1 Initializing Neural Networks using Decision Trees

Arunava Banerjee

1.1 Introduction

The task of inductive learning from examples is to find an approximate definition for an unknown function f($\mathbf{x}$), given training examples of the form $\langle x_i, f(x_i)\rangle$. Existing algorithms to solve this problem follow either of two basic approaches. Decision tree methods, such as ID3 (Quinlan, 1986) and CART (Breiman, Friedman, Olshen, and Stone, 1984) construct trees whose leaves are labeled with the predicted classification. The connectionist methods, on the other hand, apply neural network learning algorithms such as the perceptron algorithm (Rosenblatt, 1958), and the error-backpropagation algorithm (Rumelhart, Hinton and Williams, 1986) that learn through incremental changes of weights in a network consisting of elementary units called neurons.

Comparative studies (Mooney, Shavlik, Towell, and Gove, 1989; Weiss, and Kapouleas, 1989) have shown that whereas the decision tree algorithms run significantly faster during training, the connectionist methods almost always perform better at classifying novel examples in the presence of noisy data.

The studies have also demonstrated that when comparisons are confined strictly to the connectionist approach, the back-propagation algorithm outperforms the perceptron algorithm in classifying real world datasets because real world datasets are often not linearly separable.

Back-propagation however, suffers from a host of drawbacks. First, it is extremely slow to train. Second, and more important, variable parameters like learning rate, momentum, and network topology affect the accuracy of the resulting classifier. Setting these parameters to their respective optimal values is a painstaking and time-consuming process.

When it comes to solving real-world problems, restrictions on time and accuracy become of utmost importance. This is to say that most applications require accurate real-time solutions to their respective problems. The methods mentioned above either come short of attaining the required accuracy for classifying novel examples, or train far too slowly to be functional in such environments.

This paper describes an algorithm which is much faster than back-propagation, and at the same time matches it in accuracy in classifying novel examples. The key idea is to construct a decision tree, convert it into an equivalent network, and then tune the network by training it over the same dataset for a short period of time.

Many researchers have proposed converting decision trees into equivalent neural networks - see for example (Utgoff, 1989), (Sethi, 1990), (Brent, 1991), and (Cios and Liu, 1992). The difference in our approach lies in the fact that whereas all these methods generate networks that classify as accurately as the input decision trees, our method, aided by subsequent tuning, outperforms the decision tree in its accuracy in classifying new examples.

Towell and Shavlik (1993) advance a technique of symbolic rules to neural network conversion and subsequent training, that resembles our approach. The conversion algorithm they employ is however, distinct and the resultant network has a different topology. Whereas the depth of the network in their approach depends upon the input ruleset, our approach creates a network with a fixed depth.

This paper is structured as follows. Section 1.2 demonstrates how a neural network may be derived from a decision tree. In addition, it highlights certain other advantages of the proposed method. Section 1.3 clarifies the algorithm by converting an example decision tree into an equivalent neural network. In section 1.4 experimental results pertaining to three different domains are presented and the new approach is compared to back-propagation and ID3 in their context. Section 1.5 discusses future research directions based on some known shortcomings of this approach, and section 1.6 presents concluding remarks.

1.2 The Conversion Technique

As has already been stated, the central idea of the technique is to apply a decision tree (created by any decision tree algorithm, for example ID3) to initialize the neural network. This network is subsequently trained using a connectionist method on the dataset to achieve an improved predictive accuracy. In this paper, C4.5 has been chosen as a proto-typical representative of the decision tree approach. Similarly, Back-propagation (**Bp**) has been chosen as a representative of the connectionist approach.

The new technique has three major advantages. First, the initial network performs almost as well as the decision tree on which it is based. In other words, even before the process of subsequent training starts, the network is a considerably accurate classifier of the dataset. Second, the procedure that converts the decision tree into a neural network also specifies the network topology. This eliminates the guess work that goes into the creation of a network topology. Third, as will be demonstrated through the experiments, this technique fares better than the best network classifier and decision tree, even when the values for momentum and learning rates are frozen across datasets. Consequently, one could eliminate the process of repeated training required to find the optimal values for such parameters, and at the same time retain the accuracy of the classifier.

An informal sketch of the technique is presented next. This is followed by the actual algorithm.

