This is to certify that the
thesis entitled

MULTIVARIATE LINEAR AND NONLINEAR DECISION TREES

presented by

GEETHA ARUNACHALAM

has been accepted towards fulfillment
of the requirements for the

Master of Science degree in Electrical and Computer Engineering

Major Professor's Signature

December 16, 2004

Date

MULTIVARIATE LINEAR AND NONLINEAR DECISION TREES

By

Geetha Arunachalam

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

2004

ABSTRACT

Multivariate Linear and Nonlinear Decision trees

By

Geetha Arunachalam

The ID3 classification algorithm developed by Quinlan is a very popular decision tree algorithm used extensively for obtaining a classifier model for patterns described by list of discrete valued attributes. A decision tree is made of decision nodes and leaf nodes. The decision node act as a test on input patterns and determines the outgoing branches from that node. The leaf node acts as a terminal node and contains the class name or label where the pattern is finally classified. The ID3 algorithm uses information gain based criterion to select the best attribute test at decision nodes. C4.5 is an extension of the ID3 algorithm for handling continuous valued attributes. Both ID3 and C4.5 decision trees use one attribute at decision node and produce orthogonal decision boundaries. They often tend to give complex decision trees for data sets with continuous valued attributes. There is ample scope for reducing the size of the decision trees without any compromise in classification accuracy by replacing the uni-variate test with multivariate tests and orthogonal decision boundaries with non-orthogonal decision boundaries. This thesis presents two-decision tree algorithms with multivariate test at decision nodes for data sets with continuous valued attributes. The first algorithm gives a decision tree with linear decision boundaries and the second algorithm gives a decision tree with nonlinear decision boundaries. The proposed algorithm also use information criterion for selecting the best linear or nonlinear test at each decision node. Results demonstrating the benefits of generating higher order decision trees and non-orthogonal decision boundaries for classification of data sets with continuous valued attributes are presented.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards my major advisor, Dr. Lalita Udpa, for her expert guidance, support and encouragement throughout my thesis work. Her valuable suggestions helped me in completing the thesis work smoothly without any delay. I am very much grateful to her for giving me the opportunity to pursue my graduate studies.

I would like to thank my thesis committee members, Dr. Satish Udpa and Dr. Pradeep Ramuhalli. Their expert opinion and comments were very useful in fine-tuning my report and in understanding the different perspectives of the topic.

I would also like to acknowledge the support provided by Electric Power research institute *(EPRI)* in funding my graduate studies. I am also grateful to Mr. James Benson *(EPRI)* for providing the required data, which was very useful in evaluating my thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

The need to find more efficient ways of performing a given task and the sheer challenge of programming machines to imitate human intelligence has motivated extensive research in machine learning since mid 1950's. The ease with which humans recognize a face, understand spoken words, read handwritten characters belies the complex process that underlies pattern recognition [1]. Pattern recognition, which can be defined as the act of taking in raw data and taking action based on the category of the pattern, has been crucial in our day-to-day activities. Sophisticated machine learning for accurate pattern recognition has become inevitable in finger print identification, DNA sequence identification, and automated speech recognition, classification of objects and in many more applications.

Early machine learning work produced self-improving programs such as adaptive systems that monitor their own performance and improve it by adjusting parameters. Then knowledge-based systems were introduced, which emphasized learning as the acquisition of structured knowledge in the form of concepts or rules [2]. An expert system for a complex task may require hundreds or even thousands of such rules. These approaches failed to keep pace with the increasing demand on expert systems. This bottleneck has motivated further investigation of different machine learning methods as a means of explicating knowledge [3]. There are many pattern classification algorithms for dealing with various levels of complexity and different applications. The following paragraph summarizes some of the popular approaches used for pattern classification.

Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification. It makes use of the assumption that the problem is posed in probabilistic terms and all of the relevant probability values are completely or partially known [4]. Designing an optimal classifier in the presence of prior probabilities and class conditional densities is straightforward. However in reality, in most pattern recognition applications, the training data that are particularly representative of the patterns are not sufficient to provide a complete knowledge of the probabilistic structure of the problem. Though the estimation of prior probabilities is straight forward, the estimation of conditional densities becomes complex especially when the dimensionality of feature vector is large. The severity of this problem can be reduced significantly, if we know the number of parameters to be estimated and also some general knowledge about the problem for parameterizing the conditional densities in advance. Maximum likelihood and Bayesian estimation are two common methods used for parameter estimation. Maximum likelihood views the parameters as quantities whose values are fixed but unknown, whereas Bayesian method views the parameter as random variables having some known prior distribution [4]. Another way of obtaining the decision function is using non-parametric techniques. In contrast to parametric estimation, non-parametric techniques can be used with arbitrary distributions, and without any assumption regarding the underlying densities. One example of non-parametric design procedure is the nearest neighbor rule, which bypasses the probability estimation, and directly goes to decision function estimation [1]. Some non-parametric techniques estimate the probability density function from sample patterns, which if satisfactory are substituted as true density in the classifier.

Another method of obtaining the classifier model is to assume a form of the decision boundary functions and use the training data to estimate the parameters of the decision boundary. Here the knowledge of underlying probability distributions is not required and in a limited sense the approach can be called as non-parametric. Linear discriminant functions are relatively simple to compute and are attractive candidates for initial trial classifiers. The task of finding a linear discriminant function is formulated as the problem of minimizing a criterion function such as training error. In complex classification models where linear discriminants are inadequate for good classification a nonlinear decision function can yield decision boundaries with minimum error. However in these complex problems, the need to determine too many free parameters makes the problem more difficult in the absence of prior knowledge, which guides the choice of nonlinearity. In such cases multilayer neural networks or multilayer-perceptrons are found to have better performance. Neural networks learn the nonlinearity i.e. it learns the parameters governing the nonlinear mapping of the problem and gives better classifier model than linear discriminant [1]. For more complicated problems with less prior knowledge and little training data, more sophisticated models based on stochastic methods are employed. In this method randomness plays a crucial role in search and learning, for finding the parameters. The general approach is to bias the search towards the region where we expect the solution to be, and allow randomness somehow to help find optimal parameter values. Boltzmann learning is one such method. It is based on concepts and techniques from physics, specifically statistical mechanics. Another candidate is the genetic algorithm, which is based on concepts from biology specifically on mathematical theory of evolution. The Boltzmann class of techniques is highly theoretical and has

considerable success in pattern recognition whereas the latter class is more heuristic yet affords flexibility and can be attractive when adequate computational resources are available [1].

