Hierarchical Classifier Design Using Mutual Information
I. K. SETHI and G. P. R. SCHRIVER

Abstract—A nonparametric algorithm is presented for the hierarchical partitioning of the feature space. The algorithm is based on the concept of average mutual information, and is suitable for multiclass multistage pattern recognition problems. The algorithm generates an efficient partitioning tree for specified probability of error by maximizing the amount of average mutual information gain at each partitioning step. A confidence bound expression is presented for the result of the classifier. Three examples, including one of digitized numeral recognition, are demonstrated to present the effectiveness of the algorithm.

Index Terms—Beta functions, decision trees, feature extraction, mutual information, nonparametric methods, Walsh series.

I. INTRODUCTION
The problem of classifier design can be considered as one of partitioning the feature space into a number of disjoint regions. The classification then is nothing but the determination of the region to which an unknown sample belongs. In many pattern recognition problems, the pattern classes are multidimensional in nature, and it becomes difficult to use conventional partitioning procedures such as the Bayesian rule for classifier design. In such cases, the nonparametric partitioning of the feature space is usually preferred. The rationale for such hierarchical partitioning has been well summarized by Kanal [11], and optimization approaches to the design of hierarchical classifiers are described in [12] and Kanal [2].

One of the simplest nonparametric methods of partitioning the feature space is to use a set of hyperplanes parallel to feature axes. However, the main difficulty in such a partitioning is the determination of the number of hyperplanes and their locations. Henricson and Fu [3] have suggested an empirical method for finding such partitioning which can be represented as a layered structure of threshold devices. Recently, the use of the Kolmogorov-Smirnov test has been made to partition the feature space which gives rise to a binary decision tree structure for the classifier [4], [5]. The main drawback of both methods is that it requires as many decision trees as many decision trees to be developed as there are pattern classes. In another recent work, Datara and Sarma [5] suggest the use of the first- and second-order statistics of the labeled samples to discretize the feature space. A dynamic programming algorithm is then used to obtain the minimum cost decision tree for classification.

In this note, we present a simple method of hierarchical partitioning of the feature space using hyperplanes parallel to feature axes. The method is based on the concept of mutual information and is applicable to multiclass multistage pattern recognition problems. Besides presenting two illustrative examples, an example of application of the proposed method to the classifier design for numeral recognition is discussed. It shows the effectiveness of the proposed method in hierarchical partitioning. A measure of confidence on the error performance of the classifier is also discussed.

Manuscript received May 1, 1981; revised January 20, 1982.

I. K. Sethi is with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur 721 302, India.
G. P. R. Sarvarayudu is with the Jawaharlal Nehru Technological University, India, or leave at the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur 721 302, India.

REFERENCES
[1] F. Sudjadi and E. Hall, “Image registration using material infor-

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-4, NO. 4, JULY 1982

0162-8828/82/0700-0441$00.75 © 1982 IEEE
II. INFORMATION MEASURE OF PARTITIONING

Consider a two-category classification problem involving only one measurement \( x \). Let \( x \) define the partitioning of the one-dimensional feature space. If we view the measurement \( x \) taking \( n \) values greater or less than \( x \) 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

\[
H(C | X) = \sum_{x_i} \sum_{x_j} \cdot \frac{P(x_i | x_j) \log \left( \frac{P(x_i | x_j)}{P(x_i | x_j)} \right)}
\]

where \( C \) represents the set of pattern classes \( c_1 \) and \( c_2 \), having a priori probabilities \( P(c_1) \) and \( P(c_2) \), \( H(C | X) \) is the joint probability of occurrence of \( c_1 \) and \( x_1 \) and \( P(x_i | x_j) \) is the probability that the observation comes from class \( c_i \) given the outcome \( x_j \) of event \( X \). Clearly, for better recognition, the choice of threshold \( r \) should be such that we get minimum information about the classes \( c_1 \) and \( c_2 \) from the event \( X \). This means that the value which maximizes \( (1) \) should be selected over all possible values of \( r \). If both classes have unimodal distributions, \( r \) can easily be seen that this choice of threshold \( r \) turns out to be the same as that provided by the Bayesian rule for the zero-one loss function.

