IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5223

This paper appears in *Managing Modern Organizations Through Information Technology*, Proceedings of the 2005 Information Resources Management Association International Conference, edited by Mehdi Khosrow-Pour.

Building Optimal Back-Propogation Trained Neural Networks for Firm Bankruption Predictions

Gary Qing Guan
20th Century Fox, 2121 Avenue of Starts, Century City, CA 90213, USA, gary.guan@fox.com

Jim Q. Chen
St. Cloud State University, St. Cloud, MN 56301, USA, jchen@stcloudstate.edu

INTRODUCTION

There have been considerable research activities in developing optimal neural networks during the past couple of decades. Optimal neural networks have good generalization ability and incur less computational cost due to their simple structures. A simpler neural network may approximate human judgement more closely [12] and may provide fewer "rules" or simpler formulae describing the network's behavior than complex neural networks. Several techniques have been developed to obtain optimal neural network structures. They include pruning techniques [15,11], weight decay techniques [3], network construction techniques such as upstart [2] and tiling algorithm [7], and network selection techniques [4,8].

Although each method has given encouraging results both in terms of generalization and finding efficient architectures on simple test problems, it is not yet clear which of the methods described is best for a given decision-making problem [5]. Frean [2] conducted a comparative test in which the upstart algorithm used fewer units than the tiling algorithm. The weight decay method is easy to implement and easy to use because of its close relation to the back-propagation algorithm [3]. However, the pruning technique requires the user to pay constant attention to each hidden unit's output for each input pattern [15]. There is a need for more effective and robust pruning techniques.

This study focuses on the design of an optimal neural network model for firm bankrupting prediction task. A new direct pruning algorithm will be proposed.

The article is organized as follows. A brief literature review on related optimal neural network techniques will be provided in the next section. The new pruning technique for developing simplified back-propagation trained networks will be discussed. Then, the evaluations of the proposed method will be discussed. The performance of the neural network is compared to that of multivariate discriminant analysis models for matched bankruptcy samples. The article concludes with summary and future research opportunities.

LITERATURE REVIEW

One of the techniques to optimize the architecture of a network is to "prune" the network [15]. Pruning involves a process of examining a network, determining which units are not necessary to the solution, and removing those units. In this process, the network is analyzed by examining the outputs of the hidden units across all the training set inputs. Two stages of pruning have been identified. The first stage involves the removal of units that can be considered as not contributing to the solution. These units are the ones which either have approximately constant outputs across the training set, or have outputs across the training set that mimic the outputs of another unit. These units can be removed and the weights assigned to their outputs redistributed in such a way as to make almost no change to the network's performance over the training set. The second stage involves the removal of units that are independent of the other units in the layer but give information that is not required at the next layer. The primary problem with this pruning technique is that there is no mechanism built into the algorithm to identify the unnecessary hidden units. Unnecessary units must be identified via manual inspection. For a large and complex problem requiring hundreds of hidden units and thousands of patterns, manual inspection is not practical. Another approach is to have the network itself remove non-useful units during training. There are several ways to do this. Mozer and Smolensky [11] presented a "skeletonization" technique that trimmed the "fat" from a network via a relevance assessment. The relevance of unit i, μ_i, is defined as

$$\mu_i = E_{\text{without unit } i} - E_{\text{with unit } i} \quad (1)$$

where E is the error of the network on the training set. Since μ_i is difficult to calculate, a good approximation of μ_i is used in the skeletonization process. Based on approximate measures of relevance, μ_i, input or hidden units that are most critical to performance are identified, and the least relevant units are trimmed from the structure. This "skeletonization" technique can be used to simplify networks by removing units that convey redundant information; to improve learning performance by first learning with spare hidden units and then trimming the unnecessary ones away, thereby constraining generalization; and to understand the behavior of networks in terms of minimal rules. The process has been tested with a number of problems and, in most cases, has been found to perform well.

