Reprinted from

Symposium on

Machine Processing of

Remotely Sensed Data

June 3 - 5, 1975

The Laboratory for Applications of
Remote Sensing

Purdue University
West Lafayette
Indiana

IEEE Catalog No.
75CH1009-0 -C

THE DECISION TREE CLASSIFIER: DESIGN AND POTENTIAL

Hans Hauska[+] and Philip H. Swain

Luleå University of Technology, Luleå, Sweden;
School of Electrical Engineering and Laboratory
for Applications of Remote Sensing,
Purdue University, West Lafayette, Indiana 47907

I. ABSTRACT

The basic concepts of a multi-stage classification strategy, the decision tree classifier, are presented. The two main methods to design decision trees are presented and discussed along with some experimental results. An attempt is made to describe an Applicable Logic for the design of decision trees. Advantages and disadvantages of the both design approaches are discussed.

II. INTRODUCTION

The launch of LANDSAT-1 in 1972 has greatly increased the amount of data flow in multispectral remote sensing. To handle this large and continually growing amount of data efficient numerical techniques have to be developed. Conventional maximum likelihood (ML) classifiers are characterized by two significant drawbacks:

1) Only one of the possible combinations of pattern features are used in the classification.

2) Each data sample has to be tested against all classes in a classification, which leads to a relative degree of inefficiency.

Another problem often encountered is the so-called dimensionality problem. With a fixed relatively small size training set the classification accuracy may actually decrease when the number of features is increased.

Another inherent weakness of the ML procedure is that the set of pattern features used in a classification is not necessarily the optimum for all the classes. Usually the set of pattern features is selected by the criterion of maximum inter-class separability, i.e., in a multi-class multi-feature classification the set of pattern features for which the average pairwise separability is largest will be used.

The number of tests necessary in a multi-class multi-feature classification can often be significantly reduced using a sequence of tests. Several types of multi-stage classification schemes are known. It is the purpose of the present paper to discuss a class of multi-stage classifiers which we call the decision tree classifier.

The decision tree classifier is essentially a maximum likelihood classifier using multi-stage decision logic. It is characterized by the fact that an unknown sample can be classified into a class using one or several decision functions in a successive manner.

This classification strategy can be most easily illustrated by a tree diagram (Fig. 1). For processing purposes the tree is encoded as a string of symbols such that there is a unique one-to-one relationship between string and decision tree. The string is decoded in the computer and pointers are set up to find the appropriate classification path for each data sample.

A tree generally consists of a root node, a number of nonterminal nodes or decision stages and a number of terminal nodes (final classifications). The root node or starting node of the tree is the set of all classes into which a sample could be classified. A set of nodes at the same level in the tree will be called a layer. For the purpose of graphically displaying a tree we adopt the same scheme as Slagle (1971). The decision after a stage is divided into a feature node, which is numbered according to the feature set

[+]ESRO Research Fellow at the Laboratory for Applications of Remote Sensing, Purdue University

used and shown as a triangle and decision nodes, which show the possible classification paths at a certain stage. The decision nodes can contain more than one class, but for the sake of readability we will only mark the terminal nodes with the class number (Fig. 1)

We will discuss two methods to design decision trees, some experimental results and an outlook on the potentials of this classification approach when applied to the problem of classifying cover types below clouds by using multitemporal data. We will also outline a recommended procedure for designing a tree and for deciding, before actually classifying the data, if a tree will classify accurately and efficiently.

III. THE DESIGN OF DECISION TREES

To achieve the best possible performance with a classifier as described above, the design of the decision stages is of utmost importance. The choice of tree structure and the choice of appropriate feature subsets will reflect itself in performance (classification accuracy) and efficiency (computer time used for the classification). In the following sections we will discuss two approaches for the design of trees. These approaches are similar in principle, but differ widely in the way the tree is actually designed.

III.1. THE MANUAL DESIGN PROCEDURE

After the statistics for all the classes have been obtained, a graph is obtained, in which the means and variances for all the classes are plotted for each feature. It is then possible to determine from this graph suitable decision boundaries, such that all classes are separated in a finite number of decision steps. As long as one restricts the number of features used in each stage to one, this is roughly equivalent to estimating a simple distance measure between classes. The method is no longer suitable, however, when two or more features are used in a stage of the tree. For two or more features it is not possible even for an experienced analyst to accurately estimate interclass separabilities. To illustrate the procedure, Fig. 2 shows a so-called coincident spectral plot for a 8 class 13 feature classification (courtesy of L. A. Bartolucci). The data was taken by Skylab on June 5, 1973 over a test site in the San Juan Mountains, Colorado. We have chosen this particular example because it demonstrates both the advantages and disadvantages of this approach. A first inspection of the figure shows that the statistics for classes D, E, and F are rather ill-conditioned; that is, in a one-stage maximum likelihood classification scheme these feature sets could not be used because a zero variance indicates the existance of a singular covariance matrix for these classes. Also, the computation of any separability measure (such as Divergence or Bhattacharya distance) would be inhibited due to problems in matrix inversion.