Suppose we set up a decision process to recognize the pattern classes \( c_1 \) and \( c_2 \) by observing the event \( X \), i.e., whether the measurement \( x \) is greater than or less than the threshold \( r \). Let \( P_e \) be the probability of error allowed in recognition; then the following inequality determines the limit on the equivalence \( H(C | X) \) of \( C \) with respect to \( X \):

\[
H(C | X) \leq H(P_e) + P_e \log (m - 1)
\]

where \( H(P_e) \) is the entropy density and \( m \) is the number of pattern classes. Since the average mutual information can also be written as

\[
H(C | X) = H(C) - H(C | X),
\]

we get the following inequality from (2) and (3):

\[
H(C | X) \geq H(C) - H(P_e) - P_e \log (m - 1)
\]

Using the equality sign in the above, we obtain the smallest amount of average mutual information needed between \( C \) and \( X \) for a given probability of error. Denoting this by \( I_{\text{max}} \) and expanding for \( H(C) \) and \( H(P_e) \), we get

\[
I_{\text{max}} = \sum_{i=1}^{m} p(c_i) \log_2 p(c_i) + p_e \log_2 (P_e - (1 - P_e)) \log_2 (1 - P_e) - P_e \log_2 (m - 1)
\]

This equation thus relates the probability of error and the corresponding maximum value of average mutual information required for a recognition process. Let us now consider a recognition problem involving \( m \) pattern classes with \( n \) features. The type of partitioning and the results using hyperplanes parallel to feature axes can be conveniently represented in the form of a binary tree. For example, Fig. 1 shows a partitioned two-dimensional feature space and its tree representation. The average mutual information about pattern classes from such hierarchical partitioning can be obtained in terms of the mutual information available at the nonterminal nodes of the partitioning tree, since with each such node, an event is associated. Let \( x_k \) be an internal node of the partitioning tree \( T \), then the average mutual information available at the nodes \( x_k \) can be written as

\[
I(x_k) = \sum_{x_k} \cdot \frac{P(x_k | x_k) \log \left( \frac{P(x_k | x_k)}{P(x_k | x_k)} \right)}
\]

Finally, the problem we treat here is to determine an efficient hierarchical partitioning tree for a specified probability of error \( P_e \). As mentioned above, the proposed algorithm maximizes the amount of gain in average mutual information at each partitioning step. This is done by choosing a threshold \( t_k \) for the \( k \)th feature at the \( k \)th partitioning step such that \( P_{\text{single}}(X_k, X_k) \) is maximum over all possible values of \( t \) and over all possible threshold values for each \( i \). However, in the absence of the true values of the various probabilities, their estimates based on the available design samples are used to calculate the gain in average mutual information in the following fashion.

\[
G(X_k | X_k) = \sum_{x_k} \cdot \log \left( \frac{P(x_k | x_k)}{P(x_k | x_k)} \right)
\]

III. RECURSIVE PARTITIONING ALGORITHM

Formally, the problem we treat here can be stated as follows. Given a set of \( m \)-dimensional design samples \( Y_1, Y_2, \ldots, Y_n \) coming from \( m \) pattern classes, determine an efficient hierarchical partitioning tree for a specified probability of error \( P_e \). As mentioned above, the proposed algorithm maximizes the amount of gain in average mutual information at each partitioning step. This is done by choosing a threshold \( t_k \) for the \( k \)th feature at the \( k \)th partitioning step such that \( P_{\text{single}}(X_k, X_k) \) is maximum over all possible values of \( t \) and over all possible threshold values for each \( i \). However, in the absence of the true values of the various probabilities, their estimates based on the available design samples are used to calculate the gain in average mutual information in the following fashion:

\[
G(X_k | X_k) = \sum_{x_k} \cdot \log \left( \frac{P(x_k | x_k)}{P(x_k | x_k)} \right)
\]
The steps of the recursive partitioning algorithm are as follows:

1) **Initialization:**
   - Using the specified value of \( D \) calculate \( f_{\text{min}} \) using (5).
   - Initialize \( \text{ITREE} \), i.e., the average mutual information provided by the hierarchical partitioning tree, equal to zero.

2) **Initialize NODESET with \( \emptyset \).** At any instant, the NODESET contains the list of nodes which are possible candidates for further partitioning.

3) **Assign all the available design samples to LIST (1).** In general, LIST (NODE) consists of samples associated with node NODE of the partitioning tree where node NODE is a member of NODESET.

4) **Looping:** For each node of NODESET, do the following:
   - Order the samples from LIST (NODE) on all the feature axes in ascending fashion. For each such ordering, prepare an array of class labels such that \( x_j \) is the \( j \)-th order sample on the \( j \)-th feature axis. Fig. 2, for example, shows the formation of such a label array for ordered samples from three categories.
   - Examine the label arrays for possible threshold points, i.e., locations of the parallel hyperplanes. A threshold point is said to exist between \( Y_{j\min} \) and \( Y_{j\max} \) on the \( j \)-th feature axis if \( f_{\text{min}} \) is \( \geq 0 \). If such a threshold point exists, select the threshold point giving the maximum amount of gain in average mutual information over all possible threshold locations along different feature axes. In case of a tie between many threshold locations, select the one for which the quantity \( f_{\text{min}} \) is the largest. Let \( T_{\text{jmax}} \) be the threshold value of the selected point which occurs on the feature axis \( j \) (NODESET). Let AMIG (NODE) be the corresponding average mutual information gain.