1.2.1 Informal Sketch

The technique presented in this paper converts uni-variate decision trees into equivalent neural networks. The network that is created has three layers (two hidden layers and one output layer). The two hidden layers are called the *literal* layer and the *conjunction* layer. The output layer is also called the *disjunction* layer.

Any data point is assumed to be an element of R^n tagged with a concept class that is a member of the set $\{1, 2, ..., M\}$. In other words, each example is assumed to be a vector of length $(n + 1)$ that comprises of n real valued attributes and an integral class that takes a value between 1 and M.

The network generated by the technique has exactly n input nodes and M output nodes. The n input nodes correspond to the n attributes in an example, and the ordinality of the node in the output layer that has the maximum activation corresponds to the classification of the example.

The technique first creates a decision tree by applying C4.5 on the dataset. This is followed by a phase of the creation of the actual network. Corresponding to each node in the decision tree, the technique requires the construction of a node in the literal layer that replicates the decision of that node in the tree.

The next phase involves the creation of nodes that correspond to branches in the tree. This is accomplished by assigning one node in the conjunction layer to each branch such that the node replicates the decision of that branch in the tree. The penultimate phase corresponds to the act of grouping together all branches that are assigned to the same output class. This is achieved by allocating one node each in the output (disjunction) layer to a class, and by associating with it all nodes in the conjunction layer that correspond to the branches for that class.

In the final phase, a set of weak edges is superimposed on the network. These edges connect each node in a layer to every node in the previous and the next layer that remain disconnected from it after the previous step.

1.2.2 The Formal Algorithm

The validity of the technique is based on the assumption that each node in the generated network performs a hyperplane test of the form

$$b + \sum_{j=1}^{n} w_j x_j > 0? \tag{1.1}$$

and replies with a zero or one depending on whether the test succeeds. Here, b corresponds to the bias of the node, w_j corresponds to the weight on the input edge j, and x_j to the value of the input on edge j. In order to facilitate the subsequent training using **Bp**, the hard threshold is softened by replacing it with a sigmoid of the form

$$f(\vec{x}) = \frac{1}{1 + e^{-k.(b+\sum w_j x_j)}} \tag{1.2}$$

where k characterizes the softness [1] of the threshold.

The algorithm is as follows.

Begin

1 Initialize parameters σ and β to 5.0 and 0.025 respectively. [2]

2 Run C4.5 on the training dataset to generate a decision tree.

3 Traverse the decision tree to create a **dnf** (disjunctive normal form) formula for each class.

4 Eliminate all redundant literals from each disjunct.

[1]The smaller the value of k, the softer is the threshold

[2]What these parameters represent is explained further on in the paper.

5 For each distinct literal of the form $\langle attrib \rangle > \langle value \rangle$,[3] create a hidden unit in the literal layer with a bias of -$\sigma * \langle value \rangle$. Connect it to the input unit corresponding to $\langle attrib \rangle$ with a weight of σ. Connect it to all other input units with weights $+\beta$ or -β with equal probabilities.

6 For each literal of the form $\langle attrib \rangle < \langle value \rangle$, repeat step 5 with the signs for the bias and weights inverted.

7 For each disjunct in a class, create a new hidden unit in the conjunction layer. Connect it to all relevant hidden units in the literal layer with weights σ. Connect it to the rest of the hidden units in the literal layer with weights $+\beta$ or -β with equal probabilities. Set the bias to -σ*(2n-1)/2, where n stands for the number of relevant hidden units in the literal layer. (In effect, each node represents an AND.)

8 For each class, create an output unit and connect it to the relevant hidden units in the conjunction layer with weights σ. Connect it to the rest of the hidden units in the conjunction layer with weights $+\beta$ or -β with equal probabilities. Set the bias to -σ*1/2. (Each node is effectively an OR.)

End

σ and β may be treated as parameters to this technique. The values for these were fixed by cross validating on an artificially created dataset. Values for momentum and learning rate were fixed similarly to 0.1 and 0.3. These values were subsequently frozen across datasets.