In the multi-stage approach to classification these features or feature sets can be used as long as the decisions arrived at from this stage do not lead to a terminal stage for a class or a mixture of classes with singular covariance matrix for this particular feature or feature set. This behaviour is due to the fact that the discriminant function at a stage is computed from the pooled statistics of the representative classes in that node. In other words, as long as the distribution for the mixture of classes in a node does not have zero variance for any feature this feature or a set of features containing it can be used for a decision leading to this node.

As can be seen from the figure, feature No. 5 is well suited to separate classes (A,B,C) from classes (D,E,F,G,H). The first layer in our tree is thereby determined. Thereafter we have to look for a feature set that can classify the mixture of classes (A,B,C) into single classes. Feature set No. 7 or 8, or a combination of both can be selected. The mixture (D,E,F,G,H) also has to be divided into its components. Feature No. 7 shows good separation between classes H,G and the mixture (D,E,F). Again this feature can be used despite the ill-conditioned statistics of class D because this class is pooled together with E and F. Finally, to separate all classes, the mixture (D,E,F) has to be separated. Features 9 and 10 show good separation and have therefore been chosen. The tree was encoded in a string and the data classified. The result of the classification was good, but due to an existing correlation between classes D,E,F,G and H and topography, we felt it could be improved by adding an additional feature in the classification of classes (D,E,F,G,H) and (D,E,F). The tree was modified (Fig. 3) and the classification accuracy improved especially for classes D,E,F,G and H, which were of particular interest to the analyst. The example shows clearly a disadvantage of this approach for the tree design. The coincident spectral plot only permits a rough estimation of interclass separability for single features. If one desires to use a combination of several features, one has to design several different trees and try them for performance and accuracy. This trial and error method seems quite costly to us; therefore we will generally not recommend the use of this procedure. Also, it requires great skill and experience to design a tree when the number of classes is large.

Generally the Manual Design Procedure is justified due only to its applicability in cases where either one or a combination of the following circumstances occurs:

a) few classes
b) ill-conditioned statistics
c) the analyst is only interested in a subset of all classes in the data, which can be easily separated from the rest of the classes using a single feature.

This is sometimes the case,for example, in mapping of water bodies, as water usually can be easily separated from everything else. Here another advantage of the decision tree is apparent. Being interested in the classification of water only, the analyst can terminate all other nodes without having to explicitly classify the other classes. It is enough to decide if a sample is water or not. This is a good example of a case in which the decision tree classifier is much more efficient than a conventional maximum likelihood procedure using the same number of features.

III.2. THE OPTIMIZED LOGIC TREE DESIGN

In the manual procedure little concern is given to higher dimensionality feature subsets and to the cost of the classification expressed in terms of the probability of misclassification and the classification time. It is clear from the manual procedure, that there does not exist one unique decision tree. In order to determine the optimum tree under given constraints a heuristic search method is used. The method can be described as 'guided search with forward pruning' (Wu, 1975). The essential concept of this method is that a) an evaluation function is used to guide the search and b) after all possible candidate structures for a certain stage have been evaluated only the one with the highest evaluation is retained, thus decreasing storage requirements (Nilsson, 1971).

As it is normally impractical to search all possible feature subsets (the example used in the Manual Design Procedure has $2^{13}-1$ subsets) the analyst has to select a number of candidate feature subsets, which are likely to serve the purpose.

For the selected feature subsets a measure of interclass separability for all classes has to be computed using a distance processor. Either the transformed divergence (Swain et al, 1971) or the Bhattacharya-Distance can be used. For each feature subset a nonsupervised clustering is performed on the distance matrix in order to find mutually dissimilar and separable groups of classes. The cluster extraction and sorting procedures are those suggested by Bonner (1964) and modified by Wu (1975). The cost of the stage is then evaluated and a pointer is set up for the subset with the maximum evaluation. The clustering procedure is not always successful, i.e., no distinct, mutually dissimilar groups of classes could be found. In this case, the decision at this stage will be made using a conventional maximum likelihood classification with the feature subset with the largest average pairwise separability.

