Reprinted from

Symposium on

Machine Processing of

Remotely Sensed Data

June 29 - July 1, 1976

The Laboratory for Applications of
Remote Sensing

Purdue University
West Lafayette
Indiana

IEEE Catalog No.
76CH1103-1 MPRSD

AN APPROACH TO THE DESIGN OF A LINEAR BINARY TREE CLASSIFIER[†]

K. C. You and K. S. Fu

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

The overall performance of a multi-stage decision-tree classifier has been shown to be better than that of the conventional single-stage classifier with the same number of features because different feature subsets can be selected at different stages. But, the classification time increases due to the complexity of computation. The linear binary tree classifier designed by the method proposed in this study takes the advantages of the accuracy of a decision-tree classifier and uses linear discriminant functions at decision stages to reduce the classification time. An application of this method to the multispectral remotely sensed data is presented. All ten classes under consideration are assumed to be gaussian distributed. The result from a test on about 7000 samples shows that the linear binary tree classifier is more accurate and much faster than the maximum-likelihood classifier with the same number of features.

I. INTRODUCTION

Most literature of pattern recognition deals with single-stage classifiers and different types of discriminant function.[1,2] The conventional approach to multivariate and multiclass classification would be to perform tests on the unknown pattern against all classes using a particular feature subset and then assign the unknown to one of these classes. On the other hand, the decision-tree classifier assigns the unknown through a decision-tree procedure, which leads an unknown to a terminal node by serial stages of decision. That terminal node tells to which class the unknown is assigned.

We say that a tree classifier is binary, if and only if each nonterminal node in the decision tree has two subtrees. In other words, there are only two possible outcomes at each stage of decision. A binary tree classifier uses only one discriminant function $f(X)$, at each stage of decision. If the unknown falls in the region where $f(X)<0$, a decision is made to go through one subtree; if it falls in the other region where $f(X)\geq 0$, a decision is made to go through the other subtree, until a terminal node is reached.

A linear decision-tree classifier is a tree classifier of which all the decision rules can be mathematically expressed by linear functions. The tree classifier designed by this method is linear as well as binary.

The tree classifier takes advantage of using different feature subsets at different stages in contrast to the conventional single-stage classifier which uses only one fixed feature subset. It has been reported by Hauska and Swain[3] that the overall performance of a decision-tree classifier is better than that of the conventional classifier, but the classification time increases due to complexity of computation.

From the tree structure viewpoint, the conventional classifier can be expressed by Figure 1 as a single-stage classifier which uses only one particular feature subset. If each comparison is considered as a decision making, it can also be expressed by Figure 2 as a multistage classifier which can use different feature subsets at different stages. If different feature subsets are selected at different decision stages, the accuracy might be improved by proper selections, but the classification time will be long, because the probability distribution or density functions corresponding to different feature subsets for each class have to be calculated at different stages.

The structure of Figure 2 shows that m-1 decisions are necessary for classifying into m classes. And it is easily seen that there is only one non-overlap[†] class in the two next nodes of each decision stage; in other words, the information obtained in one decision stage is that the unknown does not belong to one particular class. If the number of overlap classes can be minimized, then the number of decisions for each class will be reduced, and the classification time will be reduced as well. So, the goal of constructing an efficient

[†]This work was supported by the National Science Foundation Grant ENG 74-17586.

[†]If two nonterminal nodes contain the same classes, then it is said that they have some overlap classes.

tree classifier is to find a tree, of which each nonterminal node has a minimum number of overlap classes in its next level nodes under some error bound. For example, Figure 3 has no overlap in the same level nodes, and the expected number of decisions is less than m-1. The expected number of decisions is calculated by the summation of the number of decisions for each class times the probability of that class. In the observation space, classifiers can be expressed in terms of discriminant functions. In non-parametric cases, the discriminant function can be obtained usually by an iterative method.[4,5]

Figure 4 and 5 show the 2-dimensional feature spaces of Figure 2 and 3 respectively, where the same feature subset selections and linear classifiers are provided.

In practical application, the distribution function of each class is usually unknown. A proper assumption of the form of distribution function is very important for deriving a good classifier. Fortunately, the assumption of multivariate gaussian distribution for each class is often reasonable in remote sensing problems.

Because of the linearity of the discriminant functions, the tree classifier constructed by this method is expected to be fast in classification time, though it might have some drawback in accuracy in comparison with a single-stage maximum-likelihood classifier with full features, because, in general, the discriminant function derived according to the Neyman-Pearson test[6,7] is not necessarily linear. Any discriminant function other than that from Neyman-Pearson test will have this disadvantage. But the loss in accuracy may be more or less compensated by the allowance of overlap and the different selections of feature subsets at different stages.

To store the data for a L-feature m-class gaussian distributed maximum-likelihood classifier, $m(L+1)(L+2)/2$, or $m(L^2+3L+2)/2$ memory locations are needed, since the covariance matrices are symmetric. But only about $(m-1)(2L+3)$ memory locations are necessary for storing a linear binary tree classifier. For m classes, if there is no overlap class after each stage of classification, the number of nodes in a tree is m-1, while there are 2 pointers, L feature indicators, L coefficients and one constant for each node. In case of 10 features and 30 classes, the comparison of memory locations will be 1980 to 667. Another advantage of a linear binary tree classifier is the simplicity of implementation in both hardware and software.

