CA-2 | 131

MILITARY TECHNICAL COLLEGE
CAIRO EGYPT

NEURAL NETWORK IMPLEMENTATION OF BINARY TREES

DR. Ismail A. Farag* **DR. Fawzy Ibrahim***

ABSTRACT

Multiple layer artificial neural network (ANN) structure is capable of implementing arbitrary input-output mappings. Similarly, hierarchical classifiers, more commonly known as decision trees, possess the capabilities of generating arbitrarily complex decision boundaries in an n-dimensional space. Given a decision tree, it is possible to restructure it as a multilayered neural network. The objective of this paper is to show how this mapping of decision trees into multilayer neural network structure can be exploited for the systematic design of a class of layered neural networks, called entropy nets, that have far fewer connections.

1. INTRODUCTION

Feedforward layered neural networks are finding applications in many areas. Several of these applications require mapping of the input data into one of the various decision classes. A mapping is called linear if the different data classes are separated by linear decision boundaries. Most of the interesting data mapping problems of interest need a nonlinear mapping between the input and output. While a single neuron is capable of only a linear mapping, a layered network of neurons with multiple hidden layers can generate any desired mapping.

An example of a multiple hidden layer network is shown in Fig. 1(a). Generally, all neurons in a layer are connected to all the neurons in the adjacent layers through uni-directional links. The connection strength between two neurons from adjacent layers is represented in the form of a weight value. The significance of this weight value is that it acts as a signal multiplier on the corresponding connection link. Each neuron in the layered network is typically modeled as shown in Fig. 1(b). As indicated Fig. 1 the input to a neuron is the linear summation of all the incoming signals on the various connection links. This net summation is compared against a threshold value. The difference arising due to the comparison drives an output function, usually called an activation function, to produce a signal at the output line of the neuron. The most common choice for the activation function is the sigmoid or the logistic function $1/(1+exp(-x))$. In the context of pattern recognition, such layered networks are also called multilayer perceptron (MLP) networks. It can be easily shown that two hidden layers are sufficient to from piece wise linear decision boundaries of any complexity [2,10]. However, it must be noted that two layers are not necessary for arbitrary decision regions. The first hidden layer is the

* Department of Specialized Electrical Engineering, Military Technical College Cairo, Egypt.

partitioning layer that divides the entire feature space into several regions. The second hidden layer is the ANDing layer that performs Anding of partitioned regions to yield convex decision regions for each class. The output layer can be considered as the ORing layer that logically combines the results of the previous layer to produce disjoint decision regions of arbitrary shape with holes and concavities, if needed. These networks have been used for classification tasks involving speech, text, and sonar data with performance similar to that of conventional nonparametric classifiers such as k-nearest neighbor classifier with the benefit of inherent parallelism of the neural net structure [2,7].

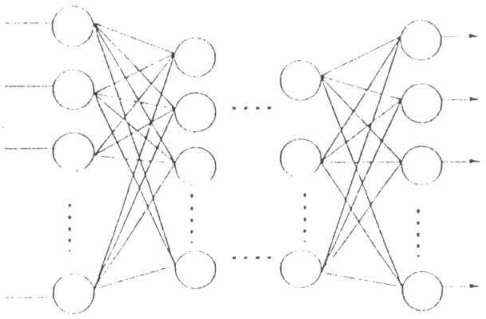

Fig. 1(a) An example of a multiple hidden layer network

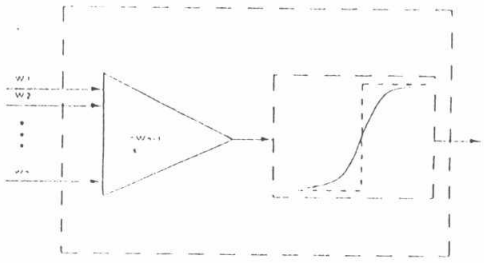

Fig. 1(b) Neuron model

These networks are trained with examples of input-output mapping pairs following the supervised mode of learning. The goal of learning is to force the network to produce the desired classification present in the training examples. During the learning process the network continuously modifies its connection strengths or weights to achieve the mapping present in the examples. While the single layer neuron training procedures have been around for thirty to forty years, the extension of these training procedures to multiple layer neuron networks proved to be a difficult task because of the so called credit assignment problem, i.e. what should be the desired output of the neurons in the hidden layers during the training phase. One of the solutions to the credit assignment problem that has gained wide acceptance is to propagate back the error at the output layer to the internal layers. The resulting back propagation algorithm [11] is the most frequently used training procedure for layered networks. It is a gradient descent procedure that minimizes the error at the output layer. Although the convergence of the algorithm has been proved only under the assumption of infinitely small weight changes, the practical implementations with larger weight changes appear to yield convergence most of the time. Because of the use of gradient search procedure, the back propagation algorithm occasionally leads to solutions that represent local minima. Some other examples of layered network training procedures are Boltzmann learning [1], counter propagation, and Medaline Rule II [19]. These training procedures including the back propagation algorithm, however, are quite slow, and need a very large number of training cycles before reaching the final set of weights. This is specially true if the input-output mapping requires disjoint regions of complex shape. In addition to the large training time, these training procedures do not specify in any way the number of neurons needed in the hidden layers. This