The evaluation function used in the search is an additive measure of efficiency and accuracy. For each candidate structure following node d_i the evaluation is computed as follows:

$$(1)\quad E(d_i) = -T(d_i) - K\cdot \varepsilon(d_i) + \sum_{j=1}^{c_i} E(d_{i+j}).$$

The evaluation for a given stage d_i consists of the sum of the negative of the classification time and the negative of the classification error at this stage. K is a user-prespecified weight constant which determines the tradeoff between efficiency and accuracy. The summation term is the predicted evaluation of further stages developed from this stage. The node d_i will have c_i immediately descending nodes d_{i+j}, where $j=1,\ldots,c_i$. In order to improve the performance of the classifier, a constraint is applied to the evaluation function at each stage. A conventional ML procedure is evaluated at each stage and is used as a lower bound for the evaluation of this stage in the tree. I.e., if none of the evaluations for the candidate structures are greater than the evaluation for the conventional structure, the conventional structure will be used at this stage. The constraint can be expressed as

$$(2)\quad E_o(d_i) = -T_{o,n_i}(d_i) - Kx\ \varepsilon_o(d_i)$$

with n_i the number of classes in d_i

Since for all candidate structures E_o is constant, it is subtracted from $E(d_i)$. This implies that evaluations smaller than zero will lead to the use of a conventional procedure, due to the constraint mentioned above. The efficiency-related terms are then normalized by dividing by the time used for classifying the data set according to a conventional ML procedure, using the highest dimensionality feature set. This is done to provide commensurability with the terms expressing accuracy. The evaluation function can then be rewritten as follows:

$$(3)\quad E^1 = \frac{1}{T_{oo}}\ [T_{o,n_i}(d_i) - T(d_i)] + \sum_{j=1}^{c_i} E(d_{i+j}) - K\cdot\varepsilon(d_i)$$

T_{oo} the time for a ML procedure (all classes, highest dim. feature subset)

The classification time used at a certain stage d_i is expressed as a product of the probability P_i that the classification path will go through d_i and the classification time for a conventional ML procedure with m_i features and c_i immediately descending nodes, (classes).

The evaluation function for stages developing further from the immediate descendents of a node is difficult to estimate, especially in terms of accuracy, as these stages have not been developed yet. The accuracy-related term for the evaluation of further stages is therefore expressed in form of a positive constant, a bias constant.

The accuracy-dependent term for the conventional procedure can also be included in the bias constant. Eq. 3 can now be rewritten

$$(4)\quad E^1(d_i) = \frac{1}{T_{oo}} \{P_i \cdot T_{o,n_i} \; P_i \cdot T(d_i) + \sum_{j=1}^{c_i} P_{i+j} \cdot T(d_{i+j})\} + K \cdot [C' - \varepsilon(d_i)]$$

This is the final form for the evaluation function for one stage of the tree.

The computation of the evaluation function is now reduced to estimates of the computation time, the probability of the classification path to go through this particular node and the probability of error for this stage. The computation times are easily estimated, as they can be experimentally determined by measuring the times used for classifying a data set of a certain size with a certain number of features and a number of classes. It is more difficult to estimate the probabilities for the classification path going through this node and to estimate the probability of error. These probabilities can be estimated by means of a good measure of pairwise interclass separability. A detailed account on the estimation of these probabilities is given by Wu (1974).

Despite the fact that the search algorithm is designed to maximize the evaluation at each stage, it cannot be assumed that the overall evaluation (i.e., the sum of the evaluations of all stages) also has a maximum. This is mainly due to

a) not all possible tree structures are evaluated
b) the evaluation at each stage is only an estimation.

Although the search described above is not strictly optimal, improvement in classification performance over conventional classifiers can be achieved. A flowchart of the search procedure is given in Fig. 4. Limiting the number of feature subsets, the search procedure is rather efficient, and more sophisticated trees than with the previously discussed manual design approach can be obtained.

III.3. EXPERIMENTAL RESULTS

A series of experiments have been performed designing a number of trees for a 19 class 4 feature classification (courtesy M. D. Fleming). The data was taken by LANDSAT-1 over Kenosha Pass, Colorado on August 15, 1973. The area classified was part of frame no. 1388-17134. As the number of available feature subsets was small (15), all subsets were considered in the search. Several parameters were systematically varied.

Performance and accuracy of the trees have been compared to classifications obtained by one-stage procedures with our classifier and with the LARSYS maximum likelihood classifier using 4, 3, and 2 features. As compared to the one-stage procedure using the layered classifier program, the slowest tree was approximately 40% more efficient. The fastest tree on the other hand was 70% more efficient than a comparable one-stage procedure with the same maximum number of features.

It is very difficult to make objective comparisons between the efficiency of decision trees and the efficiency of the LARS Maximum Likelihood classifier, as the classifying subroutine in the LARS classifier classifies a whole line of data at the time and is written in Assembler language, while our program calls a similar FORTRAN subroutine for each data point. Despite this fact, we have been able to design decision trees that give the same accuracy as the LARS classifier, but are considerably more efficient, (Table 1).

For each set of parameters, two trees, one using transformed divergence (D_T) and the other using a transformed B-distance for error estimation were used. The transformed B-distance is defined as

$$(5)\quad B_T = 2000 \cdot \mathrm{erf}(\sqrt{B})$$

The best tree designed using B-Distance and allowing selection from all 15 feature sets was approximately 16% faster than the best tree using D_T. Figures 5a and 5b show the best tree designed with D_T and B_T respectively.

