Neural Networks: Proposed Methodology for Applications Development

Thomas A. Bodnovich[1)]

[1)]Department of Computer Science and Information Systems, Youngstown State University
Youngstown, OH 44555, USA
[1)]Department of Administrative Sciences, Kent State University
Kent, OH 44242, USA
tom@cis.ysu.edu

Abstract

A review of published business neural network applications research is conducted to determine what parts of development methodologies were used or could have been used in designing, constructing, and deploying the networks. This review is limited to articles describing either the development of a prototype or an actual neural network application. Based upon this review and upon accepted development processes for creating NNs, a methodology for the complete development of neural network applications is proposed. The methodology provides a systematic view of the steps involved in the development of a successful NN business application which will allow for (1) better analysis of existing NN applications, (2) identification and development of new applications that can benefit from NN technology, and (3) development of improved NN applications.

1. Introduction

In recent years many great advances have been made in computer artificial neural networks (NNs). These advances extend the pioneering work of David Rumelhart and James McLelland [54] on back propagation NNs, Teuvo Kohonen [41] on self-organizing map NNs, and John Hopfield [31,32] on the use of NN as associative memory. This progress has contributed to the development of numerous scientific as well as business applications [77]. Furthermore, a process for developing NNs has become accepted and it is somewhat analogous to the structured design methodoloies used in designing other types of information systems [67]. The goal of this paper is to propose a methodology which expands upon this existing process of NN development. This methodology can then be used to facilitate the development of NN business applications. Hopefully, such a methodology can lead to greater successes in these applications and provide insights about the development process. This paper first discusses the importance of this research and the methodology used to conduct this study, and then it presents a methodology for implementing NN solutions.

2. Value of Research

NNs have been constructed for solving problems in a wide variety of business application areas such as accounting, auditing, finance, human resources, information systems, marketing/distribution, and production/operations. It has been demonstrated that many of these NNs improve the accuracy, consistency, effectiveness, efficiency, and/or flexibility of these applications [78]. This project will increase our understanding NNs in business problem solving by examining past research from the perspective of a NN development methodology. This will provide a systematic view of the steps involved in the development of a successful NN business application which will allow for (1) better analysis of existing NN applications, (2) identification and development of new applications that can benefit from NN technology, and (3) development of improved NN applications.

3. Prior Research

Villegas and Eberts [70] identified five steps for developing a NN. Chu and Widjaja [11] demonstrated how technical design decisions could be determined for creating a forecasting NN. Hwarng [34] investigated training strategies and data selection for recognizing cyclic data. Rangwala and Dornfield [52] and Knapp and Wang [40] investigated strategies for reducing training time. Lee and Jhee [42] investigated adding noise to input data sets. Bahrami et al. [2], Chu and Holm [12], Cui and Shin [16], Oh et al. [49], and Taha et al. [64] proposed a new NN framework or architecture for an application. While many researchers have proposed and investigated various steps in designing, constructing, and deploying a specific neural network, none have investigated or proposed expanding upon a current development process to create a more complete or comprehensive development methodology for neural network applications.

4. Research Methodology

The research methodology is based upon the technique employed by Wong et. al. [77,78] in earlier NN studies. The ABI/INFORM database was searched back to 1971 to access abstracts of articles from 800 different business-related journals using the descriptors "neural network" and "neural networks". Similarly, the *Business Periodical Index* (BPI) was searched back to 1971 with the same search argument. The BPI indexes 340 business-related journals. There is some overlap between the BPI and ABI/INFORM, so only articles not included in the ABI database will be considered for further review at this step.

Although the ABI and BPI searches provided access to a very diverse collection of journals, some additional journals were also searched. These supplementary journals were selected for review if they are known to publish articles on neural networks, artificial intelligence, or if they are important to the field of MIS [23,28,77]. These additional journals include AI Expert, Artificial Intelligence, Communications of the ACM, DATABASE, Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, IEEE Expert, IEEE Transactions on Neural Networks, IEEE Transactions on Software Engineering, IEEE Transactions on Systems, Man, and Cybernetics, Information Systems Research, International Journal of Man Machine Studies, International Journal of Production Research, Journal of Management Information Systems, Journal of Management Systems, and Omega: The International Journal of Management Science.