5) **Decision:**
   - Determine the node \( \text{max} \) from NODESET such that for any node in NODESET, \( \text{PAMIG} \) is maximized. For any node is calculated as the ratio of the number of samples in LIST (NODE) to the total number of design samples. Node \( \text{max} \) then defines the current partitioning hyperplane which is located at \( T_{\text{mm}} \) along the feature axis \( j \) (NODESET).
   - Increment ITREE by \( \text{PAMIG} \) and check whether it exceeds \( f_{\text{min}} \). If the answer is no, go to the next step; otherwise, go to the step Termination of the algorithm.

6) **Replace NODESET by its left and right descending nodes \( \text{max} \) and \( \text{min} \).** Determine the LIST (\( \text{max} \)) and LIST (\( \text{min} \)) by classifying the samples of LIST (\( \text{max} \)) with the threshold chosen above and go back to step Looping.

7) **Termination:** Denote all the nodes of NODESET as terminal nodes of the partitioning tree. Label each terminal node with the class label having the majority count in the list of design samples associated with that terminal node.

In the next section, we present three examples of the hierarchical partitioning trees obtained using this algorithm.

**IV. TREE DESIGN EXAMPLES**

Example 1: This example involves two perfectly separable pattern classes from a two-dimensional feature space. A design set consists of 200 independent bivariate observations of pattern class \( w_1 \) from the vertically hatched region of Fig. 3(a) having uniform distribution and 200 bivariate observations of pattern class \( w_2 \) drawn similarly from the horizontally hatched region of Fig. 3(a). Specifying zero error, the recursive partitioning algorithm generates the hierarchical tree of Fig. 3(b). (RC, T) for this tree is 1 bit, which is the same as \( f_{\text{min}} \) given by (5) with \( P_x \) equal to zero.

Example 2: This example utilizes the Iris data [8] of three categories in the four-dimensional space. Using the first 25 samples from each category, the above algorithm was applied to obtain a classifier with \( f_{\text{min}} \) equal to 1.28 bits, which corresponds to \( P_x \) of 5 percent. The resulting hierarchical partitioning tree is shown in Fig. 4. From the 75 samples of the design set, only one sample belonging to category Iris Versicolor was found to lie in the wrong region when the algorithm terminated. Using the remaining Iris data as a test set, the classification performance of the tree of Fig. 4 was found to be 97.33 percent. It is also seen from Fig. 4 that the first two features of the Iris data, i.e., sepal length and sepal width, do not appear at all in the partitioning tree. Thus, the algorithm has inherent feature selection capability. This fact is further illustrated by the next example.

Example 3: This example demonstrates the capability of the recursive partitioning algorithm in a real application involving handwritten numerical recognition. The above algorithm was applied to design a hierarchical classifier which is expected to be a part of the low-cost handheld numerical recognition system currently under development. The input to this classifier
is a set of features which is obtained from the coefficients of the Walsh series expansion approximating the boundary of the unknown pattern [9]. Letting \( \Phi(\lambda) \) represent the boundary information in the form of a normalized cumulative angular bend function as the pattern boundary is traced, we use the following finite Walsh series expansion to approximate it:

\[
\Phi(\lambda) \approx a_0 \text{Wal}(0, \lambda) + \sum_{i=1}^{2} a_i(0) \text{Cai}(i, \lambda) + a_i(1) \text{Sal}(i, \lambda) + a_i(2) \text{Sal}(N/2, \lambda),
\]

where

\[
a_0 = \int_0^1 \Phi(\lambda) \text{Wal}(0, \lambda) d\lambda,
\]

\[
a_i(0) = \int_0^1 \Phi(\lambda) \text{Cai}(i, \lambda) d\lambda,
\]

and

\[
a_i(1) = \int_0^1 \Phi(\lambda) \text{Sal}(i, \lambda) d\lambda.
\]

Fig. 5 shows an example of \( \Phi(\lambda) \) for the numeral four along with the unwounded cumulative angular bend function. The seven features used by the classifier are obtained as

\[
a_i = \left| a_i(0) \right|^2 + \left| a_i(1) \right|^2 + \left| a_i(2) \right|^2, \quad i = 1, 7.
\]

Each of these features thus corresponds to one of the first seven amplitudes in the sequency spectra of \( \Phi(\lambda) \). As the Walsh sequency spectra is sensitive to the choice of the starting point for the boundary trace, a normalization step in the computation of metrics is required. The distance of test is determined [9].