Similar results can be obtained using 14 feature sets in the search (feature sets

with one, two and three features). In the case that only 10 feature sets (sets with one and two features) are employed in the search, we have been able to design a decision tree that exceeds the LARS classifier in accuracy by one percent. As it would exceed the scope of this paper to explicitly discuss the experimental results, we refer to a technical report to be published in the near future.

It appears, however, that the B-Distance is a more effective measure than the transformed divergence. This is presumably due to the fact, that the B-distance is related to a tighter error bound than the transformed divergence (Swain and King, 1973; Whitsitt and Landgrebe, 1974). From the results discussed above we conclude that regardless of the higher cost of computation for the B-Distance as compared to the transformed divergence (Swain et al, 1971), the former should be used in the search processor, as the accuracies are approximately the same but the classifier efficiency tends to be better for trees designed from the B-Distance.

A few words should be said here about the advantages and disadvantages of the optimized tree design. It is clear that the trees designed by the search processor are superior to the manually designed trees both with regard to accuracy and efficiency. Also, a greater number of feature subsets can be considered, as a modern high-speed computer is capable of handling large amounts of data much faster than a human being. Another advantage is the fact that the different stages of a tree are designed by using objective criteria about the separability of classes or groups of classes. Still, we feel that the search procedure will not always produce the tree best suited for a certain problem (we noted earlier that the result is not strictly optimal).

Very little can be found in the literature about criteria to decide when a decision tree is good and when not. In the following section we will propose a logical way to procede in the tree design and how to decide whether or not a tree is accurate and efficient.

IV. SUGGESTIONS FOR AN APPLICABLE TREE DESIGN LOGIC

The approaches to the design of decision trees described above do not give the analyst a hint to decide what tree to use. In the following we will give a few rules which are not theoretically founded but appear practicable. We will try to treat both approaches in parallel rather than separately, as there are many similarities between them. As the classifier was developed in a LARSYS environment, the procedure assumes that facilities are available for LARSYS-like data processing. The rules are general, but they will have to be slightly modified under different conditions.

1. Train (determine class statistics) carefully and combine or delete those classes that have very low separability.

2. Use a feature selection algorithm to determine the feature sets to be used in the search procedure. Always obtain a coincident spectral plot for the classes and candidate features. It is recommended that not more than 30-50 feature subsets should be used. The feature subsets chosen for the search procedure do not have to fulfill the criterion of maximum separability for all classes. One may very well select subsets that provide good separability for a small subset of classes only. The feature subsets should always be selected with regard to the particular problem the analyst is working with. In the case of LANDSAT data it is recommended to use all 15 feature subsets in order to maintain acceptable accuracy.

3. Use the Distance processor (a FORTRAN program that computes either transformed Divergence or Transformed Bhattacharya distance for selected feature subsets) to obtain pairwise interclass separability for all feature sets to be used in the search. This processor is also very useful in the Manual Design Procedure, as it allows to compute separabilities for selected feature subsets only.

4. Use the search processor to design a tree, or design a tree manually with the help of the coincident spectral plot and the interclass separability information for the candidate feature sets.

5. Draw the tree.

A good tree should have the following properties:

It should have both breadth and depth. A tree that is only deep or only broad will either be less accurate or very inefficient, sometimes both. The tree might not have any terminal nodes in the first layer and hopefully as few as possible in the second layer. The more terminal nodes in the first layer, the closer the tree approaches

a conventional one-stage maximum likelihood classifier. It is not always possible though, to obtain a tree both broad and deep. The tree structure is highly dependent on the interclass separability and the distance measure used. Design therefore several different trees by keeping the threshold for interclass separability fixed and varying the tradeoff constant. Use the tree that most closely resembles the properties mentioned here in the classification.

A comparison between the two approaches for the tree design shows, that they are similar in their general concepts. In the Manual Design Procedure mutually dissimilar groups of classes are selected by the analyst. In the Optimized Logic Tree Design, the same operation is performed by the computer.

The Optimized Logic Tree Design gives the possibility to perform an exhaustive search of the given subset of the feature space. Nevertheless, the use of the search processor is not a substitute for the experience and insight of the analyst in a particular problem. We want to emphasize here, that the software discussed here has been developed as a tool to provide accurate and efficient multi-class classifications for remotely sensed data. The analyst will have to utilize this tool to the best of his knowledge.

In view of the criteria outlined above, we find that the tree we constructed for the example in Fig. 2 appears to be a rather good tree.

V. POTENTIALS OF THE DECISION TREE CLASSIFIER

In remote sensing applications the existence of clouds in a data set is a great inconvenience. A data set cannot be very effectively used in a multi-date overlay if clouds exist in one of the dates. We have selected such an overlay and performed a small experiment.