number is an important parameter that can significantly affect the learning rate as well as the over all classification performance as indicated by the experimental studies of several researchers [2,7].

There exists a class of conventional pattern classifiers that has many similarities with the layered networks. This class of classifiers are called hierarchical or decision tree classifiers. As the name implies these classifiers arrive at a decision through a hierarchy of stages.

Unlike many conventional pattern recognition techniques, the decision tree classifiers also do not impose any restriction on the underlying distribution of the input data. These classifiers are capable of producing arbitrarily complex decision boundaries. The two significant differences between the decision tree classifiers and their limited generalization capabilities in comparison with layered networks. The aim of this paper is to show how the similarities between the tree classifiers and the layered networks can be used for developing a pragmatic approach to the design and training of neural networks for classification.

2. DECISION TREE CLASSFIERS

The decision trees offer a structured way of decision making in pattern recognition. The rationale for decision tree based partitioning of the decision space has been well summarized by Kanal [8]. A decision tree is characterized by an ordered set of nodes. Each of the internal nodes is associated with a decision function of one or more features. The terminal nodes or leaf nodes of the decision tree are associated with actions or decisions that the system is expected to make. In an m-ary decision tree, are m descendants for every node. Binary decision tree form is the most commonly used tree from. An equivalent binary tree exists for any m-ary decision tree. Henceforth in this paper, a decision tree will imply a binary tree.

A decision tree induces a hierarchical partitioning over the decision space. Starting with the root node, each of the successive internal nodes partitions its associated decision region into two half spaces with the node decision function defining the dividing hyperplane equation. An example of a decision tree and the corresponding hierarchical partitioning induced by the tree are shown in Fig. 2. It is easy to see that as the depth of the tree increases, the resulting partitioning becomes more and more complex.

3. TREE-TO-NET MAPPING

In order to understand the relationship between the decision trees and multiple layer neural networks. let us consider the way classification is done with decision trees. Classification using decision tree is performed by traversing the tree from root node to one of the leaf nodes using the unknown pattern vector. The response elicited by the unknown pattern is the class or decision label attached to the leaf node that is reached by the unknown vector. It is obvious that all the conditions along any particular path the root to the leaf node of the decision tree must be satisfied in order to reach that particular leaf node. Thus each path of a decision tree implements an AND operation on a set of half spaces. If two or more leaf nodes result in same action or decision then the corresponding paths are in OR relationship. Since a layered neural network for classification also implements ANDing of hyperplanes followed by ORing in the

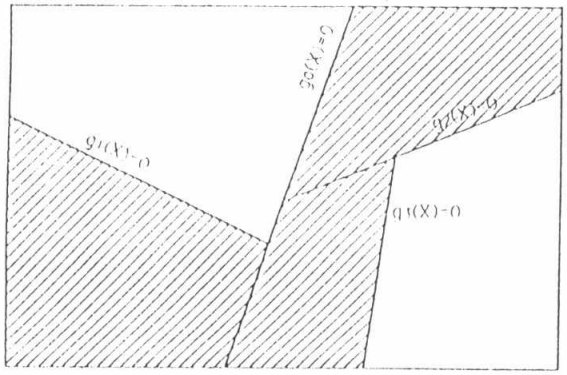

Fig. 2 (a) Decision boundaries of a two-class, two-feature pattern recognition problem.