2. A SUB-OPTIMAL LINEAR BINARY TREE CLASSIFIER

For an n-feature m-class problem, the total number of possible trees N is roughly equal to $\sum_{i=1}^{K} 2^{n \cdot m_i}$, where K is the number of different node structures, and m_i is the number of nonterminal nodes in the i-th node structure. Though K and m_i are not determined in the above expression, it is obvious that the total number N is very large. In order to reduce the number of possible trees to some extent, such that they can be searched through for a good one, the first restriction suggested is limiting the number of features selected at each stage. If the number of features in a classifier is limited to L, the feature subsets of size less than L will not be considered not only because of accuracy, but also because of uniformity of the classifier. The destruction of uniformity will make the classifier more difficult to implement and will increase the design and classification time.

For the sake of accuracy, the second restriction is the size of the tolerable error probability at each stage. In order to reduce the number of overlap classes in the next level nodes of each stage of classification, some classes might have fairly large misclassification region in the feature space, or they might have pretty large error probabilities. An example is shown in Figure 6. If ten classes are considered and a linear classifier is desired between two nonoverlap groups of classes. They are group of classes 1, 2, 5, 8 and group of classes 3, 4, 6, 7, 9, 10. The best linear decision boundary may be F1 as shown, and this decision stage is shown in Figure 7. But classes 3, 4, 8 have high error probabilities. If the total error probability is higher than some error bound, the classifier may be useless no matter how fast the classification time is. The policy of dealing with this situation is to take the class, which has the largest error probability, out of consideration and let it overlap in the next level nodes, and then find the classifier again. In the last example, class 8 is taken out of consideration and then the linear decision boundary found will be F2, and the decision stage will change to that in Figure 8. It is obvious that the misclassification probability is lowered. If the number of classes under consideration is only two, that means we cannot do better, then let it stay.

To look for a tree, which has higher accuracy, a larger limit on the number of features and a lower error bound should be specified as parameters at the beginning of the design process.

For a decision stage, at which a linear classifier is being found, there are $\binom{n}{L}$ possible feature subsets under the limitation of L features out of n total features. Suppose that the stage has p classes under consideration, there are $2^{p-1}-1$ possible ways of grouping, which consists of two groups of classes. Because each class can be assigned to either one of the two groups, and the grouping with all classes in one group is meaningless, the total number of possible combinations of feature subset and grouping is therefore $(2^{p-1}-1)\binom{n}{L}$. But we need only one of them. An algorithm is necessary to select one good grouping among $2^{p-1}-1$, so that the number of varieties to be searched can be reduced to $\binom{n}{L}$. Besides, a function is necessary for evaluating the separability of the two groups obtained from the above algorithm. Upon the separabilities calculated from this function, a good feature subset and a grouping are selected.

It is obvious that a classifier which commits the minimum error will reflect a good performance. So, after a feature subset and a grouping are selected, a function is needed to calculate the error committed by the classifier. The process to find a classifier which has minimum error will somehow become an iterative procedure starting with an initial guess. The convergence of the iterative procedure is related closely to the initial guess and the error calculation function. If the error calculation function is first order differentiable with respect to the coefficients of the linear equation of classifier, the Fletcher-Powell algorithm is recommended.[8,9]

The initial guess of the classifier depends on the grouping selected. However, the classifier corresponding to the minimum error may not be able to separate the two groups of classes in the original grouping because the grouping algorithm may not work as well as it is expected. Hence, the classes considered have to be regrouped by the classifier obtained. Since the classifier divides the space into two regions, the classes are assigned to the same group if their means are in the same region, such that they are regrouped into two groups.

After the classifier and the associated grouping have been obtained, the error commitment of this classifier is checked by the error bound. If the error commitment is too high, take one class off and find a new classifier. If the error is lower than the error bound, build up the tree; that is, create two nodes for the two groups obtained from regrouping and link them to the node whose classes are grouped. And then check the tree to see whether every terminal node contains only one class or not. If yes, the tree is completed and the process terminates. If not, apply the classifier design process to one node which needs further operation. A flow chart is shown in Figure 9.

3. LINEAR BINARY TREE CLASSIFIER FOR MULTIVARIATE GAUSSIAN MULTICLASS CASE

Before getting into the description of the method, two lemmas are stated.

Lemma 1

A hyperplane $B^TX + C = 0$ is tangent to an equiprobability surface of Gaussian distribution, with mean M and covariance Σ, at point X_o, and X_o has conditional probability $Pr(X_o|w) = Exp(K/2)$, where

$$K = -\left(S_o + \frac{(B^TM + C)^2}{B^T\Sigma B}\right)$$

$$S_o = \ln|\Sigma| + n\cdot\ln(2\pi)$$

B,X,M are (n x 1) vecters, Σ is an (n x n) matrix, C and S_o are constants, and T indicates transpose.

It can be proved by finding the maximum probability point on the hyperplane.

Suppose that a linear decision surface $B^TX + C = 0$ is tangent to the equiprobability surface of class w_i at point X_i which has conditional probability $Pr(X_i|w_i) = Exp(K_i/2)$, where $K_i = -\ln(2\pi)^n|\Sigma_i| - \frac{[B^TM_i + C]^2}{B^T\Sigma_i B}$. How is the error probability calculated? The next lemma will show it.

Lemma 2

The error committed by a linear classifier on a gaussian distributed class w_i is $\frac{1}{2} \pm \frac{1}{2}\,\mathrm{erf}(\sqrt{-(So_i + K_i)/2})$, where the error function $\mathrm{erf}(x/\sqrt{2})$ is defined as $\mathrm{erf}(x/\sqrt{2}) = \int_{-x}^{x} \frac{1}{\sqrt{2\pi}} \mathrm{Exp}\frac{-t^2}{2}\,dt$.

