Heuristic Principles For The Design Of Artificial Neural Networks

Article *in* Information and Software Technology · January 1999

DOI: 10.1016/S0950-5849(98)00116-5 · Source: CiteSeer

Steven Walczak
University of South Florida
91 PUBLICATIONS **1,409** CITATIONS

Narciso Cerpa
Universidad de Talca
103 PUBLICATIONS **841** CITATIONS

Published and copyrighted, 1999, *Information and Software Technology* 41 (2), pp. 109-119.

HEURISTIC PRINCIPLES FOR THE DESIGN OF ARTIFICIAL NEURAL NETWORKS

Steven Walczak

&

University of Colorado at Denver
College of Business & Administration
Campus Box 165, PO Box 173364
Denver, CO 80217-3364 USA
swalczak@carbon.cudenver.edu
(303) 556-6777
(303) 556-5899 fax

Narciso Cerpa

School of Information Systems
University of New South Wales
Sydney 2052
Australia
n.cerpa@unsw.edu.au
(61) 2 9385-4847
(61) 2 9662-4061 fax

Abstract

Artificial neural networks have been used to support applications across a variety of business and scientific disciplines during the past years. Artificial neural network applications are frequently viewed as black boxes which mystically determine complex patterns in data. Contrary to this popular view, neural network designers typically perform extensive knowledge engineering and incorporate a significant amount of domain knowledge into artificial neural networks. This paper details heuristics that utilize domain knowledge to produce an artificial neural network with optimal output performance. The effect of using the heuristics on neural network performance is illustrated by examining several applied artificial neural network systems. Identification of an optimal performance artificial neural network requires that a full factorial design with respect to the quantity of input nodes, hidden nodes, hidden layers, and learning algorithm be performed. The heuristic methods discussed in this paper produce optimal or near-optimal performance artificial neural networks using only a fraction of the time needed for a full factorial design.

Keywords: Artificial neural networks; Heuristics; Input vector; Hidden layer size; ANN learning method; Design.

1. Introduction

Artificial neural network (ANN) applications have exploded onto the scene in the past several years and are continuing to be developed. Industrial applications exist in the financial, manufacturing, marketing, telecommunications, biomedical, and other domains [4,12,15,16,26,27,30,31,55,56]. While business managers are seeking to develop new applications using ANNs, a basic misunderstanding of the source of intelligence in an ANN exists. Furthermore, the development of new ANN applications is facilitated by the recent emergence of a variety of neural network shells (e.g., Neuralworks Professional II Plus, @Brain, and Neuralyst) which enable anyone to produce neural network systems by simply specifying the ANN architecture and providing a set of training data to be used by the shell to train the ANN [32,41]. These shell based neural networks may fail or produce sub-optimal results unless a deeper understanding of how to use and incorporate domain knowledge in the ANN is obtained by the designers of ANNs in business and industrial domains.

The traditional view of an ANN is of a program that emulates biological neural networks and "learns" to recognize patterns or categorize input data by being trained on a set of sample data from the domain. Learning through training and subsequently the ability to generalize broad categories from specific examples [19] is the unique perceived source of intelligence in an ANN. However, experienced ANN application designers typically perform extensive knowledge engineering and incorporate a significant amount of domain knowledge into the design of ANNs even before the learning through training process has begun.

Design of optimal neural networks is problematic in that there exist a large number of alternative ANN physical architectures and learning methods, all of which may be applied to a given business problem. Artificial neural network designers must determine the set of design criteria specified in Figure 1.

1. The appropriate input (independent) variables.
2. The best learning method. Learning methods may be classified into either supervised or unsupervised learning methods. Within each of these larger classifications exists numerous alternatives, each of which works optimally on different distributions or types of data.
3. The number of hidden layers (depending on the selected learning method).
4. The quantity of processing elements (nodes) per hidden layer. To avoid both under-fitting and over-fitting the training data, which produces poor generalization or out-of-sample performance.

Figure 1. Set of Design Criteria

Each individual choice for these four design variables will affect the performance of the resulting ANN on out-of-sample data. Improper selection of the values for these four design factors may produce ANNs that perform worse than random selection of an output (dependent) value.

In this paper, a heuristic approach for incorporating knowledge into an ANN and using domain knowledge to design optimal ANNs is examined. The heuristic ANN design approach is made up of the following steps: knowledge-based selection of input values, selection of a learning method, and design of the hidden layers (quantity and nodes per layer). The majority of the ANN design steps described will focus on feedforward supervised learning (and more specifically backpropagation) ANN systems. A discussion of other learning methods is presented in Section 4. Following these heuristic methods will enable businesses to take advantage of the power of ANNs and will afford economic benefit by producing an ANN that outperforms similar ANNs with improperly specified design parameters.

2. Background

Artificial neural networks have been applied to a wide variety of business problems [1,5,10,12,26,30,41,55], especially in accounting [28,40] and finance [23,51]. Various research results have shown that ANNs outperform traditional statistical techniques (e.g., regression or logit) [1,10,40] as well as other standard machine learning techniques (e.g., the ID3 algorithm) [40] for a large class of problem types.

The ANN learning algorithm called backpropagation is the most popular design choice for implementing neural networks [8,9,14]. The popularity of backpropagation is due in part to its wide availability and support in commercial neural network shells [32,41] and to the robustness of the paradigm. Hornik et al. [21] have shown that backpropagation ANNs are universal approximators, able to learn arbitrary category mappings. Supporting this research finding, various other researchers have shown the superiority of backpropagation ANNs to different ANN learning paradigms including: radial basis function (RBF) [2], counterpropagation [11], and adaptive resonance theory [5]. Cherkassky and Lari-Najafi [8] claim that an ANN's performance is more dependent on data representation than on the selection of a learning rule. Learning rules other than backpropagation perform well if the data from the domain has specific properties. The functionality and prerequisite properties of several popular ANN learning rules are presented in Section 4. Readers interested in the mathematical specifications of the various ANN learning methods are directed to the references [9,14,17,19,21,54].