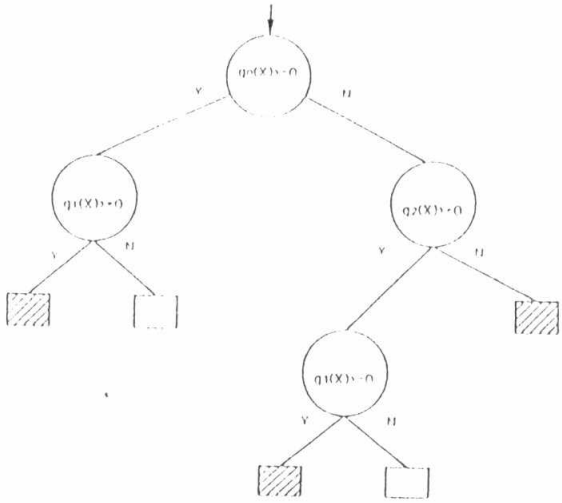

Fig. 2.(b) Decision tree for the problem of Fig. 2(a).

output layer, it is obvious that a decision tree and a layered network are equivalent in terms of input-output mapping. In fact some of the earlier literature on decision trees [6] views them as layered structure of threshold devices. In addition to a decision tree and a layered network being equivalent in terms of input-output mapping, it is also possible to restructure a decision tree as a three layer neural network structure by following certain rules. These rules can be informally stated as follows :

- The number of neurons in the first hidden layer of mapped neural network equals the number of internal nodes of the decision tree. Each of these neurons implements one of the decision functions of internal nodes. This layer is the partitioning layer.

- All leaf nodes have a corresponding neuron in the second hidden layer where the ANDing is implemented .

- The number of neurons in the output layer equals the number of distinct classes or actions. This layer implements the ORing of those tree paths that lead to some action.

- The connections between the neurons from the partitioning layer and the neurons from the ANDing layer implement the hierarchy of the tree.

An example of tree restructuring following the above rules is shown in Fig. 3 for the decision tree of Fig. 2. The shaded neurons in the ANDing layer correspond to the shaded terminal nodes of the decision tree. As this example shows, it is fairly straightforward to map a decision tree into a layered network of neurons. It should be noted that the mapping rules given above do not attempt to optimize. The number of neurons in the partitioning layer. However, the optimal mapping can be achieved by incorporating checks in the mapping rules for replications of the node decision functions in different parts of the tree to avoid the duplication of the neurons in the partitioning layer.

The most important consequence of the above mapping is that it defines exactly the number of neurons needed in each of the three layers of the neural network. Hither-to this number of neurons has been determined by empirical means and the credit assignment problem has been tackled with back propagation. In comparison with the standard layered network architecture, the mapped network has far fewer connections. Except for one neuron in the partitioning layer that corresponds to the root node of the decision tree, the remaining neurons do not have connections with all the neurons in the adjacent layers. The smaller number of connections in the mapped network is an important advantage from the VLSI fabrication standpoint. To emphasize difference in the architecture, we shall henceforth refer to the mapped network as the entropy net. The reason for this name is due to mutual information based data driven tree generation methodology that is discussed in the next section .

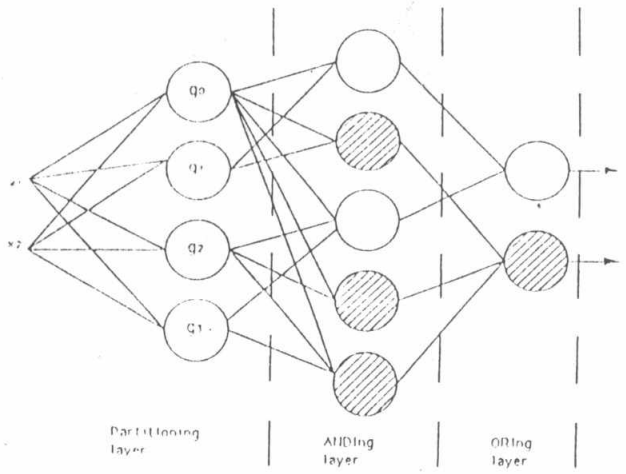

Fig 3: Mapped multiple layer perceptron (MLP) network for the decision tree of Fig 2(b).

4. MUTUAL INFORMATION BASED SELF-CONFIGURATION

Having established the equivalence between decision trees and layered neural networks, it is obvious that the design of a layered network can be done automatically by combining the mapping rules with an automatic tree generation procedure.

There exist several automatic tree generation algorithms in the pattern recognition literature where the problem of tree generation has been dealt with in two distinct ways. Some of the early approaches break the tree design process into two stages. The first stage yields a set of prototypes for each pattern class. These prototypes are viewed as entries in a decision table which is later converted into a decision tree using some optimal criterion. Examples of this type of tree design approaches can be found in [5,14]. The problem of finding prototypes from binary or discrete valued patterns are considered in [13,17]. The other tree design approaches try to obtain the tree directly from the given set of labeled pattern vectors. These direct approaches can be considered as generalization of decision table conversion approaches with all the available pattern vectors for the design forming the decision table entries. Examples of these direct tree design approaches can be found in [6,9,12,15,18]. While some of these tree

generation algorithms can handle only two classes at a time, there are several suitable for multifeature, multifeature, multiclass pattern recognition problems.