1.3 Example

Figure 1.1 depicts a typical decision tree that might be generated by running C4.5 on a training dataset comprising of examples with two attributes and two output classes. The classifier could be expressed equivalently, as the following rules in disjunctive normal form.

$$(X < 2.5) \vee ((X >= 2.5) \wedge (Y < 1.3)) \Rightarrow Class : 1. \qquad (1.3)$$

[3]Literals of the form $\langle attrib \rangle = \langle value \rangle$ can be converted to $\langle attrib \rangle > \langle value - \delta \rangle \wedge \langle attrib \rangle < \langle value + \delta \rangle$

$$(X >= 2.5) \wedge (Y >= 1.3) \Rightarrow Class : 2. \tag{1.4}$$

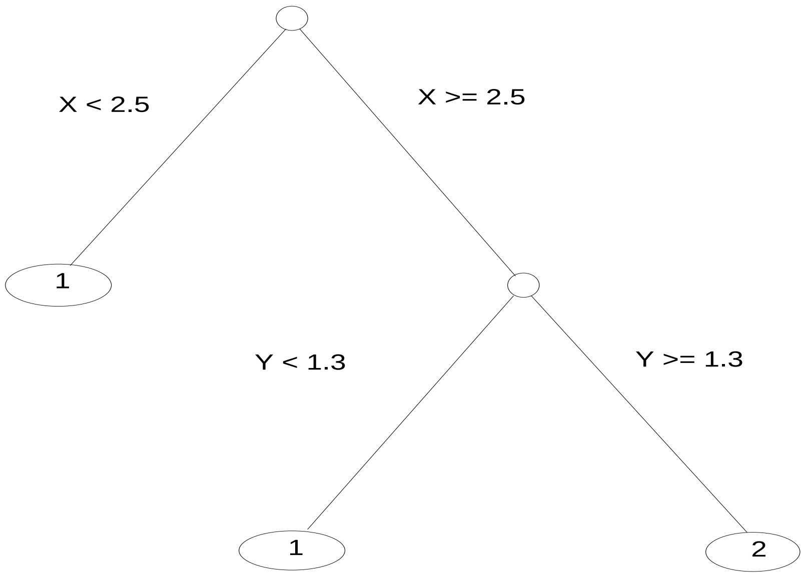


Figure 1.1
A typical decision tree.

The technique creates four nodes in the literal layer that correspond to $(X < 2.5)$, $(X >= 2.5)$, $(Y < 1.3)$, and $(Y >= 1.3)$. The literal $(X < 2.5)$ is characterized by the hidden node that has a bias of 12.5 and a weight of -5.0 on the input edge from X. The other nodes in the literal layer represent, in order, $(X >= 2.5)$, $(Y < 1.3)$, and $(Y >= 1.3)$.

Three nodes are generated in the conjunction layer; one each for $(X < 2.5)$, $(X >= 2.5) \wedge (Y < 1.3)$ and $(X >= 2.5) \wedge (Y >= 1.3)$. For each such node, the weights on the edges from relevant nodes in the literal layer are set to

5.0, and the bias of the node is set to $-(5n - 2.5)$, where n denotes the number of literals in the disjunct.

Two nodes are created in the output (disjunction) layer, one each for the two classes. The execution of this step is almost identical to the previous step. The only difference is that in this case, the bias of the nodes are set uniformly to -2.5. A final step connects each node in a layer to nodes in the previous and the next layer that remain disconnected from it after the mentioned procedure. The weights on these edges are set to $+0.025$ and -0.025 with equal probabilities. Figure 1.2 depicts the skeletal network generated prior to the final step. Any classifier that can be reduced to a propositional ruleset of the form shown may be converted into an equivalent neural network using the given method.