The generalization performance of supervised learning artificial neural networks (e.g., backpropagation) generally improves when the network size is minimized [19,33] with respect to the weighted connections between processing nodes (elements of the input, hidden, and output layers). Networks that are too large tend to over-fit or memorize the input data [2]. Conversely, ANNs with too few weighted connections do not contain enough processing elements to correctly model the input data set, under-fitting the data. Both of these situations result in poor out-of-sample generalization.

3. Input Variable Selection

The place to begin using and implementing domain knowledge in an ANN is the input vector. The old adage "garbage in, garbage out" applies to ANNs and ANN designers must spend a significant amount of time performing the task of knowledge acquisition. Selection of input variables is an important and complex task for the ANN designer [44]. Pakath and Zaveri [38] claim that ANNs as well as other artificial intelligence (AI) techniques are highly dependent on the specification of input variables. However, input variables are routinely misspecified by ANN developers.

Input variable misspecification occurs because ANN designers follow the expert system methodology of incorporating as much domain knowledge as possible into an intelligent system [29]. Hertz et al. [19] specifically state that ANN performance improves as additional domain knowledge is provided through the input variables. This is certainly true to the extent that if a sufficient amount of information representing critical decision criteria is not given to an ANN, then the ANN (or any other modeling technique) cannot develop a correct model of the domain. The common belief is that since ANNs learn, they will be able to determine those input variables that are important and develop a corresponding model through the modification of the weights associated with the connections between the input layer and the hidden layers [9].

However, Smith [43] and others [38,44,46,53] claim that noise input variables produce poor generalization performance in ANNs. The presence of too many input variables causes poor generalization when the ANN not only models the true predictors, but includes the noise variables in the model. Piramuthu et al. [40] state that the interaction between input variables produces critical differences in output values, further obscuring the ideal problem model when unnecessary variables are included in the set of input values.

Degraded performance of an ANN is not the only cost of including too many input variables. Bansal et al. [1] have noted that data (training, test, and standard use) represents an important and recurring cost for information systems in general and ANNs in particular.

3.1. Heuristic Determination of Input Variables

As indicated above and shown in the following sections, both under and over specification of input variables produces sub-optimal performance. No formal methods currently exist for selecting input (independent) variables for an ANN solution to a domain problem.

The first step in determining the optimal set of input variables is to perform standard knowledge acquisition. Typically, this involves consultation with multiple domain experts. Various researchers [25,51,52] have indicated the requirement for extensive knowledge acquisition utilizing domain experts to specify ANN input variables. The primary purpose of the knowledge acquisition phase is to guarantee that the input variable set is not under-specified, providing all relevant domain criteria to the ANN.

Once a base set of input variables is defined through knowledge acquisition, the set can be pruned to eliminate variables that contribute noise to the ANN and consequently reduce the ANN generalization performance. Smith [43] claims that ANN input variables need to be predictive, but should not be correlated. Correlated variables degrade ANN performance by interacting with each other as well as other elements to produce a biased effect. A first pass filter to help identify "noise" variables is to calculate the correlation of pairs of variables (Pearson correlation matrix). Alternatively, a chi-square test may be used for categorical variables. If two variables have a high correlation, then one of these two variables may be removed from the set of variables without adversely affecting the ANN performance. The cutoff value for variable elimination is a heuristic value and must be determined separately for every ANN application, but any correlation absolute value of 0.20 or higher indicates a probable noise source to the ANN.

Additional statistical techniques may be applied, depending on the distribution properties of the data set. Stepwise regression (multiple or logistic) and factor analysis provide viable tools for evaluating the predictive value of input variables [23,34] and may serve as a secondary filter to the Pearson correlation matrix. Multiple regression and factor analysis perform best with normally distributed linear data, while logistic regression assumes a curvilinear relationship [34].

3.2. Effect of Variable Set on ANN Performance

Although efficient design of the input variable set has not been a previous research topic, several researchers have shown that smaller input variable sets can produce better generalization performance by an ANN. Lenard et al. [28], in developing a neural network model to predict an auditor's going concern for a corporation, reduced their input set size by 50 percent going from eight to four input variables with a corresponding improvement in performance of 1.2 to 9 percent. The variable reduction performed in Lenard et al.'s research is accomplished by selecting only a single variable for representing each of liquidity, solvency, and profitability instead of the two to three variables used for each measure in the original ANN design. Lenard et al.'s variable reduction successfully eliminates highly correlated variables.

Jain and Nag [23] however, when developing a neural network model to predict the value of initial public stock offerings, indicate that reducing their input variable set from eleven to four variables did not result in an improvement in the ANN's performance, but actually decreased the performance level. Further analysis of the Jain and Nag research shows that a four input variable set tested on a single ANN hidden layer architecture did perform similar to two of the five different architectures used to evaluate the eleven variable model. Additionally, unlike the Lenard research, variables are eliminated in Jain's research based on a statistical measure of predictiveness instead of eliminating correlated variables. Several of the variables used in this research have high correlations, such as a correlation of 0.75 between two variables used in both the eleven and four variable models. Use of the recommended correlated variable elimination heuristic may have produced an ANN with better or at least equal performance to the larger ANN model, with a corresponding economic savings in the data costs for the network.

Tahai et al. [46] report on both the under and over specification effects of input variables on an ANN designed to assist new ambulance medical personnel in determining the need for application of cardio-pulmonary resuscitation. When only a partial set of the necessary input variable set determined from knowledge acquisition interviews was used, the ANN was incapable of learning when to advise (predict) the application of artificial respiration. When all of the specified variables were used, the ANN performed with 100 percent accuracy. Upon adding two additional input variables to the ANN input vector (these new variables were all acquired by ambulance technicians as part of their routine patient evaluation), then the performance of the ANN dropped by 7 percent.

