounting and finance problem.

Vellido *et al*. (1999) surveyed 123 articles from 1992 through 1998. They included 8 articles in accounting and auditing, and 44 articles in finance (23 on bankruptcy prediction, 11 on credit evaluation, and 10 in other areas). They provided information on the ANN model applied, the method used to validate training of the model, the sample size and number of decision variables, the comparative parametric/linear technique used as a benchmark, and main contribution of the article. They summarize the most frequently cited advantages and disadvantages of the ANN models. The paper does provide a sense of which ANN methodologies have been applied to accounting and finance problems, but offer little insight into how to choose and develop an appropriate ANN model.

The objective of this paper is to provide a broader review of the literature on ANNs applied to accounting and finance problems, focusing on the modeling issues, and summarizing the 'suggestions' from this literature. It is more like a tutorial on modeling issues than a critical analysis. A subsequent paper will provide a more critical look at these modeling issues in the accounting and finance articles.

The first section will review the basic foundation of ANNs to provide a common basis for further elaboration. Based on this foundation, we suggest criteria that should be used to determine whether it is appropriate to use an ANN. For a more detailed description of ANNs, we refer the reader to numerous other articles that provide insights into various networks (Anderson and Rosenfeld, 1988; Hecht-Nielsen, 1990; Hertz *et al.*, 1991; Hoptroff *et al.*, 1991; Lawrence, 1991; Rumelhart and McClelland, 1986; Waite and Hardenbergh, 1989; Wasserman, 1989).

The second section of the paper discusses development of ANN models. Once the decision to use an ANN is made, the researcher faces numerous modeling decisions concerning:

- The selection of the learning algorithm

- The choice of the error and transfer functions
- The specification of the architecture
- The appropriate preparation of the data to match the architecture and
- The approach used to train of the network.

The final section presents some general guidelines, conclusions on research progress and open research questions.

FOUNDATIONS

Background

ANNs are structures of highly interconnected elementary computational units. They are called *neural* because the model of the nervous systems of animals inspired them. Each computational unit (see Figure 1) has a set of input connections that receive signals from other computational units and a bias adjustment, a set of weights for each input connection and bias adjustment, and a transfer function that transforms the sum of the weighted inputs and bias to decide the value of the output from the computational unit. The sum value for the computational unit (node j) is the linear combination of all signals from each connection (A_i) times the value of the connection weight between node j and connection i (W_{ji}) (equation (1)). Note that equation (1) is similar to the equation form of multiple regression: $Y' = B0 + \Sigma_i [B_i * X_i]$. The output for node j is the result of applying a transfer function g (equation (2)) to the sum value (Sum_j).

$$\text{Sum}_j = \Sigma_i [W_{ji} * A_i] \quad (1)$$

$$O_j = g(\text{Sum}_j) \quad (2)$$

If the transfer function applied in equation (2) is linear, then the computational unit resembles the multiple regression model. If the transfer function applied in equation (2) is the sigmoid (see description below), then the computational unit resembles the logistic regression model. The only difference between the ANN and regression models is the manner in which the values for the weights are established. ANNs employ a dynamic programming approach to iteratively adjust the weights until the error is minimized while the regression models compute the weights using a mathematical technique that minimizes the squared error.

Most ANNs applied in the literature are actually a network of these computational units (hereafter referred to as nodes) interconnected to function as a collective system. The *architecture* of the network defines how the nodes in

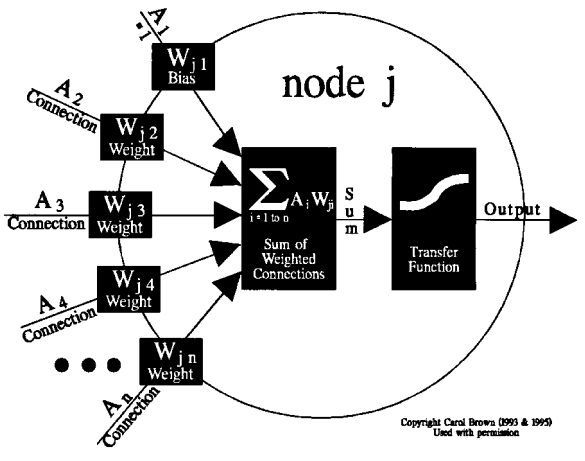

Figure 1 Structure of a computational unit (node j)

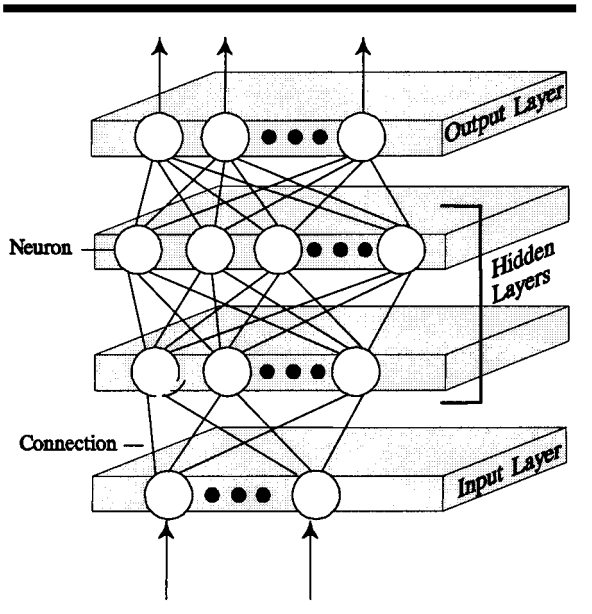

Figure 2 Feed-forward neural network structure with two hidden layers

a network are interconnected. A multi-layer, feed-forward architecture is depicted in Figure 2. The nodes are organized into a series of layers with an input layer, one or more hidden layers, and an output layer. Data flows through this network in one direction only, from the input layer to the output layer.

What is the advantage of a multi-layered network? An ANN model with no hidden layers (a single computation unit) can separate data that falls on opposite sides of a hyperplane (see the left panel in Figure 3). Compared to a linear model, the hyperplane generated by the ANN model will be non-linear (note that it has a large linear section, but curves at the endpoints). In this example, the linear and non-linear ANN models would produce comparable classification accuracies (each would misclassify four of the 49 data points).

If a single hidden layer is added, then each node in the hidden layer will form a hyperplane. The nodes in the output layer of this single hidden layer network combine the hyperplanes to create convex open or closed regions. An ANN model with two nodes in a single hidden layer is depicted in the middle panel in Figure 3. In this example, a single node in the output layer selects which of the hyperplanes to apply. The classification accuracy of this one hidden layer model has increased to 48 of the 49 data points.

If two hidden layers are used, then the second hidden layer combines the hyperplanes from the first hidden layer into convex regions, and the nodes in the output layer now combine the convex regions to form concave regions.

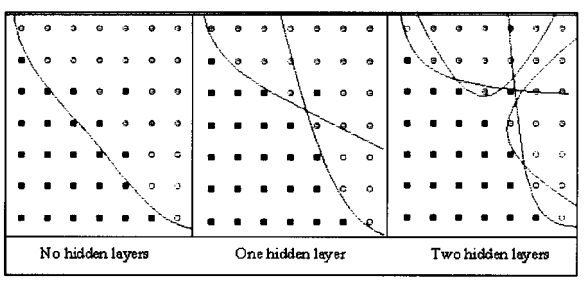

Figure 3 Example to demonstrate the advantages of hidden layers

The ANN model depicted in the right panel of Figure 3 would have six nodes in the first hidden layer and four nodes in the second hidden layer. The single node in the output layer would select among the four convex regions. Note that we are now able to attain 100% classification accuracy with the ANN model.

Research has shown that an ANN with two hidden layers can approximate a particular set of functions to a given accuracy (Cybenko, 1988; Lapedes and Farber, 1987). Research has also shown that it is possible to use a single hidden layer to approximate a continuous function and achieve the desired accuracy (Cybenko, 1989; Hecht-Nielsen, 1990; Hertz *et al.*, 1991; Hornik *et al.*, 1989). Since a single hidden-layer model is less complex, most researchers have favored these over two hidden layer models. We were not able to find any references in the literature when more than two hidden layers were used. Thus, we will limit the discussions in this paper to one and two hidden-layer models.

Networks that contain feedback connections are said to be *recurrent*. Recurrent networks recirculate previous outputs back to the inputs, similar to the concept of using lagged variables in forecasting. The recurrent feature of an ANN inherently considers the moving average factor in a time series and outperforms a multi-layered network (Jhee and Lee, 1993). However, most ANN applications in accounting and finance have employed a multi-layer, feed-forward architecture.

Linear versus Non-linear Models

As discussed above, an ANN with no hidden layers is similar to a generalized linear model. Feedforward ANNs with hidden layers are a subset of the larger class of non-linear regression and discrimination models. Generally, one chooses a non-linear model over a linear model when the underlying relationships between the variables are either known to be non-linear, or are not known, *a priori*.

Linear economic models are not able to capture non-linear patterns and trends in the

relationships between and within most econometric predictor variables used in accounting and finance research. For example, a time series can be broken down into four components: secular trend, cyclical variation, seasonal fluctuation and irregular fluctuation. The irregular fluctuation can be further divided into deterministic chaotic behavior and stochastic noise. Conventional linear techniques cannot distinguish between these two sub-components of random noise and non-linear relationships (Wong, 1991).

Austin *et al.* (1997) used an ANN to identify mathematical relationships between current values of macroeconomic and financial variables and future stock market valuations. They found that linear economic models were not able to capture non-linear patterns and trends in the relationships between and within stock and bond price movements. The linear techniques could not distinguish between random noise and non-linear relationships.

When applied to classification tasks, ANNs have the ability to induce algorithms for recognizing patterns. Thus, the knowledge base is inferred from the facts presented to train the network. The disadvantage, however, is that these networks do not have facilities to explain how a conclusion is reached. Hence, they are best suited for rather straightforward discrimination and classification problems involving complex relationships. Curram and Mingers (1994) found that ANNs provided better classification rates than linear discriminant analysis on data sets that were non-linear, but were generally slightly worse where the data were linearly separable. When using real data where the degree of non-linearity is often not known, they suggest using both methods and comparing the performance.

A limitation of using financial ratios with linear discrimination models is that the ratio must provide a dichotomous signal. For example, a ratio value above or below a given threshold must always signal financial distress. In practice, a ratio may signal distress both when it is higher than normal and when it is lower than normal (Coats and Fant, 1993). So in these situations, ANNs may provide a better model to capture the underlying relationships between the financial ratios and the dependent variable.

In general, an ANN should not be used to model inherently linear relationships. At best, the ANN will turn out to be a far more complex and cumbersome approach to achieve comparable results. ANNs should be applied when there is some evidence of non-linearity in the relationships between the dependent and explanatory variables.