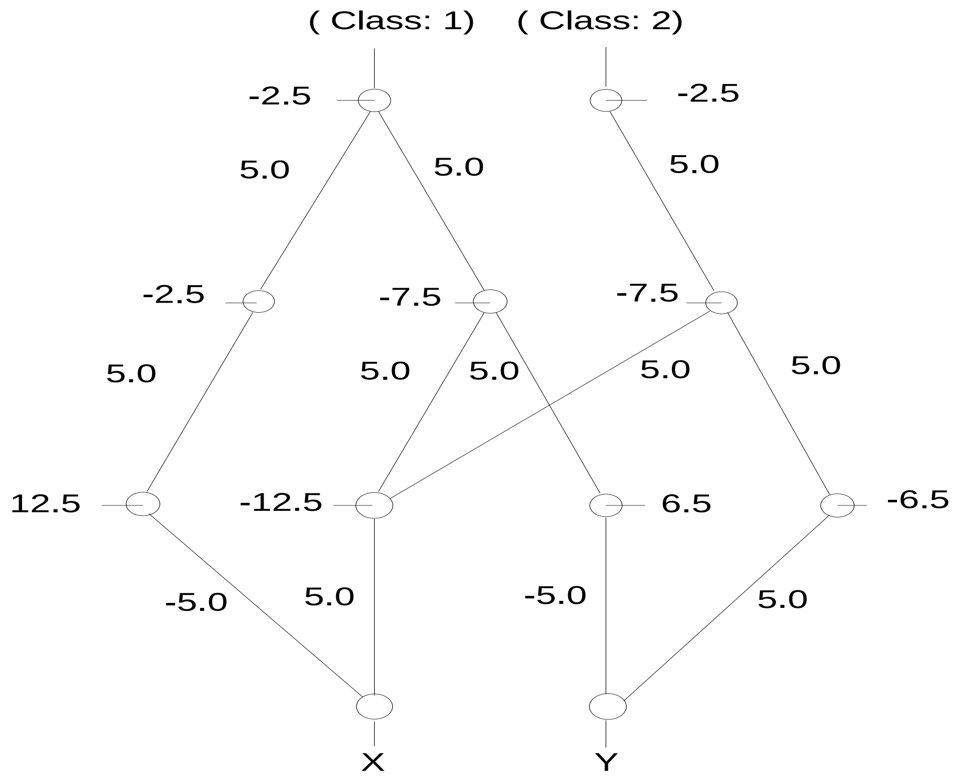


Figure 1.2
The skeletal neural network.

1.4 Experiments and Results

The new technique was compared to C4.5 and two separate instances of **Bp**. The first network consisted of a single hidden layer with as many hidden units as were created by the new technique (strawman:1), and the second network employed the same topology as the network created by the technique (strawman:2). In both cases, weights on the edges were initialized randomly.

Experiments were conducted on three separate benchmark datasets. The first dataset was Iris, the second was a thyroid gland dataset donated to the UCI machine learning repository by Stefan Aeberhard and the third was a glass identification database donated to the UCI repository by Vina Spiehler.

In all experiments a 5-fold cross validation was performed with each training conducted thrice. Results were then averaged over the fifteen runs. **Bp** remained in a permutation mode of training through all the experiments. The values of momentum and learning rate for the network created by the technique were fixed at 0.1 and 0.3 respectively, across datasets. However, for the other networks, good values for momentum and learning rates were chosen. The number of epochs where the networks achieved their best predictive accuracies were also noted. Finally, statistical significance tests were conducted to calibrate the classification accuracy of the various networks.

The results are presented graphically in figures 1.3, 1.4, and 1.5, and in a tabular form in figure 1.6. Figures 1.3 through 1.5 contain curves corresponding to the classification accuracy of each of the three networks. The curves represent the error rates for each network trained over a specific number of epochs, averaged over fifteen runs. The range of epochs presented in the figures are restricted to a region where the optimal performance of the all the networks were observed. The performance of C4.5 on these datasets are noted in figure 1.6.

1.4.1 Iris Dataset

When run on the Iris dataset, C4.5 generated a tree with an expected error rate of 8.5%. All three networks that were trained on the same dataset, however, surpassed C4.5 in accuracy for classifying novel examples. The new technique when trained to its optimal state, generated the best classifier with an expected error rate of 3.88%. The accuracy of the new technique was found to be significantly higher than strawman:2, but not significantly higher than strawman:1 (with a confidence level of 99%). The new technique also reached its optimal state quicker than the other networks. Whereas strawman:1 and strawman:2 achieved their optimal states at 220 and 320 epochs respectively, the technique arrived at its optimal state at 140 epochs. Finally, the network with the same number of hidden units as used by the technique (strawman:1) performed better than the network with the same topology as the technique (strawman:2).