Articles from ABI, BPI, and the additional journals were then reviewed to determine if they describe either the development of a prototype or an actual neural network application. References from these articles are also examined. Those articles which satisfied this criteria were then examined to determine what parts (if any) of a development methodology was used (or could have been used) in creating the NN.

5. Neural Network Development Methodology

The design, construction, and implementation of over 200 business neural networks were examined to gain an understanding of how a development methodology may assist practitioners and researchers in the creation of neural networks. Applying the results of this examination to a conventional NN development process presented by Turban [67], the following methodology is proposed.

5.1 Identify problem as suitable for a neural network solution

All problems are not equally good candidates for neural network solutions. Obviously well structured problems with explicit solution steps are better suited for traditional computer program solutions. But when reasoning or inference is required, then some type of artificial intelligence solution may be appropriate. Several possible choices include expert systems, genetic algorithms, and neural networks. For problems which have no sample solutions and no well-defined rules, but which have known characteristics of good solutions, genetic algorithms may be a suitable technology[25]. Other problems which require reasoning and for which rules may be well-defined are probably better suited for expert system solutions [60]. Lastly, problems which are suited for intuitive decision making for which rules do not or can not be identified may be appropriate for neural network solutions [60]. NNs can uncover relationships when provided with sufficient data and thereby provide new insights for problems [5]. They can also continue to learn after being deployed in a production environment. In a dynamic environment, this continual learning may make NNs superior to causal models or static techniques (i.e., linear regression or discriminant analysis).

Applying the well established decision processing phase model of Simon [59], NNs seem better suited to the intelligence and choice phases. Schocken and Ariav [57] reveal NNs are very well suited for the intelligence phase of the decision process. In this phase, many NN applications have successfully identified problems, such as operations process fault diagnosis [62], machine fault diagnosis [7], computer vision inspection [9], and bankruptcy prediction [65]. Wong et al. [78] report that almost one third of NN applications support the choice phase of decision making. Such applications include selecting optimal or near optimal job shop schedules [56,58] and LIFO/FIFO accounting choice [43]. Whereas there are many examples in the literature of NNs supporting the intelligence and choice phases of decision making, there are relatively few examples of NNs supporting the design phase.

NN learning algorithms are inductive, and therefore require masses of data and repetitive examples [57]. Valasco and Rowe [68] used 2000 values for training a BP NN for analyzing quality control charts, while Jain and Nag [35] used 552 new issues to train a NN for pricing IPOs. Villegas and Eberts [70] used 4182 training patterns, Wong and Long [79] used 400 training patterns, and Altman et al. [1] used 808 examples. Because of the need for large amounts of data, some problems which appear to be potential candidates for NN based solutions must be eliminated because of an insufficient amount of data.

Lastly, sometimes simpler is better. Barr and Mani [3] report that a NN becomes more effective as it becomes more task specific. They investigate the use of NNs to manage investments and suggest that a NN which predicts a market's trend and amplitude will be less effective than one which only predicts its trend.

5.2 Identify relevant input fields

Wang et al. [72] use linear regression to identify relevant input fields. But since NNs can model nonlinear relationships [53,22], linear regression may not be an appropriate technique for identifying all inputs. Other techniques for recognizing appropriate input fields are genetic algorithm preprocessing [47] and non-parametric regression.

5.3 Collect and prepare the data

NNs require numeric quantities as inputs. Values such as *greater, less, yes, no, A, B,* or *C* must therefore be converted to numbers. Numeric input data may also require some type of transformation. Many researchers suggest normalizing input data [80].

Kaastra and Boyd [37] use mean/standard deviation scaling to scale all variables between 0 and 1. Any observations which are at least two standard deviations less than the mean are mapped to 0, similarly any of those observations which are at least two standard deviations greater than the mean are mapped to 1. This scaling reduces outlier effects and provides a more uniform distribution. Wong and Long [79] achieve optimal performance by pre-processing input data to remove linear trends, seasonal variations, and outliers.

5.4 Divide data into training, testing, and validation sets

Input data for developing a NN is divided into two or three sets. A training set is used to train the NN. The weights of the NN are adjusted as it learns or discovers the relationships in this data. Later, the NN is presented with the testing set. The data in this set is used to evaluate how well the NN will perform with new or unseen data. Lastly, some developers create a validation set. This set is the combination of the training and testing sets. It may also be used to measure the performance of the network.