So far all the pattern classification models discussed above involve feature vectors of real valued numbers and use some notion of metric to be minimized. In classification problems with patterns described by list of attributes, the classifier model has to move beyond the idea of continuous probability distributions and metric, and move towards the discrete problems that are addressed by rule based or syntactic pattern recognition methods. The simple and intuitive way to classify a pattern is via sequence of questions, which in turn can be displayed in a directed decision tree. In a basic decision tree the classification proceeds from top to bottom. The decision at each node concerns a particular property of the pattern and the branches correspond to possible values. The tree is grown inductively with decision nodes until a terminal or leaf node is reached. Leaf node has the class name of the patterns.

Quinlan's Itemized Dichotomizer 3 (ID3) algorithm and C4.5 are very popular decision tree algorithms, based on information theory concepts. The ID3 algorithm is developed from basic CLS-Concept Learning System concepts, which constructs a decision tree by minimizing the cost of classifying an object and recursively divides each of the partitioned sets. In the ID3 algorithm the cost driven approach is replaced by information driven evaluation [2]. ID3 algorithm uses Shannon entropy function to automatically determine the attribute with most significant amount of discriminating information. It uses a greedy search algorithm to obtain best separation and builds the tree with best attributes. C4.5 is an extension of ID3, which can handle continuous attributes and

training data with missing attribute value. Both ID3 and C4.5 decision tree algorithms use univariate test at each decision node and construct the trees with orthogonal decision boundaries resulting in a complex tree.

This thesis uses the entropy concept in a variation of the basic ID3 algorithm to obtain non-orthogonal linear decision boundaries and nonlinear decision boundaries for a two class problem. In chapter two the basic 1- D ID3 algorithm and the need for non-orthogonal boundaries for continuous valued attributes are explained. In the third chapter two of the previously developed multivariate decision tree algorithms JE and LAC1 with two attributes at each decision node are discussed. In chapter four the proposed linear attribute combination algorithm for continuous valued attributes is discussed. This algorithm assumes that each class is distributed in and around the vicinity of its mean. The algorithm systematically identifies attribute combinations with maximum interclass separation in the corresponding two dimensional feature space.

Chapter five presents an algorithm for obtaining a nonlinear decision boundary for nonlinearly separable classes. In this algorithm at each decision node a nonlinear test is used to partition the data and the test with minimum entropy is selected for building the decision tree. The results of all five algorithms on real field data are discussed and contrasted in chapter six. Chapter seven summarizes all five algorithm implemented in this thesis and presents some concluding remarks and possible directions for future work.

CHAPTER 2: BASIC RULES FOR DECISION TREE

2.1 Decision Tree

A decision tree is generally defined as a tree whose internal nodes are tests on input patterns and whose leaf nodes are categories or classes. The basic decision tree employs a top down greedy search through all possible branches. Top down Induction decision trees are characterized by their representation of acquired knowledge as decision nodes. They use simple knowledge formalism instead of complex expressive power of semantic networks and as a result these systems have less complex learning methodologies capable of solving difficult problem of practical significance [2].

The conventional incremental learning method analyzes each instance at a time for developing a classification model [5]. A basic decision tree uses non-incremental learning strategy, in which the decision tree is developed with a set of cases relevant to classification tasks. The set of values or attributes associated with training instances are used to express decision node in the tree. The decision trees are built starting with the root of the tree and proceeding down its leaves. The process is repeated for each sub tree rooted at the new node. Each decision node in the tree represents condition or expression with attributes in the given data set. Based on the outcome of test on the set is partitioned into branches, which forms leaf node or another decision node.

The training set used for developing the decision tree should not contain conflicting instances with same attribute values and yet belonging to different classes. In such cases the attribute will be inadequate for building decision trees by the induction task. With adequate attributes a decision tree that correctly classifies each instance in

training set can be obtained. Leaves of decision tree are class names, whereas other nodes represents attributes test with a branch for each possible outcome. Usually there will be many such correct decision trees, but the trick lies in selecting a tree, which is minimal and more likely to capture the inherent structure in the problem and correctly classifies training data as well as other unseen objects. An ideal decision tree should have good classification on training as well as testing data and yet be simple with minimum number of nodes. By selecting attributes, which have more information at each decision node, a simple and good classifier model can be obtained. The following example illustrates how the concept of selecting attributes with more information at decision nodes gives simpler tree with good classification.

Table 2.1: Training Data

Instances	Attribute X	Attribute Y	Class
1	0	0	0
2	1	0	1
3	0	1	1
4	1	1	0

In the above binary example we have two attributes X and Y, two classes 0, 1 and four instances. Two different decision trees can be built by choosing different attributes at the root node as shown in Figure 2.1. The decision tree shown in Figure 2.1 (a) is simple with one decision node. Attribute X is used at the root node. Depending on the values of attribute X the data set is partitioned into two subsets. The instances belonging to each of the subsets are from the same class so we can label the subsets as leaf or terminal node without any further test nodes. In contrast the decision tree in Figure 2.1 (b) has 3 decision nodes. This is because the Attribute Y at root node results in subsets with

instances, which do not belong to same class. In order to obtain the leaf nodes with instances of same class, we need two more test nodes for each subset. Further test on each subset with attribute X, results in the leaf nodes with correct classification. It is obvious from this example that by selecting attribute X with more information we can construct a decision tree, which is simpler.