The most promising and effective approaches in building an optimal network structure for a particular task are weight decay techniques [3]. Under these techniques, the network architecture is simplified gradually as unnecessary units and connections are removed from the network.

During training, the network is supposed to preferentially remove less useful connection weights. This is accomplished by introducing a mechanism making it possible for each connection weight w_{ij} to decay to zero. The desired result is that those connections receiving insufficient reinforcement will disappear automatically.

The weight decay method is easy to implement and easy to use [3]. However, there are some difficulties in using this method. During the initial training phase for almost all back-propagation networks, connection weights increase very slowly.

$$w_{ij}^{*} = (1 - \varepsilon) w_{ij} \quad (2)$$

Even when the decay parameter ε is very small, all of the weights decay to zero after only a few iterations over the set of training patterns. The vectors of hidden unit weights become equal with further training. Because the weight vectors become equal, the network degenerates to a single hidden unit. In general, the single hidden-unit function network can never be trained successfully because most tasks require more than one hidden unit. This suggests that weight decay methods are of more practical value when applied to the network only after it has been trained.

Most of the weight pruning methods tend to remove the smaller weights without making large changes in error function E_0. These methods are sensitive to the decay parameter ε, and the learning rate η setting. Also, it is hard to control the final output fitting rate, or error. Clearly, there is a need for a more effective and robust pruning technique.

BUILDING OPTIMA NETWORK WITH DIRECT PRUNING METHOD

Because of the difficulties with the weight decay methods, we introduce a new pruning method, which directly eliminates the smallest and least influential weights. Due to page limitation, we will skip the mathematical foundation in this manuscript.

A Direct Pruning Algorithm

The direct pruning algorithm for back-propagation trained neural network consists of seven steps. According to the algorithm, the weight with the smallest magnitude is pruned away sequentially as long as the fitting rate (defined as the fraction of the input samples for which the network gives acceptable outputs) of each newly simplified network remains better than a prescribed value. Because 0.5 is the average of the normalized input signals, and therefore of all unit inputs in the network, each single weight removal is compensated for by subtracting one half of the value of the deleted weight from the threshold of the unit that loses the zeroed weights' input contribution. This weight pruning can be continued until the prescribed fitting rate is no longer satisfied. A hidden unit, with all zero inputs, has a constant output. Such a hidden unit can be removed by compensating for the threshold of following layer's units for these constant contributions. An input unit can be removed if all its connections are trimmed away during the simplification process.

- Step 1. Search all the weights in all layers of the network to find the weight having the smallest magnitude and delete it.
- Step 2. Subtract one half of the deleted weight from the threshold for the unit that was fed by the weight found in step 1.
- Step 3. Test this pruned network using the entire training data set. If the computed fitting rate is better than the externally prescribed value, accept this pruned network and repeat the process starting with step 1; otherwise, undo the last change performed in steps 1 and 2 and proceed to step 4.
- Step 4. Search all the weights in the whole network except those connected to the output layer to find the weight having the smallest magnitude and delete it.
- Step 5. Subtract one half of the deleted weight from the threshold for the unit that was fed by the weight found in step 4.
- Step 6. Test this pruned network using the entire training data set. If the computed fitting rate is better than the externally prescribed value, accept this pruned network and repeat the process starting with step 4. Otherwise, undo the last changes in steps 4 and 5, and proceed to step 7.
- Step 7. Delete all the hidden and input units j for which all input weights are zero or all weights leading from their output are zero. For the units having all zero input weights, reduce the threshold of unit i at the following layer by $w_{ij}\ f(-\theta_j)$, where f is the activation function for hidden unit, θ_j is the threshold value of deleted unit j, and $f(-\theta_j)$ is therefore the constant output of this unit j.