There is no single procedure for determining the proportion of data that should be placed in the training and testing sets. Many researchers use 50 percent of the data for training the NN and 50 percent for testing [83,24,35], although Jensen [36] used 60% for training and 40% for testing, and Wong and Long [79] reported using 93% for training and only 7% for testing. Based upon published NN research, a good rule of thumb would be to include at least 50% of the data in the training set.

Data in the testing set should be representative of the population. Data in the training set, however, should be selected so each concept has roughly an equal number of examples, regardless of the distribution of each concept in the population. Wilson and Sharda [76] demonstrated that such equal weighting of concepts resulted in NNs with superior differentiation. They suggest smoothing the training set distribution, regardless of the population distribution to obtain a better model.

5.5 Select a paradigm for the NN

Three components determine a NN's paradigm: the architecture, the transfer function, and the learning rule [8}. The architecture specifies the structure of a NN. It is characterized by the number of hidden layers, the number of processing elements in each layer, and the way that the processing elements are interconnected. The transfer function governs the production of an output by a processing element. Lastly, the learning rule specifies how the weights of a NN are updated.

A commonly used transfer function is the sigmoid defined as equation (1)

$$g(a) = \frac{1}{1 + e^{-a}} \tag{1}$$

Cybenko [17}, Funahashi [21], White [75] and Barron [4] have proved that NNs with at least one hidden layer using this transfer function can fit any function and its derivative. Another common transfer function is the hyperbolic tangent defined as equation (2)

$$g(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \tag{2}$$

Bishop [6] shows that both of these functions are essentially equivalent with the application of a linear transformation, but notes that the hyperbolic tangent converges faster with many training algorithms.

The learning algorithm greatly affects the performance of the network [1] and more than 100 different learning rules are available [67]. For a taxonomy of these algorithms, consult Lippman [45]. The learning rules can be classified into two broad classes: supervised learning and unsupervised learning. Supervised learning is used when the desired outputs are known. The most frequently used supervised learning rule is backpropagation (BP) [37,2]. Altman et al. [1] and Coats and Fant [13] used BP for financial distress forecasting and Pugh [50,51], Smith and Dagli [61], and Velasco and Rowe [68] used BP for process control. This learning rule allows a NN with a sufficient number of hidden neurons to

satisfactorily approximate any measurable function [33] by minimizing the squared error between the computed and desired outputs.

When the desired outputs are unknown, unsupervised learning is used. This type of learning rule lets the NN organize itself to identify different groupings of inputs. Commonly used unsupervised learning rules are adaptive resonance theory (ART) and Kohonen self-organizing maps (SOM). ART has frequently been used for problems such as part family machine grouping [38,44,69]. SOM has been shown to be superior to non-NN clustering techniques [10].

5.6 Design the NN

The final step in selecting the NN paradigm is determining an appropriate number of hidden layers and hidden nodes for the network structure. There is no single procedure for determining the optimal number of hidden nodes. Several researchers report selecting the number of hidden nodes based upon systematic experimentation [35,81,83] with Yoon et al. [83] reporting that NN performance improves with the addition of each hidden node up to a certain limit, and then performance begins to degrade. Other researchers suggest a rule of thumb for determining the number of hidden nodes. Some of these heuristics are based upon a ratio of hidden nodes to input and/or output nodes. Salchenberger et al. [55] recommends that the hidden layer should contain 75% as many nodes as the input layer, Jensen [36] suggests that the hidden layer should have 50% of the total number of neurons in the input and output layers, Eberhat and Dobbins [19] propose that the number of hidden units in a layer should be roughly the square root of the number of input and output units, and Hecht-Nielson [29] suggests 2N + 1 hidden nodes for a single hidden-layer network with N inputs. Other heuristics use a ratio of the number of connections, which increase with additional hidden units, to the number of training patterns. Villegas and Eberts [70] suggest that the number of training patterns should be at least four times the number of connections between input and hidden units, while Kaastra and Boyd [37] report that overfitting is likely when the number of training patterns is not greater than two times the number of weights in a NN.

Table 1 shows the tradeoffs to be made when considering an appropriate number of hidden nodes.