The data consists of an overlay of five data sets over the same geographical area, in three of which clouds are present. In this first experiment we have chosen two out of these five data sets - one with clouds and one that was completely cloud-free. We have then designed a tree for each of the two data sets. The tree for the first data set included the classes 'clouds' and 'cloud shadow'. The tree for the second data set did not contain these classes. As a basis of the classification we chose the data set containing clouds. The tree for classifying this data set was then modified in the following manner: in the tree for the data set with clouds, the terminal nodes for clouds and cloud shadow were eliminated and in their place the root node for the tree without clouds was inserted. In other words, in cases where the classification path originally would lead to a class 'clouds' or 'cloud shadow' the second date was used to classify the actual ground-cover. Figure 6a shows the trees for the respective parts of the overlay and Fig. 6b shows the combined tree.

We believe, that this method of inserting one tree into another can be easily applied in all cases of multi-temporal overlays with no more than 30% cloud cover in a single date. The data from LANDSAT are collected every 18 days over a certain region and the probability of clouds existing in the scene is (averaged over a whole year for the central USA) 0.5. If we use 30% cloud cover in a conservative estimate we find that the probability of a certain point on the ground covered with clouds in both dates reduces to 0.0225. Using a third date reduces this probability even more. This line of reasoning is valid as long as we can assume the cloud cover in a scene taken with 18 days interval as an independent parameter.

Another field of potential applications appears to be data sets which are of a higher complexity than just remotely sensed reflectance data. We are here thinking about an overlay of spectral, geomagnetic, gravitational and other data used for instance for exploration purposes of natural resources. Unfortunately we have not been able yet to work with such a data set experimentally, but we hope to be able to do this very soon.

VI. CONCLUSIONS

The discussions of the manual procedure and the optimized logic tree design have shown that both are very similar in nature, although the latter approach is more general. We have shown, that the manual approach is useful in certain special situations, e.g., existing danger of encountering singular covariance matrices.

Generally we prefer the Optimized Logic Tree Design, as the criteria used in the search are not guided by rules of thumb. We will therefore generally design a better tree using the search processor than is possible in the Manual Design Procedure.

We have tried to give an a priori rule how to determine the eventual performance of a tree. The criterion lacks objectivity and is therefore not regarded as entirely satisfactory. Experiments have shown that

the decision tree classifier can be designed to be much more efficient with maintained accuracy than a conventional ML procedure. The gain in efficiency generally being on the order of 30-50%.

In a small experiment we have shown that it is possible to classify ground cover types using multitemporal overlays containing clouds, which is not possible using a conventional ML classifier.

ACKNOWLEDGEMENT

This work was sponsored by the National Aeronautics and Space Administration under Contract No. NAS9-14016. The Skylab data cited in Figures 2 and 3 was made available through NASA Contract No. NAS9-13380. One of us (HH) is grateful for a postdoctoral grant by the European Space Research Organization. We are indebted to Dr. C. L. Wu on whose original work our continuing research is based.

REFERENCES

Bonner, R. E., 1964. On Some Clustering Techniques. IBM Journal, January, 1964.

Nilsson, N. J., 1971. Problem Solving Methods in Artificial Intelligence, McGraw-Hill, New York.

Slagle, J. R. and R. C. T. Lee, 1971. Application of Game Tree Searching Techniques to Sequential Pattern Recognition. Comm. ACM, 14, 2, February 1971.

Swain, P. H., T. V. Robertson, and A. G. Wacker, 1971. Comparison of the Divergence and B-Distance in Feature Selection, LARS Information Note 020871.

Swain, P. H., and R. C. King, 1973. Two Effective Feature Selection Criteria for Multispectral Remote Sensing, Proc. First International Joint Conf. on Pattern Recognition, Washington, November 1973.

Whitsitt, S. J., and D. A. Landgrebe, 1974. Simulation Techniques for Estimating Error in the Classification of Normal Patterns. LARS Information Note 040174.

Wu, C. L., 1975. The Decision Tree Approach to Classification, Ph.D. Thesis, School of Electrical Engineering, Purdue University, May, 1975.

Table 1. Table shows best trees using 15, 14 and 10 feature subsets in the search. Both Transformed Divergence and Bhattacharya-Distance are used. LARSYS means Lars ML classifier. One-stage is Decision Tree Classifier program. Efficiency comparisons have to be made with caution (see text).

MAXIMUM NUMBER OF FEATURES	DISTANCE MEASURE	THRESHOLD	TRADE OFF CONSTANT	CLASSIFICATION TIME IN SECONDS	TRAINING FIELD PERFORMANCE		CLASSIFICATION TIME IN SECONDS		TRAINING FIELD PERFORMANCE	
					OVERALL	BY CLASS	LARSYS	ONE-STAGE	OVERALL	BY CLASS
4	B_T	1950	20.0,10.0	545	93.5	94.2	920	1574	93.7	94.5
4	D_T	1950	10.0	655	93.6	94.3				
3	B_T	1950	10.0	440	92.9	93.3	650	1036	93.0	93.7
3	D_T	1950	25.0	520	92.9	93.3				
2	B_T	1950	5.0	450	91.6	92.0	360	650	90.2	91.1
2	B_T	1850	5.0	390	90.4	91.1				
2	D_T	1850	5.0	435	92.2	91.3				