This pruning method removes the weights by first calculating their contributions to the performance of the network. Then the isolated hidden and input units are trimmed. While it is similar to the skeletonization method [11], our direct pruning method possesses distinct advantages. First, unlike skeletonization method which trims hidden units on the basis of some additional measurement of the contributions of each hidden unit, our method requires no additional computation and is simpler and much easier to control. Second, the direct pruning method can create a simplified network structure in which some units are not fully connected.

EVALUATION OF THE PROPOSED METHOD

The proposed direct pruning algorithm is implemented in C programming language along with the back-propagation procedure. The programs were run on an HP9000 mainframe machine with UNIX operating system.

The method was evaluated first by using two common benchmark problems: XOR and Parity. Then, it was applied to a real-world business application: firm bankruptcy prediction. The performance of the simplified neural networks was compared to that of multivariate discrimination analysis models. Due to page limitation, we will not discuss XOR and Parity evaluations and the intermediate steps of network training and simplifications.

Prediction of Firm Bankruptcy

The same procedure for developing a simplified network is used for the firm bankruptcy prediction problem. The testing process will be embedded in the training and simplification process. The inputs to the neural network are financial variables associated with a firm, and the output is the status of the firm: bankrupt or non-bankrupt. Many past bankruptcy prediction studies using discriminant analysis regard the Altman [1] study as the standard of comparison. To validate the neural network's performance, the same five financial ratios used in the Altman study are used in this study.

Collect Data

The data sample consists of the five financial ratios on 129 firms that either were in operation or went bankrupt during the period 1975-1982. The data were obtained from Moody's Industrial Manuals. Among the 129 firms, 65 went bankrupt during the period and 64 firms, matched on industry and year to bankrupt firms, were non-bankrupt. Data used for the bankrupt firms is from the last financial statements issued before the firms declared bankruptcy. For training purposes, the output of the network is a pair of binary digits, (1,0) or (0,1). Bankrupt is represented as (1,0) and non-bankrupt as (0,1). The training set, bankrupt1.trn (see Appendix), consists of 35 bankrupt firms and 34 non-bankrupt firms. The testing set, bankrupt1.tes (see Appendix), consists of 30 bankrupt firms and 30 non-bankrupt firms.

Network Structure Design

The design of the network structure follows naturally from the selection of inputs and outputs. Five input units represent the financial ratios labeled as x_1, x_2, x_3, x_4 *and* x_5. The two output units are interpreted as bankrupt (1,0) or non-bankrupt (0,1). According to Kolmogrov's theory [6], the maximum number of hidden units required to perform the classification is 11. The actual number of hidden units chosen for this study was 10. See Figure 1.

Network Training and Simplification

All the connections in the fully connected network were randomized before training. Then the training samples were presented to the network in random order to maximize performance and to minimize the introduction of bias.

The training and simplification process generated five progressive versions of the network, labeled from Net_0 to Net_4. Figure 2 shows the final network Net_4 structure. The initial network has five (5) input units,

Figure 1. Initial Network (Net_0) Structure for Firm Bankruptcy Prediction

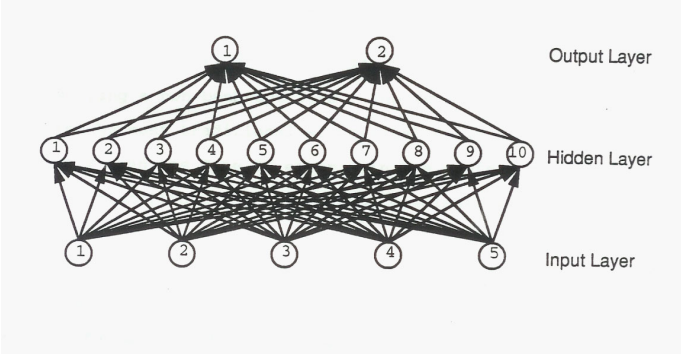
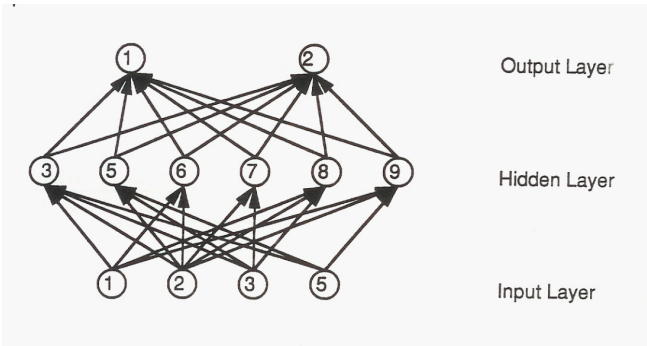