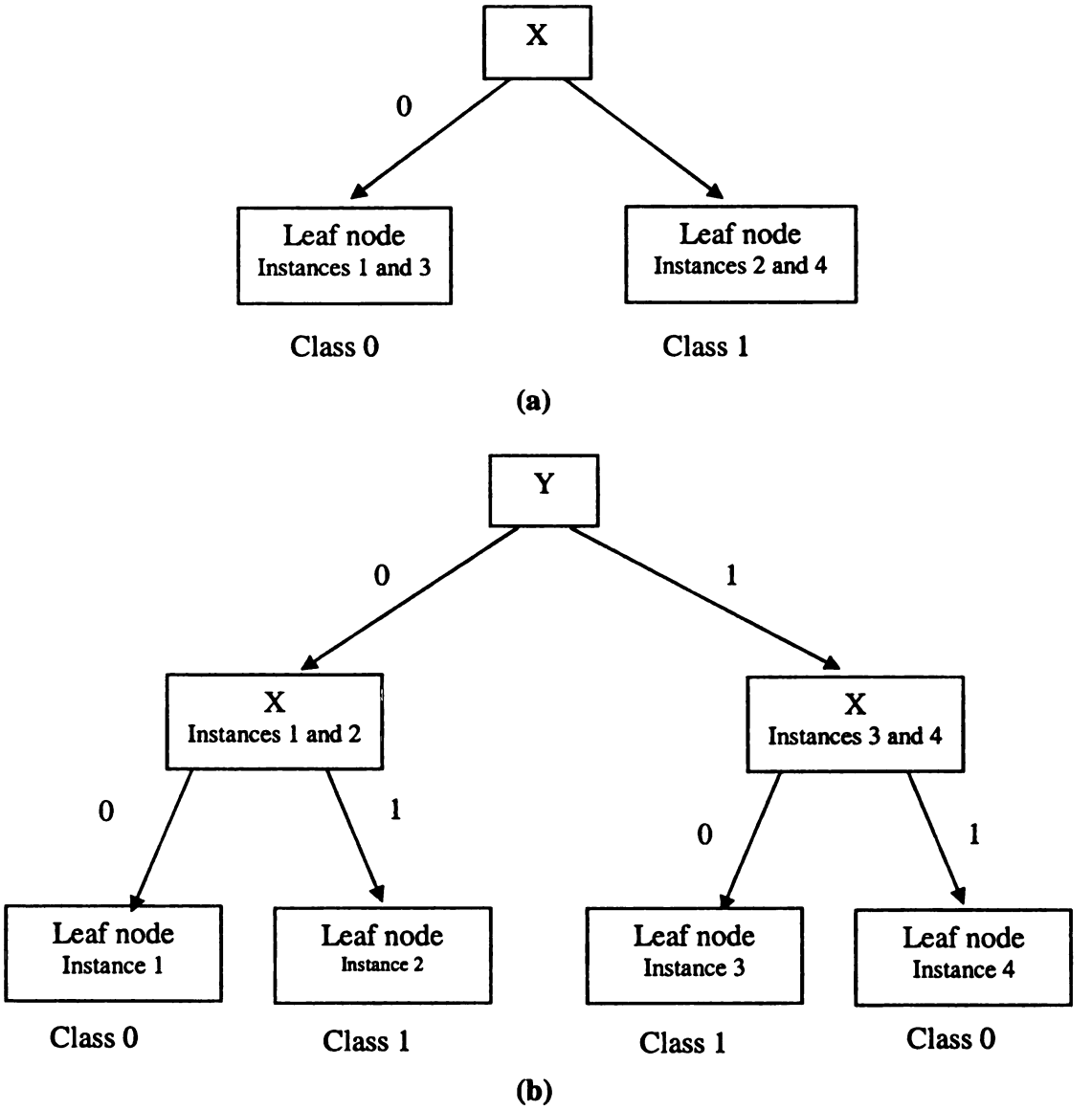

Figure 2.1: Different Decision Trees for Table 2.1

The crucial task in designing a decision tree is in selecting which attribute test or query is appropriate at each node. The attribute test at each node should aim at partitioning the data into subsets that are as pure as possible, in other words belonging to one particular class. Lesser the impurity at a node, better the classification will be.

There are many mathematical measures of impurity, Variance impurity, Gini impurity, Misclassification impurity and Entropy impurity [1]. Let $i(N)$ denote the impurity of node N and $P(\omega_i)$ is the fraction of patterns at node N belonging to class ω_i. For a two category classification model, the variance impurity is given by the following expression.

$$i(N) = P(\omega_1)\ P(\omega_2) \qquad 2.1$$

The expression goes to zero indicating zero impurity whenever the node represents only patterns of one class. Gini impurity is the generalization of variance impurity to two or more classes. Gini impurity is given by the expression as shown below.

$$i(N)=\sum_{i\neq j} P(\omega_i)\ P(\omega_j)=1-\sum_{j} P^2(\omega_j) \qquad 2.2$$

This gives the expected error rate at node N when the node is labeled as a leaf node with randomly selected class from the data distribution at node N. The Misclassification impurity measures the minimum probability of misclassifying a training pattern at node N. Misclassification impurity is expressed by the following equation.

$$i(N)= 1-\max_{j} P(\omega_j) \qquad 2.4$$

Finally the Entropy impurity, which is the most popular is based on Shannon's information theory concept and is given by the expression below.

$$i(N) = -\sum_j P(\omega_j) \log_2 P(\omega_j) \qquad 2.5$$

This expression will result in zero when all the patterns belong to one class and results in a positive value when patterns of different classes are present. Maximum value occurs when the different classes are equally likely at a node. ID3 algorithm employs entropy impurity for selecting the best attribute at each node. Shannon's measure of information contained in a message based on information theory concept is widely used in computer science, communications, information processing and in data compression and storage. The information in a message depends on a priori expectations. Smaller the prior probability of the message more the information contained, in other words the more probable the message, the less information it conveys [7].

2.2 Entropy Test

Shannon defines the information content of a message as a function of the probability of occurrence of the message [6]. In a set S of n symbols in a message with probabilities of $p_1, p_2, \ldots p_n$, the information contained in a message n_i of probability p_i is given by

$$I(p_i) = -\log_2 p_i \text{ bits} \qquad 2.6$$

Total information in the message with all n symbols is expressed as summation of all information contained in them as below.

$$I(p_1, p_2, \ldots p_n) = -\sum_{i=1}^{n} \log_2(p_i) \quad \text{bits} \tag{2.7}$$

The expected value of the information of the set S also known as entropy of the message is given by the expression

$$H(S) = H(p_1, p_2, \ldots p_n) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.8}$$

The expected information or entropy for a test X on set S with k outcomes such that S is partitioned into k subsets is the weighted sum over the subsets as shown below

$$H_X(S) = -\sum_{i=1}^{k} p_i I(p_i) \tag{2.9}$$

The information gain by applying test X on set S is

$$gain(X) = H(S) - H_X(S) \tag{2.10}$$

Lower the entropy for a test X on set S, higher the information gain. Information gain is precisely the measure used by ID3 to select the best attribute at each step in growing the tree.

2.3 ID3 Algorithm

The entropy concept discussed above is used for attribute selection at each node in the ID3 algorithm where a decision tree is built from top to bottom with the best possible separation at each test node. ID3 strives to obtain a decision tree from an arbitrary collection of objects in a set. If the set C is empty or contains instances from one class, a decision tree with leaf node assigned with the class is obtained. On the other hand if the

set C has more than one class, let T be any test on the set C with $O_1, O_2 .... O_n$ as the outcomes. The test T on the set C produces the partition as $\{C_1, C_2 .... C_n\}$ as the outcome. The graphical representation is shown in Figure 2.2.

