[3] K. Lang, A. Waibel, and G. Hinton, "A Time Delay Neural Network Architecture for Isolated Word Recognition," *Neural Networks*, vol. 3, no. 1, pp. 23–43, 1990.

[4] A. Rajavelu, M. Musavi, and M. Shirvaikar, "A Neural Network Approach to Character Recognition," *Neural Networks*, vol. 2, no. 5, pp. 387–394, 1989.

[5] Y. L. Cun *et al.*, "Handwritten Zip Code Recognition with Multilayer Networks," in *Proceedings of the 10th International Conference on Pattern Recognition*, pp. vol. 2, 35–40, 1990.

[6] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[7] D. Parker, "Learning Logic," Tech. Rep. TR-47, MIT, Center for computational research in economics and management science, 1985.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.* Belmont,CA: Wadsworth international group, 1984.

[9] J. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[10] R. Mooney, J. Shavlik, G. Towell, and A. Gove, "An Experimental Comparison of Symbolic and Connectionist Learning Algorithms," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[11] L. Atlas, R. Cole, Y. Muthusamy, A. Lippman, J. Connor, D. Park, M. El-Sharkawi, and R. M. II, "A Performance Comparison of Trained Multi-Layer Perceptrons and Trained Classification Trees," *Proceedings of the IEEE*, vol. 78, October 1990.

[12] A. C. Tsoi and R. Pearson, "Comparison of Three Classification Techniques, CART, C4.5, and Multi-Layer Perceptrons." Proceedings of the 1990 IEEE Conference on Neural Information Processing Systems, Denver.

[13] D. Fisher and K. McKusick, "An Empirical Comparison of ID3 and Back-propagation," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[14] A. Sankar and R. Mammone, "Neural Tree Networks," in *Neural Networks: Theory and Applications* (R. Mammone and Y. Zeevi, eds.), Academic Press, 1991.

[15] L. Hyafil and R. Rivest, "Constructing Optimal Decision Trees is NP-Complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.

[16] N. J. Nilsson, *Learning Machines.* McGraw-Hill, 1965.

[17] M. Minsky and S. Papert, *Perceptrons.* Cambridge, Mass.: MIT Press, 1969.

[18] S. Gallant, "Perceptron-based Learning Algorithms," *IEEE Transactions on Neural Networks*, pp. 179–191, June 1990.

[19] A. Sankar and R. Mammone, "A Fast Learning Algorithm for Tree Neural Networks," in *Proceedings of the 1990 Conference on Information Sciences and Systems*, pp. 638–642, March 1990.

[20] A. Sankar and R. Mammone, "Tree Structured Neural Networks," Tech. Rep. TR-122, Rutgers University, 1990.

[21] A. Sankar and R. Mammone, "A New Fast Learning Algorithm for Feedforward Neural Networks using the L1 Norm of the Error," Tech. Rep. TR-115, Rutgers University, 1990.

[22] A. Sankar and R. Mammone, "Pruning Neural Tree Networks for Good Generalization." in preparation.

[23] A. Robinson, *Dynamic Error Propagation Networks.* PhD thesis, CAmbridge University Engineering Department, 1989.

Classifier	MLP	MLP	MLP	RBF	RBF	GN	GN	CART	NTN
Number of hidden neurons	88	22	11	528	88	528	88	–	59 [2]
Percent Correct	51	45	44	53	48	55	53	44	54

Table 1: Results on Speaker Independent Vowel Recognition

log area parameters calculated using six 512 sample Hamming windowed segments from the steady part of the vowel. Details of this data set can be obtained from [23]. This particular database represents a difficult problem and the best performance of any classifier was only around 55%. Experiments have been performed on this data set using various MLP neural networks with different numbers of hidden nodes and different types of nodes, such as radial basis function nodes and guassian nodes [23]. The performance of CART, one of the most popular decision tree methods, on this data set has been reported in [12]. These results are summarized in table 1. The table also shows the performance of the best pruned NTN on this data set. These results have been averaged over 20 runs of the NTN algorithm. We see that the performance of the NTN is much better than CART but is similar in classification accuracy to MLPs with radial basis function (RBF) nodes or guassian nodes (GN). As shown in the table, if the number of hidden nodes is not chosen correctly, the performance of MLPs is quite poor. The NTN algorithm, however, grows the correct number of neurons while training. Table 1 also shows the number of neurons used in the various methods. The number of neurons used in the NTN is less than that of MLPs with similar performance. The average depth of the NTN was only 3.1, which is quite a shallow tree. The retrieval time for a test pattern is thus only around 3 to 4 time units. An MLP with one hidden layer and implemented in parallel requires two time units–one to fire the hidden layer nodes and the second for the output nodes. The NTN is therefore only very slightly slower than the MLP for retrieval. However, for the NTN, only one processor needs to be implemented [14], whereas all the processors must be implemented in the case of the MLP.