Table 1 Number of Hidden Node Tradeoffs

Fewer hidden nodes	many hidden nodes
< ——————————————— >	
High training error	increased training time
Less training data required	poor generalization
	overfitting

Just as there is no single rule for selecting the number of hidden nodes, there is also no one procedure for determining the number of hidden layers. Jensen [36] reports that a two hidden layer NN converged faster than a single hidden-layer NN for a credit scoring application. Altman et al. [1] achieves an overall classification accuracy between 92% and 98% with multiple hidden layer NNs. These networks are constructed of simpler neural networks and interestingly these simpler networks have only moderate success on their own. But while some researchers recommend multiple hidden layers, many others suggest that only one hidden layer should be used [18,20,55].

5.7 Select the implementation environment and construct the NN

Neural networks can be developed with either a general purpose programming language, such as C or Pascal, or with a neural network development tool, such as NeuralWorks Professional Plus or BrainMaker. A special purpose tool will ease the development process, but may place some constraints on the developer. For example, Predict by NeuralWare completely automates the development and implementation of a NN, but does not allow modifications to the finished network. On the other hand, using a programming language will allow greater flexibility at the expense of automation and time.

In an effort to compare the overall success of NNs developed with programming languages with those developed with NN tools, the research articles reported in a comprehensive bibliography of NN business applications by Wong et al. [77] are examined. The bibliography includes 25 articles reporting the development of NNs with some type of programming language and 29 articles reporting the development of NNs with a NN development tool. Only 55% of the networks developed with a tool result in improved decision making or an improved technique, while 72% of the networks developed with a traditional programming language yield improved decision making or an improved technique. Based upon these results, programming languages are recommended over NN development tools, particularly when the developer has sufficient time and programming expertise.

5.8 Train the NN with the training set

There is no universal rule for determining when NN training is complete, but NN training should proceed until some type of performance criterion is satisfied. Velasco and Rowe [68] trained a quality control analsysis BP NN until its root mean square error value was .01. Villegas and Eberts [70], Yih et al. [81], and Yoon et al. [83] trained a NN until there was no significant additional decrease in the error. Altman et al. [1] trained for exactly 1000 learning cycles, although they believe that further improvement is likely with additional training.

Sometimes the NN may learn the training data too well. This condition is called overfitting and results in poor generalization with new/unseen data. It is caused by having too many connections, which can be caused by having too many hidden nodes. Several solutions have been proposed for overfitting. These include weight elimination [73] and pruning [48]. Weight elimination eliminates non-significant weights in a NN by utilizing a complexity term which penalizes networks with excessive numbers of small weights. Pruning, on the other hand, eliminates those weights which are below some minimum threshold. Another way to reduce complexity other than eliminating weights is to remove nodes. Yamashina et al. [80] suggest that hidden neurons which are associated with irrelevant output can be removed without affecting performance.

An alternative to reducing the complexity of a NN by removing unnecessary or unimportant weights is to start with a simple network and gradually increase its complexity. Wang [71] uses a dynamic training procedure for a feedforward NN with one hidden layer. Starting with a small number of hidden nodes, additional nodes are added during training whenever the training error exceeds a certain threshold and the decreasing rate of error is lower than another threshold. Wang et al. [72] use another dynamic training procedure, reporting that a decreasing learning rate outperforms a constant rate in their BP NN.

5.9 Test and validate the NN

Some type of testing or validation is required to measure the performance of the NN in terms of making predictions, classifications, clusters, etc. This testing may be performed with only a hold-out test set or it may be performed with a validation set comprised of testing and training data.

Each NN should be tested or validated according to some criteria such as most correct (or fewest incorrect) classifications or closest predictions. Depending upon their importance, some types of outcomes might be weighted more heavily. For example, in a lending decision it would probably be more important to a bank to have very few financially distressed firms improperly classified as good credit risks, even if it means that some strong companies are incorrectly classified as poor credit risks. Rather than evaluating NNs solely as a function of correct or incorrect classifications, developers can also use sensitivity analysis to get a better understanding of their NNs [3].

In addition to testing for the successful outcome of a NN, the developer may also want to test the importance of the inputs. Yoon et al. [82] propose a method for ranking the relative importance of input variables in a NN. For a single-hidden layer NN, the following relative strength, RS_{ji} , between the ith input and jth output variables is computed as equation (3)

$$RS_{ji} = \Sigma(W_{ki}U_{jk})/(\Sigma ABS\{\Sigma(W_{ki}U_{jk})\}) \tag{3}$$

where W_{ki} is the weight between the kth hidden unit and the ith input unit and U_{jk} is the weight between the kth hidden unit and the jth output unit. The developer might consider creating a revised version of the NN without an unimportant input.