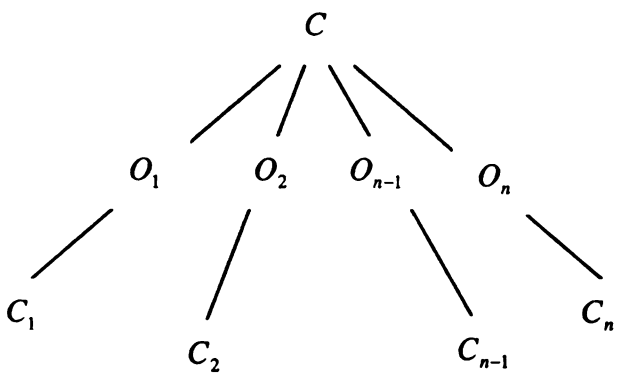

Figure 2.2: A tree structuring of objects in C

The decision tree for the whole set C can be obtained if each partition is replaced by a decision tree. As long as two or more subsets from test node are non-empty and each subset is smaller than the set at the decision node, this divide- and- conquer method will lead to a subset containing instances belonging to one class. Such a subset is then assigned as leaf or terminal node. The test at a decision node is important for a simple decision tree. The ID3 algorithm based on information theory depends on two assumptions as explained below [1].

a) Any correct decision tree for set C will classify instances in the same proportion as their representation in set C. An arbitrary instance belonging to class P is given by $\frac{p}{(p+n)}$ and belonging to class N is $\frac{n}{(p+n)}$.

b) A decision tree returning a class while classifying an instance can be regarded as a source of a message 'P' or 'N'. The expected information to generate the message is given by

$$H(p,n) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right) \qquad 2.11$$

Attribute test A with values $\{a_1, a_2 ... a_v\}$ at root node will partition the training set C into $C_1, C_2 .... C_v$ where C_i contains those objects in C that have value a_i for attribute A. Let C_i contain p_i instances of class P and n_i instances of class N. The expected value of information of subset C_i is given by $H(p_i, n_i)$ and the expected value of information of the tree with attribute A as root is given by weighted average

$$H(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} H(p_i, n_i) \qquad 2.12$$

where the weight of the i^{th} branch is in proportion to the number of instances in C that belong to C_i. The information gain for this test on attribute A is given by

$$gain(A) = H(p,n) - H(A) \qquad 2.13$$

ID3 examines all the possible partitions with all candidate attributes and selects the attribute that has the maximum gain. The attribute with maximum gain is used as test node and the procedure is repeated recursively to form decision trees for each of the subsets formed at the test node. The procedure is illustrated with the example training data shown in Table 2.2. The training set has 14 instances and indicates whether golf can be played or not in a particular weather condition. There are four attributes namely Outlook, Temperature, Humidity and Windy.

The possible values that different attributes can take are as follows.

1) Outlook = {sunny, overcast, rainy}

2) Temperature ={hot, mild, cool}

3) Humidity ={high, normal}

4) Windy = {true, false}

Table 2.2 Golf Training Data Set 1

Instances	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	Sunny	Hot	High	False	N
2	Sunny	Hot	High	True	N
3	Overcast	Hot	High	False	P
4	Rain	Mild	High	False	P
5	Rain	Cool	Normal	False	P
6	Rain	Cool	Normal	True	N
7	Overcast	Cool	Normal	True	P
8	Sunny	Mild	High	False	N
9	Sunny	Cool	Normal	False	P
10	Rain	Mild	Normal	False	P
11	Sunny	Mild	Normal	True	P
12	Overcast	Mild	High	True	P
13	Overcast	Hot	Normal	False	P
14	Rain	Mild	High	True	N

For simplicity the construction of a decision tree with ID3 algorithm is illustrated with discrete valued attributes. The different classes we have in this example are 'Play' P and 'Don't Play' N. Out of 14 instances 9 belong to class P and 5 belong to N.

Therefore the average information of this whole data set is given by

$$H(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940 \quad \text{bits}$$

The attribute 'Outlook' can take 3 values namely {sunny, overcast, and rainy}. By choosing 'Outlook' as the test attribute we will have three subsets with five instances for attribute value 'sunny', five instances for 'rainy' and four instances for 'overcast'. The expected value of information for three subsets is given below

Let p_i and n_i be the number of instances in a subset indicating the two classes and $p = 9$ and $n = 5$ be the total number of instances in each of the two classes.

'sunny': $p_1 = 2; \quad n_1 = 3; \quad H(p_1, n_1) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$ bits

'overcast': $p_2 = 4; \quad n_2 = 0; \quad H(p_2, n_2) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$ bits

'rainy': $p_3 = 3; \quad n_3 = 2; \quad H(p_3, n_3) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$ bits

The expected value of information of the tree with attribute 'Outlook' as root is given by weighted average as follows.

$$H(outlook) = \sum_{i=1}^{3} \frac{p_i + n_i}{p + n} H(p_i, n_i) = 0.694 \text{ bits}$$

The information gain for attribute 'outlook' is

$Gain(outlook) = H(S) - H(outlook)$ = 0.940-0.694 = 0.246 bits

The information contained in the whole set $H(S)$ is constant for all attribute tests, therefore the information gain for each attribute test is equivalent to minimizing the expected value of information or entropy at the test node with the attribute test.

In a similar manner if we calculate the average information and gain of other three attributes we will get

$H(temperature) = 0.911$ bits; $Gain(temperature) = 0.029$ bits

$H(humidity) = 0.79$ bits; $Gain(humidity) = 0.151$ bits

$H(windy) = 0.892$ bits; $Gain(windy) = 0.048$ bits

The ID3 chooses the attribute with highest information gain for test node. In this example, attribute 'outlook' has maximum information gain and it is used as the test attribute at root node.