4 Summary and Conclusions

We have developed a new neural architecture and learning algorithm called Neural Tree Networks (NTN). This network can be implemented in a very efficient manner. The new algorithm grows the network while learning and finds the correct number of neurons. The NTN algorithm is also superior to standard decision tree algorithms which rely on exhaustive search methods. Simulation results on a speaker independent vowel recognition task show that the new method is superior to both MLPs and decision trees.

Acknowledgments

The research reported here was made possible through the support of the New Jersey Commission on Science and Technology and the Center for Computer Aids for Industrial Productivity at Rutgers University.

This research was performed in partial fullfillment of the requirements for the Ph.D. degree of Ananth Sankar

References

[1] D. Rumelhart and J. McClelland, *Parallel Distributed Processing.* MIT Cambridge Press, 1986.

[2] A. Waibel, "Modular Construction of Time Delay Neural Networks for Speech Recognition," *Neural Computation*, vol. 1, March 1989.

[2] The neurons in the NTN are not hidden neurons. Rather this number refers to the total number of neurons used in the NTN

section 2.1. Each subset is assigned to a child node. The algorithm is then repeated for each child node. There are two stopping conditions for this recursive training algorithm: 1) The training set in R_t contains only one class ($N_t = 1$). In this case t becomes a leaf node and is labeled by the appropriate class. 2) The neural network in t is unable to split the training set into subsets even though $N_t > 1$. This is a problem though we have found that it occurs very rarely in the applications we have tried. t again becomes a leaf node but it is labeled by the class which has maximum membership in the training set contained in R_t. A solution to this problem is given in [19].

The learning algorithm grows the NTN so that the training set is classified with few errors. However for good generalization, it is necessary to prune the NTN. The pruning algorithm is described in the next section.

2.3 Pruning

Let the NTN grown by the learning algorithm be denoted by T. A branch, T_t of T consists of the node $t \in T$ and all the descendants of t. Pruning the branch T_t from T consists of deleting all the descendants of t. This pruned tree is called a pruned subtree of T. The node t now becomes a leaf node and it is labeled according to that class which has maximum membership in the training data contained in R_t. Let $\tilde{T}$ denote the set of leaf nodes of T and $\mid \tilde{T} \mid$ be the number of leaf nodes of T. $\mid \tilde{T} \mid$ is a measure of the complexity of the NTN. The pruning algorithm uses a cost function that is a weighted sum of the classification error of the tree and its complexity. This cost function is given by:

$$D_T + \alpha \mid \tilde{T} \mid \tag{4}$$

where D_T is a measure of the classification error of T and α is the penalty attached to each leaf. Pruning the NTN consists of generating a sequence of subtrees that minimize the above cost function for different values of α. Thus, if $\alpha = 0$, then the optimally pruned subtree is T itself and if $\alpha = \infty$, then the optimally pruned subtree is the root node of T, denoted by $\{t_1\}$. It is an interesting fact that the minimizing sequence of subtrees is a nested sequence [8]. Thus, as the value of α is increased, each new pruned subtree is found by pruning the current pruned subtree. This property allows for a very elegant pruning algorithm. An optimal pruning algorithm for binary trees based on the cost function of equation 4 has been given in [8]. We extend this algorithm to the case of arbitrary trees such as the NTN. The full derivation of this algorithm and proof of optimality is given in [22].