Proof:

Let us rewrite the two equations from lemma 1 for class w_i

$$\frac{[B^TM_i + C]^2}{B^T\Sigma_i B} + So_i + K_i = 0$$

$$B^TX_i + C = 0$$

A two dimensional case is illustrated in Figure 10.

The distance between linear classifier and the mean is $|M_iQ_i|$, and

$$|M_iQ_i| = \left[\frac{-(So_i + K_i)\,B^T\Sigma_iB}{B^TB}\right]^{\frac{1}{2}}$$

Suppose that the variance of class w_i in the direction of the linear classifier is σ_i^2,

$$\sigma_i^2 = \frac{B^T\Sigma_iB}{B^TB}$$

It is known that the error probability committed by $B^TX + C = 0$ on class w_i is E_i, where

$$E_i = \int_{B^TX + C = 0}^{\infty} Pr(X|w_i)dX \text{ or } \int_{-\infty}^{B^TX + C = 0} Pr(X|w_i)dX$$

It depends on which side of the space of X is claimed to be the region of class w_i.

Refer to Figure 10. If left side of $B^TX + C = 0$ is claimed to be the region of class w_i, the error will be the region from T_i to ∞, while M_i is projected onto zero point. Thus

$$E_i = \int_{T_i}^{\infty} \frac{1}{\sqrt{2\pi}\,\sigma_i} \mathrm{Exp}\left[\frac{-t_i^2}{2\sigma_i^2}\right] d\,t_i$$

$$= \tfrac{1}{2}\left[1 - \operatorname{erf}\left(\frac{T_i}{\sigma_i}/\sqrt{2}\right)\right]$$

$$= \tfrac{1}{2} - \tfrac{1}{2}\operatorname{erf}[\sqrt{-(So_i + K_i)/2}]$$

If the other side of the space is claimed to be the region of this class, then the error will be

$$E_i = \tfrac{1}{2} + \tfrac{1}{2}\operatorname{erf}[\sqrt{-(So_i + K_i)/2}]$$

This situation should be avoided.

The overall performance of the classifier depends on the total error commitment of all classes to be classified at each stage instead of that of a single class. What is to be minimized should be the average error E_T.

$$E_T = \sum_{\text{(All the classes under consideration)}} \Pr(w_i)E_i$$

The procedure of design of the linear binary tree classifier is described as follows:

(A) Grouping Algorithm

With the previous two lemmas, the error commitment of a linear classifier at all stages can be calculated. Besides error calculation, one more algorithm is needed to group the classes under consideration into two groups with respect to each feature subset. And it is assumed that the grouping algorithm will produce two groups with pretty high linear separability among the $2^{p-1}-1$ possibilities, while p classes are under consideration. Figure 11 is the flow chart of the grouping algorithm we used.

(B) Separability Measurement

Through the grouping algorithm, the classes will be grouped in different ways with respect to different feature subsets. In order to choose a proper one among them, a measurement is needed to evaluate their respective separabilities. Bhattacharyya distance[10] between two gaussian distributed classes can be expressed as

$$B = Bm + Bv$$

$$Bm = \frac{1}{8}(M_2 - M_1)^T\left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1}(M_2 - M_1)$$

$$Bv = \tfrac{1}{2}\ln\left\{\left|\frac{\Sigma_2 + \Sigma_2}{2}\right| \Big/ |\Sigma_1|^{\frac{1}{2}}|\Sigma_2|^{\frac{1}{2}}\right\}$$

where Bm is due to the different of means and Bv is due to covariances. Since we are looking for a linear classifier, Bm is more significant than Bv. So, Bm is introduced to the method as a measurement of linear separability between classes.

The linear separability between two groups of classes is estimated by the summation of Bm's of all possible pair of classes between two groups resulting from the grouping algorithm. Among the $\binom{n}{1}$ possible combinations of grouping result and feature subset, the one which has the highest separability is chosen.

(C) Error Minimization Procedure

With some initial guess, the Fletcher-Powell algorithm is performed to get the classifier corresponding to the minimum value of error. The details of implementation of this part are given in the Appendix.

(D) Error Bound Checking and Regrouping

In concerning with the accuracy in performance, the error commitment of the classifier obtained from the procedure (C) is checked with the error bound as described in Section 2. If the error is higher than the error bound, the class which has the largest error probability will be taken out of consideration and will be in both groups after the classifier is found. If it is lower than the error bound, the classes under consideration have to be regrouped by the classifier obtained, as described in Section 2.

After the linear classifier and the associated regrouping have been obtained, as described in Section 2, the binary tree is updated and checked to see whether it is completed or not. If yes, the design process terminates; if not, the node design process is applied to the next node of which a linear classifier is to be found. A flow chart for the overall design procedure is given in Figure 12.

4. COMPUTATIONAL RESULTS AND REMARKS

The method has been applied to the pattern recognition problem of remotely sensed data of twelve spectral bands and ten classes (see Table 1). All computations were carried out on CDC 6500 at the Purdue Computing Center. All ten classes are assumed to be multivariate gaussian distributed.