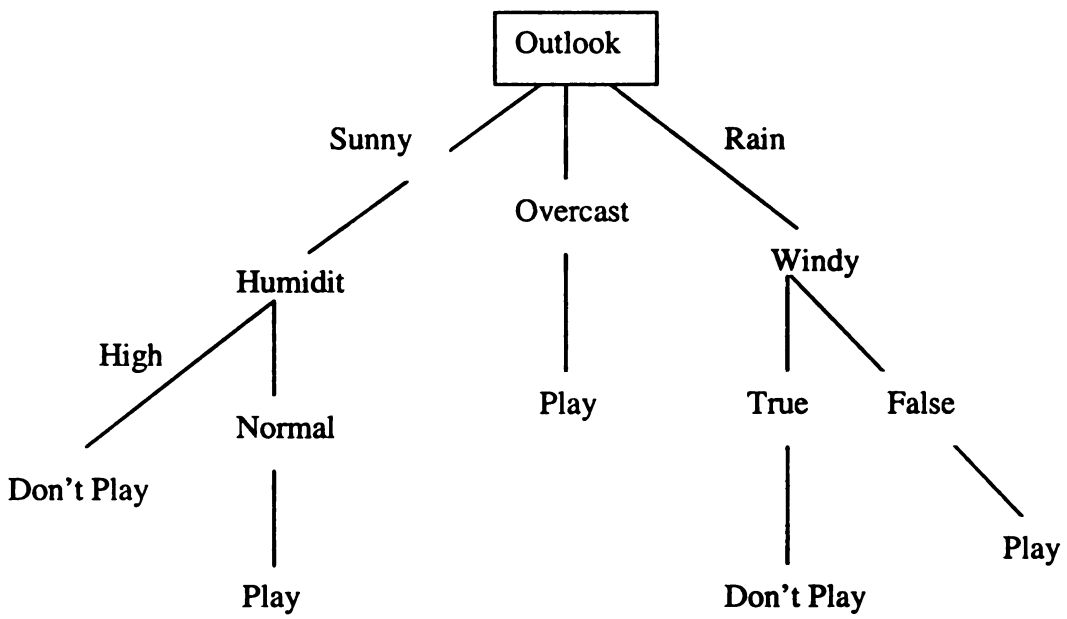

Figure 2.3: The Decision tree for Table 1

After the test with attribute 'outlook' the training set S is divided into subsets based on the attribute values. Each subset is either assigned as leaf node if all of its instances belong to one class or a decision tree is developed recursively using the same procedure. The tree is continued until all the instances are correctly assigned to a leaf node. Developing a complete decision tree for the given (golf) training data using the ID3 algorithm will result in a decision tree as shown in Figure 2.3.

Given a training data set S with attributes set A and classes set C to ID3 algorithm it executes the following steps. All the instances in training data set S is assigned a class.

If all entries in S are from the same class

{ Return a leaf node with that class name as label }

else if S is empty

{ Return a single node with value failure }

else

{Let A_j be the attribute from set A that has the highest gain, with best classification of training set S.

Then the decision attribute at the root node is A_j. For each value v_i of A_j, add a new branch below with root corresponding to the test $A_j = v_i$.

Let S_{vi} be the subset of S that has value v_i for A_j.

If S_{vi} is empty

{Then add a leaf node below this branch with label = most frequently occurring class in set S}

else

{Add a new sub tree below this branch and repeat the procedure for other subsets}

end

end

Figure 2.4: ID3 Algorithm

2.4 Improvements on Basic ID3 Algorithm

2.4.1 Gain ratio

The information gain approach has a natural bias towards attributes having many values over those with few values [7]. For example if we add an attribute Date which will be different for each instance, then we would end up selecting Date as the attribute with maximum gain as it classifies all instances correctly. The resulting tree will have a single decision node with number of outcomes or in other words leaf nodes equal to number of instances. Such a decision tree would be useless when a test data is applied although it correctly classifies the training data. To handle this problem, Quinlan [7] suggested the concept of gain ratio, which normalizes the information of the attribute using split information. Split information is defined as

$$SplitInformation\ (S, A) = -\sum_{i=1}^{c} \left|\frac{S_i}{S}\right| \log_2 \left|\frac{S_i}{S}\right| \qquad 2.14$$

where A is the attribute with c different values and S is the data set with c subsets $\{S_1, S_2, ...S_c\}$ due to partitioning by attribute A. Gain ratio is then defined as

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)} \qquad 2.15$$

The attribute with maximum gain ratio is selected for a decision node. The split information of the attribute should be small, in other words the number of values that an attribute can take should not be very large.

2.4.2 Continuous valued attributes

The attribute humidity can also take continuous values. The basic ID3 introduced initially was designed to predict classes, which are discrete valued, and to handle attributes with discrete values. The second restriction of handling the discrete valued

attribute can be relaxed and continuous-valued decision attributes can be incorporated in the decision tree [8]. For example for an attribute A with continuous value the test node can be assigned as $A < c$ where the threshold c lies within the minimum and maximum of the values taken by attribute A. For obtaining the threshold value for an attribute under examination, all possible values it takes are sorted. If the attribute take n different values, then it is sufficient if we examine $n-1$ thresholds by taking the midpoint of two consecutive sorted values. For example, in the training set in Table 2.2 if humidity is a continuous attribute with values {85,90,78,96,80,70,65,95,70,80,70,90,75,80}, the information gain for $n-1$ thresholds are calculated and the threshold value that partitions the data with minimum entropy is selected. The best partition with minimum entropy occurs at 'Humidity'=75 and the test for 'Humidity' results in binary partition with one subset having instances satisfying 'Humidity' ≤ 75 and other subset containing instances satisfying 'Humidity' >75. C4.5 is an extension of ID3, which operates on continuous valued attributes by following the above procedure.

2.4.3 Noise

The golf training set explained above deals with data, which is noise free. In most real world data the description of instances may include attributes based on measurements or subjective judgments that results in errors in the values of attributes. For example if the attribute of outlook of instance 1 is incorrectly recorded as overcast, then the instances 1 and 3 will have identical descriptions but belong to different classes. This kind of non-systematic errors either in the values of attributes or class information is

usually referred to as noise and the attributes in the training set becomes inadequate for building a decision tree. Two essential modifications for any tree-building algorithm capable of operating in noise-affected training set are as follows [7].

a) The algorithm must work with inadequate attributes, because noise can cause even the most comprehensive set of attributes appear inadequate.

b) The algorithm must be able to decide that testing further attributes will not improve the predictive accuracy of the decision tree. It should refrain from increasing the complexity of the decision tree to accommodate a single noise-generated special case.

The first requirement is necessary to handle situations when further testing is ruled out for a subset with instances of different classes. This case arises when the attributes are inadequate or unable to discriminate among the instances, or when each attribute has been judged to be irrelevant to the class of instances in the subset.

In above scenario, it is necessary to produce a terminal node with class information, though the objects in the subset are not of the same class. One approach is to select the majority class name and assigning it to the leaf node. This minimizes the sum of the errors over all instances in a subset.

The second requirement of deciding when an attribute is really relevant to classification can be obtained by chi-square test for stochastic independence. Using this test based on statistical properties we can determine the confidence of how much the attribute is independent or dependent on the class of objects in a subset [7].

2.4.4 Unknown Attributes

Sometimes the data can have missing attribute values. This happens when the values are not relevant to a particular case, and not recorded when the data was collected or due to some human error in compiling the data. Either significant amount of the data with unknown attributes has to be discarded with some test cases misclassified or the algorithm should be modified to handle unknown attribute values. To incorporate the necessary modification the following facts have to be considered [7].

a) Two tests using different numbers of unknown values should be weighted appropriately with respect to their relative desirability.

b) After selecting a test, the training instances with unknown values of relevant attribute cannot be associated with a particular outcome of the test, and so cannot be assigned to a particular subset. We need to find a way to partition the data in such cases.

c) When the decision tree is used to classify an unseen case, how should the system proceed, if the case has an unknown value for the attribute tested in the current decision node?