Parametric versus Non-parametric

Parametric models, and regression models in particular, have become abused statistical methods. Tests are routinely performed and inferences made without verifying normality of errors, independence of errors and constancy of error variance (Marques *et al.*, 1991). ANNs can be used as parametric models. Previous research has shown that back-propagation will minimize the least squared error if (1) the model does not get trapped in a local optimal and (2) there are an adequate number of nodes in the hidden layer. If an ANN is used as a parametric model, the same sort of distributional assumptions are required for error terms as for a statistical model. These distributional assumptions of noise that is independent between each case and normally distributed with equal variance affect the ability to make statistical inferences about the results.

However, ANNs applied as non-parametric models can easily incorporate multiple sources of evidence without simplifying assumptions concerning the functional form of the relationship between output and predictor variables. In many cases, the predictor variables are not normally distributed and customary transformations have been unsatisfactory. Markham and Ragsdale (1995) found that the data typically used in the study of bank failures did not fit normal distributions even after data transformations. When such statistical assumptions (distribution, independence of multiple features, etc.) are not valid, an ANN that does not rely on these assumptions provides better generalization properties and seems to be better suited to handle small sample problems (Niles *et al.*, 1989).

Parametric statistical models require the developer to specify the nature of the functional relationships between dependent and independent variables. ANNs use the data to develop an internal representation of the relationship between the variables so that *a priori* assumptions about underlying parameter distributions are not required. Thus, better results would be expected from the ANN if the known relationships between the variables do not fit the appropriate parametric statistical model.

Most parametric models require that the input variables be linearly separable. That is, no two input variables can have similar influences on the dependent variables. When financial ratios and aggregate account balances are used as input, this requirement can be easily violated.

Within a parametric model, outliers in a data set influence the size of the correlation coefficient, the average value for a group, or the variability of scores within a group. Univariate outliers, cases that are extreme with respect to one of the input variables, are relatively easy to detect. There are situations where the outlier is caused by the interaction of two or more variables. These multivariate outliers are harder to detect since the values for each individual variable are within bounds (they were not detected as a univariate outlier). There are numerous aspects of an ANN that make them more robust with respect to outliers (non-linear transfer functions within each computational unit, moving average weight adjustments, etc.). Research has shown that ANNs are more robust than regression when outliers are present in the data (Marques *et al*., 1991; Subramanian *et al*., 1993).

In summary, ANNs applied as non-parametric models are not as constrained by distribution-related requirements as most traditional statistical models. The non-parametric ANN model may be preferred over traditional parametric statistical models in those situations where the input data do not meet the assumptions required by the parametric model, or when large outliers are evident in the dataset.

Selection Criteria

Most ANNs used in the accounting and financial literature have been applied as statistical methods, not as artificial intelligence techniques that attempt to mimic the basic architecture and functioning mechanisms of the human brain. We consider an ANN as a non-linear statistical method that offers certain advantages over its parametric counterparts. As such, the first fundamental decision that should be made by a researcher is whether it is appropriate to use an ANN to solve accounting and finance problems, as opposed to a traditional parametric statistical model.

DETERMINE APPROPRIATE PARAMETRIC MODEL

As a first step, the researcher should determine the research question being asked and the appropriate parametric modeling approach that would be used if all the underlying assumptions were valid. Tabachnick and Fidell (1983) offer a classification scheme relating research questions to the appropriate parametric models. We have extended this classification scheme to distinguish the nature of the output variable(s) (continuous versus discrete). We have also categorized ANN applications to accounting and finance problems according to this scheme (see Table 1).

The first category assesses the strength of the relationship between the input and output variables. The greater the strength of the relationship, the greater ability to predict the value of the output variable. When the output variables are continuous, the traditional parametric models applied to answer these research questions have been regression-based and econometric approaches (Box–Jenkins ARIMA). The accounting and finance problems which fall into this category include the application of forecasting techniques to stock prices and other market-related indicators, cost estimation, and other managerial forecasting studies. When the output variables are discrete, then logistical regression models must be applied. However, the accounting and finance studies which have

Table 1 Classification scheme for research questions

Research question type	Type of output	Parametric model	Accounting and finance ANN application
Assess degree of relationship (forecasting, pattern recognition)	Continuous	Regression Econometric approaches (Box–Jenkins ARIMA).	• Stock price and market-related studies (White, 1988; Kamijo and Tanigawa, 1990; Kimoto *et al.*, 1990; Yoon and Swales, 1991; Bergerson and Wunsch, 1991; Wong, 1991; Trippi and De Sieno, 1992; Kryzanowski *et al.*, 1993; Grudnitski and Osburn, 1993; Charitou and Charalambous, 1995; Austin *et al.*, 1997; Kohara *et al.*, 1997) • Cost estimation (Hoyt and Lay, 1995) • Analytic review (Coakley and Brown, 1993; Coakley, 1995) • Bankruptcy (Udo, 1993) • Managerial forecasting studies (Jhee and Lee, 1993; Choi *et al.*, 1997)
Predict group membership (two group and multiple group classification)	Discrete Discrete	LOGIT LOGIT Discriminant functions usually includes tests of statistical significance and the evaluation of the various predictor variables	See Two-group classification Two-group classification studies: • Predictions of management fraud (Fanning *et al.*, 1995; Green and Choi, 1997; Fanning and Cogger, 1988) • Auditor's going concern uncertainty opinion (Hansen and Messier, 1991; Coats and Fant, 1993; Lenard *et al.*, 1995; Anandarajan and Anandarajan, 1999) • Bank failures (Bell *et al.*, 1990; Chung and Tam, 1992; Tam and Kiang, 1992; Subramanian *et al.*, 1993; Merkham and Ragsdale, 1995; Bell, 1997; Etheridge and Sriram, 1997) • Bankruptcy (Boucher, 1990; Raghupathi *et al.*, 1991; Salchenberger *et al.*, 1992; Odom and Sharda, 1992; Brocket *et al.*, 1994; Fanning and Cogger, 1994; Wilson and Sharda, 1994; Boritz *et al.*, 1995; Boritz and Kennedy, 1995; Huang *et al.*, 1994; Barniv *et al.*, 1997; Jo *et al.*, 1997) • Credit evaluation and underwriting (Collins *et al.*, 1988; Reilly *et al.*, 1991; Jensen, 1992; Grudnitski *et al.*, 1995; Torsun, 1996) • Bond ratings (Dutta and Shekhar, 1988; Surkan and Singleton, 1991) Multiple-group classifications: • Bond ratings (Kwon *et al.*, 1997; Daniels *et al.*, 1997; Maher and Sen, 1997). Prediction of group membership only
Assess underlying structure	Continuous	Principal components analysis Factor analysis	Traditional parametric techniques in this category were applied to achieve data reduction. ANN models with Hebbian learning are closely related to principal components analysis. No studies were found that applied ANN models in this category
	Discrete	Cluster analysis	• Prediction of bank failures (Etheridge and Sriram, 1997) Categorical Learning ANN was used but results were compared with logit and discriminant analysis rather than a ***k***-means clustering algorithm. • Bankruptcy (Chen *et al.*, 1995; Serrano-Cinca, 1996). Kohonen networks are very similar to ***k***-means cluster analysis.

Table 1 Continued

Research puestion type	Type of output	Parametric model	Accounting and finance ANN application
			• None found
Significance of group differences	Continuous	ANOVA	Limitations of the ANN models is the computational complexity required to assess the effect of an input variable on the output variables
	Discrete	PROBIT	

Note: A short summary of each study cited in this table can be found in Coakley *et al*. (1995).

used logistic regression have focused more on the two-group classification problem and are included as classification studies.

The second category measures the degree to which group membership can be predicted from the various input variables. The groups act as discrete levels of the output variable (hence there are no comparable models for continuous output variables). This type of research question is analyzed with logistic regression and discriminant functions, and usually includes tests of statistical significance and the evaluation of the various predictor variables. The ANN models focus on the prediction of group membership only. We further subdivide the accounting and finance studies in this category into two-group and multiple-group classification problems. The two-group classification studies include predictions of management fraud, the auditor's going concern uncertainty opinion, bank failures, bankruptcy, and credit evaluation and underwriting. The studies on predicting bond ratings range from two-group classification to multiple-group classifications.

The third category deals with finding an underlying structure inherent in a set of financial data. These types of problems are answered with principal components analysis (and factor analysis) when the output variables are continuous, and cluster analysis when the output variables are discrete. ANN models with Hebbian learning are closely related to principal components analysis. In the accounting and finance literature, the traditional parametric techniques in this category were applied to achieve data reduction. We did not find any studies where these type of ANN models were applied to accounting and finance problems, although determining the information content of financial statements seems a likely candidate.

There were two studies that applied Kohonen networks to perform k-means cluster analysis. Although Etheridge and Sriram (1997) did apply a Categorical Learning ANN, the study focused on the prediction of bank failures and the results were compared with logit and discriminant analysis (versus a *k*-means clustering algorithm). Chen *et al*. (1995) evaluated bankruptcy data and found that the Self Organizing Map achieved higher classification accuracy than conventional clustering methods (84.8% compared to 54.3%). Serrano-Cinca (1996) applied a Self Organizing Feature Map to cluster firms into zones of bankruptcy, zones of solvency, and a zone of doubtful insolvency. This zone of doubtful insolvency contained the firms that were mis-classified in previous studies using the same data set (Odom and Sharda, 1992; Wilson and Sharda, 1994). In addition to improved classification accuracy, the SOFM provides graphic output depicting the risk of bankruptcy and the financial characteristics of the firm.

The final category measures the significance of group differences. The emphasis is on inferring population differences among groups. The classic parametric model is Analysis of Variance (ANOVA) when the output variable is continuous, and PROBIT when the output variable is discrete. One of the limitations of the ANN models is the computational complexity required to assess the effect of an input variable on the output variables. To date, there are no known attempts to apply ANN models in this manner.

Validate Underlying Assumptions

Once the nature of the research question has been established, and the appropriate parametric model is determined, the next step is to validate the underlying assumptions for the parametric model.

(1) If the validity of these assumptions is questionable, then an ANN may provide better results.
- When applied to determine the degree of relationships with continuous outputs, ANNs have been shown to outperform ARIMA (Choi *et al*., 1997), and multiple regression (Kohara *et al*., 1997; White, 1988).
- When applied to determine the degree of relationships with discrete outputs (two-group classifications), ANNs have been shown to produce lower type I errors (e.g. 0.13 for ANN, 0.536 for logit and 0.533 for discriminant analysis), but also higher type II errors (0.09 for ANN, 0.012 for logit and 0.014 for discriminant analysis). In general, the combined type I and II errors are lower for the ANNs over the other methods. However, the comparison of the methods becomes dependent on the relative cost of the type I versus type II errors (Etheridge and Sriram, 1997).