5.10 Repeat from steps 5 or 6, creating alternative models

It may be advantageous to create additional NNs since it will allow the developer flexibility in selecting a good solution. Collins and Evans [14] created five models for predicting property values and Creese and Li [15] constructed three models for an engineering cost estimation application. Additional models should also be created when there are large differences between the predicted and expected outputs. Guiver and Klimasauskas [27] report that it is not unusual to have correlations near .90 between predicted and actual values of a NN, although this may be high for many applications.

For constructing additional NNs, the developer may (1) alter part of the existing NN paradigm and retrain and retest with existing data, (2) alter the training (and possibly testing) data and retrain with the existing NN paradigm, or both (1) and (2). As an example, if the network generalizes poorly with the test data, reducing the number of hidden nodes may be appropriate. Alternatively, the training data may need revised to ensure that all important concepts are equally represented or to ensure that there are sufficient amounts of all concepts.

5.11 Select the best model and deploy

One or more of the NN models is deployed. A single *best* NN may be selected by ranking all NNs according to some criteria such as most correct (or fewest incorrect) classifications or closest predictions. The criteria used for testing and validating each NN can be used as the selection criteria for determining the best NN.

Sometimes a single best NN is not the best solution; a combination of multiple models may provide better results than any given model. Multiple NNs may be combined through averaging or rule synthesis. For predicting the TOPIX index, Kimota et al. [39] increased the correlation with the training data by averaging the outputs of several networks. Trippi and DeSieno [66] used rule synthesis, combining the outputs of five NNs with boolean operators, and thereby increased the expected return in a trading decision system.

After the NN has been implemented, its performance should be monitored. If the performance decreases below some level, then a new NN should be built and trained on data typifying the present conditions.

6. Contribution to Knowledge

This study provides a methodology which may be used by both academicians and practitioners to guide the systematic development of NN applications. It synthesizes this methodology by examining the reported results of hundreds of neural network researchers. The methodology can be used by developers to plan and justify the necessary steps for creating a NN application and to gauge their progress in the development process. Hopefully, this methodology will lead to increased NN development and improve the likelihood of developing successful NN applications.

7. Limitations and Key Assumptions

The methodology proposed in this study is derived from business NNs presented in academic/professional journal articles. It therefore does not consider all NN applications. Conference proceedings and doctoral dissertations are excluded, assuming that high quality research is eventually published in academic/professional journals.

8. Future Research

The proposed methodology must now be used and evaluated by NN developers. Refinements in the methodology should also be explored. Although this study proposes a methodology for all NN development, methodologies tailored to specific NN paradigms or application areas should also be investigated.

References

[1] Altman, Edward I., Giancarlo Marco, and Franco Varetto. Corporate distress diagnosis: comparisons using linear discriminant analysis and neural networks (the Italian experience). *Journal of Banking and Finance*, 18.3 (May 1994) 505-529.

[2] Bahrami, A., M. Lynch C. H. Dagli. Intelligent design retrieval and packaging system: application of neural networks in design and manufacturing. *International Journal of Production Research*, 33.2 (1995) 405-426.

[3] Barr, Dean S. and Ganesh Mani. Using neural nets to manage investments. *AI Expert*, 9 (February 1994) 16-21.

[4] Barron, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information*, 38 (1992).

[5] Beaverstock, Malcolm and Kenneth Wolchina. Neural network helps G-P Ashdown Mill improve brownstock washer operation. *Pulp and Paper*, 66.9 (September 1992) 134-136.

[6] Bishop, Christopher M. *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.

[7] Burke, Laura I. An unsupervised neural network approach to tool wear identification. *IIE Transactions*, 25.1 (January 1993) 16-25.

[8] Caudill, M. Neural networks primer. *AI Expert*, 3 (1988) 55-61.

[9] Chang, C. Alec and Chao-Ton Su. A comparison of statistical regression and neural network methods in modeling measurement errors for computer vision inspection systems. *Computers and Industrial Engineering*, 28.3 (July 1995) 593-603.