Figure 2. Net_4 Network Structure for Firm Bankruptcy Prediction



ten (10) hidden units, and two (2) output units, indicated as (5,10,2) in the figure. The arrow links with labels such as "training" or "simplifying" indicate the facilitating processes involved in the evolution. When two processes are indicated at the same juncture, the one appearing above the arrow was applied first. For example, Net_2 progresses to Net_3 through the application of, first, simplification and then retraining. The letter "F" indicates a fully connected network and "P" a partially connected network. The percentage figures in each version are the combined classification accuracy for both bankruptcy and non-bankruptcy samples.

Table 3 shows the training and testing results before the network was simplified. At a tolerance rate of 0.05, the performance of Net_0 on the training set is presented in parts a and b of Table 3. Net_0 is extremely accurate in classifying 97.1% of the training sample correctly. The Type I error rate proved to be only 5.7%, while the Type II error rate was 0%. The standard deviation of L_2 norm error is 0.044. The Type I error is the probability of misclassifying a bankrupt firm; the Type II error is the probability of misclassifying a non-bankrupt firm. The Type I error is considered by most traditional statistical analyses as being more costly than the Type II error. In the context of firm bankruptcy classification, the cost of misclassifying a bankrupt firm is greater than that of misclassifying a non-bankrupt firm. To keep the Type I error rate smaller is our ultimate goal.

The discriminating ability of the network for the testing set is presented in parts c and d of Table 1. Although the accuracy of the classification is reduced, 90% correct assignment is evidence that Net_0 can be used to predict firm bankruptcy. Because no learning takes place on the testing set, both the Type I error rate and the standard deviation of L_2 norm error are increased considerably.

Table 2 lists the classification matrix for Net_4. At a tolerance rate of 0.05, Net_4 retains the classification accuracy of 97.1% and has a smaller Type I error rate of 2.9% for the training set. For the testing set, compared to an accuracy of 90% of Net_0 for a Type I error rate 16.7%, Net_4 has superior classification ability of 96.7% and committed no Type I errors. Net_4 has a standard deviation of L_2 norm error of 0.241 for the training set and 0.211 for the testing set.

Table 3 summarizes the performance of the networks. For bankrupt firms in the testing set, which was not used to train the network, the classification accuracy gradually increased as the evolution of the network progressed. This is particularly important because the goal of building the simpler neural network is to improve the accuracy of predicting future bankruptcies.

Multivariate Discriminant Analysis

The same two sets of sample data used in training and testing the simpler neural network model were used to develop the multivariate discriminant models. The discriminant models were developed using SPSS. The bankrupt firms are classified as B, and the non-bankrupt ones are classified as NB. The discriminant function for the first set of sample data is as follows:

$$Z_1 = -0.125 + 0.302\,x_1 + 3.034\,x_2 + 2.134\,x_3 + 2.707\,x_4 - 2.377\,x_5 \qquad (3)$$

Where $x_i (i = 1,2,3,4,5)$ represents the same five financial ratios used for the network design.