(2) If complex non-linear relationships are expected between the predictor variables, then a non-linear ANN should provide better results than a linear model. If the non-linear relationships do not exist, the ANN results should be similar to linear model results (Charitou and Charalambous, 1996).

(3) Marques *et al*. (1991) suggest that ANNs will perform better than their parametric counterparts under conditions of high noise and low sample size. The ANNs become less competitive when there is low noise or large samples. Slightly different results were obtained by Markham and Rakes (1998). Using simulated data, they compared linear regression and ANN models across 20 datasets varying the sample size and the variance in the error. With low variance, linear regression outperformed ANN models across all sample sizes. With high variance, ANN models were better. With medium variance, regression performed better with small sample sizes, while ANNs performed better with large sample sizes.

(4) With respect to classification problems, ANNs perform better as the number of variables and groups increase, and are more robust to changes in sample size and proportion (Subramanian *et al*., 1993).

If the underlying distributional properties of the data are not known *a priori*, it is recommended that both ANN and parametric models be developed and compared. The use of an ANN model does not guarantee better results over a parametric model. Poor performance with both models could indicate the lack of a deterministic relationship between inputs and outputs. In general, failures in ANN performance can be attributed to the use of an inadequate learning algorithm, the use of an inappropriate model and/or architecture, invalid data representation to the model, or inadequate training of the network. These factors are discussed in the next section.

DEVELOPMENT OF THE ANN MODEL

Once the decision to use an ANN is made, the researcher faces numerous decisions including selection of the learning algorithm, choice of the error and transfer functions, specification of the architecture, preparation of the data to match the architecture, and training of the network.

Selection of the Learning Algorithm

ANNs effectively filter input to produce output. More specifically, an ANN looks for patterns in a set of examples applied to the input layer of the network, and learns from those examples to produce new patterns, the output. Knowledge within the ANN is maintained in the weights. The process of learning is implemented by changing the weights until the desired response is attained at the output nodes. In an ANN with linear transfer functions, the weights can be derived using matrix

manipulation (much like solving the general linear model). In an ANN with non-linear transfer functions, two learning mechanisms can be used to derive the weights: unsupervised learning and supervised learning.

Unsupervised learning is analogous to a cluster analysis approach. The network self-organizes to identify the salient properties of the input data set. The input patterns are classified according to their degree of similarity, with similar patterns activating the same output pattern.

Supervised learning accepts input examples, computes the output values, compares the computed output values to the desired output values (termed *target values*), and then adjusts the network weights to reduce the difference (error). The learning process is repeated until the difference between the computed and target output values are at an acceptably low value.

The most common supervised learning algorithm is *back-propagation*. Back-propagation employs a gradient-descent search method to find weights that minimize the global error from the error function. The error signal from the error function is propagated back through the network from the output layer, making adjustments to the connection weights that are proportional to the error. The process limits overreaction to any single, potentially inconsistent data item by making small shifts in the weights. A complete description of the algorithm and its derivations can be found in numerous sources, including Hertz *et al*. (1991) and Rumelhart *et al*. (1986).

The commonly cited disadvantages of the back-propagation learning algorithm are that the training time usually grows exponentially as number of nodes increases, and there are no assurances that a global minimum will be reached. The use of partial derivatives in gradient descent algorithm makes the assumption that the error surface is locally linear. At points of high curvature this linearity assumption does not hold and divergent behavior might occur. To compensate for this possibility, an infinitesimally small step size (learning rate) must be used to guarantee asymptotic convergence to a minimum point.

But, small learning coefficients lead to slow learning. The momentum factor (delta weight) was added to back-propagation algorithm to act as a low-pass filter on the weight adjustment terms. General trends are reinforced whereas oscillatory behavior cancels itself out. It allows a low learning coefficient with faster learning. But, incorrect specification of these parameters (learning rate and momentum term) can lead to either extremely slow convergence or to oscillatory behavior without convergence.

One technique suggested to speed up learning is the use of the conjugate-gradient algorithm. This algorithm uses the second derivative of the error function to exploit information regarding both the slope and curvature of the response surface. When compared to back-propagation, the conjugate-gradient algorithm has been shown to produce comparable results with much faster training times (Charitou and Charalambous, 1996; Wong, 1991).

A major drawback of the gradient descent algorithms is that there is no way of determining in advance whether the architecture and selected methods will converge. Real error surfaces, with multiple weight dimensions, tend to have very complex features including dents and ravines. Although the gradient-descent method always follows the steepest path towards the bottom of the error surface, it may get stuck within a large dent or ravine on the side of the surface. Numerous methods are available to compensate for this tendency to find local (non-optimal) minima. Some methods adjust the weight derivation process to maintain the momentum established by previous adjustments. Other methods involve starting from a different point on the error surface (using a different set of initial weights) and ensuring that the results are similar. Still other methods involve the dynamic adjustment of the network architecture (trimming nodes or connections between nodes).

Evolutionary programming is a stochastic optimization technique that has been used in accounting and finance problems as an alternative to the conventional gradient methods. Evolutionary programming involves two processes—mutation and selection. The algorithm starts with an original population of weight sets that are evaluated by examining the corre-

sponding outputs. Random mutation of the parents creates new solutions. Specifically, a Guassian random variable with mean zero and variance equal to mean squared error of the parent perturbs each weight and bias term. Each offspring is evaluated, and the n 'best' solutions are retained (selected) as parents for the next iteration.

Genetic algorithms extend this mutation and selection process by adding a recombination phase to create the child nodes that are evaluated. Each child is formed as a cross between two parents, hence the term 'crossover'. Goldberg (1994) cites numerous advantages of genetic algorithms: they can easily solve problems that have many difficult-to-find optima, they are noise tolerant, and they use very little problem-specific information. They work with the coding of the parameter set, not the specific values of the parameters. The major disadvantage cited for genetic algorithms is the difficulty in specifying the optimal parameter settings for the model.

A genetic algorithm was used by Huang *et al* (1994) to predict financially distressed firms using unbalanced groups (the group of financially distressed companies represents less than 3% of the population). The ANN clearly dominated discriminant analysis and Logic methods applied to the same data set. Levitan and Gupta (1996) applied a genetic algorithm to the cost driver optimization (CDO) problem in activity-based costing. CDO is a combinatorial optimization problem that is NP-hard. Compared to a greedy algorithm proposed by Babad and Balachandran to solve the CDO problem, the genetic algorithm was less complex (used fewer cost driver parameters) and calculated more accurate costs.

Genetic algorithms have been demonstrated as possible techniques to aid in the development of the ANN model. Back *et al*. (1996) used a genetic algorithm to select the best input variables for an ANN model applied for bankruptcy prediction. Hansen (1998) demonstrated the use of a genetic algorithm to find the most effective ANN architecture.

Optimal Estimation Theory has also been applied to accounting and finance problems as an alternative learning algorithm to back-propagation. Optimal Estimation Theory (OET), introduced by Shepanski (1988), uses a least squares estimator to calculate the optimal set of connection weights for the presented training set. This method significantly reduces the time required to train a network, but requires having the same number of nodes in the input and hidden layers. This method was used by Boritz *et al*. (1995) to predict bankruptcy filings. They compared the performance of the OET algorithm with back-propagation, and found mixed results. In general, when the numbers of bankrupt vs. non-bankrupt firms in the training sets were equal, the back-propagation algorithm provided higher classification accuracy. However, Boucher (1990) also used training data sets with equal numbers of bankrupt and non-bankrupt firms and found that the OET algorithm provided higher overall classification accuracy than the back-propagation algorithm.

It appears that the OET algorithm does train faster, but it is not known whether it achieves similar performance. The discussion below on network size suggests that the requirement to maintain a fixed number of nodes in the hidden layer may be a limitation of the OET algorithm. Larger or smaller networks trained with the back-propagation algorithm may achieve better results.

In a survey of ANNs in business, Velido *et al.* (1999) found that the gradient descent back-propagation model dominated other methods (of 92 reported papers, 74 used back-propagation). Given the often-cited disadvantages of this approach, it still seems to be the favored one. The algorithm is intuitively simple, and applications incorporating back propagation are freely available on numerous web sites. In addition, researchers have seen a ten-fold increase in processing power at the desktop in the past decade that may have diminished some of the previous concerns regarding slow training. Back-propagation will be the focus in the remainder of this paper.

Choice of the Error and Transfer Functions

Error Functions

The sum-of-squared-error (SSE) error function is the one most widely applied in the account-

ing and finance literature. Since the SSE function is differentiable and minimized when its arguments are equal, the resulting error surface (in three dimensions) looks like a bowl. The bowl's bottom corresponds to the set of weights that produce the minimum error. At any given point on the bowl's surface, the derivative of the error function provides the slope of the surface with respect to the weights. To minimize the error, the weights are adjusted to move down the slope towards the bowl's bottom. The SSE function uniformly weights each training trial error in accordance with the square of the magnitude of the error vector (*Target* − *Output*). This error-measurement scheme ensures that large errors receive much greater attention than small errors. Also, the SSE takes into account the frequency of occurrence of particular inputs. It is much more sensitive to errors made on commonly encountered inputs than it is to errors on rare inputs (Hecht-Nielsen, 1990).

However, any differentiable function of the target and output variables, F(*T, O*), that is minimized when its arguments are equal could be used as an error function in the back-propagation model. The more important consideration is that the error function should reflect the statistical nature of the original data as well as any assumptions about measurement errors.

Coakley (1995) proposes that an alternative error function that maximizes the relative entropy between the input data streams may be more appropriate for classification tasks. This error function, based on information theory and proposed by Hertz *et al.* (1991), involves learning the correct probabilities of a set of hypotheses represented by the output unit. Information theory is applied to derive the expected information content of an input signal, which is expressed as a conditional likelihood function. The entropy error function is formed by maximizing the conditional likelihood function, and corresponds to the maximum conditional likelihood function commonly used in Logit analysis (Loh, 1992; Palepu, 1986). The maximum conditional likelihood function is appropriate for dichotomous output variables. As a result of using the entropy error function, the ANN will place more weight on those signals that produce complementary information, and less weight on those signals that provide redundant or conflicting information.

Transfer Functions

The transfer function is used to derive the output of a node given its weighted-adjusted input. The use of linear transfer functions requires that the input patterns be linearly independent. If non-linear transfer functions are used, linear independence of the input patterns is not required. Thus, non-linear transfer functions allow ANN models to be applied to a wider range of problems (Hertz *et al.*, 1991).