Another explicit example of the effect of both under and over specification of the input variable set is derived by extending the research of Walczak [51]. The purpose of the ANNs in this research is to forecast the change to the future cable-rate values for exchanging US dollars to either British pounds or Japanese yen. The variable set determined by Walczak (using traditional knowledge acquisition techniques, such as interviewing domain experts) to produce the best forecast values is the one day lags for ten cross rate values for the dollar, pound, yen, mark, and Swiss franc and nine rates of returns on international bonds (three each for Japan, Britain, and Germany), resulting in nineteen input variables. New ANNs are designed to attempt to reduce the nineteen input variables by eliminating either the international bond variables or the cross rate variables. Next, a reduced set is derived by eliminating all variables that are not directly related to the British pound. Finally, other financial indicators (the S&P 500 and the CRSP indexes, and oil and non-ferrous metals indexes) are added to the original set of input variables to provide related noise variables.

A partial Pearson correlation matrix of input variables is given in Table 1 (correlations of .20 or higher are displayed in bold). Although all of the cable rate cross rates have a correlation

	$/P	$/Y	$/M	Brt. bond	Jap. bond	Ger. bond	S&P	CRSP	Oil	Non-Ferrous
$/Pound	1	.30	.80	.19	-.08	.14	.20	.15	.25	.15
$/Yen	**.30**	1	.37	-.06	-.02	.10	-.11	-.06	-.06	.08
$/Mark	**.80**	**.37**	1	.24	-.07	.20	.13	.12	.15	.19
B 1 month Bond	.19	-.06	**.24**	1	.02	.48	.06	.25	.04	.23
J 1 month Bond	-.08	-.02	-.07	.02	1	.02	-.06	-.05	-.05	-.08
G 1 month Bond	.14	.10	**.20**	**.48**	.02	1	.12	.33	.07	.16
S&P	**.20**	-.11	.13	.06	-.06	.12	1	.56	.27	.09
CRSP	.15	-.06	.12	**.25**	-.05	**.33**	**.56**	1	.15	.10
Oil	**.25**	-.06	.15	.04	-.05	.07	**.27**	.15	1	.12
Non-Fe	.15	.08	.19	**.23**	-.08	.16	.09	.10	.12	1

Table 1: Pearson correlation matrix for variables in cable-rate forecast ANN.

greater than .20, this information redundancy is unavoidable since each of the represented variables is composed of two pieces of information, the currencies being traded, and the dollar value are a common element for each of these cross rates. As a further illustration of the required correlations, the dollar/pound cross rate has a correlation value of -.05 compared to the franc/mark cross rate and the dollar/yen has a correlation of 0.001 to the pound/mark cross rate.

Various hidden layer architectures are implemented for each of the input variable sets just described. All hidden layer architectures consist of two layers following the design of [51]. The various hidden node quantities are used to reduce the opportunity for introducing non-input-variable dependent error, such as under-fitting or over-fitting the data [2]. Backpropagation is the learning method for all of the ANNs. The training data for all ANNs consists of the corresponding values from January 2, 1993 until October 31, 1993 (143 cases) and the test data consists of the corresponding values from November 1, 1993 until December 31, 1993 (37 cases). Each ANN is trained until the in-sample MSE is less than .05 or the MSE does not change over the presentation of 2000 training examples. Forecasting results of the various networks are displayed in Table 2 as (*1-MAPE*), where a value of 1.0 indicates perfect forecasting. Following the results of the November to December 1993 test set, an additional test set covering the period from January 1994 through April 1994 was also tested with the original ANN and produced nearly identical results. The extension of the test time period serves to demonstrate the robustness of the ANN forecasting model.

Input variables [quantity]	Best Performance Pound	Yen [nodes per layer]	Average Performance of All Networks Tested	
Cross Rates (CR) and Bond Returns (BR) [19]	.5676[†]	.5405	.5541[†]	.4956
	[_,18,3,2]		(8 ANNs)	
CR only [10]	.4762	.5135	.4445	.4886
	[_,9,6,2]	[_,6,3,2]	(3 ANNs)	
BR only [9]	.3810	.5135	.3333	.4728
	[_,6,3,2]	[_,9,6,2]	(3 ANNs)	
CR and BR for Pound only [7]	.5135	N/A	.5135	N/A
	[_,6,3,1]		(1 ANN)	
CR and BR plus S&P 500 & CRSP [21]	.5405	.4865	.5169	.4662
	[_,18,6,2]		(8 ANNs)	
CR and BR plus Oil and Non-Ferrous [21]	.5405	.4595	.5067	.4122
	[_,18,3,2]		(8 ANNs)	

[†] statistically significant at .05 level

Table 2: Forecasting performance for the various input variable sets.

An MAPE value of .50 represents chance, so the ideal networks will have a *1 - MAPE* value greater than .50. Table 2 indicates that the combination of cross rates and bond return values (Model 1) produces the optimal forecasting results for all of the ANNs with different input vectors. In fact, for the dollar/pound cable rate forecasts, none of the other ANN variable sets surpasses the average performance of all Model 1 ANN architectures evaluated. The test set for forecasting the pound only using Model 1 is extended to April 30, 1994, bringing the total test cases to 120, and produces similar results. The similarity of the forecasting performance for the British pound on the second test set (using the same training set), demonstrates the robustness of the ANN for producing out-of-sample forecasts.

The same cannot be said of the dollar/yen forecasts, shown in Table 2. However, the best and average ANN forecasting performance for the yen, of the Model 1 ANNs, consistently outperforms the corresponding measurement for all other variable set models.

As noted above, high correlation values of variables that share a common element need to be disregarded. Identification of the single element components of the ten cross rate variables that are correlated requires a comparison of similar correlation values. In the full Pearson matrix, the correlation values for the Swiss franc and German mark are highly similar. The franc-yen and mark-yen cross rates have an average difference in correlation values of .013 when correlated against identical variables and a similar result occurs for the pound-franc and pound-mark variables. The only other two variables with this type of similarity in the full correlation matrix are the non-ferrous metals index and the CRSP index however, these two variables do not appear together in any of the ANN models. Removal of the three Swiss franc variables (cross rates against the dollar, pound, and yen) yields ANNs with identical best case performances to the Model 1 ANNs.