[10] Chen, S. K., P. Mangiameli, and D. West. The comparative ability of self-organizing neural networks to define cluster structure. *Omega, The International Journal of Management Science*, 23.3 (June 1995) 271-279.

[11] Chu, Chao-Hsien and Djohan Widjaja. Neural network system for forecasting method selection. *Decision Support Systems*, 12.1 (August 1994) 13-24.

[12] Chu, Xuening and Hans Holm. Product manufacturability control for concurrent engineering. *Computers in Industry*, 24.1 (1994) 29-38.

[13] Coats, Pamela K. and L. Franklin Fant. A neural network approach to forecasting financial distress. *The Journal of Business Forecasting*, 10.4 (Winter 1991-92) 9-12.

[14] Collins, Alan and Alec Evans. Aircraft noise and residential property values: an artificial neural network approach. *Journal of Transport Economics and Policy*, 28.2 (May 1994) 175-197.

[15] Creese, Robert C. and Li Li. Cost estimation of timber bridges using neural networks. *Cost Engineering*, 37.5 (May 1995) 17-22.

[16] Cui, Xianzhong and Kang G. Shin. Intelligent coordination of multiple systems with neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 21.6 (November/December 1991) 1488-1497.

[17] Cybenko, G. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2 (1989) 303-314.

[18] Dutta, Soumitra and Shashi Shekhar. Bond-rating: a non-conservative application of neural networks. *Proceedings of the IEEE International Conference on Neural Networks*, 2 (1988) 443-450.

[19] Eberhat, Russell C. and Roy W. Dobbins. *Neural Network PC Tools*. San Diego: Academic Press, 1990.

[20] Fletcher, Desmond and Ernie Goss. Forecasting with neural networks: an application using bankruptcy data. *Information and Management*, 24.3 (March 1993) 159-167.

[21] Funahashi, K. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2 (1989) 183-192.

[22] Gemen, S., E. Beienenstocks and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4 (1992) 1-58.

[23] Gillenson, Mark L. and Joel D. Stutz. Academic issues in MIS: journals and books. *MIS Quarterly,* 15.4 (December 1991), 447-452.

[24] Gorman, R. and T. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1.1 (1988) 75-89.

[25] Grefenstette, J. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16.1 (1982).

[26] Grossberg, S. How does a brain build cognitive code? *Psychological Review*, 87 (1980) 1-51.

[27] Guiver, John P. and Casimir C. Klimasauskas. Applying neural networks: part iv. *PC AI*, 5.4 (July 1991) 34-41.

[28] Hamilton, S., and Ives, B. The journal communication system for MIS research. *Database,* 14,2 (April 1983), 3-14.

[29] Hecht-Nielsen, R. Kolmogorov's mapping neural network existence theorem. *Proceedings of the 1st IEEE Annual International Conference on Neural Networks*, San Diego (1987) III.11-III.14.

[30] Hinton, Geoffrey, James L. McClelland and David E. Rumelhart. Distributed representations. In David E. Rumelhart and James L. McClelland (eds,), *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, MIT Press, Cambridge (1986) 77-109.

[31] Hopfield, John J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79 (1982) 2554-2558.

[32] Hopfield, John J. Neurons with graded responses have collective computational properties like those of two stated neurons. *Proceedings of the National Academy of Sciences*, 81 (1984) 3088-3092.

[33] Hornik, K., M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2 (1989) 359-366.

[34] Hwarng, H. Brian. Proper and effective training of a pattern recognizer for cyclic data. *IIE Transactions*, 27.6 (December 1995) 746-756.

[35] Jain, Bharat A. and Brian N. Nag. Artificial neural network models for pricing initial public offerings. *Decision Sciences*, 26.3 (May 1995) 283-302.

[36] Jensen, Herbert L. Using neural networks for credit scoring. *Managerial Finance*, 18.6 (1992) 15-26.

[37] Kaastra, Iebeling and Milton S. Boyd. Forecasting futures trading volume using neural networks. *The Journal of Futures Markets*, 15.8 (December 1995) 953-970.

[38] Kaparthi, Shashidar and Nallan C. Suresh. Machine-component cell formation in group technology: a neural network approach. *International Journal of Production Research*, 30.6 (1992) 1353-1367.