Four non-linear transfer functions have been proposed in the literature for use with ANN models: sigmoid (logistic), half-sigmoid, sine (or cosine), and hyperbolic tangent (see Anderson and Rosenfeld, 1988; Hecht-Nielsen, 1990; Hertz *et al.*, 1991; Lawrence, 1991; Rumelhart and McClelland, 1986; Stornetta and Huberman, 1987; Waite and Hardenbergh, 1989; Wasserman, 1989). We did not find any accounting and finance papers that reported the use of the sine transfer function, so have excluded that function from the analysis.

The sigmoid function (equation (3)) is a bounded differentiable real function that is defined for all real input values and has a positive derivative everywhere. As depicted in Figure 4, the sigmoid is a semi-linear function that makes a rapid yet smooth transition from a nearly proportional center section to the upper and lower limits. The sigmoid function is centered at 0.5, and provides an output range of zero to one.

$$g(h) = (\text{High} - \text{Low}) / (1 + \exp(-\text{Gain} * (h - \text{Center}))) + \text{Low} \qquad (3)$$

where:

- High is the upper limit for the output.
- Low is the lower limit for the output.
- Center is the value of the input at which the output is equal to (High + Low)/2.
- Gain is directly proportional to the derivative of the function at the Center point. With high gain (Gain >> 1), the sigmoid function approximates a step function, while at low

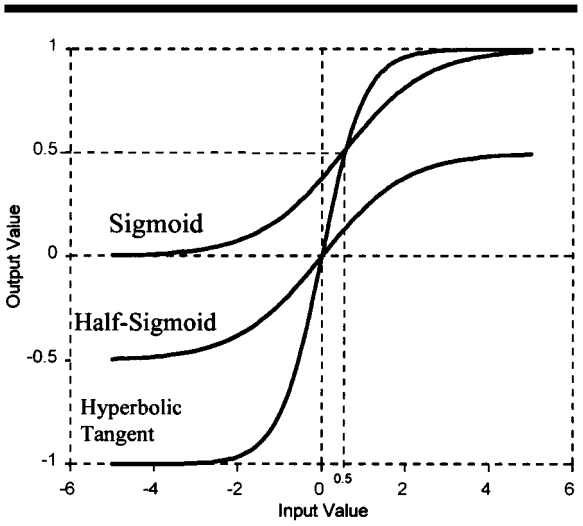

Figure 4 Comparison of sigmoid, Half-Sigmoid and hyperbolic tangent Transfer functions

gain (Gain << 1) it approximates a linear function.

- h is the sum of the weighted connections.

Stornetta and Huberman (1987) suggest that the conventional sigmoid function may not be optimal for all data sets. With the back-propagation algorithm, the magnitude of a weight adjustment at a particular node is proportional to the output level of that node. With a sigmoid function, it is possible to get an output value of zero from a node. These appropriate output values would result in no weight modification. They propose an alternative that changes the output range of the sigmoid transfer function by adding a bias of $-\frac{1}{2}$. Thus in equation (3), the High and Low values would be set to (0.5, –0.5) respectively. This modified sigmoid function, commonly termed the 'Half-Sigmoid', places less weight on those data values close to the mean, yet provides an improved capability for explaining those input data values close to the lower bound. Note in Figure 4 that the Half-Sigmoid function is centered at zero and has a range of $\pm\frac{1}{2}$.

Another alternative is to use the hyperbolic tangent function (equation (4)). Like the half-sigmoid, it is centered at zero. However, the range for the output values is wider (± 1 versus $\pm\frac{1}{2}$). This function is more linear in nature and produces an effect similar to truncating outliers in a linear model.

$$g(h) = \tanh(h)$$

Coakley *et al.* (1992) compared the performance of these three transfer functions when used in a multi-layer, feed-forward, back-propagation network. The study was limited to evaluating the forecasting performance of the ANN when applied to a computer-generated time series with seasonal variations and random noise. In general, they found that the hyperbolic tangent transfer function had faster convergence and slightly better predictive accuracy than the sigmoid and half-sigmoid functions. There were no consistent differences between the sigmoid and the half-sigmoid function in terms of either performance or speed of convergence.

For an ANN with no hidden layers and a linear transfer function, Brown *et al.* (1993) present a theoretical explanation for why learning is faster when the input values are in the range -b to b rather than 0 to b. They extend their arguments to suggest that the use of the hyperbolic tangent transfer function in the hidden layer of an ANN is also better.

Selection Considerations

There are no clear criteria regarding which transfer function to use. The best guidance that can be offered is to consider how the transfer function relates to the error function. To solve the back-propagation algorithm, the weights in the network must be adjusted to minimize the error function. Using the gradient-descent method, the change in the weights (ΔW_{ji}) is derived from the derivative of the error function E with respect to the weights W_{ji} (equation (5)). For the derivations presented here, subscript j refers to processing units in the output layer, subscript i refers to the processing units in the input layer, and there are no hidden layers in the model. The equations for the output processing units do not change in form if hidden layers are added to the model. In that case, the subscript i would refer to the processing units in the hidden layer immediately preceding the output layer:

$$\Delta W_{ji} = -\eta \frac{\partial E}{\partial W_{ji}} \quad (5)$$

where:

- Eta (η) is the learning coefficient
- E = f(Target Value − Output Value), f any function with characteristics discussed in the following section
- Output Value = $g(\Sigma_i [W_{ji} * A_i])$ (from equations (1) and (2))

Equation (5) is easier to solve with certain combinations of transfer function and error function. For example, the sigmoid transfer function and sum-of-squared-error error function combination provide a simple solution, as do the combination of the hyperbolic-tangent transfer function and relative-entropy error function.

With the SSE error function, the partial derivative in equation (5) has a tendency to bound the maximum values of the weights. As Figure 5 depicts, the adjustment to the weights will tend to zero when the target and predicted values are equal, as expected, but will also be zero when the target and predicted values are at the extreme opposites (target = 0, predicted = 1 for the sigmoid case).

One interesting difference to note in the two graphs in Figure 5 is the relative values of the delta weights. For the sigmoid function, the delta weight values range from −0.15 to +0.15. For the hyperbolic tangent function, the values range from −1.2 to +1.2. This difference has caused some researchers to recommend the sigmoid function for classification problems that involve learning about average behavior, and the hyperbolic tangent function if the problem involves learning about deviations from the average (such as forecasting). This difference in delta weight values also becomes extremely important when the learning coefficient and momentum factors are set. Lower values need to be used if the hyperbolic tangent transfer function is to be used (Coakley *et al.*, 1992).

For the entropy error function, a different story emerges. For the sigmoid transfer function, the delta weight is only bounded when the predicted value approaches zero (top panel in Figure 6). When combined with the hyperbolic tangent transfer function, the delta weight adjustment is linear (bottom panel in Figure 6).

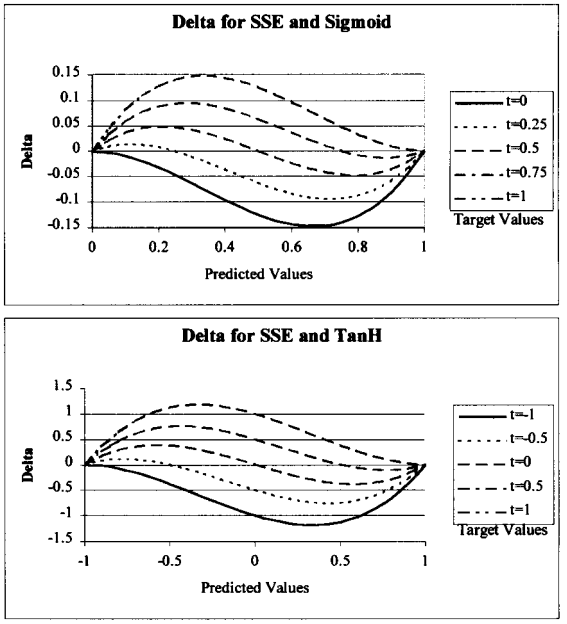

Figure 5 Delta for SSE error function

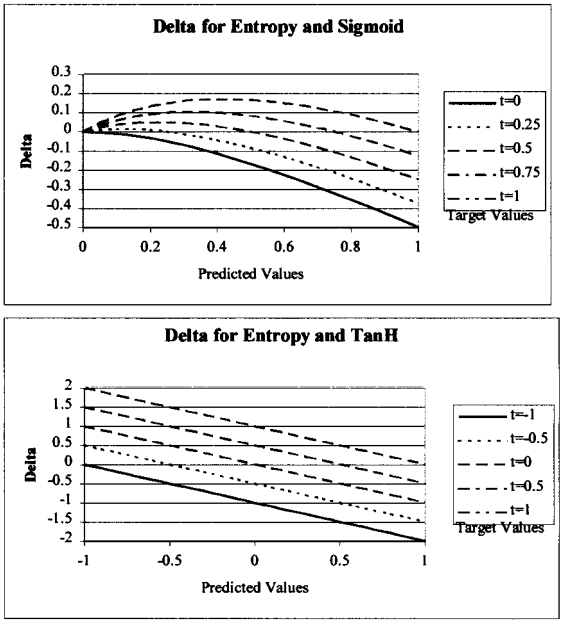

Figure 6 Delta for entropy error function

The adjustment is still zero when the target and predicted values are equal, but the adjustment increases linearly as the difference between the target and predicted values increase. The implication is that if the learning coefficient and momentum adjustment factors are incorrectly specified, the ANN model will blow up very early in the training process.

Specification of the ANN Architecture

Determining the appropriate ANN architecture is not a simple, straightforward task. Since numerous alternative architectures exist, the first task is to determine which architecture is appropriate for the problem. Most of the research in accounting and finance has used the multi-layer, feed-forward model. Thus, our discussion will be limited to this architecture.

Input and Output Layers

With the feed-forward architecture, the interface with the environment usually establishes the number of processing nodes in the input and output layers. For example, many authors (Boritz *et al.*, 1995; Boucher, 1990; Coats and Fant, 1993) used an ANN to perform financial distress classifications based on the Altman (1968) model. Since the Altman model uses five financial ratios, each of these ANN architectures has a five-node input layer and a single-node output layer that represents the dichotomous classification of distressed vs. non-distressed.

When applied for forecasting, the number of input nodes corresponds to the number of lagged observations used to discover the underlying time-series patterns. This is a very critical decision since it captures the complex autocorrelation structures in the data. According to Zhang *et al*. (1998) there is systematic way to determine the number of lagged variables to include. They report that a number of studies are using genetic algorithms to determine the optimal input architecture.