Consider the node t and branch T_t. The cost of t is given by $D_t + \alpha$ and the cost of T_t is given by $D_{T_t} + \alpha \mid \tilde{T}_t \mid$. As long as $D_t + \alpha > D_{T_t} + \alpha \mid \tilde{T}_t \mid$, the branch T_t is preferable to the node t, and is kept in the tree. When $D_t + \alpha = D_{T_t} + \alpha \mid \tilde{T}_t \mid$, the node t becomes preferable to the branch T_t since it has fewer leaf nodes and thus is less complex. This value of α is given by

$$\alpha_t = \frac{D_{T_t} - D_t}{1 - \mid \tilde{T}_t \mid} \tag{5}$$

The pruning algorithm works by finding this value of α for all the nodes $t \in T - \tilde{T}$ and then pruning off the branch T_{t^*} corresponding to the minimum value of $\alpha_t = \alpha_{t^*}$. This process is repeated starting with the NTN, $T_1 = T - T_{t^*}$ and so on until only the root, $\{t_1\}$, is left. Thus the algorithm produces a sequence of NTNs $T, T_1, T_2 \ldots \{t_1\}$. An independent test set of data is run through each of these pruned NTNs to find the best pruned NTN. If the test set is large enough, this gives a relatively unbiased estimate of the classification error of the NTN [8].

3 Simulation Results

In this section, we present simulation results on a speaker independent vowel recognition task. The training set consists of 528 feature vectors representing 11 vowel sounds obtained from 4 male speakers and 4 female speakers. The problem is to train the system on this data and then test it on an independent set of 462 feature vectors obtained from a different set of 4 male speakers and 3 female speakers. For both the training and testing set, each vowel is repeated 6 times by each speaker. The features are 10

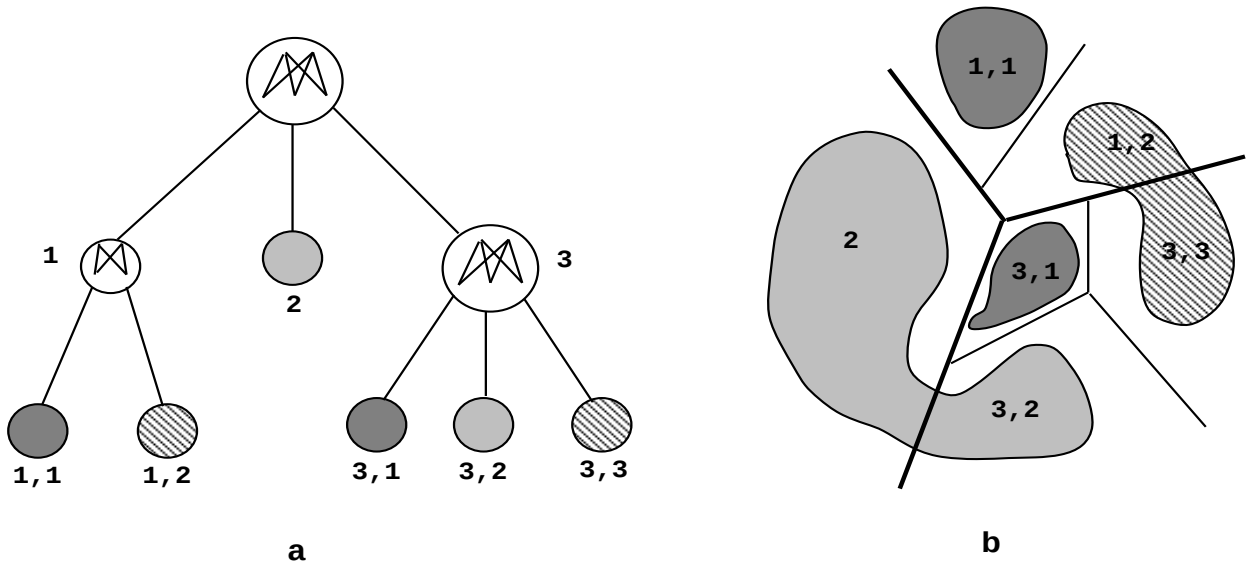

Figure 1: (a) Two level NTN with neural nets in internal nodes. Leaf nodes shaded according to appropriate class. (b) Feature space regions: first number refers to first level split of feature space, and second number to a second level split. First level region boundaries are shown by thick lines.