[39] Kimoto, T., K. Asakawa, M. Yoda, and M. Takeoka. Stock market prediction system with modular neural networks. *Proceedings of the International Joint Conference on Neural Networks*, San Diego: IEEE Network Council, 1 (1990) 1-6.

[40] Knapp, Gerald M. and Hsu-Pin Wang. Machine fault classification: a neural network approach. *International Journal of Production Research*. 30.4 (1992) 811-823.

[41] Kohonen, T. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43 (1982) 59-69.

[42] Lee, Jae Kyu, and Won Chul Jhee. A two-stage neural network approach for ARMA model identification with ESACF. *Decision Support Systems*, 11.5 (June 1994) 461-479.

[43] Liang, Ting-Peng, John S. Chandler, Ingoo Han, Jinsheng Roan. An empirical investigation of some data effects on the classification accuracy of probit, ID3, and neural networks. *Contemporary Accounting Research*, 9.1 (Fall 1992) 306-328.

[44] Liao, T. Warren and Kwan S. Lee. Integration of a feature-based CAD system and an ART1 neural model for GT coding and part family forming. *Computers and Industrial Engineering*, 26.1 (January 1994) 93-104.

[45] Lippman, R. P. Review of neural networks for speech recognition. *Neural Computing*, 1 (1989) 1-38.

[46] Masson, Egill and Yih-Jeou Wang. Introduction to computation and learning in artificial neural networks. *European Journal of Operations Research*, 47.1 (July 1990) 1-28.

[47] NeuralWare Technical Publications Group. *NeuralWorks Predict: Complete Solution for Neural Data Modeling*. Pittsburgh: Neural Ware, 1995.

[48] NeuralWare Technical Publications Group. *Reference Guide: Software Reference for Professional II/PLUS and NeuralWorks Explorer*. Pittsburgh: Neural Ware, 1995.

[49] Oh, Se-Young, Hyoung-Gwen Park and Soo-Hyuk Nam. A neural network-based real-time robot tracking controller using position sensitive detectors. *Expert Systems*, 12.2 (May 1995) 115-122.

[50] Pugh, G. Allen. Synthetic neural networks for process control. *Computers and Industrial Engineering*, 17.1-4 (1989) 24-26.

[51] Pugh, G. Allen. A comparison of neural networks to SPC charts. *Computers and Industrial Engineering*, 21.1-4 (1991) 253-255.

[52] Rangwala, Sabbir S. and David A. Dornfield. Learning and optimization of machining operations using computing abilities of neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 19.2 (March/April 1989) 299-314.

[53] Refenes, A. N. Comments on 'neural networks: forecasting breakthrough or passing fad' by C. Chatfield. *International Journal of Forecasting*, 10.1 (June 1994) 43-46.

[54] Rumelhart, David E., James L. McClelland, and the PDP Research Group (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, MIT Press, Cambridge.

[55] Salchenberger, Linda M., E. Mine Cinar and Nicholas A. Lash. Neural networks: a new tool for predicting thrift failures. *Decision Sciences*, 23.4 (July/August 1992) 899-916.

[56] Satake, Tsuyoshi, Katsumi Morikawa and Nobuto Nakamura. Neural network approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics*, 33.1-3 (January 1994) 67-74.

[57] Schocken, Shimon and Gad Ariav, Neural networks for decision support: problems and opportunities, *Decision Support Systems*, 11.5 (June 1994) 393-414.

[58] Sim, S. K., K. T. Yeo and W. H. Lee. An expert neural network system for dynamic job shop scheduling. *International Journal of Production Research*, 32.8 (1994) 1759-1773.

[59] Simon, Herbert A. *The New Science of Management Decision Making*. New York: Harper & Row, 1960.

[60] Slater, John R., Susan J. Hazen and Sachi Sakthivel. On selecting appropriate technology for knowledge systems. *Journal of Systems Management*, 44.10 (October 1993) 10-15+.

[61] Smith, Alice E. and Cihan H. Dagli. Controlling industrial processes through supervised, feedforward neural networks. *Computers and Industrial Engineering*, 21.1-4 (1991) 247-251.

[62] Sorsa, Timo, Heikki N. Koivo, and Hannu Koivisto. Neural networks in process fault diagnosis. *IEEE Transactions o Systems, Man, and Cybernetics*, 21.4 (July/August 1991) 815-825.