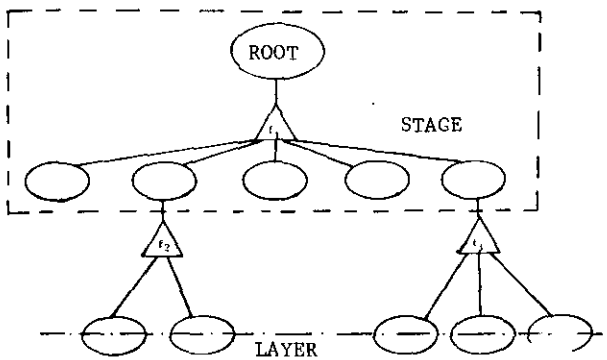

Figure 1 (above). A simple tree structure. Decision Nodes are drawn as ellipsies. Feature nodes are drawn as circles. The nodes inside the dashed line represent a stage. Nodes along the dot-dashed line represent a layer.

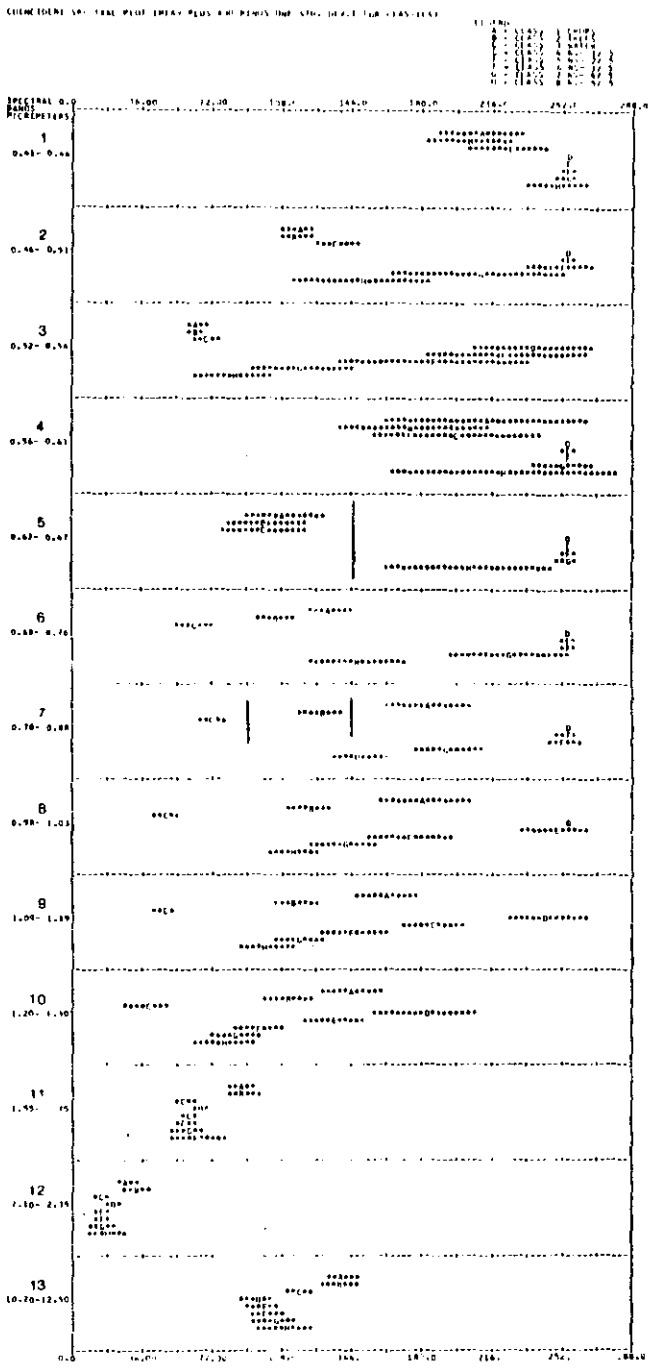

Figure 2 (right). Coincident Spectral Plot. The 8 classes are marked by letters A through H. The features are numbered from 1 through 13. Decision boundaries are outlined for features 5 and 7.

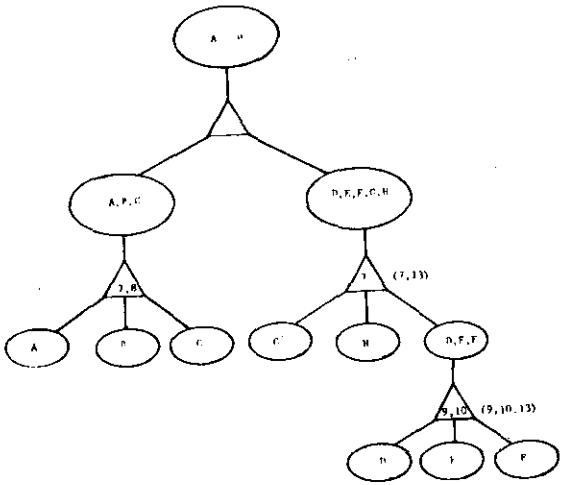

Figure 3. Tree for 8-class Skylab classification. Classes are marked by letters A through H. Feature sets in parentheses are used in improved version of the tree.