Elimination of the Swiss franc values does not improve the ANN forecasting performance, but instead yields the secondary benefits of reduced data acquisition costs (by eliminating the need for the four Swiss franc variables) and a reduction in the complexity of the ANN model (producing shorter training times as discussed in Section 5).

In this section, research examples are provided from accounting, finance, and medicine which indicate that many ANN systems can be improved through variable reduction. Smaller input variable sets frequently improve the ANN generalization performance and reduce the net cost of data acquisition for development and usage of the ANN. However, care must be taken when removing variables from the ANN's input set to insure that a complete set of non-correlated predictor variables is available for the ANN, otherwise the reduced variable sets may worsen generalization performance.

4. ANN Learning Method Selection

After determining a heuristically optimal set of input variables using the methods from the previous section, a learning method must be selected. Learning methods can be divided into two distinct categories: unsupervised learning and supervised learning. Both types of ANN learning require a collection of training examples that enable the ANN to model the data set and produce accurate output values. Unsupervised learning systems; such as adaptive resonance theory (ART) [6], self organizing map (SOM) [24], or Hopfield [20] networks; do not require that the output value for a training sample be provided at the time of training, while supervised learning systems; such as backpropagation (multi-layer perceptron), radial basis function (RBF) [35], counterpropagation [18], or fuzzy ARTMAP [7] (a supervised learning extension of the ART learning method) networks; require that a known output value for all training samples be provided to the ANN.

Unsupervised learning methods determine output values (classifications) directly from the input variable data set. Because the "answers" must be directly learnable from or contained within the input values, most unsupervised learning methods have less computational complexity and less generalization accuracy than supervised methods [14]. Therefore, unsupervised learning techniques are typically used for classification problems, where the desired classes are self-descriptive. ART networks are a favorite technique for object recognition in pictorial or graphical data. The advantage of using unsupervised learning methods is that these ANNs can be designed to learn much more rapidly than supervised learning systems [14,26].

As discussed in the Background Section, backpropagation is the most common learning method for ANNs and because of its generality (robustness) and ease of implementation [32] it is the best choice for a majority of ANN systems. Backpropagation is the superior learning method when a sufficient number of noise/error free training examples exist, regardless of the complexity of the specific domain problem. Although backpropagation networks can handle noise in the training data (and may actually generalize better if some noise is present in the training data), too many erroneous training values may prevent the ANN from learning the desired model.

A heuristic lower bound on the number of training examples required to train a backpropagation ANN, is four times the number of weighted connections contained in the network. Therefore, if a training database contains only 100 training examples, the maximum size of the ANN is 25 connections or approximately (depending on the ANN architecture) ten nodes. While the general heuristic of four times the number of connections is applicable to most categorization problems, time series problems, including the prediction of financial time series (e.g., stocks or commodities values), are more dependent on “business cycles”. Walczak’s [49] research claims that only a maximum of one or two years of data is required to produce optimal forecasting results for neural networks performing financial time series prediction. Wilson and Sharda [56] make a similar claim, stating that the neural network training set should be representative of the population-at-large. Therefore, the training set size of 10 months of data used in this articles on-going example to predict foreign exchange rates is very near to the optimal value specified by Walczak.

For ANN applications that provide only a few training examples or very noisy training data, other supervised learning methods should be selected. RBF networks perform well in domains with limited training sets [2] and counterpropagation networks perform well when a sufficient number of training examples is available, but may contain very noisy data [11]. Walczak [48] performs an analysis of six different neural network supervised learning methods on the resource allocation problem of assigning workers to perform mission critical tasks. Walczak concludes that for the specific domain problem being addressed, backpropagation produced the best results (although the first appearance of the problem indicated that counterpropagation might outperform backpropagation due to anticipated noise in the training data set). Hence, although properties of the data population may strongly indicate the preference of a particular training method, because of the strength of the backpropagation network [21,54], this type of learning method should always be tried in addition to any other methods proscribed by domain data tendencies.

The dollar to pound and dollar to yen cable rate forecasting ANN described in Section 3 is implemented using backpropagation. The domain for the cable rate forecasting ANN has a large collection of relatively error free historical examples with known outcomes, suiting it for backpropagation ANN implementations. A comparison of different ANN learning models is performed using the training and test data from the Model 1 (CR and BR variables) ANN to illustrate the correctness of the learning method choice. An ART ANN and an RBF ANN are both implemented and to the extent possible with hidden layer constraints imposed by the ART and RBF learning methods, the ANN hidden layer architectures is duplicated. The results of the backpropagation ANN for the original Model 1 input set and the best results for the two new learning method ANNs are shown in Table 3. Both the ART and RBF ANNs have worse performance than the backpropagation ANN performance for this specific domain problem.

ANN Learning Method	Forecast Accuracy Dollar/Pound	Forecast Accuracy Dollar/Yen
Backpropagation	.5676	.5405
Adaptive Resonance Theory (ART)	.4054	.4595
Radial Basis Function (RBF)	.3784	.4595

Table 3: ANNs with different learning methods for predicting cable rates.

Many other ANN learning methods exist and each is subject to constraints on the type of data that are best processed by that specific learning method. For example, general regression neural networks [45] are capable of solving any problem that can also be solved by a statistical regression model, but does not require that a specific model type (e.g., multiple linear or logistic) be specified in advance. However, regression ANNs suffer from the same constraints as regression models [14], such as the linear or curvilinear relationship of the data with heteroscedastic error [34]. Likewise, learning vector quantization (LVQ) networks [24] try to divide input values into disjoint categories similar to discriminant analysis and consequently have the same data distribution requirements as discriminant analysis. The resource/employee allocation example described earlier [48] indicated that LVQ neural networks produced the second best allocation results, which indicated the previously unknown perception that the categories used for allocating employee resources were unique (unlike the previous assumption that the two categories of job were correlated).