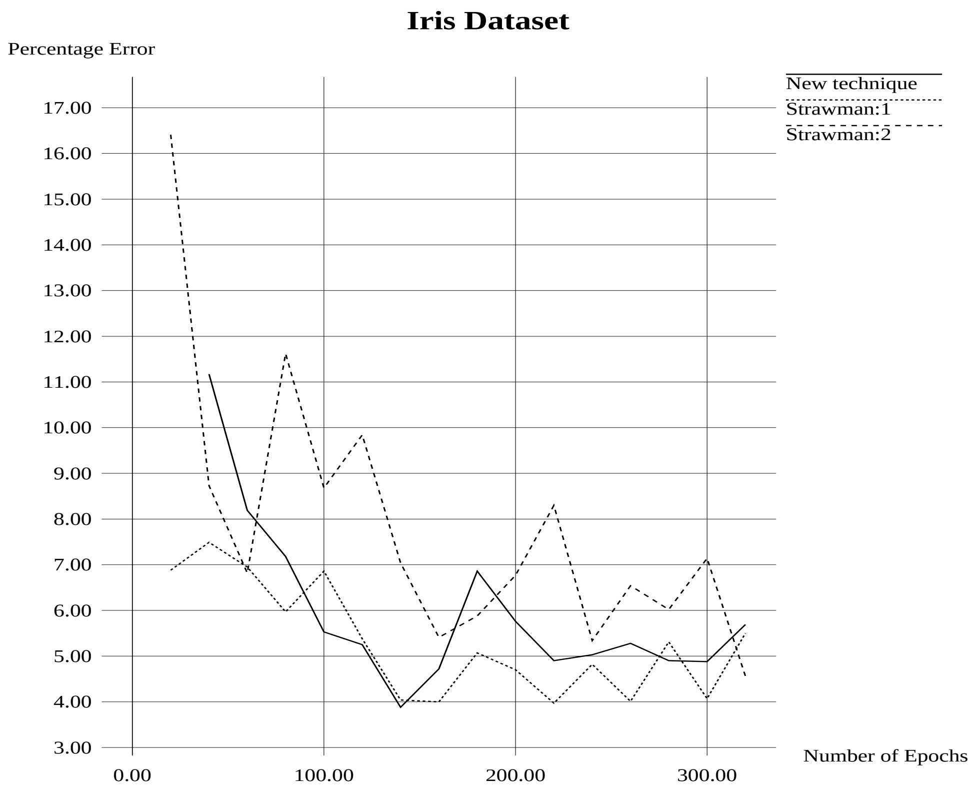


Figure 1.3
Results from the iris dataset.

1.4.2 Thyroid Gland Dataset

In this case, the new technique performed significantly better than C4.5 and the two other networks. The technique created a classifier with an expected error rate of 3.68% that surpassed, by far, the 4.37%, 5.08%, and 8.2% error rate classifiers created by strawman:2, strawman:1, and C4.5 respectively. Moreover, strawman:2 outperformed strawman:1 in this case unlike the previous instance.

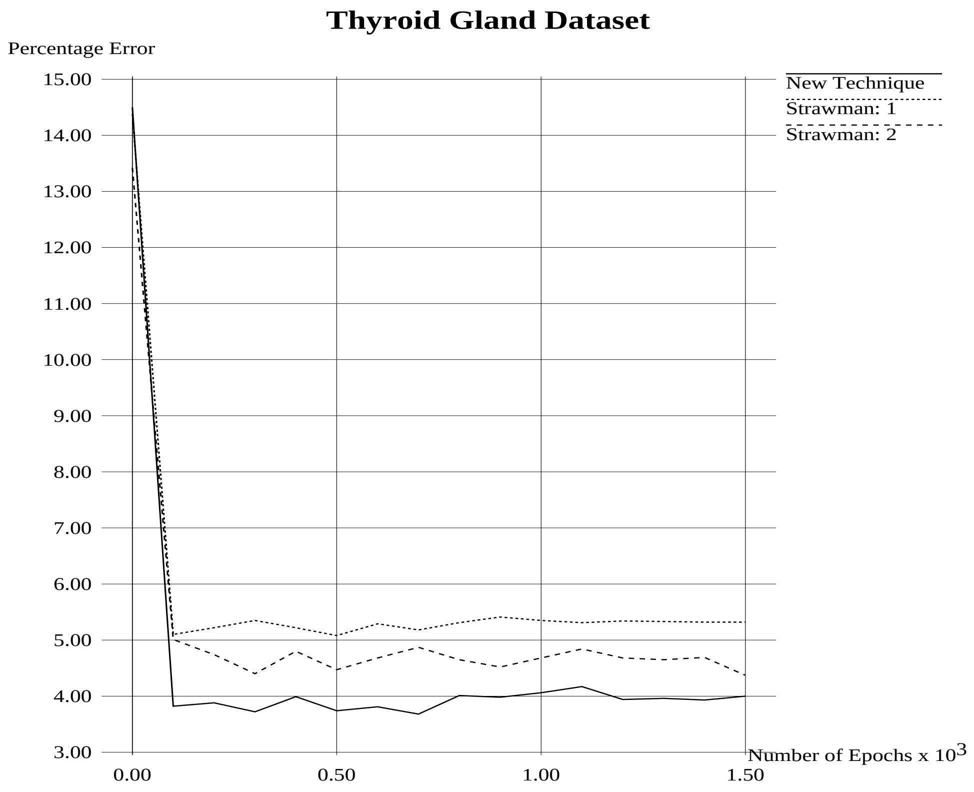


Figure 1.4
Results from the thyroid gland dataset.