Taking 160 digitized samples of handwritten numerals, 16 each from categories 0-9, the above seven features for each pattern were extracted, and the resulting set of 160 seven-dimensional pattern vectors was passed on to the recursive partitioning algorithm. Because of the size of the design set and earlier results of the classification on these data [9], \( P_w \) was specified to be 20 percent. This corresponds to \( \alpha \) of 1.96 assuming equal a priori probability for all pattern classes.

The hierarchical partitioning tree generated by the above algorithm is shown in Fig. 6 where Roman numerals at each interval indicate the sequence in which these nodes were generated. Also indicated are the number of samples and the respective labels present in the LIST node for terminal nodes at the instant of termination of the algorithm. This indicates an average correct classification rate of 82.5 percent on the design set. The independent test set of 40 samples yielded a correct classification rate of 70 percent. This compares favorably with the classification performance of the nearest neighbor classifier on the same data. In addition, the following important observations can be made from the tree of Fig. 6:

1) Numerical categories 2, 3, and 4 show a great deal of resemblance in the normalized Walsh sequency spectra domain. This is an expected result because of the nature of the sequency computation normalization step [9].

2) Numerals 0 and 8 also show a lot of similarity. However, this similarity is due to the size of the quantization grid, and is again expected.

3) Features \( A_3 \) and \( A_4 \) do not appear as all in the partitioning tree, while feature \( A_4 \) appears only once. Thus, it can be safely said that the first four features are the most powerful, and possibly one can work with them only. This indicates the usefulness of the algorithm as far as feature selection and ordering are concerned.

Because of the small size of the design set, although not conclusive, can be placed in the tree of Fig. 6. The fact that the proposed algorithm has a good capability of producing an effective partitioning and bringing out the similarity between patterns.

V. CONFIDENCE BOUNDS ON ERROR PROBABILITY

Since the structure of the hierarchical classifier generated by the algorithm of Section III is similar to the empirical classifier of Henrichs and Fo [31], the bounds obtained by them are directly applicable here. Thus, the probability of error of the hierarchical classifier can also be expressed by the following inequality in addition to (5), i.e.,

\[
P_e \leq \sum_{i=1}^{n} \frac{\nu_i}{\nu_i} P_i y_i \]

where \( y_i \) is the probability that a pattern comes from category \( c_i \), \( u_i \) is a random variable having beta distribution as

\[
\nu_i \sim \text{Beta}(\gamma_i, \eta_i, a_i + 1)
\]

where \( \gamma_i \) is the total number of samples of category \( c_i \) in the design set, and \( \eta_i \) is the sum of misrecognized design samples from category \( c_i \) and the smallest number of terminal nodes of the hierarchical tree when its terminal nodes are reclassified as of category \( c_i \) or not, and merged to yield minimum nodes in the partitioning tree. Assuming feature independence, the upper bound on the probability of error can be determined to any degree of confidence by the following relationships:

\[
P_e(P_e \leq \frac{\nu_i}{\nu_i} P_i y_i u_i \gamma_i \eta_i > \beta^\gamma_i)
\]
VI. CONCLUSIONS

An algorithm has been proposed to partition the feature space for multifeature multicategory problems. The algorithm maximizes the mutual information gain at each partitioning step in the local sense, and therefore gives rise to a locally optimal decision tree. In case the globally optimum tree is desired, one can make use of available algorithms [10] which produce an optimal tree once the partitioning is specified. The resulting partitioning tree produced by the proposed algorithm can be used for classification as well as for data interpretation and pattern similarity analysis. Moreover, the classifier can also be implemented easily in hardware using threshold circuits.

REFERENCES


Fuzzy Tree Automata and Syntactic Pattern Recognition

EDWARD T. LEE

Abstract—An approach of representing patterns by trees and processing these trees by fuzzy tree automata is described. Fuzzy tree automata are defined and investigated. The results include that the class of fuzzy root-to-frontier recognizable Z-trees is closed under intersection, union, and complementation. Thus, the class of fuzzy root-to-frontier recognizable Z-trees forms a Boolean algebra. Fuzzy tree automata are applied to processing fuzzy tree representation of patterns based on syntactic pattern recognition. The grade of acceptance is defined and investigated. Manuscript received August 20, 1981; revised January 25, 1982. The author is with the Department of Mathematical Sciences, Memphis State University, Memphis, TN 38152.