There re three basic tasks that need to be solved during the tree design process: (i) defining the hierarchical ordering of the node decision functions, (ii) choosing the node decision functions, and (iii) setting up a decision rule at each terminal node. In its complete generality, the decision tree design problem is a difficult problem and no optimal tree design procedure exists. Some of the tree design difficulties are simplified in practice by enforcing a binary decision based on a single feature at each of the nonterminal nodes. This results in the decision space partitioning with the hyperplanes that are parallel to the feature axes. In terms of the tree design the problem is then to find the ordering and the location of these hyperplanes. It should be noted that while the use of single feature decision function at every nonterminal node reduces the computational burden at the tree design time, it usually leads to larger trees.

One of the popular approach for ordering and locating the partitioning hyperplanes is based on defining a *goodness measure* of partitioning in terms of mutual information. Consider a two-class problem with only one measurement x. *Let* $x = t$ define the partitioning of the one-dimensional feature space. If we view the measurement x taking on values greater or less than threshold t as two outcomes x_1 and x_2 of an event **X**, then the amount of average mutual information obtained about the pattern classes from the observation of event **X** can be written as

$$I(C;X) = \sum_{i=1}^{2} \sum_{j=1}^{2} p(c_i, x_j) \log_2 [p(c_i / x_j) / p(c_i)] \qquad (1)$$

where C represents the set of pattern classes and the p (.) 's have the usual meaning. Clearly, for better recognition, the choice of the threshold t should be such that we get as possible from the event X. This means that the value which maximizes (1) should be selected over all possible values of t. Average mutual information thus provides a basis for measuring the goodness of a partitioning.

Let us now extend the average mutual information partitioning measure to a decision tree. Let I_k be an internal node of the decision tree having an event X_k associated with it. Let C_k be the subset of m pattern classes tied up with the event X_k. If we let P_k denote the probability of the pattern class subset C_k, then the average mutual information provided by the decision tree T can be written as a weighted sum of mutual information provided by the individual internal nodes, i.e.

$$I(C;T) = \sum_{k=1}^{L} P_k I_k (C_k ; X_k) \qquad (2)$$

where $L \geq m-1$ is the number of internal nodes of the tree T, and $I_k (C_k ; X_k)$ is the average mutual information associated with the node I_k.

The above partitioning measure immediately suggests a top down recursive procedure for the tree design. The AMIG (Average Mutual Information Gain) algorithm [15] is one such example of the recursive tree design procedure that seeks to maximize the amount of mutual information gain at every stage of tree development. Unlike many other algorithms that either operate on discrete features or two classes, the AMIG algorithm is capable of generating decision trees for continuous valued multifeature, multiclass pattern recognition problems from a set of labeled pattern vectors. The AMIG algorithm at any stage of tree development essentially employs a brute force search technique to determine the best feature for that stage along with its best threshold value to define an event for the corresponding node. Since the orientation of dividing hyperplanes is restricted, i.e. only one feature is used at any internal node, the search space for maximizing the average mutual information gain is small. The search is made efficient by ordering the labeled patterns along different feature axes to obtain a small set of possible candidate locations along each axis. The AMIG algorithm or its variants have been used by numerous researchers to automatically design decision trees in problems such as character recognition, target recognition etc. The differences in the various algorithms pertain to the stopping criterion. The stopping criterion used in AMIG algorithm is based on the following inequality [3] that determines the lower limit on the average mutual information to be provided by the tree for the specified error performance P_e:

$$I(C\,;\,T) \geq H(C) - H(P_e) - P_e \log_2 (m-1) \qquad (3)$$

where H(C) and $H(P_e)$ respectively represent the pattern class and the error entropy. The criterion used in [18] is to test the statistical significance of the mutual information gain that results from further splitting a node. Recently, Goodman and Smyth [4] have derived several fundamental bounds for mutual information based recursive tree design procedures and suggested a new stopping criterion which is claimed to be more robust in the presence of noise. Once the complete tree has been developed, the terminal nodes are assigned class labels following the majority rule criterion.

Summarizing the above discussion, it is clear that there exist several automatic tree generation procedures that are driven by the example pattern vectors. Any of these procedures in conjunction with the decision tree to layered network mapping discussed earlier can be used to design an entropy network for a given input-output mapping problem.

While the design of layered networks through decision tree mapping eliminates the guesswork about the number of neurons in different layers and provides a direct method of obtaining connection strengths, the resulting network still has some of the limitations that were exhibited by the *MEDALINE* [19] type of early multilayer ANN models; there is no adaptability beyond the first hidden layer .