1.4.3 Glass Identification Database

The Glass identification database has been known to be a dataset wherein C4.5 typically performs better than **Bp**. Results from experiments on this dataset, however, demonstrated that the new technique achieved a predictive accuracy that surpassed C4.5 and both the strawmen, the difference being significant to a 99% confidence level. This result could be attributed to the network's initialization through C4.5 (28.3% error.).

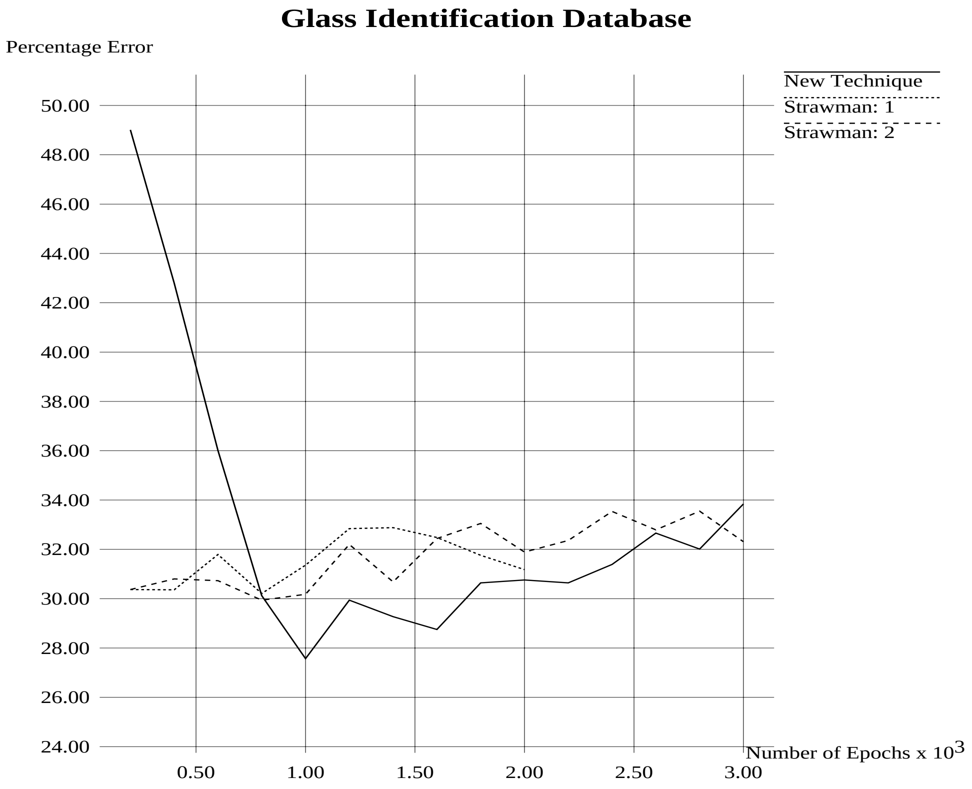
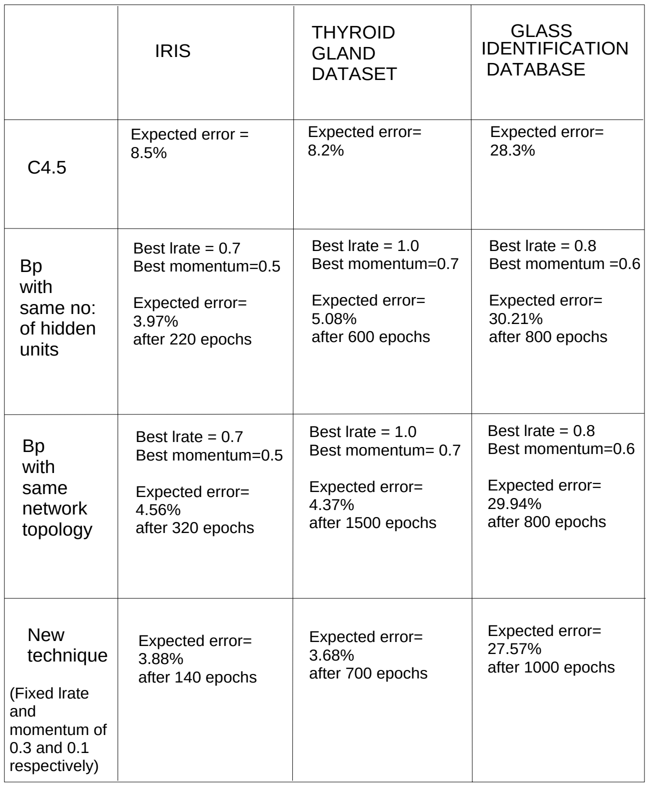