The selection of a learning method is an open problem and ANN designers must use the constraints of the training data set for determining the optimal learning method. If a reasonably large quantity of relatively noise-free training examples are available, then backpropagation provides an effective learning method which is relatively easy to implement.

5. Design of Hidden Layers

The architecture of an ANN consists of the number of layers of processing elements (or nodes); including input, output, and any hidden layers; and the quantity of nodes contained in each layer. Heuristic design of the input vector is discussed in Section 3 and the output vector is normally predefined by the problem to be solved with the ANN. Design of hidden layers is dependent on the selected learning algorithm. For example, unsupervised learning methods such as ART normally require a first hidden layer quantity of nodes equal to the size of the input layer. Supervised learning systems are generally more flexible in the design of hidden layers. The remaining discussion focuses on backpropagation ANN systems or other similar supervised learning ANNs. The primary questions for the ANN designer concerning hidden layers are:

- how many hidden layers should exist in the ANN architecture and

- how many nodes should be present in the hidden layer(s)?

5.1. Number of Hidden Layers

While it is possible to design an ANN with no hidden layers, these types of ANNs can only classify input data that is linearly separable [17], which severely limits their application. Artificial neural networks that contain hidden layers have the ability to deal robustly with nonlinear and complex problems and therefore can operate on more interesting problems [32]. The quantity of hidden layers corresponds to the complexity of the domain problem to be solved. Single hidden layer ANNs create a hyperplane. Two hidden layer networks combine hyperplanes to form convex decision areas and three hidden layer ANNs combine convex decision areas to form convex decision areas that contain concave regions [14]. The convexity or concavity of a decision region corresponds roughly to the number of unique inferences or abstractions that are performed on the input variables to produce the desired output result.

Increasing the number of hidden unit layers enables a trade-off between smoothness and closeness-of-fit [2]. A greater quantity of hidden layers enables an ANN to improve its closeness-of-fit, while a smaller quantity improves the smoothness or extrapolation capabilities of the ANN. Several researchers have indicated that a single hidden layer architecture with an arbitrarily large quantity of hidden nodes in the single layer, is capable of modeling any categorization mapping [21,54], while others [19,51] have demonstrated that two hidden layer networks outperform their single hidden layer counterparts for specific problems. A heuristic for determining the quantity of hidden layers required by an ANN is: “As the dimensionality of the problem space increases (higher order problems), the number of hidden layers should increase correspondingly”. The number of hidden layers is heuristically set by determining the number of intermediate steps, dependent on previous categorizations, to translate the input variables into an output value. Therefore, domain problems that have a standard non-linear equation solution are solvable by a single hidden layer ANN [1].

5.2. Quantity of Nodes Per Hidden Layer

A trade-off exists when choosing the number of nodes to be contained in a hidden layer between training time and the accuracy of training. A greater number of hidden unit nodes results in a longer training period, while fewer hidden units provide faster training at the cost of having fewer feature detectors [9]. Too many hidden nodes in an ANN enable the ANN to memorize (over-fit) the training data set, which produces poor generalization performance [22]. Several quantitative heuristics exist for selecting the quantity of hidden nodes for an ANN such as: using 75 percent of the quantity of input nodes [23,28], using 50 percent of the quantity of input and output nodes [40], or using $2n + 1$ hidden layer nodes where n is the number of nodes in the input layer [13,39]. These algorithmic heuristics do not utilize domain knowledge for estimating the quantity of hidden nodes and may be counterproductive.

Jain and Nag [23] use the 75 percent rule, for their pricing of initial public offerings ANN, to show that an eleven input variable ANN with six or seven hidden nodes produces the best results on the training examples, but an ANN with twelve hidden nodes produces the best generalization result. As with the knowledge acquisition and elimination of correlated variables heuristic for defining the optimal input node set, the number of decision factors (DF) heuristically determines the optimal number of hidden units for an ANN. Knowledge acquisition or existing knowledge bases may be used to determine the DF for a particular domain and consequently the hidden layer architecture and optimal quantity of hidden nodes [1,14,47]. Decision factors are the separable elements which serve to form the unique categories of the input vector space. The DF can be equated to the collection of heuristic production rules used in an expert system.

The NETTalk neural network research [42] provides an illustration of the DF design principle. NETTalk has 203 input nodes representing seven textual characters and 33 output units representing the phonetic notation of the spoken text words. Hidden units are varied from zero to 120. The researchers claim that improved output accuracy is obtained as the number of hidden units is increased from zero to 120, but only a minimal improvement in the output accuracy is observed between 60 and 120 hidden units. This indicates that the quantity of DF for the NETTalk problem was close to 60 and adding hidden units beyond 60 served to increase the training time while not providing any appreciable difference in the ANN's performance.

Other research provides additional examples to illustrate that understanding the DF for a domain enables efficient design of hidden layers. Patuwo et al. [39] use various quantities of hidden nodes (3,6,8) in an ANN that solves a two-group classification problem given two input variables. The DF for this problem can be inferred to be two. While Patuwo et al. did not use a hidden layer with two nodes, the best generalization performance is reported to come from the ANN with three hidden nodes. A similar two-group classification problem, with an inferred DF of two, is addressed by Hung et al. [22] and they report that an ANN with two hidden nodes produces the best generalization. The two hidden node architecture used by Hung et al. produced a performance accuracy two to three times better than ANNs with three to six hidden nodes.

Another example is provided by Fletcher and Goss [13] who vary the quantity of hidden nodes from three to six, in a neural network to predict bankruptcy. Their results indicate that ANN performance improves by more than two percent from three to four hidden nodes and then dramatically the performance falls (9.8 percent decrease) as additional hidden nodes are added. The additional nodes beyond four enabled Fletcher and Goss's ANN to memorize the training data set, by providing a one-to-one mapping between input values and the corresponding output values, which decreased the generalization performance of the ANN.