In order to obtain information about prior probability of each class, training samples are selected periodically throughout the whole data set, and the prior probability is calculated by the ratio of number of samples from a particular class to the whole training sample size. Means and covariances of each class are calculated according to the sample mean and sample variance[1]. If the losses of misclassification of all classes are assumed to be equal, it has been proved that the Bayes decision rule with respect to some prior distribution is the maximum-likelihood decision rule[7], i.e., the unknown X is assigned to class w_i, if $\Pr(w_i)\cdot\Pr(X|w_i) \geq \Pr(w_j)\cdot\Pr(X|w_j)$, for $j = 1,\ldots,m$.

If the assumption of gaussian distribution is correct, and training samples are adequate, the highest accuracy which can be found will be that of maximum-likelihood test using full feature size; that is, twelve.

In Table 2, the training sample set and testing sample set do not contain any common sample. The 94.48% accuracy of performance of maximum-likelihood classifier with twelve features indicates that the assumption of gaussian distribution appears to be a good approximation of the true distribution. The linear binary tree classifier

with one or two features at each stage is much better in performance and is about twelve times faster in classification speed than the conventional single stage maximum-likelihood classifier with the same number of features, although it takes about one minute to design the tree.

The linear binary trees of 1, 2, and 3 features are shown on Figure 13, 14, and 15. It is seen that features 1, 3, 5, and 7 have never been selected in Figure 13; features 1, 6 have never appeared in Figure 14, and features 1, 3, 5 have never shown up in Figure 15.

Although the minimization of error commitment is emphasized in finding the linear classifier for the purpose of accuracy, the balance of the tree is also important in reducing the average classification time. Figure 13 is not balanced, but Figure 14 is better and is two levels less than Figure 13. The only difference in prespecification of the design is the number of features.

Suppose that a tree classifier has M nonterminal nodes, each nonterminal node has probability $Pr(N_i)$ and conditional error probability $E(N_i)$. The total error commitment of the tree classifier is less or equal to $\sum_{i=1}^{M} Pr(N_i)\, E(N_i)$, because misclassification regions of different stages may be partially overlapped. If the total error of the classifier has to be under a certain bound, a dynamic bound is suggested instead of the prespecified fixed error bound for all stages. For example, if we are dealing with the k-th nonterminal node and we have p more nonterminal nodes to do, the error bound for the current stage can be calculated by a function,

$$EB(k) = \frac{1}{p}\left[ET - \sum_{i=1}^{k-1} EB(i)\right],$$

where ET is the preset total error bound for the classifier.

Since the grouping algorithm employed may not work as well as desired, and the linear separability measurement used may not provide the true linear separabilities, they could be improved by selecting more appropriate grouping and feature subset, and a better result could consequently be obtained. Furthermore, the backtracking techniques could be introduced to attain the capability of looking ahead in designing the sub-optimal tree.

APPENDIX

A Brief Description of Fletcher-Powell Algorithm and Its Application to This Case.

The idea of the Fletcher-Powell Algorithm can be briefly described as follows:

* Problem: Find an $X = (x_1, x_2, \ldots, x_n)$ to minimize $E(X) = E(x_1, x_2, \ldots, x_n)$.

$X^{(0)}$ Iterative Method: Start with an initial guess and proceed iteratively until a local minimum is reached.

$$X^{(i+1)} = X^{(i)} + \lambda_i D_i$$

where $D_i = - H_{i-1} G_i$

$$\lambda_i = \min_{\lambda} E(X^{(i)} + \lambda D_i)$$

$$G_i = \left(\frac{\partial E}{\partial X_1}, \frac{\partial E}{\partial X_2}, \ldots\ldots, \frac{\partial E}{\partial X_n}\right)\Big|_{X = X^{(i)}}$$

$$H_i = H_{i-1} + \lambda_i \frac{D_i D_i^T}{G_i^T H_{i-1} G_i} - \frac{H_{i-1} Y_i Y_i^T H_{i-1}}{Y_i^T H_{i-1} Y_i}$$

$$Y_i = G_{i+1} - G_i$$

H_0 can be any positive definite matrix; for example, the unit matrix. The reader may refer[8,9] for details of this algorithm.

To apply this algorithm to our case, the derivatives of E_T have to be determined. From lemma 2,

$$E_T = \sum Pr(w_i)\left[\tfrac{1}{2} - \tfrac{1}{2}\, erf\left(\sqrt{-(So_i + K_i)}/2\right)\right]$$

where $-(So_i + K_i) = \dfrac{[B^T M_i + C]^2}{B^T \Sigma_i B}$

For the purpose of keeping the classifier in the region between two centers, U1 and U2, of the grouping, let it pass through some point G between U1 and U2.

$$B^T G + C = 0$$

and $$G = \sin^2\theta \cdot U1 + \cos^2\theta \cdot U2,$$

so $$C = - B^T[\sin^2\theta \cdot U1 + \cos^2\theta \cdot U2].$$

The derivatives of E_T with respect to $B = (b_1, b_2, \ldots, b_L)$, and θ are as following equations.

$$\frac{\partial E_T}{\partial B} = \sum Pr(w_i) \frac{\partial E_i}{\partial B}$$

$$\frac{\partial E_i}{\partial B} = \bar{W}_i\left[M_i - \frac{B^T M_i + C}{B^T \Sigma_i B} \sum_i B\right]$$

$$\frac{\partial E_T}{\partial \theta} = \sum Pr(w_i) \frac{\partial E_i}{\partial \theta}$$

$$\frac{\partial E_i}{\partial \theta} = \bar{W}_i[B^T(U2 - U1)]\sin(2\theta)$$

where $\bar{W}_i = \dfrac{-1}{\sqrt{2\pi}}\left[\exp\left(\dfrac{So_i + K_i}{2}\right)\right] \dfrac{1}{\sqrt{-(So_i + K_i)}} \left[\dfrac{B^T M_i + C}{B^T M_i B}\right]$

REFERENCES

1. K. S. Fu, Sequential Method in Pattern Recognition and Machine Learning, Academic Press, 1968.

2. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley, 1973.