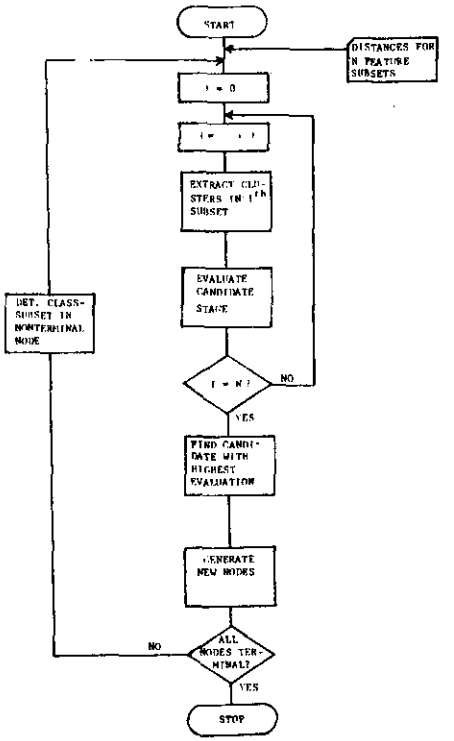

Figure 4. Flowchart for Optimized Logic Tree Design

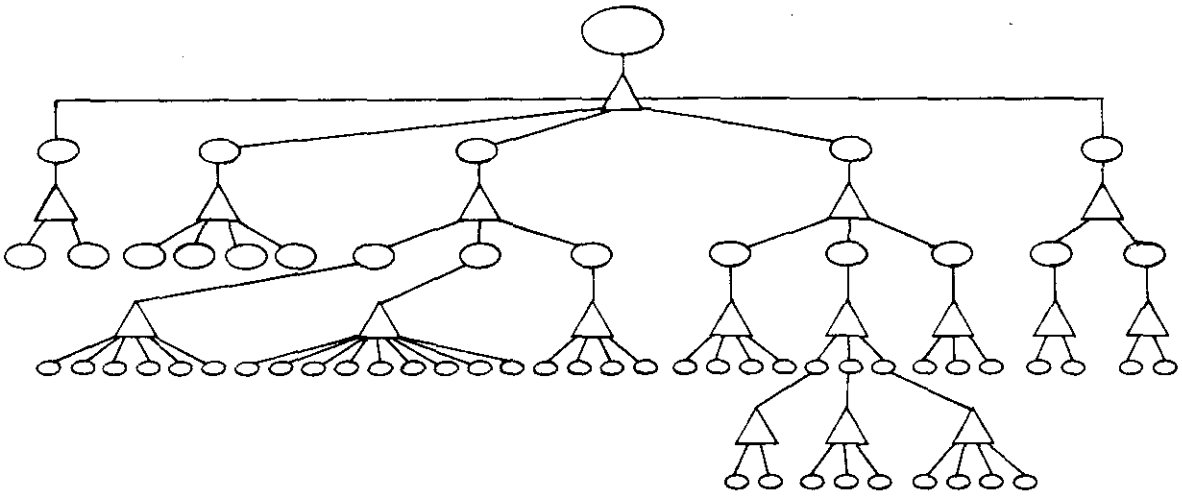

Figure 5a. Graph of structure of best tree designed by Optimized Logic Tree Design using Transformed Divergence and all 15 feature subsets. For results see Table 1, second row. Feature sets and classes not shown for clarity.