In each of the examples discussed above, the ANNs performed poorly until a sufficient number of hidden units were available to represent the correlations between the input vector and the desired output values and increasing the number of hidden units beyond the sufficient number served to increase training time without a corresponding increase in output accuracy. Knowledge acquisition is necessary to determine the optimal input variable set to be used in an ANN system. During the knowledge acquisition phase, additional knowledge engineering can be performed to determine the DF and subsequently the minimum number of hidden units required by the ANN architecture. The ANN designer must acquire the heuristic rules or clustering methods used by domain experts, similar to the knowledge that must be acquired during the knowledge acquisition process for expert systems [50]. The number of heuristic rules or clusters used by domain experts are equivalent to the DF used in the domain.

For example, the Model 1 ANN for predicting currency exchange cable rates described in Section 3 has three DF: the anticipated change to the exchange rate, anticipated changes in relative interest rates, and anticipated action by the Federal Reserve or other government entity [51]. The DF indicate the quantity of nodes for the final hidden layer (layer 2 in this case). Because the

Model 1 cable rate forecasting ANN predicts two cable rate values, the optimal quantity of hidden nodes is 3 or 6. Additionally, the decision making process in forecasting cable rate changes is a multi-step process indicating the need for two (or three) hidden layers. The quantity of hidden nodes required in the first layer is heuristically estimated to be from 16 to 19 (16 if the Swiss franc and German mark are in fact correlated). The Model 1 ANN architectures briefly presented in Section 3 are extended to include models with a first hidden layer of 9 to 21 nodes in the first layer and 3 to (n - 3) in the second layer (except for the 21 node ANN that was stopped after 12 nodes in the second layer), where n is the quantity of hidden nodes in the first layer. Results for these ANN architectures for predicting the dollar/pound cable rate only, shown in Table 4, indicate that 15 to 18 first layer nodes and 6 second layer nodes, corresponding to the DF (3 DF * 2 output values) produce the best forecast accuracies. The poor performance of the ANN architectures with 21 nodes in the first layer is caused by over-fitting (memorizing) the data.

Recent research [3,36] has demonstrated techniques for automatically producing an ANN architecture with the exact number of hidden units required to model the DF for the problem space. The general philosophy of these automatic methods is to initially create a neural network architecture with a very small or very large number of hidden units, train the network for some pre-determined number of epochs and evaluate the error of the output nodes [14,47]. If the error exceeds a threshold value then a hidden unit is added or deleted respectively and the process is repeated until the error term is less than the threshold value. Another method to automatically determine the optimum architecture is to use genetic algorithms to generate multiple ANN architectures and select the architectures with the best performance [37]. Determining the optimum number of hidden units for a ANN application is a very complex problem [36] and a method for automatically determining the DF quantity of hidden units without performing the corresponding knowledge acquisition remains a current research topic.

Hidden Units First Layer	Hidden Units Second Layer	Forecast Accuracy Dollar/Pound
9	3	0.5135
9	6	0.5405
12	3	0.5405
12	6	0.5405
12	9	0.5405
15	3	0.5405
15	6	**0.5676**
15	9	0.5405
15	12	0.5405
18	3	**0.5676**
18	6	**0.5676**
18	9	0.5405
18	12	0.5405
18	15	0.5405
21	3	0.4867
21	6	0.5135
21	9	0.5135
21	12	0.4324

Table 4: Effect of hidden units on Model 1 forecast accuracy.

In this section, the heuristic design principle of acquiring decision factors to determine the quantity of hidden nodes and the configuration of hidden layers has been presented. A quantity of hidden nodes equal to the quantity of the DF is required by an ANN to perform robustly in a domain and produce accurate results. This concept is similar to the principle of a minimum size input vector determined through knowledge acquisition presented in Section 3. The knowledge acquisition process for ANN designers must acquire the heuristic decision rules or clustering methods of domain experts. The DF for a domain are equivalent to the heuristic decision rules used by domain experts. Further analysis of the DF to determine the dimensionality of the problem space enables the knowledge engineer to configure the hidden nodes into the optimal number of hidden layers for efficient modeling of the problem space.

6. Conclusions

General guidelines for the development of artificial neural networks (ANNs) are few, so this paper presents several heuristics for developing ANNs that produce optimal generalization performance, as outlined in Figure 2. Extensive knowledge acquisition is the key to the design of ANNs. First the correct input vector for the ANN must be determined by capturing all relevant decision criteria used by domain experts for solving the domain problem to be modeled by the ANN and eliminating correlated variables. Next, the selection of a learning method is an open problem and an appropriate learning method can be selected by examining the set of constraints imposed by the collection of available training examples for training the ANN. Finally, the architecture of the hidden layers is determined by further analyzing a domain expert's clustering of the input variables or heuristic rules for producing an output value from the input variables. The collection of clustering/decision heuristics used by the domain expert has been called the set of decision factors (DF). The quantity of DF is equivalent to the minimum number of hidden units required by an ANN to correctly represent the problem space of the domain.

1. Perform extensive knowledge acquisition. This knowledge acquisition should be targeted at identifying the necessary domain criteria (information) required for solving the problem and identify the decision factors that are used by domain experts for solving the type of problem to be modeled by the ANN.
2. Once the relevant domain information/criteria are identified as potential input values to the ANN, remove noise variables.
   - A. Identify highly correlated variables via a Pearson correlation Matrix or Chi-square test (and keep only one correlated variable).
   - B. Identify and remove non-contributing variables (depending on data distribution and type) via discriminant/factor analysis or step-wise regression.
3. Analyze the demographic features of the data and decision problem to select an ANN learning method. If supervised learning methods are applicable then implement backpropagation in addition to any other method indicated by the data demographics (i.e., radial-basis function for small training sets or counterpropagation for very noisy training data).
4. Make sure that an adequate quantity of training data is available for the selected method. Recall that time-series models differ from standard categorization models (1 to 2 years of data vs. 4 times the number of weighted connections).
5. Analyze the complexity (number of unique steps) of the traditional expert decision making solution to determine the number of hidden layers. If in doubt, then use a single hidden layer, but realize that additional nodes (processing elements) may be required to adequately model the domain problem.
6. From the previous knowledge acquisition step (1.), set the quantity of hidden nodes in the last hidden layer equal to the decision factors (expert heuristics) used by domain experts to solve the problem.