3. H. Hauska and P. H. Swain, "The Decision Tree Classifier: Design and Potential," Proceeding of Second Symposium on Machine Processing of Remotely Sensed Data, 1975.

4. N. J. Nilsson, Learning Machines, McGraw-Hill, New York, 1965.

5. E. G. Henrichon and K. S. Fu, "A Nonparametric Partitioning Procedure for Pattern Classification," IEEE Trans. on Computer, Vol. C-18, No. 7, July, 1969.

6. J. Neyman and E. S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypothesis," Philos. Trans. Roy. Soc. London Series A 231, pp. 289-337, 1933.

7. T. S. Ferguson, Mathematical Statistics, Academic Press, 1967.

8. R. Fletcher and M. J. D. Powell, "A Rapid Descent Method for Minimization," Computer Journal, Vol. 6, ISS. 2, 1963, pp. 163-168.

9. J. Kowalik and M. R. Osborne, Methods for Unconstrained Optimization Problems, American Elsevier, 1968.

10. H. C. Andrews, Introduction to Mathemetical Techniques in Pattern Recognition, Wiley, 1972.

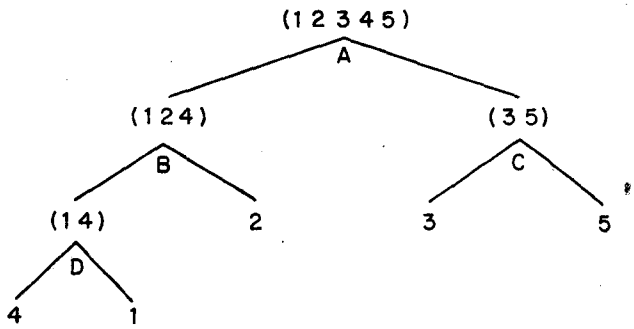

Figure 3

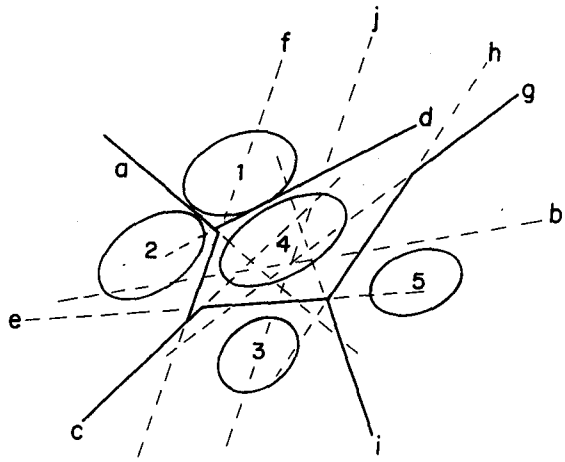

Figure 4

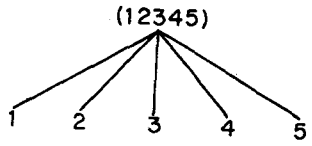

Figure 1

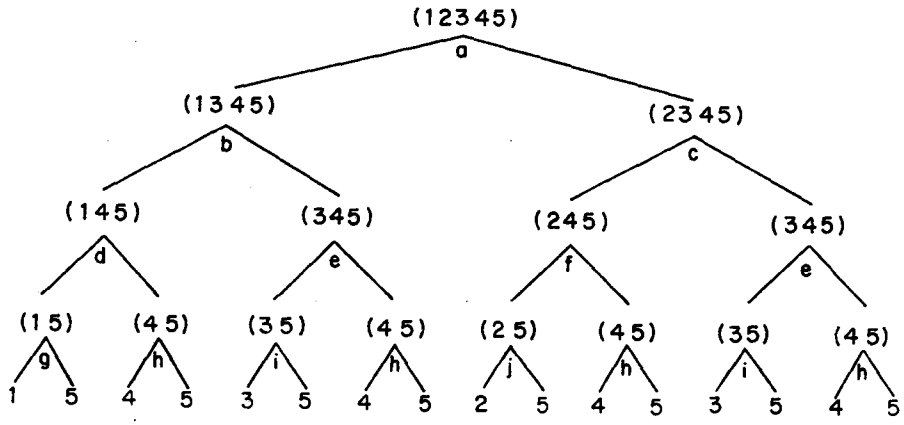