The number of output nodes to use is not always straightforward. For example, Yoon and Swales (1991) used a model with one output node for distressed firms and a second output node for non-distressed firms. This does add additional information in that the network can distinguish firms that exhibit characteristics that are clearly associated with distressed firms, firms that exhibit characteristics that are clearly associated with non-distressed firms, and firms that exhibit characteristics that are associated with both. In the multiple-group classification problem of predicting bond ratings, Kwon *et al*. (1997) used a single output node for each bond rating category (for a total of five nodes in the output layer) while Maher and Sen (1997) used a single output node to distinguish the six ordinal bond rating categories. Since the data sets are different, it is difficult to do direct comparison of the results. But both models reported similar classification accuracy, and each model outperformed its parametric counterpart (multiple discriminant analysis in the Kwon study and logit in the Maher and Sen study).

Internal Architecture

Specifying the internal architecture requires determining the number of hidden layers, and the number of processing nodes in each hidden layer. These two parameters affect the overall performance of the ANN, and usually vary from one implementation to the next.

The ability of a multi-layer network to represent non-linear models has been established through an application of an existence theorem proved by Kolmogorov (1963) and restated for back-propagation neural networks by Hecht-Nielsen (1990). Additional research applied this general theorem to the back-propagation multi-layer feed-forward model:

- At most two hidden layers are needed to approximate a particular set of functions to a given accuracy (Cybenko, 1988; Lapedes and Farber, 1987). The arbitrary accuracy is obtained by including enough processing nodes in each of the hidden layers. Research suggests that an architecture with n input data streams will require at most $(2n + 1)$ processing nodes per hidden layer to achieve the desired accuracy (Cybenko, 1989; Hecht-Nielsen, 1990; Hertz *et al*., 1991; Hornik *et al*., 1989).
- It is possible to approximate a continuous function that may achieve the desired accu-

racy with a single hidden layer (Cybenko, 1989; Hecht-Nielsen, 1990; Hertz *et al.*, 1991; Hornik *et al.*, 1989). Extra hidden layers are detrimental due to the increased processing requirements to train and evaluate the model. Therefore, an architecture with a single hidden layer would be preferred as long as the desired accuracy can be obtained.

- A rule of thumb for obtaining good generalization from a network is to use the smallest system that will fit the data (Reed, 1993). One approach is to train successively smaller networks until the smallest one is found that will adequately learn the data. However, these smallest feasible networks tend to be sensitive to initial conditions and learning parameters and may be more likely to become trapped in local minima. So Reed's recommended approach is to train a network that is larger than necessary and remove parts that are not needed. Determining which parts are not needed is usually a trial and error process. Parts are selected for removal using a heuristic (e.g. remove the node with the smallest sum of weights). The smaller network is then tested to see if it continues to produce acceptable results. The process is similar to the 'Backward Elimination Procedure' used with regression models. The large initial size allows the network to learn reasonably quickly with less sensitivity to initial conditions while the reduced complexity of the trimmed system favors improved generalization.

These recommendations for smaller networks can be substantiated with a comparison of results between Curram and Mingers (1994) and Hart (1992). Both studies used similar data sets. Hart (1992) achieved a 29% error rate with a 20-hidden-node network while Curram and Mingers (1994) attained a 26% error rate with a 10-hidden-node network. This suggests that Hart's results may be subject to overfitting of the data caused by using more hidden nodes than necessary.

When applied for classification tasks, the ANN architectures should have fewer nodes in the hidden layer than in the input layer. The rationale for this heuristic is conceptually simple. If a Hebbian-learning ANN model was developed with the same number of nodes in the input, hidden and output layers, then at convergence, the weights would resemble an identity matrix. If only one node was used in the hidden layer, the weights would become collinear with the principal eigenvector of the input data. Having fewer nodes in the hidden layer than in the input layer creates a bottleneck that forces the model to perform data compression, and the resulting weights capture the principal components that form the basis for many parametric methods (Diamantaras and Kung, 1996).

Subramanian *et al*. (1993) recommend that the number of nodes in a single hidden layer should range from k to $n+1$, where k is the number of output nodes and n is the number of input variables. Another rule of thumb is that the number of nodes in the middle layer should be 75% of the number of nodes in the input layer (Salchenberger *et al.*, 1992). Generally speaking, too many nodes in the middle layer (too many connections) produce a network that memorizes the input data and lacks the ability to generalize. Patuwo *et al.* (1993) also found that the number of nodes affected the classification accuracy. As the number of nodes decreases, the average classification rate decreases in the training sample but increases in the test samples. Thus, increasing the number of nodes in the middle layer will degrade the network's ability to classify outside the training set (overfitting). Hence, excess nodes in the hidden layer will reduce the ability of the network to generalize.

Another consideration is that as the number of hidden nodes in a network is increased, the number of variables and terms are also increased. If the network has more degrees of freedom (the number of connection weights) than the number of training samples, there are not enough examples to constrain the network. This is similar to fitting a high-order polynomial though a small number of points.

De Villiers and Barnard (1992) conducted experiments comparing networks with one and two hidden layers and the same total number of connections (weights). Their results suggest that there is no statistically significant differ-

ence between the optimal performance of networks with one or two hidden layers, that networks with a single hidden layer do a better classification on average, and that networks with two hidden layers train easier if the number of nodes in each of the two hidden layers is equal.

Masters (1994) has proposed a geometric progression rule to determine the number of nodes in the hidden layers. If a single hidden layer is used, then the number of nodes = sqrt(n* m), where n = number of input nodes and m = number of output nodes. If two hidden layers are used, then first compute the parameter $r = (n/m)^{1/3}$. The number of nodes in the first layer = mr^2 and the number in the second layer = mr.

Torsun (1996) used Masters (1994) geometric progression rule to determine number of nodes in the initial model, and then nodes were progressively deleted until the best performance was attained. Since further pruning of a network can lead to better generalization, they continued by deleting weights that had the least effect on the solution. The final ANN achieved a classification accuracy of approximately 92%.

To overcome problems associated with the 'trial and error' approach to determining the appropriate internal network architecture, researchers have developed algorithms that adaptively adjust the architecture during training. Most of the algorithms proposed in the literature and applied to accounting and finance problems have started small and added connections:

- Malakooti and Zhou (1994) have developed an adaptive strategy algorithm which starts with a small number of hidden nodes and connections, then adaptively increases the number of connections first, and the number of nodes second, until the 'proper' topology is obtained.
- Fahlman and Lebiere (1990) have developed the Cascade-Correlation (Cascor) training paradigm. Cascor begins with no hidden nodes and then incrementally creates and installs hidden nodes to improve the network's ability to categorize. Coats and Fant (1993) found several advantages of the Cascor network: it determines its own size and design, it retains the structure it has built even if the training set changes, and it does not require back-propagation of error signals through the connections of the network. This eliminates the need for human trial and error experimentation to determine the number of hidden nodes. They found the Cascor network provided more effective financial distress pattern classifications than multiple discriminant analysis.
- Fanning and Cogger (1994) have developed the Generalized Adaptive Neural Network Architecture (GANNA) where the architecture of the network is learned by trial and error. This does not require *a priori* specification of number of layers, numbers of nodes, or other design choices. The GANNA uses an evolutionary mechanism to grow networks, with new nodes added provided they offer improvement.

Other examples, termed 'pruning' algorithms, start with a large network and reduce the number of connections during training:

- Armstrong *et al.* (1991) provide a trimming approach called the Adaptive Logic Network (ALN). The ALN is a Boolean logic-based tree framework that evolves by deleting those branches that are not providing any additional information. Fanning *et al.* (1995) found that the ALN approach produced similar results as the GANNA processor, and superior results to the Logit model, when applied to detect management fraud.
- Jhee and Lee (1993) applied a weight-decay technique that adds an extra term to the cost function that penalizes network complexity. This additional term pushes the smaller-valued weights to zero while having minimal impact on the larger-valued weights. In essence, their algorithm deletes connections within the model by freezing the value of the weights to zero.

In conclusion, specification of the internal architecture involves tradeoffs between fitting accuracy and generalization ability. While numerous expansion/pruning methods have been proposed, these methods are usually quite

complex, difficult to implement, and cannot guarantee selection of the optimum architecture. Most of the heuristics that are offered are based on simulations derived from limited experiments. None of the heuristics works well for all problems.

The design of an ANN is basically problem-dependent and requires an experimental trial-and-error approach. We do know that larger architectures are required for complex response surfaces, larger architectures are required (refer back to Figure 3 earlier in the paper to visually see the result of adding additional hidden layers). Multiple-group bond rating classification is a complex problem. Papers addressing this problem have used large architectures. Kwon *et al*. (1997) used an ANN with 41 nodes in the input layer, two hidden layers with 25 and 15 nodes each, and 5 nodes in the output layer. The Maher and Sen (1997) model used a single hidden layer with $2n+1$ nodes. At the other extreme, simple classification problems can be modeled with relatively simple architectures. Charitou and Charalambous (1996) used 9 nodes in the input layer, a single hidden layer with 4 nodes, and one node in the output layer to predict the sign of earnings change.

Preparation of the Sample Data

ANN training requires three sets of data: a training set which must be representative of the entire domain, an evaluation set which is used to determine when the best generalization ability has been obtained, and a test set which is used to evaluate the prediction/classification accuracy of the model. This tends to cause problems in some accounting and financial applications since the number of instances is sometimes limited.

Sample Size

Baum and Haussler (1989) have estimated that the minimum training set size required for typical classification tasks is N/I, where N is the number of connections (non-zero weights) in the network and I is the fraction of the test examples that include incorrect classifications. For the typical financial distress problem using the five Altman (1968) ratios, and a single layer network with five nodes, there will be 36 connections (presuming that a bias node is included at both the input and hidden layer). If a balanced set of training data is used, the fraction of incorrect classifications will be 0.5, suggesting that the training set should include 72 observations. Additional, independent observations will also be required in the evaluation and testing sets.

Patuwo *et al.* (1993) found that sample size in the training set affects the classification accuracy. As sample size increases, the average classification rate decreases in the training sample but increases in the test samples. The smaller the sample size, the easier it is for the networks to adjust themselves to the idiosyncrasies of the individual observations in the sample.

Data Transformations

It is generally recommended that the input data for the ANN be manipulated to match the input constraints on the transfer function. That is, data is scaled such that the midpoint of the data range is at the center of the transfer function, and the range of the input data is adjusted to match the input range of the transfer function. For the commonly used sigmoid transfer function, using an input data range of (0–1) will scale the data to the midpoint and limit the calculations in the initial iterations to the near-linear portion of the function. With wider input data ranges, those training samples at or near the extremes will be clipped at the upper or lower bound of the transfer function. Eventually, the derived weights should compensate for any differences between the input data range and input range of the transfer function. However, correct specification of the input data range facilitates training of the ANN.

Shanker *et al*. (1996) evaluated two data transformation methods: linear transformation to the zero to one range, and statistical standardization. They found that data standardization improves classification rate but requires more computation time. The improvement was found to diminish as the network became large, and as the sample size increased.