Figure 2. Heuristic Principles for Design of Artificial Neural Networks

Use of the knowledge-based design heuristics enables an ANN designer to build a minimum size ANN that is capable of robustly dealing with specific domain problems. The future may hold automatic methods for determining the optimum configuration of the hidden layers for ANNs. Minimum size ANN configurations guarantee optimal results with the minimum amount of training time.

References

[1] Bansal, A., Kauffman, R. J., & Weitz R. R. "Comparing the Modeling Performance of Regression and Neural Networks As Data Quality Varies: A Business Value Approach", *Journal of Management Information Systems*, Vol. 10 No. 1, 1993, pp. 11-32.

[2] Barnard, E., & Wessels, L. "Extrapolation and Interpolation in Neural Network Classifiers", *IEEE Control Systems*, Vol. 12 No. 5, 1992, pp. 50-53.

[3] Bartlett, E. "Dynamic Node Architecture Learning: An Information Theoretic Approach", *Neural Networks*, Vol. 7 No. 1, 1994, pp. 129-140.

[4] Bejou, D., Wray, B., & Ingram, T. "Determinants of Relationship Quality: An Artificial Neural Network Analysis", *Journal of Business Research*, Vol. 36, 1996, pp. 137-143.

[5] Benjamin, C. O., Chi, S., Gaber, T., & Riordan, C. A. "Comparing BP and ART II Neural Network Classifiers for Facility Location", *Computers and Industrial Engineering*, Vol. 28 No. 1, 1995, pp. 43-50.

[6] Carpenter, G. A., & Grossberg, S. "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network", *Computer*, Vol. 21 No. 3, 1988, pp. 77-88.

[7] Carpenter, G. A., Grossberg, S., Markuzon, N., & Reynolds, J. H. "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Learning of Analog Multidimensional Maps", *IEEE Transactions on Neural Networks*, Vol. 3 No. 5, 1992, pp. 698-712.

[8] Cherkassky, V., & Lari-Najafi, H. "Data Representation for Diagnostic Neural Networks", *IEEE Expert*, Vol. 7 No. 5, 1992, pp. 43-53.

[9] Dayhoff, J. *Neural Network Architectures: An Introduction*, New York: Van Nostrand Reinhold, 1990.

[10] Falas, T., Charitou, A, & Charalambous, C. "The Application of Artificial Neural Networks in The Prediction of Earnings", *Proceedings IEEE International Conference on Neural Networks*, 1994, pp. 3629-3633.

[11] Fausett, L., & Elwasif, W. "Predicting Performance from Test Scores Using Backpropagation and Counterpropagation", *Proceedings of IEEE International Conference on Neural Networks*, 1994, pp. 3398-3402.

[12] Fish, K. E., Barnes, J. H., & Aiken, M. W. "Artificial Neural Networks: A Methodology for Industrial Market Segmentation", *Industrial Marketing Management*, Vol. 24, No. 5, 1995, pp. 431-438.

[13] Fletcher, D., & Goss, E. "Forecasting With Neural Networks: An Application Using Bankruptcy Data", *Information and Management*, Vol. 24 No. 3, 1993, pp. 159-167.

[14] Fu, L. *Neural Networks in Computer Intelligence*, New York: McGraw-Hill, 1994.

[15] Grudnitski, G., & Osburn, L. "Forecasting S&P and Gold Futures Prices: An Application of Neural Networks", *The Journal of Futures Markets*, Vol. 13, No. 6, 1993, pp. 631-643.

[16] Hammerstrom, D. “Neural Networks At Work”, *IEEE Spectrum*, Vol. 30 No. 6, 1993, pp. 26-32.

[17] Haykin, S. *Neural Networks: A Comprehensive Foundation*, New York: Macmillan, 1994.

[18] Hecht-Nielsen, R. “Applications of Counterpropagation Networks”, *Neural Networks*, Vol. 1, 1988, pp. 131-139.

[19] Hertz, J., Krogh, A., & Palmer, R. *Introduction To The Theory of Neural Computation*, Reading, MA: Addison-Wesley, 1991.

[20] Hopfield, J. J., & Tank, D. W. “Computing With Neural Circuits: A Model”, *Science*, Vol. 233 No. 4764, 1986, pp. 625-633.

[21] Hornik, K., Stinchcombe, M., & White, H. “Multilayer Feedforward Networks Are Universal Approximators”, *Neural Networks*, Vol. 2 No. 5, 1989, pp. 359-366.

[22] Hung, M. S., Hu, M. Y., Shanker, M. S., & Patuwo, B. E. “Estimating Posterior Probabilities in Classification Problems With Neural Networks”, *International Journal of Computational Intelligence and Organizations*, Vol. 1 No. 1, 1996, pp. 49-60.

[23] Jain, B. A., & Nag, B. R. “Artificial Neural Network Models for Pricing Initial Public Offerings”, *Decision Sciences*, Vol. 26 No. 3, 1995, pp. 283-302.

[24] Kohonen, T. *Self-Organization and Associative Memory*, Berlin: Springer-Verlag, 1988.

[25] Kou, C., Shih, J., Lin, C., & Lee, Z. “An Application of Neural Networks to Reconstruct Crime Scene Based on Non-Mark Theory - Suspicious Factors Analysis”, In P. K. Simpson (Ed.), *Neural Networks: Theory, Technology, and Applications*, New York: IEEE Press, 1996, pp. 537-543.

[26] Kulkarni, U. R., & Kiang, M. Y "Dynamic grouping of parts in flexible manufacturing systems - A self-organizing neural networks approach", *European Journal of Operational Research*, Vol. 84, No. 1, 1995, pp. 192-212.