To answer the first query we can modify the apparent gain obtained by considering all cases to the value obtained by looking at cases with known values of the relevant attribute, multiplied by the fraction of known cases in training set. Let S be the training set and T a test based on some attribute. Suppose the value of attribute A is known in fraction F of the cases in S. Let *info (S)* be the information of the set S and $info_S(T)$ be the information obtained by applying the test T on S. The information

gain can be modified to consider only case with known attribute values as shown below

$$\begin{aligned} gain\,(T) = \;& P(A\ known) \times (info(S) - info_S(T) + P(A\ not\ known) \times 0 \\ =\;& F \times (info\,(S) - info_S(T)) \end{aligned} \qquad 2.16$$

To deal with the second problem stated above, we can associate with each case in each subset a weight representing the probability that the case belongs to each subset. Let $O_1, O_2, \ldots O_n$ be the outcomes of the test T on training set S. Weight w is associated with each case in each subset representing the probability that the instance belongs to each subset S_i. The weight w is 1 when a case with known outcome O_i is assigned to subset S_i and 0 in all other subsets. For unknown outcomes the weight is taken as the probability of the outcome O_i at that point. In general a case from S with weight w whose outcome is not known is assigned to each subset T_i with weight

$$w \times P(O_i) \qquad 2.17$$

For addressing the third issue of classifying unseen instances, which arises when the decision node encounters test attribute with unknown value, the system is modified to explore all possible outcomes and combines the resulting classification arithmetically. Once the total class distribution for an unseen instance has been established, the class with highest probability is assigned as the predicted class.

2.4.5 Pruning of decision trees

The decision tree generated by the ID3 algorithm continues to subdivide the set until all training instances are correctly classified and no additional test offers any

improvement. This may lead to over fitting of the data, which may not perform, well on test data. To overcome this drawback pruning is done on decision trees to remove the parts of the tree that do not contribute to classification accuracy on unseen instances, resulting in a tree that is less complex and hence more comprehensible. When a decision rule identifies greater error rate in a sub-tree than in a single leaf, that particular decision node can be pruned by replacing the whole sub-tree by a leaf node. We have two families of techniques; the first uses a training data set for pruning and the second uses cost complexity and reduced error as the basis for pruning. Consider the test data shown in Table 2.3. The last three instances in the test data will be misclassified as 'Don't Play' if we use the decision tree in Figure 2.3. Though we do not have any misclassification in training data we will have 3 misclassifications in the test data.

Table 2.3: Test data

Instances	**Attributes**				**Class**
	Outlook	**Temperature**	**Humidity**	**Windy**	
1	Sunny	Hot	High	False	N
2	Rain	Hot	High	True	P
3	Rain	Hot	High	True	P
4	Rain	Mild	High	True	p

If the tree is pruned by replacing the sub-tree for attribute 'Windy' by a leaf node 'Play', then all instances in test data will be correctly classified but with 2 misclassification in training data. Although we have 2 misclassifications on the training data on the whole

with training and test data together there will be only 2 misclassifications, instead of 3 misclassifications. The pruned decision tree is shown in Figure 2.5(b).