The range of the target (or known output) values **must** be adjusted to lie within the output range of the transfer functions in the output layer. Otherwise the error function may have local minima besides the global minima. Target values at or near the boundary will cause saturation of the network. Because non-linear transfer functions have asymptotic limits, a very large net input value is required to cause the transfer function to produce an output value that lies on the boundary. The algorithm will attempt to adjust the weights to large values to achieve the large net input value. A 'better' fit is usually obtained if the target values are scaled to 90% of the output range of the transfer function. So for the sigmoid transfer function, target values should be scaled into the range of 0.05 to 0.95. However, if the ANN is being use to predict group membership, scaling the target values to the range of the transfer function, in this case, 0 to 1, allows the output from the network to be directly interpreted as posterior probabilities.

Coding of Categorical Variables

For non-ordered categories, multiple variables must be used (regardless of whether it is an input or output variable). In parametric modeling, all input variables must be linearly separable. Thus, c categories must be coded with c-1 variables. If the data has been coded with 'effects coding' for the associated parametric model, then the same data can be used with the ANN. ANNs are less sensitive to linear combinations of the input data, so c categories could be coded with c variables. Coding ordinal categories with a single variable makes it easier for the network to use the order of the category to generalize.

Initializing Connection Weights

The initial connection weights must also be specified prior to training. Generally, these weights are randomly assigned within a prespecified range. Experiments by Wessels and Barnard (1992) suggest that the initial training weights should occupy the range of $\pm 3A/\text{sqrt}(N)$, where A is the standard deviation of the inputs to the node and N denotes the number of weights leading to a particular node. Results indicate that this method performs close to optimally when the value of A is approximately one.

Training of the Network

Training of the ANN involves propagating the error (generated from the forward processing of the input training records) to adjust the set of weights to minimize the error function. The momentum factor in the algorithm acts as a moving average of the weight adjustments. Thus, if the error from each input record is propagated before the next record is processed, the network will capture the temporal information between the individual record sets (e.g. single-period autocorrelation). If multiple input records are forward processed before propagation, the model captures the temporal information between those record sets (e.g. seasonal fluctuations). If the ANN is being used for classification, then all of the input records should be processed before the error is propagated. If this is unreasonable due to computational constraints (e.g. memory requirements), then the sequence of input patterns should be randomized with each epoch to avoid capturing inadvertent autocorrelations in the data.

The back-propagation algorithm guarantees that total error in the training set will continue to decrease as the number of epochs increases. With each epoch, the weights are modified to decrease the error on the training patterns. As training progresses, the amount of change in the error function becomes smaller. Convergence occurs when the change in the error function is less than a specified threshold.

However, training with repeated applications of the same data set may result in the phenomenon of overtraining. Similar to the concept of overfitting discussed in the 'Internal Architecture' section above, overtraining occurs when the ANN attempts to exactly fit the limited set of points and loses its ability to interpolate between those points (Hecht-Nielsen 1990). As training progresses, there is always an intermediate stage at which the algorithm reaches a good balance between accurately fitting the training set examples and still providing a reasonably good interpolation capability.

According to Hecht-Nielsen (1990), the back-propagation model starts with a very flat surface and does a good job at interpolating between points on the surface.

However, the training data may be incomplete, or may contain spurious and misleading regularities due to sampling errors and/or omissions. At some point in later stages of learning, the network starts to take advantage of these idiosyncrasies in the training data. As training progresses, the surface becomes convoluted as it attempts to 'better' fit the set of training points and the ability to interpolate between the points diminishes.

This problem is demonstrated in Figure 7. An ANN was developed for time-series prediction. The first 48 periods were used to train the network, and the last 12 periods were used for prediction. After 10 iterations, the ANN results resembled the average of the data points (heavy-dashed line). The 'best' predictive ability occurred after 300 iterations. Notice that the best-fit line does not exactly fit the data in either the training or prediction periods. After 3000 iterations, the ANN almost exactly fits the data points in the training periods. However, the fit in the prediction period is very poor, and is clearly worse than the 300-iteration results.

The problem created from overtraining is determining when sufficient iterations have been accomplished to achieve the desired prediction accuracy. The 'best' predictive performance should be obtained with the set of weights that produces the minimum value for the error function in the testing set of data. Iterations beyond that point will not improve predictive performance. However, since the testing set of data is a holdout sample used to evaluate the performance of the network, it cannot be used to determine when to stop training. Thus, performance is evaluated on independent set of known observations, termed an evaluation set, after each epoch. The error on this evaluation set will tend to decrease in step with the training error as the network generalizes from the training data to the underlying function. As the network begins to 'overtrain' to the training set of data, the error for the evaluation set starts to increase again even though the training error continues to decrease. The algorithm should be

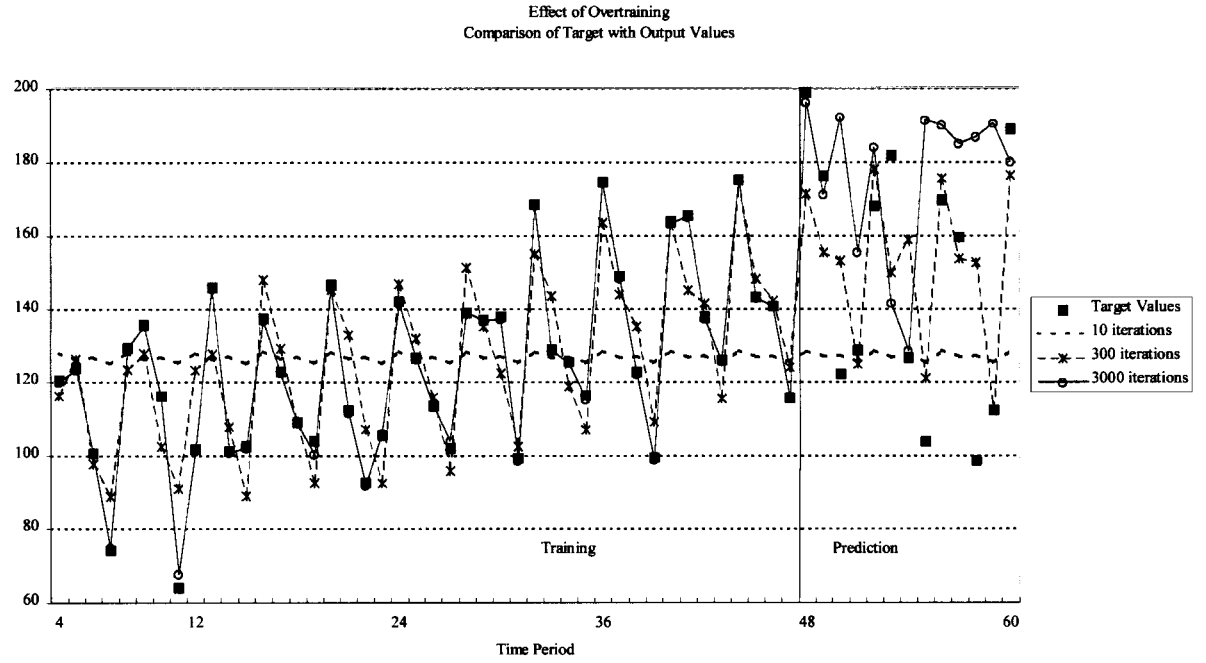

Figure 7 Effect of overtraining on time series data

terminated when performance on the evaluation set begins to deteriorate.

This technique may not be practical when only a small amount of data is available since validation data cannot be used for training (Reed 1993). An alternative approach that has been applied by Chung and Tam (1992) and Coakley (1995) is the use of a jackknife method to derive the evaluation set of data. In Chung and Tam (1992), examples were excluded from the training sample one at a time, with the remaining examples serving as the training set. This process is repeated n times (where n is the number of observations in the training set) and the average misclassification rate is calculated. Lachenbruch (1967) has shown that this method provides an unbiased estimate for the probability of misclassification. An extension of this approach is to use *k*-fold cross-validation when the data set is divided into *k* equal parts. The ANN would be trained *k* times, using *n-k* data points for training and *k* data points for evaluation. The results of the *k* models are then averaged. Coakley (1995) used a different approach of drawing a random sample from the training set at the beginning of each epoch.

CONCLUSIONS

The non-linear characteristics of ANNs make them a promising alternative to traditional linear and parametric methods. The challenge faced by ANN researchers is that there are no formal theories for determining, *a priori*, the optimal network model. Development of an ANN model is a complex process that is currently more art than science. Thus, for an ANN applied to a specific problem, experiments must be conducted to determine the performance differences between alternative models.

Based on an extensive survey of ANNs applied to accounting and finance problems, the following guidelines are suggested:

(1) Determine the type of research question and comparable parametric model that would be used. If the properties of the data do not match the distributional assumptions required by the parametric model, or there is some evidence of possible non-linearity, then development of an ANN model is warranted
(2) The most widely implemented learning algorithm is back-propagation. For classification problems, there are no clear performance differences between back-propagation and other learning algorithms. Thus, the selection will most likely be driven by the availability of alternative learning algorithms within the software package being used. For forecasting problems, the recurrent algorithms appear to improve predictive ability.
(3) There are no clear guidelines on the selection of the error and transfer functions. The most widely used combination is the SSE error function and sigmoid transfer function. It has been suggested that the relative entropy error function is more appropriate for classification problems that produce dichotomous output. With this error function, the hyperbolic tangent transfer function seems most appropriate. Another consideration is robustness of the transfer function. During error propagation, rounding errors could produce negative values for the hidden neurons. The sigmoid function cannot handle these values, while the hyperbolic tangent function can. Finally, some researchers recommend the sigmoid function for classification problems that involve learning about average behavior, and the hyperbolic tangent function if the problem involves learning about deviations from the average (such as forecasting).
(4) In terms of architecture, selection of the number of nodes in the input and output layers is driven by the problem and the nature of the input data. If a classification task, using a single output node forces classification along an ordinal scale, while using multiple output nodes allows duplicate and mis-classifications.
(5) Determining the number of layers, and the number of nodes per layer, is still part of the 'art' of neural networks.

- Assess the complexity of the problem and the resulting response surface. Complex response surfaces require larger networks.

- Assess the noise within the data. Smaller networks should be used with noisy data to increase generalization ability and avoid overfitting to the noise.
- If being applied for classification tasks, the number of nodes in the hidden layer should be less than the number of nodes in the input layer.
- Although numerous heuristics have been suggested for determining the number nodes, they do not apply across all the reported studies.
- Some algorithms have been developed that adaptively adjust the architecture during training to overcome the 'trial and error' approach typically associated with back-propagation ANNs. The expansion and pruning algorithms both achieve improved accuracies.