[27] Lacher, R. C., Coats, P. K., Sharma, S. C., & Fant, L. F. "A neural network for classifying the financial health of a firm", *European Journal of Operational Research,* Vol. 85, No. 1, 1995, pp.53-65.

[28] Lenard, M. J., Alam, P., & Madey, G. R. "The Application of Neural Networks and a Qualitative Response Model to the Auditor's Going Concern Uncertainty Decision", *Decision Sciences*, Vol. 26 No. 2, 1995, pp. 209-227.

[29] Lenat, D. B., & Feigenbaum, E. A. "On the Thresholds of Knowledge", *Technical report AI-126-87*, Austin, TX: Microelectronics and Computer Technology Corporation (MCC), 1987.

[30] Li, E. Y. "Artificial neural networks and their business applications", *Information & Management,* Vol. 27, No. 5, 1994, pp. 303-313.

[31] McLeod, R. W., Malhotra, D. K., & Malhotra, R. "Predicting Credit Risk: A Neural Network Approach", *Journal of Retail Banking*, Vol. 15, No. 3, 1993, pp. 37-40.

[32] Medsker, L., & Liebowitz, J. *Design and Development of Expert Systems and Neural Networks,* New York: Macmillan, 1994.

[33] Mehra, P., & Wah, B. W. *Artificial Neural Networks: Concepts and Theory,* New York: IEEE Press, 1992.

[34] Mendenhall, W., & Sincich, T. *A Second Course in Statistics: Regression Analysis,* Upper Saddle River, NJ: Prentice Hall, 1996.

[35] Moody, J., & Darken, C. J. “Fast Learning in Networks of Locally-Tuned Processing Elements”, *Neural Computation*, Vol. 1 No. 2, 1989, pp. 281-294.

[36] Nabhan, T., & Zomaya, A. “Toward Generating Neural Network Structures for Function Approximation”, *Neural Networks*, Vol. 7 No. 1, 1994, pp. 89-99.

[37] Opitz, D., & Shavlik, J. “Genetically Refining Topologies of Knowledge-Based Neural Networks”, *International Symposium on Integrating Knowledge and Neural Heuristics*, Pensacola, FL, 1994, pp. 57-66.

[38] Pakath, R., & Zaveri, J. S. “Specifying Critical Inputs in a Genetic Algorithm-driven Decision Support System: An Automated Facility”, *Decision Sciences,* Vol. 26 No. 6, 1995, pp. 749-779.

[39] Patuwo, E., Hu, M. Y., & Hung, M. S. “Two-Group Classification Using Neural Networks”, *Decision Sciences*, Vol. 24 No. 4, 1993, pp. 825-845.

[40] Piramuthu, S., Shaw, M., & Gentry, J. “A classification approach using multi-layered neural networks”, *Decision Support Systems*, Vol. 11 No. 5, 1994, pp. 509-525.

[41] Schocken, S. & Ariav, G. “Neural Networks for Decision Support: Problems and Opportunities”, *Decision Support Systems*, Vol. 11 No. 5, 1994, pp. 393-414.

[42] Sejnowski, T., & Rosenberg, C. “Parallel Networks That Learn to Pronounce English Text”, *Complex Systems*, Vol. 1 No. 1, 1987, pp. 145-168

[43] Smith, M. *Neural Networks for Statistical Modeling*, New York: Van Nostrand Reinhold, 1993.

[44] Soulié, F. "Integrating Neural Networks for Real World Applications", In Zurada, Marks, & Robinson (Eds.), *Computational Intelligence: Imitating Life*, New York: IEEE Press, 1994, pp. 396-405.

[45] Specht, D. F. "A General Regression Neural Network", *IEEE Transactions on Neural Networks*, Vol. 2 No. 6, 1991, pp. 568-576.

[46] Tahai, A., Walczak, S., & Rigsby, J. T. "Improving Artificial Neural Network Performance Through Input Variable Selection", In P. Siegel, K. Omer, A. deKorvin, & A. Zebda (Eds.) *Applications of Fuzzy Sets and The Theory of Evidence to Accounting II*, Stamford, Connecticut: JAI Press, 1998, pp. 277-292.

[47] Towell, G. G., Shavlik, J. W., & Noordewier, M. O. "Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks", *Proceedings Eighth National Conference on Artificial Intelligence*, Boston, 1990, pp. 861-866.

[48] Walczak, S. "Neural Network Models for A Resource Allocation Problem", *Transactions on Systems, Man and Cybernetics*, Vol. 28 B No. 2, 1998a, pp. 276-284.

[49] _____ "Information Effects on Neural Network Forecasting Model Accuracy", *1998 Proceedings Decision Sciences Institute Annual Meeting*, 1998, in press.

[50] _____ "A Modified Decision Tree Approach for Evaluating the Potential for Application of Neural Networks and Expert Systems", *Journal of Computer Information Systems*, Vol. 36 No. 4, 1996, pp. 1-6.

[51] _____ "Developing Neural Nets for Currency Trading", *Artificial Intelligence in Finance*, Vol. 2 No. 1, 1995, pp. 27-34.

[52] _____ "Categorizing University Student Applicants With Neural Networks", *Proceedings of IEEE International Conference on Neural Networks*, 1994, pp. 3680-3685.

[53] Weigand, A. S., & Zimmermann, H. G. "The Observer-Observation Dilemma in Neuro-Forecasting: Reliable Models From Unreliable Data Through CLEARNING", Proceedings of *Artificial Intelligence Applications on Wall Street*, New York, 1995, pp. 308-317.

[54] White, H. "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings", *Neural Networks*, Vol. 3, 1990, pp. 535-549.

[55] Widrow, B., Rumelhart, D., & Lehr, M. "Neural Networks: Applications in Industry, Business, and Science", *Communications of the ACM*, Vol. 37 No. 3, 1994, pp. 93-105.

[56] Wilson, R. L., & Sharda, R. "Bankruptcy prediction using neural networks", *Decision Support Systems*, Vol. 11, No. 5, 1994, pp. 545-557.