Again, consider node t of the NTN, corresponding to the sub-region, R_t, of feature space. As described in section 2.1, the neural network at node t has N_t neurons. The N_t classes are labeled by a local encoding scheme. Thus class i is labeled by the binary vector that has a 1 in its ith component, all the other components being 0. Training the neural network consists of sequentially presenting the network with a feature vector $\mathbf{X}$ and its corresponding label $\mathbf{T}$. The error between the actual network output, $\mathbf{Y}$, and the target, $\mathbf{T}$, is used to update the network weight vector $\mathbf{W}$. If the training data in R_t is multi-class linearly separable as defined in [16], then an optimal weight vector can be found by using the perceptron learning rule extended to the multi-class case [16, 17]. The perceptron rule, however, does a very poor job if the training set is not linearly separable. The pocket algorithm [18] produces a better solution in the non-linearly separable case but it is computationally expensive.

The approach used here is to treat the problem as one of sequential robust surface fitting. The surface to be fit at each step has the form of a sigmoid corresponding to the sigmoid activation function of the neurons, and the data to be fit is the training data. A gradient descent technique is used to update the neuron weights. We minimize the L1 norm of the errors rather than the L2 norm as is normally used in backpropagation. The L1 norm is less sensitive to outliers, and hence does a better job than the L2 norm in the non-linearly separable case [19, 20]. Furthermore, minimizing the L1 norm also results in faster training [21]. The L1 norm does not have a derivative at the origin. However this does not affect the gradient descent method since no weight updates are made when the output and the target agree exactly. The weight update rule for neuron i is given by

$$w_{j,i}^{n+1} = w_{j,i}^{n} - \eta y_i(1 - y_i)\mathrm{sgn}(t_i - y_i)x_j \quad 1 \leq j \leq d \tag{3}$$

where $w_{j,i}$ is the weight from input j to neuron i, y_i is the output of neuron i, t_i is the target output for neuron i and x_j is the jth component of the input feature vector $\mathbf{X}$. η is a gain term and d is the dimension of the feature space. The sgn function gives the sign of its argument, and is, in fact, the derivative of the L1 norm.

At each NTN node t, the training data contained in R_t is used to train the corresponding neural network. The training is continued until the output error has converged. Convergence can be detected when the difference in error between two consecutive weight updates is less than some pre-determined threshold. The network is then used to split the training data into at most N_t subsets as detailed in

Recently there have been many empirical studies comparing the performance of Neural Networks and Decision Trees [10, 11, 12, 13]. One particularly interesting test bed for pattern classifiers is Speech Recognition. The relative performances of MLPs and Decision Trees have been evaluated on a speaker independent vowel recognition task in [11, 12]. These studies show that while MLPs sometimes have a smaller classification error, they usually take longer to learn. We present simulation results of the NTN on a speaker independent vowel recognition task. These results show that the NTN is superior to both feedforward neural networks and decision trees.

In section 2 we describe the architecture and learning rule for the NTN. Section 3 compares the new method with MLPs trained by backpropagation, and decision trees on a speaker independent vowel recognition task. The summary and conclusions of the paper are presented in section 4.

2 Neural Tree Network (NTN)

2.1 Architecture

The NTN uses a tree structure with a feedforward neural network with no hidden neurons at each internal (non-leaf) node. The same input feature vector is presented to all the neural networks. Each leaf node corresponds to a class and is labeled accordingly. The neural network at the root of the NTN is used to divide the feature space into N regions, where N is the number of classes. Each region is assigned to a child node. The neural networks at each child node again divide these regions and this process is repeated until each region represents a single class. These single class regions correspond to leaf nodes. Thus classification is effected by using the neural networks at the internal NTN nodes to route the input feature vector to the appropriate leaf node. Figure 1 shows an NTN and the corresponding division of feature space for a 3–class pattern recognition problem. The neural network at the root divides the feature space into 3 regions corresponding to 3 child nodes. One of these regions has only one class in it and thus the corresponding child node is a leaf node. The other two regions are further divided into subregions by the neural networks at the child nodes. This architecture can be implemented with a single processor and a look-up table, thus offering a significant implementation advantage over MLPs [14].