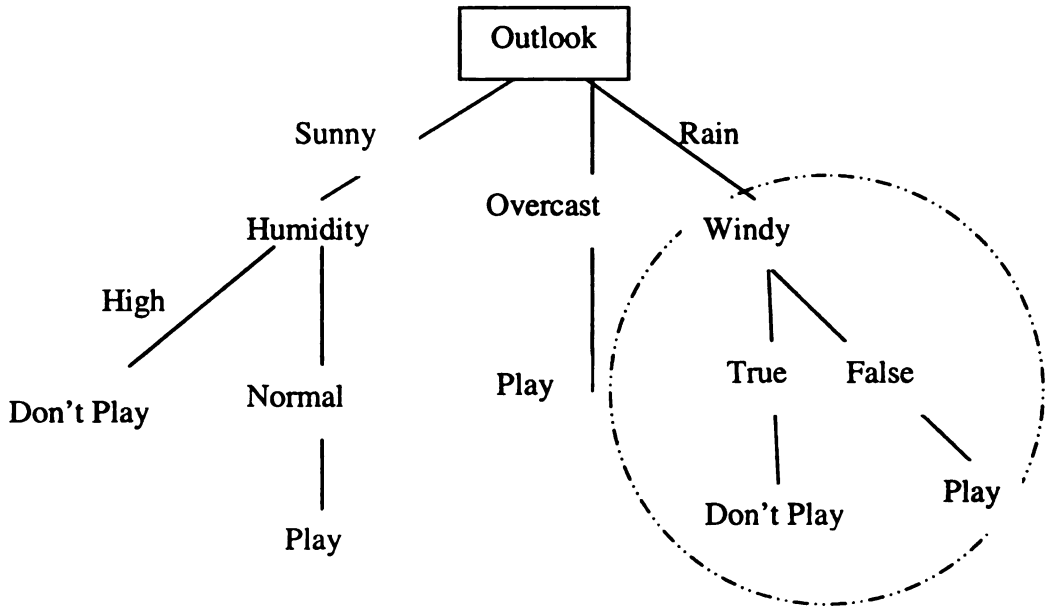

Figure 2.5(a): Original Decision tree

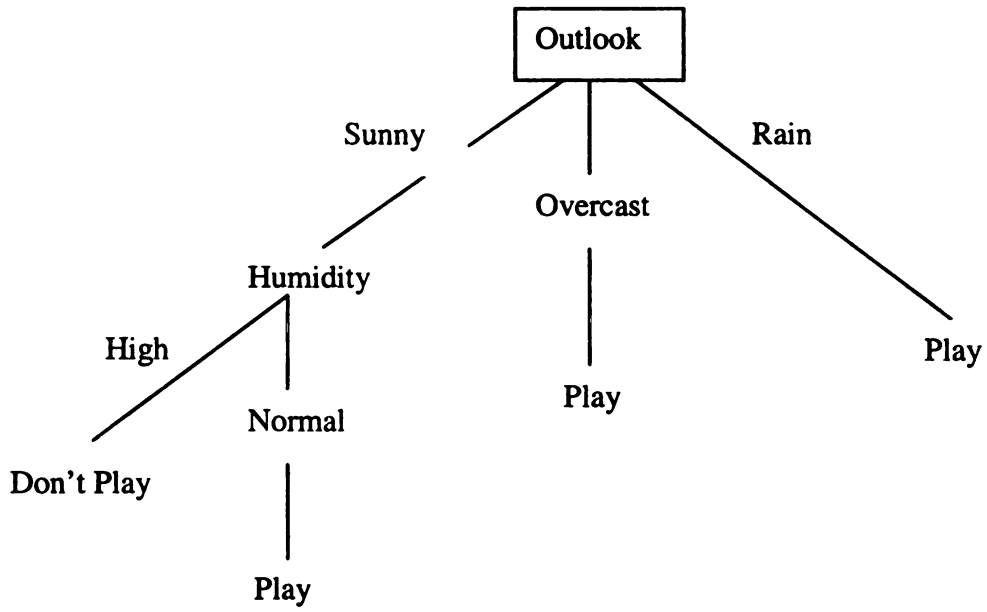

Figure 2.5(b): Pruned Decision tree

2.4.6 Multivariate test at decision nodes

The ID3 algorithm considers only one attribute test at any decision node. If more than one attribute are considered at each decision node we can obtain a simpler decision tree and possibly better classification performance. Another drawback in ID3 is that the decision boundaries are all orthogonal to feature axes. By applying a test using a linear combination of two or more attributes at decision nodes we can obtain a very simple decision tree with non-orthogonal tests [9]. In a similar manner we can replace the linear combination test with a nonlinear test, which further simplifies the decision tree by generating a more complex decision boundary. The next chapter discusses some of the previous approaches attempted to obtain a simple decision tree by considering more than one attribute at each node.

CHAPTER 3: MULTIVARIATE DECISION TREE

3.1 Need for Multivariate Test

The ID3 algorithm employs only univariate test at any decision node, which could potentially result in a complex, over-fitted decision boundaries. Secondly the correlation between the attributes is not considered in building the decision tree. By employing a test at decision node, that exploits the correlation between two attributes we can get a more powerful classifier model than the ID3 algorithm. The algorithms discussed in this chapter involve continuous valued attributes.

This chapter describes two decision tree algorithms based on the combination of two continuous valued attributes at each decision node. The first algorithm called Joint Entropy *(JE)* algorithm gives orthogonal decision boundaries, but in contrast to the ID3 algorithm which partitions the data into two subsets for continuous valued attributes, the JE algorithm divides the data into four subsets [10]. The second algorithm called Linear Attribute Combination (LAC) algorithm generates non-orthogonal decision boundaries. Only the test employed at the decision node for partitioning the data varies while the criterion for selecting the attributes is the same as in ID3. Once the subsets are obtained by applying the test, the entropy is calculated and the test with minimum entropy is chosen at a particular node.

3.2 Joint Entropy Algorithm JE

Assume that A and B are two attributes with values $\{a_1, a_2, \ldots a_n\}$ and $\{b_1, b_2, \ldots b_m\}$ respectively at a decision node. The total number of possible combinations that a test can have is $\{n \times m\}$. Each combination will divide the group of instances at the

test node into four subsets [10]. For example if attribute A takes the value a_r and attribute B takes the value b_s, we can partition the training data into four subsets. This is obtained by combination of instances having attribute $A<a_r$ and $A \geq a_r$ with instances having attribute $B<b_s$ and $B \geq b_s$.

The expected value of information in the tree with attributes A and B having values a_r and b_s at the decision node is given by the joint entropy of the pair AB expressed as [10].

$$H(A,B) = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{p_{ij} + n_{ij}}{p+n} H(p_{ij}, n_{ij}) \qquad 3.1$$

The following example illustrates the algorithm. Consider the data set in Table 3.1. The training set has two continuous valued attributes X and Y and two discrete classes 0 and 1. Here instead of sorting the values in an attribute and examining the midpoints as thresholds, the algorithm examines the actual values in the attributes for threshold selection. The number of values that attribute X can take is 7 and the number of values attribute Y can take is 8. The entire algorithm will examine 7×8 combinations at the test node for selecting the best attribute. Let us examine the combination of attribute X with threshold value 2 and attribute Y with threshold value 3. Then we will have two groups for each attributes with instances $X<2$, $X \geq 2$ and $Y<3$, $Y \geq 3$. The expected value of information or the joint entropy for each combination is given as

'$X<2$' *and* '$Y<3$' $H(p_{11}, n_{11}) = -\frac{6}{6}\log_2\frac{6}{6} - \frac{0}{6}\log_2\frac{0}{6} = 0$ bits

'$X<2$' *and* '$Y \geq 3$' $H(p_{12}, n_{12}) = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$ bits

'$X \geq 2$' *and* '$Y<3$' $H(p_{21}, n2_{21}) = -\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1} = 0$ bits

'$X \geq 2$' *and* '$Y \geq 3$' $H(p_{22}, n_{22}) = -\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3} = 0$ bits

The Joint entropy of the tree with attributes X and Y with values 2 and 3 respectively at the decision node is given by weighted average

$$H(A,B) = \sum_{i=1}^{2}\sum_{j=1}^{2}\frac{p_{ij}+n_{ij}}{p+n}H(p_{ij},n_{ij}) = 0 \text{ bits.}$$

Table 3.1: Training data with non-orthogonal boundary

Instances	Attribute *X*	Attribute *Y*	Class
1	1	1	0
2	1.5	0.5	0
3	0.5	2.2	0
4	0.5	0.5	0
5	1.5	1.5	0
6	1	2	0
7	1	4	1
8	2	3	1
9	2.5	3	1
10	3	2	1
11	3.5	3.5	1
12	1	3	1

Similar to the above procedure we can find the joint entropy for different threshold combinations. Finally the attribute combination with minimum entropy is chosen as the test at the decision node. For the given training set the attribute combination explained above has the minimum entropy and it is used at the root node for partitioning the data. The Figure 3.1 shows the decision tree generated by the JE algorithm for the training set in Table 3.1.