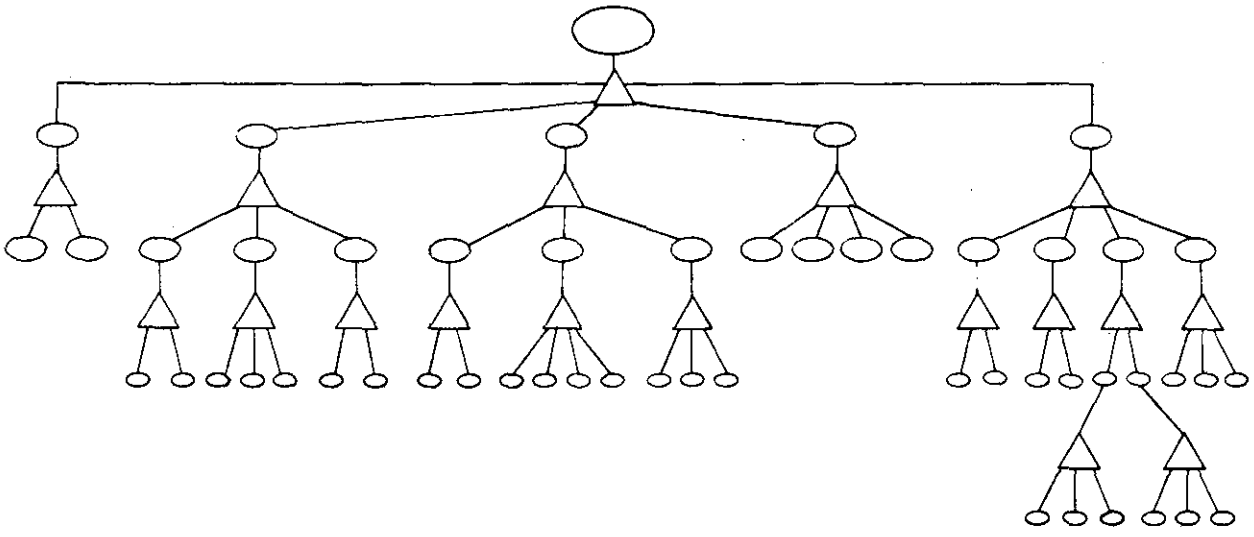

Figure 5b. Graph of structure of best tree designed by Optimized Logic Tree Design using transformed Bhattacharya-Distance and all 15 feature subsets. For results see Table 1, first row. Tree structure shown is for Trade-Off Constant 10.0. Feature sets and classes not shown for clarity. Note significant differences in third layer as compared to Figure 5a.

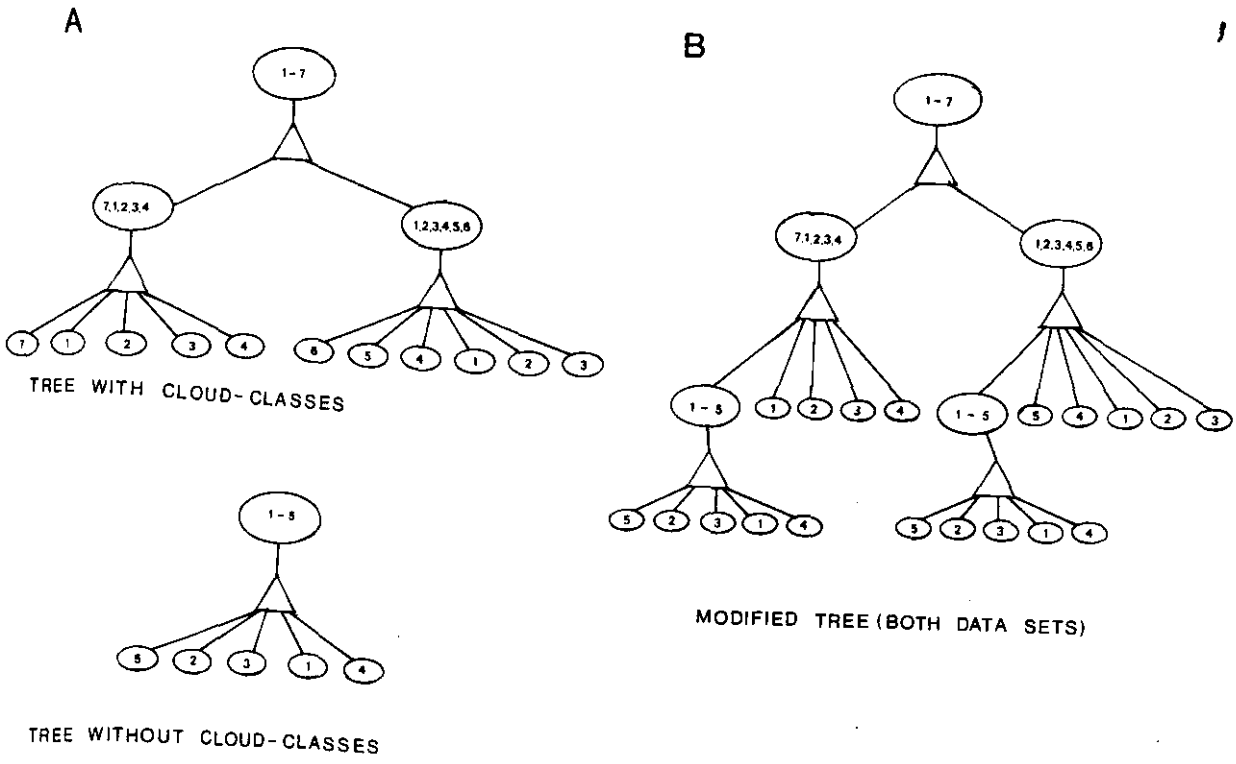

Figure 6. A) Tree structure for data set with clouds (top) and without clouds (bottom).

B) Modified tree. Tree for data set without clouds was interconnected with tree for data set with clouds. (class 7 - clouds, class 6 - cloud shadow).