Table 4 lists some statistics relevant to the discriminant function (3). The quantity $1-\lambda$ (Wilks' λ) indicates the proportion of variance of the group (B and NB) accounted for by the each individual variable (part a of Table 4). The F test determines the significance of each ratio in the discriminant function. The level of significance for the F test is also listed in the table. The relative contribution of each variable to the total discriminating power of function (3) is listed in part b of Table 6. Variable 4 (x_4) has the lowest contribution to the discriminant function (part b of Table 4) and its F statistic is not significant at level 0.05 (part a of Table 4). This finding is supported by the neural network's feature selection. For the second set of sample data, the discriminant function is as follows:

$$Z_2 = -0.582 - 0.528\,x_1 + 7.163\,x_2 + 2.856\,x_3 - 1.709\,x_5 \qquad (4)$$

Because the second set of data does not includes the fourth variable (x_4), the discriminant function is the linear combination of the remaining four variables. Table 5 shows the relevant statistics for equation (4).

Two tests were performed on each of the two models using the selected samples. The first test was conducted using the model training sample, and the second test was conducted using the testing sample.

Comparisons

The performance of the neural network models was compared to that of the MDA models for the matched samples. The number of firms correctly classified, and the Type I & II errors committed by each model, are reported in Tables 6, 7, and 8.

For model evaluating purposes, the p-values from the nonparametric test of equality of proportions [10] have also been reported in these tables. The null hypothesis that the proportion of firms correctly classified by each technique (or matched models) is the same as tested using a nonparametric test since the data is categorical.

Original Network Model Net_0 vs Simplified Network Model Net_4

In Table 6, the number of firms correctly classified, along with Type I & II error rates, for Net_0 and Net_4 are presented for both the training and testing set. The classification accuracy of the simplified network is the same as the original network for the training set. However, a significant difference was detected at the 15 percent level between the two networks' classifications abilities using the testing set. Furthermore, for a p-value as small as 0.02, there is strong evidence that the risk of misclassifying a bankrupt firm as a non-bankrupt one (Type I error) is much smaller for the simplified network than for the original network. These results support the claim of this study that a simplified network

model has better generalization ability than a complex network model.

Original Network Model Net_0 vs Five-Variable MDA Model

Net_0 misclassifications for the training and testing sets were compared to misclassifications by the five-variable MDA model. The results appear in Table 7. There is strong evidence (p=0.01) for rejection of the null hypothesis for the training set, but no significant difference is found between the classification rates made by these two models for the testing set. It is important to note that a significant difference at the 10% level was indicated for the Type I error rate and at the 5% level for the Type II error rate for the training set. There is no significant difference for the testing set.

Simplified Network Model Net_4 vs Four-Variable MDA Model

The comparative results of the simplified network Net_4 and four-variable MDA model were listed in Table 8. Although the null hypothesis is true in terms of overall accuracy for the testing set, there is a significant difference at the 10% level for the Type I error rate. Considering the cost of committing a Type I error, the simplified neural network is the preferable prediction technique. For the training set, the simplified network committed lower overall misclassification as well as Type I errors, and the differences are significant at the 1% and 2% level.

In summary, the neural network approach provided better firm bankruptcy prediction accuracy than the MDA method. In most cases, the classification results of the neural network models were significantly superior to the matched MDA models. Furthermore, the simplified neural network model enjoyed the highest overall classification accuracy and the lowest Type I error rate. It has been demonstrated that a neural network's generalizing ability can be improved through simplification of its structure using the pruning process proposed in this study.

CONCLUSION

A process to build a simpler network structure was proposed. The proposed process was evaluated using a real-world application problem: firm bankruptcy prediction. The performance of neural networks was compared to that of multivariate discrimination analysis models for matched bankruptcy samples. The simplified neural network structure offered a superior modeling approach for firm bankruptcy prediction. For each matched data set examined in this application, the neural network with the simplified structure performed as well as or better than both the non-simplified neural network and the MDA models. The simpler network structure committed significantly fewer Type I errors than both the non-simplified network and the MDA models. This is an important result due to the high cost associated with the commitment of a Type I error.

** Tables, figures and references are available upon request*