(6) Data-preparation requirements for the ANN model are relatively well defined. The input data should be scaled to match the input side of the selected transfer function, while the specified target values should be scaled to match the output side. Most software packages will perform scaling of the input data, and will automatically generate the initial training weights.

(7) If the ANN is used to model time series data, then data sets should be presented in order, and weights should be adjusted after each data set. If the ANN is used for classification tasks, then the entire data set should be presented before weights are adjusted.

(8) Obtaining a sufficient sample size may be problematic since three separate data sets are required (training, evaluation, and testing). Jackknife and cross validation methods have been shown to be effective techniques that allow use of smaller data sets.

The availability of sample data appears to be a driving factor in the ANN research performed to date in the accounting and finance areas. Many of the cited articles applied ANNs as an extension of a previous statistical model-based study, or used publicly available stock market data. Of the 123 studies cited by Velido *et al.* (1999), only 9 were considered to be examples of ANNs applied to a 'real-world' case. The commercial applications of ANNs in the private sector (see Brown *et al.*, 1995) seem to focus on a wider range of problems. We believe the limited applications in published academic research are due to lack of access to appropriate data sets.

It appears that ANNs show promise in problem areas where traditional statistical techniques are inadequate. ANNs provide a non-linear dimension that captures the underlying representations within the data sets. Thus, future applications should focus on problem areas where non-linear relationships are believed to exist within the data.

However, much more research is needed in the following areas:

- How do we systematically build an appropriate ANN model for specific types of accounting and finance problems? Although we have attempted to derive some guidelines from the literature, we are still generalizing across broad problem categories. Replication of the studies using different algorithms, architectures and data preparation techniques is needed to refine these guidelines within specific application areas.
- Are any of the alternative training methods and algorithms 'best' for a specific type of accounting and finance problem? Many of the studies are first attempts to demonstrate the applicability of a technique within a problem area. It is difficult to generalize from these limited experiments. Again, replication of the experiments is needed.

References

Altman E. 1968. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance* **23**: No. 4, 589–609.

Altman E. 1983. *Corporate Financial Distress—A Complete Guide to Predicting, Avoiding and Dealing with Bankruptcy*. John Wiley: New York.

Anandarajan M, Anandarajan A. 1998. A comparison of machine learning techniques with a qualitative response model for auditor's going concern reporting. *Expert Systems with Applications* **16**: 385–392.

Anderson JA, Rosenfeld E. 1988. *Neurocomputing: Foundations of Research*. MIT Press: Cambridge, MA.

Armstrong W, Dwelly A, Liang J, Lin D, Reynolds S. 1991. Learning and generalization in adaptive logic networks. *Artificial Neural Networks, Proceedings of the 1991 International Conference on Artificial Neural Networks*. 1173–1176.

Austin M, Looney C, Zhuo J. 1997. Security market timing using neural networks. *The New Review of Applied Expert Systems* **3**: 3–14.

Babad YM, Balachandran BV. 1993. Cost driver optimization in activity-based costing. *The Accounting Review* **68**: No. 3, 563–575.

Back B, Laitinen T, Sere K. 1996. 'Neural networks and genetic algorithms for bankruptcy predictions. *Expert Systems with Applications* **11**: No. 4, 407–413.

Barniv R, Agarwal A, Leach R. 1997. Predicting the outcome following bankruptcy filing: a three-state classification using neural networks. *International Journal of Intelligent Systems in Accounting, Finance and Management* **6**: 177–194.

Baum EB, Haussler D. 1989. What size net give valid generalization. *Neural Computation 1*. MIT Press: Cambridge, MA; 151–160.

Bell TB. 1997. Neural Nets or the Logit Model? A comparison of each model's ability to predict commercial bank failures. *Intelligent Systems in Accounting, Finance and Management* **6**: 249–264.

Bell TB, Ribar GS, Verchio J. 1990. Neural nets vs. logistic regression: a comparison of each model's ability to predict commercial bank failures. *Proceedings of the 1990 Deloitte Touche/University of Kansas Symposium on Auditing Problems*. 29–53.

Bell TB, Szykowny S, Willingham J. 1993. Assessing the likelihood of fraudulent financial reporting: a cascaded logic approach. Working Paper, KPMG Peat Marwick: Montvale, NJ.

Bergerson K, Wunsch DC. 1991. A commodity trading model based on a neural network-expert system hybrid. *Proceedings of the IEEE International Conference on Neural Networks*, Seattle, WA. I289–I293.

Boritz JE, Kennedy DB. 1995. Effectiveness of neural network types for prediction of business failure. *Expert Systems with Applications* **9**: No. 4, 503–512.

Boritz JE, Kennedy DB, Albuquerque AM. 1995. Predicting corporate failure using a neural network approach. *Intelligent Systems in Accounting, Finance and Management* **4**: No. 2, 95–111.

Boucher J. 1990. *Artificial neural systems and the prediction of corporate failure*. Master's Project, School of Accountancy, University of Waterloo.

Brockett PL, Cooper WW, Golden LL, Pitaktong U. 1994. A neural network method for obtaining an early warning of insurer insolvency. *The Journal of Risk and Insurance* **61**: No. 3, 402–424.

Brown CE, Coakley JR, Phillips ME. 1995. Neural networks enter the world of management accounting. *Management Accounting* **LXXVI**: No. 11, May, 51–57.

Brown M, An PC, Harris CJ, Wang H. 1993. How biased is your multi-layer perceptron? *Proceedings of the 1993 World Congress on Neural Networks*. 507–511.

Charitou A, Charalambous C. 1996. The prediction of earnings using financial statement information: empirical evidence with Logit models and artificial neural networks. *International Journal of Intelligent Systems in Accounting, Finance and Management* **5**: 199–215.

Chen SK, Mangiameli P, West D. 1995. The comparative ability of self-organizing neural networks to define cluster structure. *Omega* **23**: No. 3, 271–279.

Choi HR, Kim W, An SY. 1997. Recurrent and decomposed neural network-based hotel occupancy forecasting. *The New Review of Applied Expert Systems* **3**: 121–136.

Chung HM, Tam KY. 1992. A comparative analysis of inductive-learning algorithms. *Intelligent Systems in Accounting, Finance and Management* **2**: No. 1, 3–18.

Coakley JR. 1995. Using pattern analysis methods to supplement attention-directing analytical procedures. *Expert Systems with Applications* **9**: No. 4, 513–528.

Coakley JR, Brown CE. 1993. Neural networks applied to ratio analysis in the analytical review process. *International Journal of Intelligent Systems in Accounting, Finance and Management* **2**: No. 1, 19–39.

Coakley JR, McFarlane DD, Perley WG. 1992. Alternative criteria for evaluating artificial neural network performance. TIMS/ORSA Joint National Meeting (Orlando, FL).

Coats PK, Fant LF. 1993. Recognizing financial distress patterns using a neural network tool. *Financial Management* **22**: No. 3, 142–155.

Collins E, Ghosh S, Scofield C. 1988. An application of a multiple neural-network learning system to emulation of mortgage underwriting judgments. *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 2. 459–466.

Curram SP, Mingers J. 1994. Neural networks, decision tree induction and discriminant analysis: an empirical comparison. *Journal of the Operational Research Society* **45**: No. 4, 440–450.

Cybenko G. 1988. Continuous valued neural networks with two hidden layers are sufficient. Technical Report, Department of Computer Science, Tufts University, Medford, MA.

Cybenko G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems* **2**: 303–314.

Daniels H, Kamp B, Verkooijen W. 1997. Modeling non-linearity in economic classification with neural networks. *International Journal of Intelligent Systems in Accounting, Finance and Management* **6**: 287–301.

de Villiers J, Barnard E. 1992. Backpropagation neu-

ral nets with one and two hidden layers. *IEEE Transactions on Neural Networks* **4**: No. 1, 136–141.
Diamantaras KI, Kung SY. 1996. *Principal Component Neural Networks: Theory and Applications*. Wiley: New York.
Duliba KA. 1991. Contrasting neural nets with regression in predicting performance in the transportation industry. *Proceedings of the 24th Annual Hawaii International Conference on Systems Sciences*, Vol. IV. 163–170.
Dutta S, Shekhar S. 1988. Bond rating: a non-conservative application of neural networks. *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 2. 433–450.
Elman JL. 1988. *Finding Structure in Time*. Technical Report, CRL #88-1, Center for Research in Language, University of California, San Diego.
Etheridge HL, Sriram RS. 1997. A comparison of the relative costs of financial distress models: artificial neural networks, Logit and multivariate discriminant analysis. *International Journal of Intelligent Systems in Accounting, Finance and Management* **6**: 235–248.
Fahlman SE, Lebiere C. 1990. *The Cascade-Correlation Learning Architecture*, Technical Report: CMU-CS-90-100, Carnegie-Mellon University.
Fanning KM, Cogger KO. 1994. A comparative analysis of artificial neural networks using financial distress prediction. *International Journal of Intelligent Systems in Accounting, Finance and Management*, **3**: No. 4, 241–252.
Fanning KM, Cogger KO. 1998. Neural network detection of management fraud using published financial data. *International Journal of Intelligent Systems in Accounting, Finance and Management* **7**: 21–42.
Fanning K, Cogger KO, Srivastava R. 1995. Detection of management fraud: A neural network approach. *Intelligent Systems in Accounting, Finance and Management* **4**: No. 2, 113–126.
Goldberg DE. 1994. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, **37**: No. 3, 113–119.
Green BP, Choi JH. 1997. Assessing the risk of management fraud through neural network technology. *Auditing: A Journal of Practice and Theory* **16**: No. 1, 14–28.
Grudnitski G, Osburn L. 1993. Forecasting S&P and gold futures prices: An application of neural networks. *The Journal of Futures Markets* **13**: No. 6, 631–643.
Grudnitski G, Do AQ, Shilling JD. 1995. A neural network analysis of mortgage choice. *Intelligent Systems in Accounting, Finance and Management* **4**: No. 2, 127–135.
Hansen JV. 1998. Comparative performance of backpropagation networks by genetic algorithms and heuristics. *International Journal of Intelligent Systems in Accounting, Finance and Management* **7**: 69–79.
Hansen JV, Messier WF. 1991. Artificial neural networks: foundations and application to a decision problem. *Expert Systems with Applications* **3**: No. 1, 135–141.
Hart JA. 1992. Using neural networks for classification tasks—some experiments on data sets and practical advice. *Journal of the Operational Research Society* **43**: 215–226.
Hecht-Nielsen R. 1990. *Neurocomputing*. Addison-Wesley: Reading, MA.
Hertz J, Krogh A, Palmer RG. 1991. *Introduction to the Theory of Neural Computation*. Addison-Wesley: Reading, MA.
Hoptroff R, Hall T, Bramson MJ. 1991. Forecasting economic turning points with neural nets. *Proceedings of the International Joint Conference on Neural Networks*. IEEE Service Center: Piscataway, NJ, II.347–II.352.
Hornik K, Stinchcombe M, White H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* **2**: 359–366.
Hoyt RE, Lay CM. 1995. Neural network cost estimates for heart bypass surgery. *Expert Systems with Applications* **9**: No. 4, 529–542.
Huang, CS, Dorsey RE, Boose MA. 1994. Life Insurer financial distress prediction: a neural network model. *Journal of Insurance Regulation* 131–167.
Jensen HL. 1992. Using neural networks for credit scoring. *Managerial Finance* **18**: No. 6, 15–26.
Jhee WC, Lee JK. 1993. Performance of neural networks in managerial forecasting. *Intelligent Systems in Accounting, Finance and Management* **2**: No. 1, 55–71.
Jo H, Han I, Lee H. 1997. Bankruptcy prediction using case-based reasoning, neural networks, and discriminate analysis. *Expert Systems with Applications* **13**: No. 2, 97–108.
Kamijo K, Tanigawa T. 1990. Stock price pattern recognition: a recurrent neural network approach. *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vol. I. (San Diego, CA, 215–221.
Kimoto R, Asakawa K, Yoda M, Takeoka M. 1990. Stock market prediction system with modular neural networks. *Proceedings of the International Joint Conference on Neural Networks*, Vol. 1. 1–6.
Kohara K, Ishikawa T, Fukuhara Y, Nakamura Y. 1997. Stock price prediction using prior knowledge and neural networks. *International Journal of Intelligent Systems in Accounting, Finance and Management* **6**: 11–22.
Kolmogorov AN. 1963. On the representation of continuous function of many variables by superposition of continuous functions of one variable and addition. *American Mathematical Society Translation* **28**: 55–59.
Kryzanowski L, Galler M, Wright DW. 1993. Using artificial neural networks to pick stocks. *Financial Analysts Journal* **49**: No. 4, 21–27.
Kwon YS, Han I, Lee KC. 1997. Ordinal pairwise partitioning (OPP) approach to neural networks