Figure 2

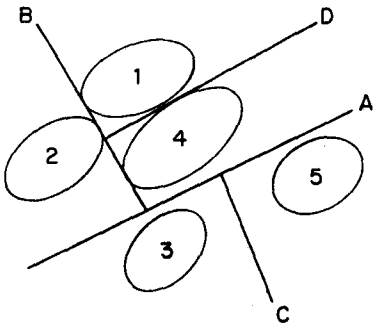

Figure 5

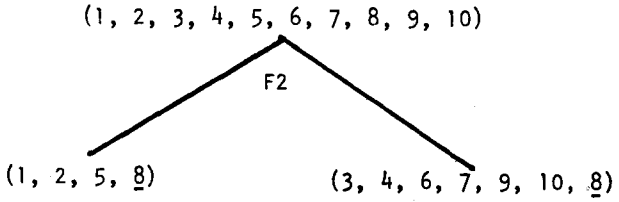

Figure 8

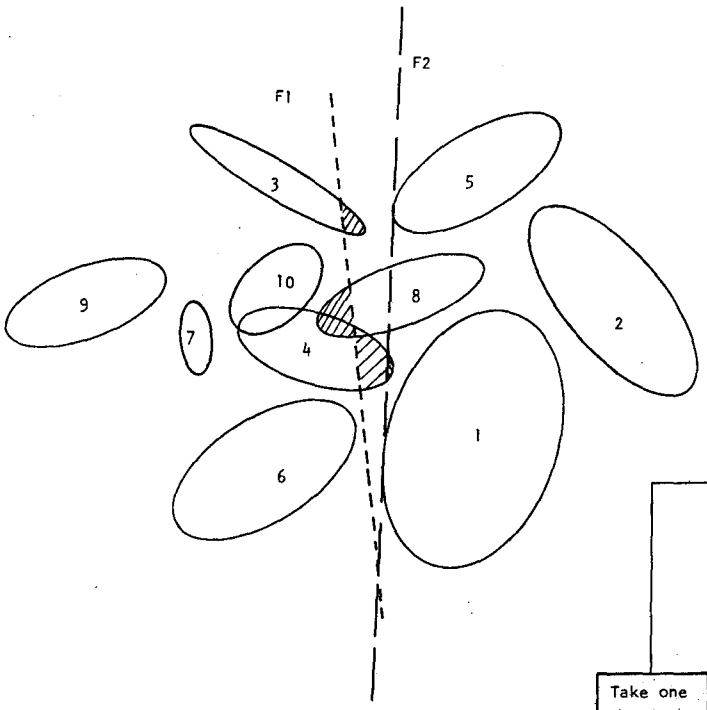

Figure 6

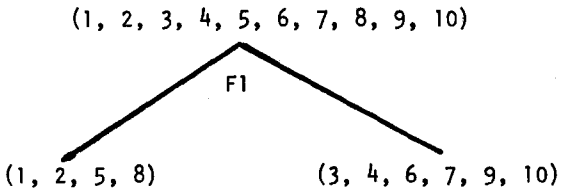

Figure 7

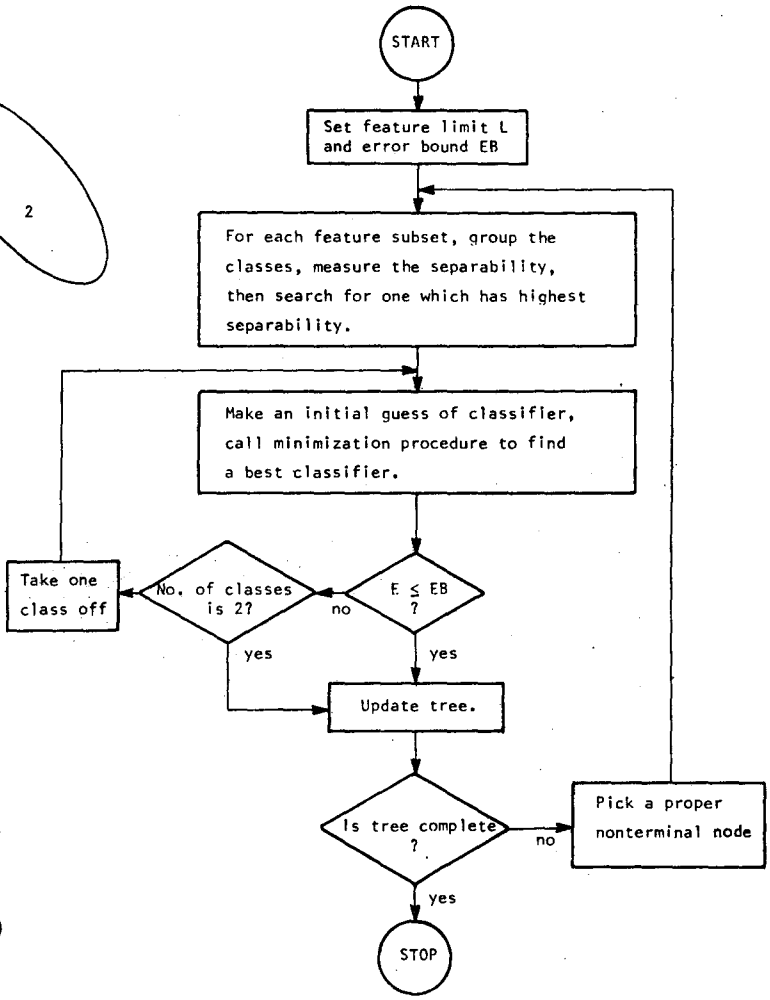

Figure 9

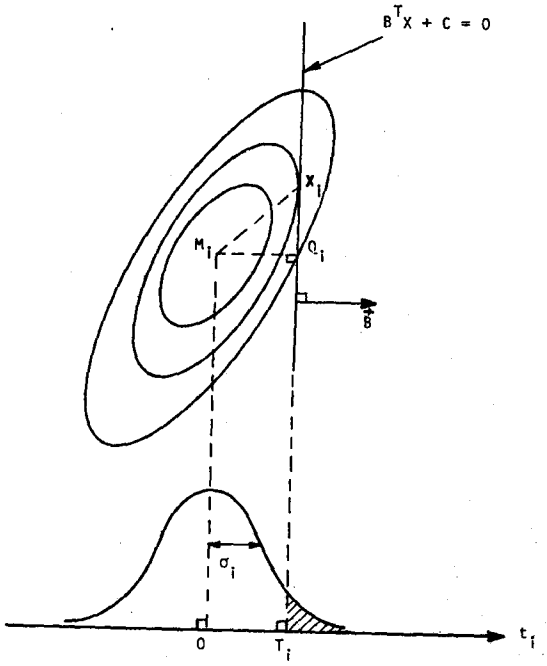

Figure 10

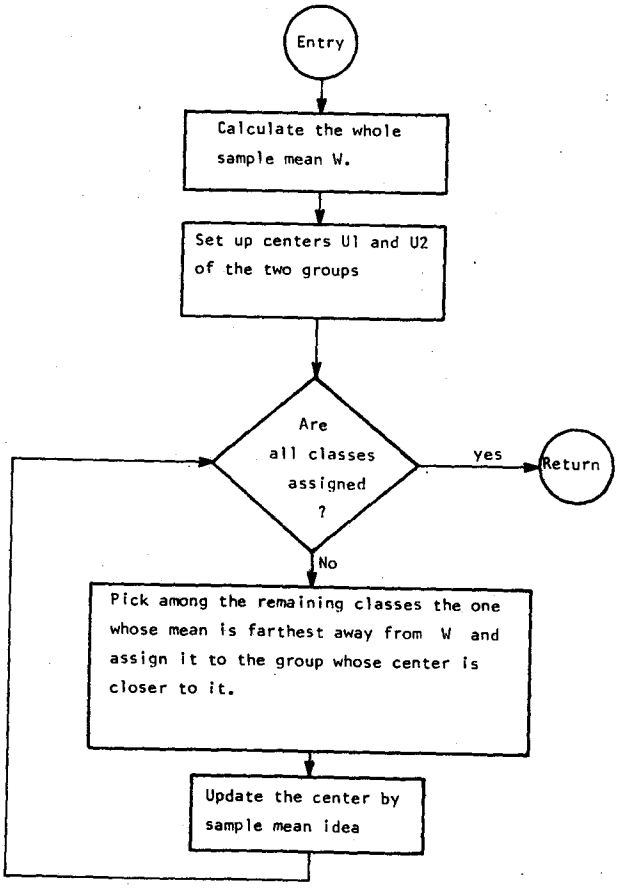

Figure 11

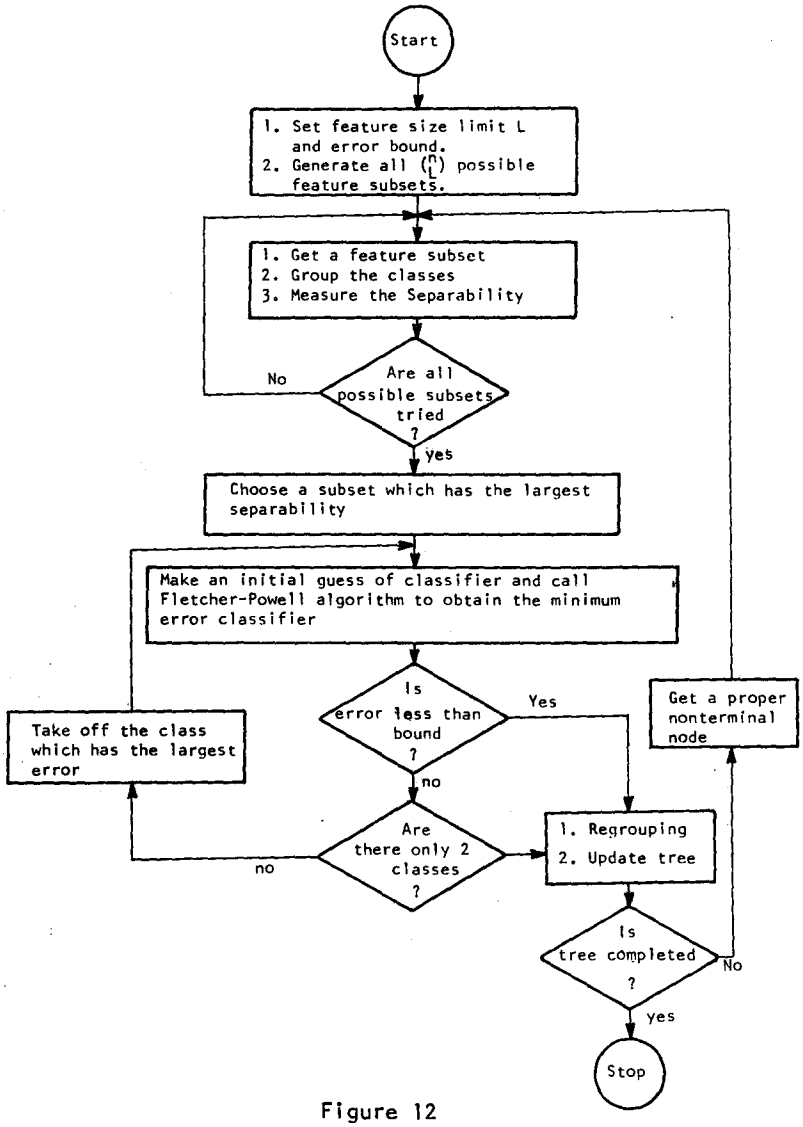

Figure 12

Class	Type of Ground Coverage
1	Corn
2	Soybeans
3	Hay
4	Oats
5	Non-Farm
6	Woods
7	Wheat
8	Pasture
9	Sudex
10	Set-Aside