Let t be an NTN node, R_t the corresponding region of feature space and N_t the number of classes in the training set contained in R_t. A neural network with N_t nodes is used at node t. The network uses a competition rule to classify an input feature vector, $\mathbf{X}$. Thus $\mathbf{X}$ is classified as class i iff,

$$O_i(\mathbf{X}) \geq O_j(\mathbf{X}) \quad \forall j \neq i, 1 \leq i, j, \leq N_t \tag{1}$$

where O_i denotes the output of node i for input $\mathbf{X}$ and is given by

$$O_i(\mathbf{X}) = \frac{1}{1 + e^{\mathbf{W}_i.\mathbf{X} + \theta_i}} \tag{2}$$

where $\mathbf{W}_i$ is the weight vector for neuron i and θ_i is a bias term. The neural network splits R_t into N_t regions by assigning all feature vectors that are classified as class i to a separate region, $R_{t,i}$, $1 \leq i \leq N_t$. However, only those regions that contain training set examples correspond to valid child nodes. Therefore the neural network splits the training data into at most N_t child nodes. It can be shown that the regions, $R_{t,i}$, formed by equation 1, are convex [14]. Thus the NTN successively splits the feature space into convex regions until each region corresponds to a single class. Unlike many standard decision tree methods such as CART [8], the NTN is not a binary tree.

2.2 Learning Algorithm

As described in the previous section, the NTN successively divides the feature space into convex subregions. Each division of feature space adds a level to the NTN. It is desired to find the smallest NTN that correctly classifies the training data. This problem has been shown to be NP-complete [15]. Thus we must look for good heuristic solutions. The criterion used in this paper is to do the best job of classification at each node of the NTN, hence reducing the number of missclassified training vectors at each step.

Speaker Independent Vowel Recognition using Neural Tree Networks [1]

Ananth Sankar and Richard J. Mammone
CAIP Center and Dept. of Electrical Engineering
Rutgers University
P.O. Box 1390
Piscataway, NJ 08855-1390

Abstract

Speaker independent vowel recognition is a difficult pattern recognition problem. Recently there has been much research using Multi-Layer Perceptrons (MLP) and Decision Trees for this task. This paper presents a new approach to this problem. A new neural architecture and learning algorithm called Neural Tree Networks (NTN) are developed. This network uses a tree structure with a neural network at each tree node. The NTN architecture offers a very efficient hardware implementation as compared to MLPs. The NTN algorithm grows the neurons while learning as opposed to backpropagation, for which the number of neurons must be known before learning can begin. The new algorithm is guaranteed to converge on the training set whereas backpropagation can get stuck in local minima. A gradient descent technique is used to grow the NTN. This approach is more efficient than the exhaustive search techniques used in standard decision tree algorithms. We present simulation results on a speaker independent vowel recognition task. These results show that the new method is superior to both MLP and decision tree methods.

1 Introduction

In recent years Neural Networks have offered an alternative approach to pattern recognition. Multi-Layer Perceptrons (MLP), in particular, have been extensively researched for various classification tasks [1, 2, 3, 4, 5]. The most commonly used algorithm to train MLP's is backpropagation [1, 6, 7]. Since this is a stochastic gradient descent technique, it tends to get stuck in local minima. Another problem is that backpropagation is very slow. In addition, the number of neurons and the interconnections between them must be determined before learning can begin. This is a problem since there is no known optimal way to determine the network architecture and it is usually arrived at by trial and error. Decision trees offer another approach to pattern recognition [8, 9]. The pattern to be classified enters the decision tree at its root. At each internal decision tree node, a hyperplane test is evaluated to route the pattern to one of the child nodes. This process is repeated for each successive child node until, finally, the pattern reaches a leaf node. The leaf nodes classify the pattern. The hyperplanes at the internal tree nodes are formed during the training stage. Most decision tree learning algorithms like CART [8] and ID3 [9] generate a set of candidate hyperplanes and then use a distortion metric to search for the best hyperplane for each tree node. Since the candidate hyperplanes are arbitrarily generated, this technique is not efficient. Decision tree learning algorithms, however, always converge to a solution which correctly classifies the training set.

In this paper, we describe a new neural architecture and learning algorithm called Neural Tree Network (NTN) which combines Neural Networks and Decision Trees in a very effective way. The NTN uses a decision tree structure with a feedforward neural network with no hidden neurons at each tree node. We develop a new learning algorithm for this network which always converges to a solution. The algorithm grows the neurons as it learns and finds the correct number of neurons needed for the classification task. The neural network at each tree node allows for an incremental approach to place the hyperplanes optimally as opposed to the search methods used in standard decision trees.

[1] To appear in the Proceedings of IJCNN, Seattle, 1991