training in bond rating. *International Journal of Intelligent Systems in Accounting, Finance and Management* **6**: 23–40.
Lachenbruch P. 1967. An almost unbiased method of obtaining confidence intervals for the probability of misclassification in discriminant analysis. *Biometrics* December, 639–645.
Lapedes A, Farber R. 1987. *Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling*. Los Almos National Laboratory Report LA-UR-87-2662.
Lawrence J. 1991. *Introduction to Neural Networks*. California Scientific Software: Grass Valley, CA.
Lawrence J, Andriola P. 1992. Three-step method evaluates neural networks for your application. *EDN* 93–100.
Lenard MJ, Alam P, Madey GR. 1995. The application of neural networks and a qualitative response model to the auditor's going concern uncertainty decision. *Decision Sciences* **26**: No. 2, 209–227.
Levitan A, Gupta M. 1996. Using genetic algorithms to optimize the selection of cost drivers in activity-based costing. *International Journal of Intelligent Systems in Accounting, Finance and Management* **5**: No. 3, 129–146.
Liang TP, Chandler JS, Han I, Roan J. 1992. An empirical investigation of some data effects on the classification accuracy of probit, ID3, and neural networks. *Contemporary Accounting Research* **9**: No. 1, 306–328.
Lippman RP. 1987. An introduction to computing with neural nets. *IEEE ASSP Magazine* April, 4–22.
Loh L. 1992. Financial characteristics of leveraged buyouts. *Journal of Business Research* **24**: 241–252.
Maher JJ, Sen TK. 1997. Predicting bond ratings using neural networks: a comparison with logistic regression. *International Journal of Intelligent Systems in Accounting, Finance and Management* **6**: 59–72.
Malakooti B, Zhou YQ. 1994. Feedforward artificial neural networks for solving discrete multiple criteria decision making problems. *Management Science* **40**: No. 11, 1542–1561.
Markham IS, Ragsdale CT. 1995. Combining neural networks and statistical predictions to solve the classification problem in discriminant analysis. *Decision Sciences* **26**: No. 2, 229–242.
Markham IS, Rakes TR. 1998. The effect of sample size and variability of data on the comparative performance of artificial neural networks and regression. *Computers and Operations Research.* **25**: No. 4, 251–263.
Marques L, Hill T, Worthley R, Remus W. 1991. Neural network models as an alternative to regression. *Proceedings of the 24th Annual Hawaii International Conference on Systems Sciences*, Vol. IV. 129–146.
Masters T. 1994. *Practical Neural Networks*. Academic Press: New York.
Niles L, Silverman H, Tajchman G, Bush M. 1989. How limited training data can allow a neural network classifier to outperform an 'optimal' statistical classifier. *IEEE Proceedings 1989 International Conference* 1. Glasgow, Scotland, 17–20.
Odom M, Sharda R. 1992. A neural network model for bankruptcy prediction. In *Neural Networks in Finance and Investing*, Trippi RR, Turban (eds). Probus Publishing: Chicago, IL.
Ohlson J. 1980. Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research* **18**: No. 1, 109–131.
O'Leary DE. 1998. Using neural networks to predict corporate failure. *International Journal of Intelligent Systems in Accounting, Finance and Management* **7**: 187–197.
Pal SK, Mitra S. 1992. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks* **3**: No. 5, September, 683–697.
Palepu KG. 1986. Predicting takeover targets: A methodological and empirical analysis. *Journal of Accounting and Economics* **8**: 3–35.
Patuwo E, Hu MY, Hung MS. 1993. Two-group classification using neural networks. *Decision Sciences* **24**: No. 4, 825–845.
Proto VW, Fogel DB, Fogel LJ. 1995. Alternative neural network training methods. *IEEE Expert* **10**: No. 3, 16–22.
Raghupathi W, Schkade LL, Raju BS. 1991. A neural network application for bankruptcy prediction. *Proceedings of the 24th Annual Hawaii International Conference on Systems Sciences,* Vol. IV. 147–155.
Reed R. 1993. Pruning algorithms—a survey. *IEEE Transactions on Neural Networks* **4**: No. 5, 740–747.
Reilly DL, Collins E, Scofield C, Ghosh S. 1991. Risk assessment of mortgage applications with a neural network system: An update as the test portfolio ages. *Proceedings of the IEEE International Conference on Neural Networks*. Piscataway, NJ, 1991, II479–II482.
Rumelhart DE, Hinton GE, Williams RJ. 1986. Learning representations by back-propagating errors. *Nature* **323**: 533–536.
Rumelhart DE, McClelland JL. 1986 *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press: Cambridge, MA.
Salchenberger LM, Cinar EM, Lash NA. 1992. Neural networks: a new tool for predicting thrift failures. *Decision Sciences*, **23**: No. 4, 899–916.
Schocken S, Ariav G. 1994. Neural networks for decision support: Problems and opportunities. *Decision Support Systems* **11**: 393–414.
Serrano-Cinca C. 1996. Self organizing neural networks for financial diagnosis. *Decision Support Systems*, **17**: 227–238.
Shanker M, Hu MY, Hung MS. 1996. Effect of data standardization on neural network training. *Omega, International Journal of Management Science* **24**: No. 4, 385–397.
Sharda R. 1994. Neural networks for the MS/OR

analyst: an application bibliography. *Interfaces* **24**: No. 2, March–April, 116–130.
Shepanski JF. 1988. Fast learning in artificial neural systems: Multilayer Perceptron training using optimal estimation. *Proceedings of the IEEE Second International Conference on Neural Networks*, Vol. 4. San Diego, CA, July.
Stornetta WS, Huberman BA. 1987. An improved three-layer backpropagation algorithm. *Proceedings of the IEEE First International Conference on Neural Networks*. San Diego, CA.
Subramanian V, Ming SH, Hu MY. 1993. An experimental evaluation of neural networks for classification. *Computers and Operations Research* **20**: No. 7, 769–782.
Surkan A, Singleton J. 1991. Neural networks for bond rating improved by multiple hidden layers. *Proceedings of the IEEE International Conference on Neural Networks*, Vol. II. 163–168.
Tabachnick BG, Fidell LS. 1983. *Using Multivariate Statistics*. Harper & Row: New York, NY.
Tam KY, Kiang MY. 1992. Managerial applications of neural networks: the case of bank failure predictions. *Management Science* **38**: No. 7, 926–947.
Taudes A, Natter M, Trcka M. 1998. Real option valuation with neural networks. *International Journal of Intelligent Systems in Accounting, Finance and Management* **7**: 43–52.
Torsun IS. 1996. A neural network for a loan application scoring system. *The New Review of Applied Expert Systems* **2**: 47–62.
Trippi RR, DeSieno D. 1992. Trading equity index futures with a neural network. *The Journal of Portfolio Management* **19**: No. 1, 27–33.
Trippi RR, Turban E. (eds). 1992. *Neural Networks in Finance and Investing*. Probus Publishing: Chicago, IL.
Udo G. 1993. Neural network performance on the bankruptcy classification problem. *Computers and Industrial Engineering* **25**: 377–380.
Velido A, Lisboa PJG, Vanghan J. 1999. Neural networks in business: a survey of applications (1992–1998). *Expert Systems with Applications* **17**: 51–70.
Waite T, Hardenbergh H. 1989. Neural nets. *Programmer's Journal* **7**: No. 3, 10–22.
Wasserman PD. 1989. *Neural Computing: Theory and Practice*. Van Nostrand Reinhold: New York.
Wessels L, Barnard E. 1992. Avoiding false local minima by proper initialization of connections. *IEEE Transactions on Neural Networks* **3**, No. 6, 899–905.
White H. 1988. Economic prediction using neural networks: the case of IBM daily stock returns. *Proceedings of the IEEE International Conference on Neural Networks* Vol. II. 451–458.
Widrow B, Rumelhart DE, Lehr, MA. 1994. Neural networks: applications in industry, business and science. *Communications of the ACM* **37**: No. 3, 93–105.
Wilson RL, Sharda R. 1994. Bankruptcy prediction using neural networks. *Decision Support Systems* **11**: 545–557.
Wong BK, Bodnovich TA, Selvi Y. 1997. Neural network applications in business. A review and analysis of the literature (1988–95). *Decision Support Systems* **19**: 301–320.
Wong FS. 1991. A 3D neural network for business forecasting. *Proceedings of the 24th Annual Hawaii International Conference on Systems Sciences*, Vol. IV. 113–123.
Yoon Y, Swales G. 1991. Predicting stock price performance: a neural network approach. *Proceedings of the 24th Annual Hawaii International Conference on Systems Sciences*, Vol. IV. 156–162.
Zhang G, Patuwo BE, Hu MY. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* **14**: 35–62.