Table 1

Classifier	No. of Features	Accuracy	Classification Time per Sample	Tree Design Time
M.L.C.	12	94.48 %	3.74×10^{-2} sec.	
M.L.C.	Best 1	60.87 %	4.39×10^{-3} sec.	
L.B.T.C.	1	75.68 %	2.75×10^{-4} sec.	74.4 sec.
M.L.C.	Best 2	77.99 %	5.34×10^{-3} sec.	
L.B.T.C.	2	82.82 %	4.05×10^{-4} sec.	41.2 sec.
M.L.C.	Best 3	83.95 %	6.81×10^{-3} sec.	
L.B.T.C.	3	83.08 %	5.34×10^{-4} sec.	74.3 sec.

No. of Classes 10

Training Samples 7906 Testing Samples 7288

Table 2

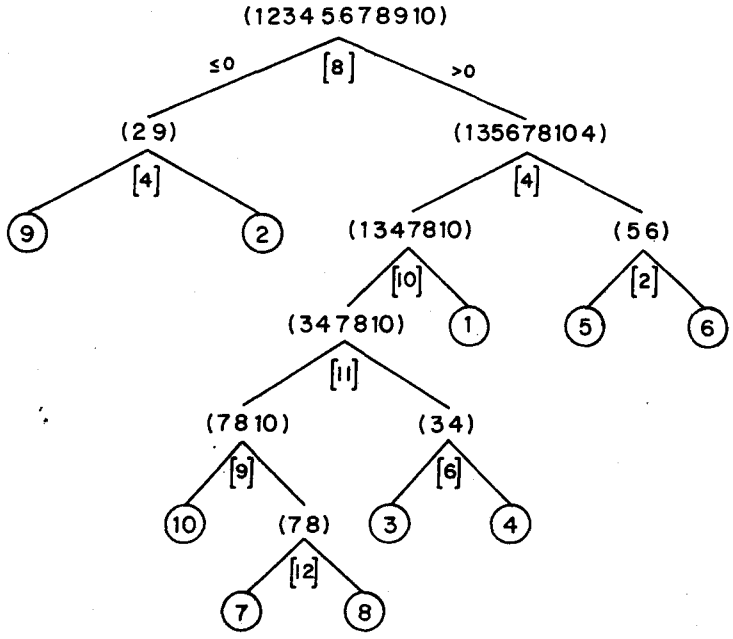

Figure 13. Tree structure with limit of 1 feature.

( ) Nonterminal node

◯ Terminal node

[ ] Feature subset selected

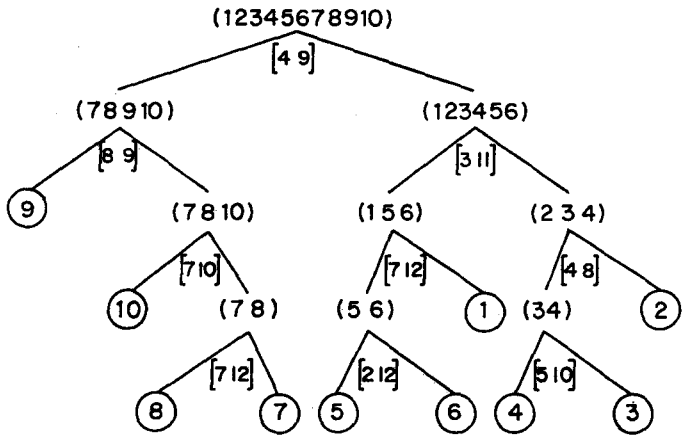

Figure 14. Tree Structure with Limit of 2 Features.

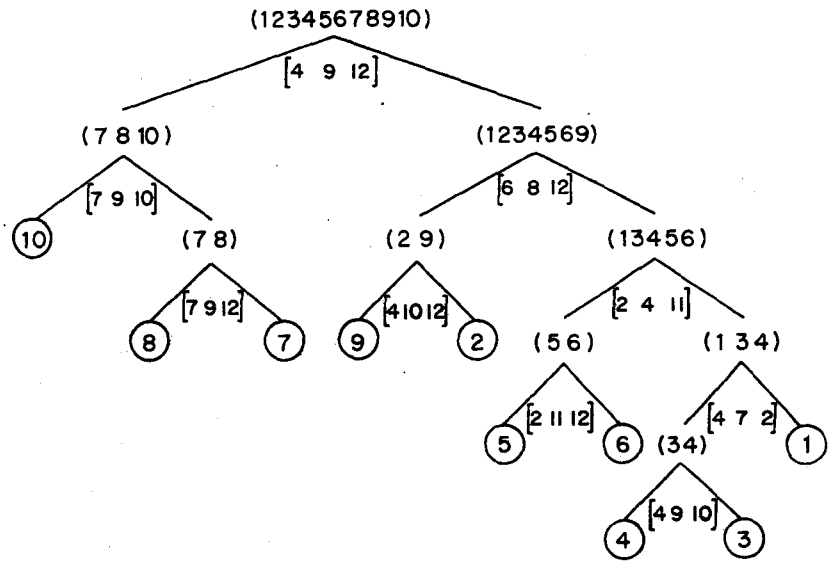

Figure 15. Tree Structure with Limit of 3 Features.