[63] Szu, H. Three layers of vector outer product neural networks for optical pattern cognition. *SPIE Optical and Hybrid Computing*, 634 (1986) 312-330.

[64] Taha, Mahmoud A., Sang C. Park and Jeffrey S., Russell. Knowledge-based DSS for construction contractor prescreening. *European Journal of Operational Research*, 84.1 (July 1995) 35-46.

[65] Tam, Kar Yan and Melody Y. Kiang. Managerial applications of neural networks: the case of bank failure predictions. *Management Science*, 38.7 (July 1992) 926-947.

[66] Trippi, Robert R. and Duane DeSieno. Trading equity index futures with a neural network. *The Journal of Portfolio Management*, 19.1 (Fall 1992) 27-33.

[67] Turban, Efraim. *Decision Support and Expert Systems: Management Support Systems*. Englewood Cliffs, New Jersey: Prentice Hall (1995).

[68] Velasco, Tomas and Mark R. Rowe. Back propagation artificial neural networks for the analysis of quality control charts. *Computers and Industrial Engineering*, 25.1-4 (September 1993) 397-400.

[69] Venugopal, V. and T. T. Narendran. Machine-cell formation through neural network models. *International Journal of Production Research*, 32.9 (1994) 2105-2116.

[70] Villegas, Leticia and Ray E. Eberts. A neural network tool for identifying text-editing goals. *International Journal of Human-Computer Studies*, 40 (May 1994) 813-833.

[71] Wang, Jun. A neural network approach to multiple-objective cutting parameter optimization based on fuzzy preference information. *Computers and Industrial Engineering*, 25.1-4 (September 1993) 389-392.

[72] Wang, Qiwen, Xiaoyun Sun, Bruce L. Golden, Lenore DeSilets, Edward A. Wasil, Scott Luco, and Adam Peck. A neural network for the wire bonding process. *Computers and Operations Research*, 20.8 (October 1993) 879-888.

[73] Weigand, A. S., D. E. Rumelhart, and B. A. Huberman. Generalization by weight elimination with application to forecasting. In R. P. Lippmann, J. Moody, and D. S. Touretzky, eds., *Advances in Neural Information Processing Systems, Volume 3*, (1991) Morgan Kaufman, San Mateo.

[74] White, H. Some asymptotic results for learning in single hidden-layer feedforward network models. *Journal of American Statistical Association*, 84.408 (1989) 1003-1031.

[75] White, H. Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, 3 (1990) 535-549.

[76] Wilson, Rick L. and Ramesh Sharda. Bankruptcy prediction using neural networks. *Decision Support Systems*, 11.5 (June 1994) 545-557.

[77] Wong, Bo K., Thomas A. Bodnovich, and Yakup Selvi. A bibliography of neural network business applications research: 1988-September 1994, *Expert Systems: International Journal of Knowledge Engineering and Neural Networks*, 12.3 (August 1995) 253-262.

[78] Wong, Bo K., Thomas A. Bodnovich, and Yakup Selvi. Neural network applications in business: a review and analysis of the literature (1988-1995), *Decision Support* Systems, 19.4 (April 1997) 301-320.

[79] Wong, Shee Q. and J. Allen Long. A neural network approach to stock market holding period returns. *American Business Review*, 13.2 (June 1995) 61-64.

[80] Yamashina, Hajime, Hiromitsu Kumamoto, Susumu Okumura and Takahiro Ikesaki. Failure diagnosis of a servovalve by neural networks with new learning algorithm and structure analysis. *International Journal of Production Research*, 28.6 (1990) 1009-1021.

[81] Yih, Yuehwern, Ting-Peng Liang and Herbert Moskowitz. Robot scheduling in a circuit board production line: a hybrid OR/ANN approach. 25.2 (March 1993) 26-33.

[82] Yoon, Youngohc, George Swales Jr. and Thomas M. Margavio. A comparison of discriminant analysis versus artificial neural networks. *Journal of Operational Research Society*, 44.1 (January 1993) 51-60.

[83] Yoon, Youngohc, Tor Guimaraes and George Swales. Integrating artificial neural networks with rule-based expert systems. *Decision Support Systems*, 11.5 (June 1994) 497-507.

[84] Zahedi, Fatemeh. An introduction to neural networks and a comparison with artificial intelligence and expert systems. *Interfaces*, 21.2 (March/April 1991) 25-38.