Figure 1.5
Results from the glass identification dataset.

One must note that the new technique performed better than the rival methods in spite of the basis for comparison being biased against it. Whereas the learning rate and momentum for the technique were frozen across datasets, good values for the parameters were chosen for the other methods.

Actual numerical values that were obtained through the experiments are recorded in a tabular form in figure 1.6. Results indicate that the complete process of finding the optimal values for momentum and learning rate through repeated training and cross validation can be discarded by applying, instead, the fixed parameter version of the new technique. Results also indicate that the efficacy of the technique can not be attributed to either the topology or the initial weights in isolation. It happens to be the consequence of the mutually supportive effects of both.



	IRIS	THYROID GLAND DATASET	GLASS IDENTIFICATION DATABASE
C4.5	Expected error = 8.5%	Expected error= 8.2%	Expected error= 28.3%
Bp with same no: of hidden units	Best lrate = 0.7 Best momentum=0.5 Expected error= 3.97% after 220 epochs	Best lrate = 1.0 Best momentum=0.7 Expected error= 5.08% after 600 epochs	Best lrate = 0.8 Best momentum =0.6 Expected error= 30.21% after 800 epochs
Bp with same network topology	Best lrate = 0.7 Best momentum=0.5 Expected error= 4.56% after 320 epochs	Best lrate = 1.0 Best momentum= 0.7 Expected error= 4.37% after 1500 epochs	Best lrate = 0.8 Best momentum=0.6 Expected error= 29.94% after 800 epochs
New technique (Fixed lrate and momentum of 0.3 and 0.1 respectively)	Expected error= 3.88% after 140 epochs	Expected error= 3.68% after 700 epochs	Expected error= 27.57% after 1000 epochs

Figure 1.6
Complete tabulated results.

1.5 Future Research

In spite of the fact that the results from the experiments were quite promising, the new technique has been found to suffer from certain drawbacks. In all the experimented datasets, the decision trees generated by C4.5 were fairly small; trees that had fewer than 25 nodes. It is however, not very difficult to conceive of a dataset where any decision tree algorithm performs poorly and creates a complex tree with over a 100 internal nodes. Since, in the new technique the number of hidden units increase linearly with the number of internal nodes in the decision tree, it would not be at all surprising if the technique created a very large network topology for certain datasets.

Applying the technique to multi-variate decision trees provides only a partial solution to this problem. Even though the hypothesis space for multivariate decision trees is richer than that for uni-variate trees, it nevertheless remains weaker than the hypothesis space for a network with at least one hidden layer. This is to say that the initial network generated by any decision tree approach has many more nodes than is actually required to classify the dataset. The only solution to this problem lies in pruning the decision tree. Methods for pruning thus need to be investigated in this context.

1.6 Conclusions

To summarize, a host of drawbacks have been identified with the **Bp** algorithm. First, it is extremely slow to train. Second and more important, there does not exist any formal technique that can decide beforehand the network topology and parameter settings that would be optimal for the network to classify a given dataset. This paper presents a novel technique that takes a decision tree classifier and transforms it into a network topology that classifies data almost as well as the tree. This process not only provides a good classifier to start with, but also chooses a suitable topology for the network. A short period of training then creates a classifier that is more accurate than **Bp**, and at the same time, runs faster. Finally, experimental comparisons against C4.5 and **Bp** demonstrate the efficacy of this new technique.

Acknowledgments

I would like to thank Haym Hirsh for stimulating discussions on this topic. I would also like to thank Jyoti Parmar for her critiques on an earlier draft of this paper.