In terms of the decision trees, this limitation is best described by saying that once a wrong path is taken at an internal node, there is no way of recovering from the mistake. The layered networks of neurons avoid this pitfall because of their adaptability beyond the partitioning layer that allows some corrective actions after the first hidden layer. The same adaptive capability is possible in the entropy net if we replace the hard nonlinearity of the decision tree nodes with soft nonlinearities like the sigmoid functions and train the network with the available examples of input-output mapping pairs[16] .

5. SUMMARY

A tree-to-network mapping has been presented in this paper to show an equivalence between multiple layer neural networks and decision trees. The most important advantage of this mapping is that it provides a self-configuration capability to the neural net design process. Additionally, the mapping also allows us to understand the feedforward networks in much as using the tree-to-net mapping for solving the credit assignment problem and the incremental training of the entropy net that need attention. Our current research is looking at these aspects.

REFERENCES

[1] D. H. Ackley, G. E. Hinton, and T.J. Sjnowski, "learning algorithm for Boltzmann machines," *Cognitive Science* Vol , 9 . pp. 147 -169 , 1985.

[2] D. J. Burr, "Experiments on neural net recognition of spoken and written text, " *IEEE Trans.* ASSP, Vol. l, 36. July 1988, pp. 1162-1168.

[3] R. M. Fano, *Transmission of Information,* John Wiley & Sons, New York , 1963.

[4]R. M. Goodman and P. Symth, "Decision tree design from a communication theory standpoint, " *IEEE Trans . Inform. Theory,* Vol. IT-34, pp. 979-994, Sept. . 1998.

[5] C.R.P Hartmann, P. k. Varshney, K. G. Mehrota, and C. L. Gerberich, "Application of information theory to the construction of efficient decision trees," *IEEE Trans. Inform. Theory*, Vol. IT=28, pp. 565-577, July 1982.

[6] E. G. Henrichon and K. S. Fu , " A nonparametric partitioning procedure for pattern classification ,"*IEEE Trans . Computer.*, Vol. C-18, pp. 614-624 ,July 1982.

[7] W.Y. Hunang and R. P. Lippmann, " comparison between neural net and conventional classifiers ,"*Proc . IEEE First International Conf. Neural Networks*, Vol. IV, San Diego, June 1987, pp. 485-493.

[8] L. K. Kanal, "Problem-solving models and search strategies for pattern recognition," *IEEE Trans Pattern Anal , Machine Intell.,* Vol . PAMI-1, pp. 194-201, Apr. 1979.

[9] X. Li and R.C. Dubes, "Tree classifier design with a permutation statistic ,"*Pattern Recognition,* Vol. 19, pp. 229-235, 1986.

[10] D. E. Lippmann, "An introduction to computing with neural nets ," *IEEE ASSP Magazine*, April 1987, pp. 4-2 .

[11] D.E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in D. E. Rumelhart & J. L. McClelland (Eds.), *parallel distributed processing :Explorations in the microstructure of Cognition* Vol. 1: Foundations., MIT Press 1986.

[12] E. M. Rounds, "A combined nonparametric approach to feature selection and binary decision tree design, "*Pattern Recognition* Vol. 12 pp. 313-317, 1980.

[13] I. K. Sethi and B. Chaterjee, "A learning classifier scheme for discrete variable pattern recognition problems, " *IEEE Trans Systems, Man, and Cyber.*, Vol SMC-8, pp. 49-52, Jan. 1987.

[14] I. K. Sethi and B. Chaterjee, "Efficient decision tree design for discrete variable pattern recognition problems, " *Pattern Recognition*, Vol. 9, pp. 197-206 ,1978

[15] I. K. Sethi and G.P.R Sarvarayuda , "Hierarchical classifier design using mutual information , "*IEEE Trans PAMI,* Vol. PAMI-4, July 1982, pp. 441-445.

[16] I. K. Sethi, "Entropy nets: from decision trees to neural nets ," *Proc. IEEE, August, 1990.*

[17] J. C. Stoffel, "A classifier design technique for discrete variable pattern recognition problems, "*IEEE Trans. computer,* Vol. C-23, pp. 428-441, 1974.

[18] J. L. Talmon, "A multiclass nonparametric partitioning algorithms, " in E. S. Gelsema & L. N. Kanal (Eds), *Pattern Recognition in Practice II* Elsevier Science Pub. B. V. (North Holland), 1986.

[19] B. Widrow, R.G. Winter, and R.A. Baxter, "Layered neural nets for pattern recognition," *IEEE Trans ASSP,* Vol. 36 , July 1988, pp. 1109-1118.