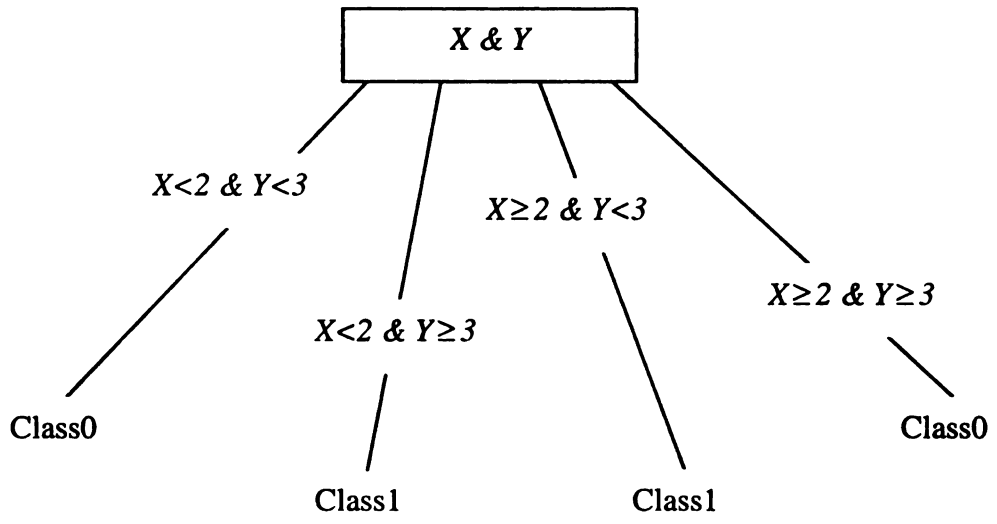

Figure 3.1: Decision Tree using JE Algorithm

The two dimensional data distribution is plotted in Figure 3.2. It can be seen that the two classes can be separated by a line having equation $y = -x+3.5$. The decision boundary obtained using the *JE* algorithm, also shown in Figure 3.2, is orthogonal in nature. Although the decision boundary is orthogonal and not much different from that obtained using the ID3 algorithm, JE classifies the data using a single decision node where as ID3 requires two separate nodes for classifying the training data. Figure 3.3 shows the decision tree obtained using the ID3 algorithm on the data set in Table 3.1. The ID3 algorithm with one attribute at a decision node gives complicated and over fitted decision boundaries [10]. Consequently when the test data is applied to the decision tree, the JE algorithm results in much better classification performance in terms of both classification accuracy and simplicity of the decision tree compared to corresponding results of conventional ID3 algorithm. These results on test data are summarized in chapter seven.

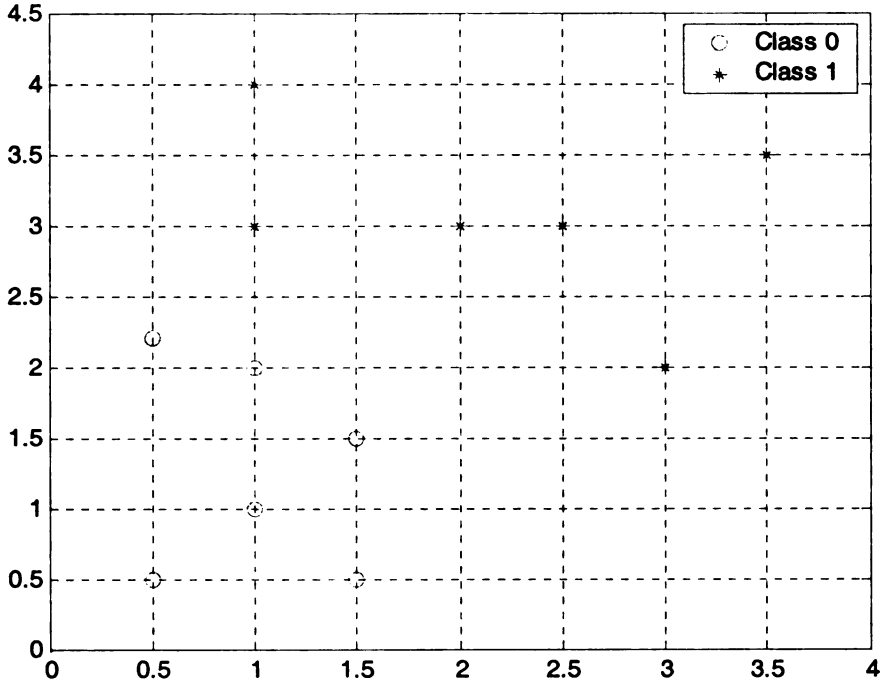

Figure 3.2: Data Distribution of Table 3.1 with JE decision Boundary

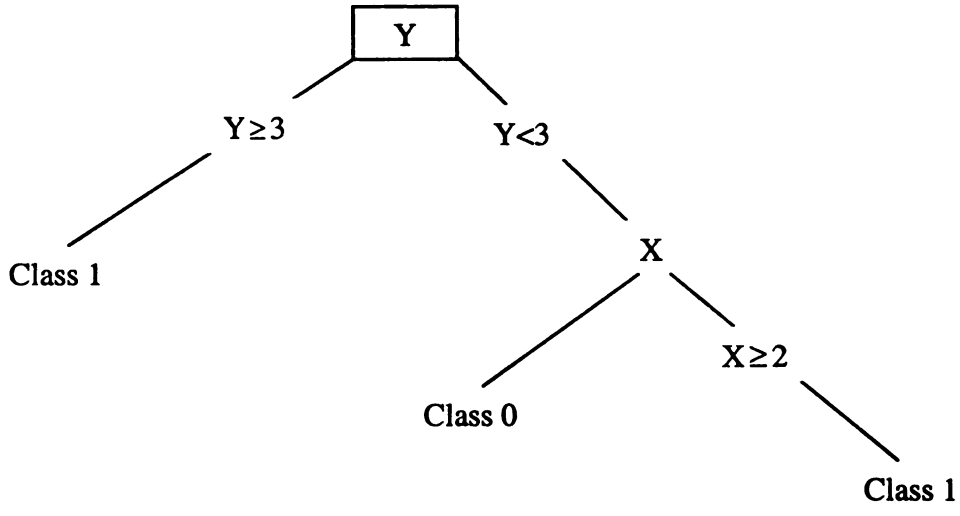

Figure 3.3: Decision Tree obtained by using ID3 algorithm

Total attribute combination set AB is determined by all possible pair wise combination of attributes. All the instances in training data set S is assigned a class.

Given a training data set S, attributes set A, classes C and total attributes combination set AB as input to the Joint Entropy algorithm (JE) it executes the following procedure.

If all entries in S are from the same class

{ Return a leaf node with that class name as label }

else if S is empty

{ Return a single node with value failure }

else

{Let a_j, b_k be the attribute pair that has the highest information gain or minimum joint entropy, with best classification of training set S.

Then the decision attribute at the root node is a_j, b_k. For each subsets with instances that satisfies the test condition $a_j < v$ & $b_k < x$; $a_j < v$ & $b_k \geq x$; $A_j \geq v$ & $b_k < x$; $a_j \geq v$ & $b_k \geq x$; where v is threshold value for attribute A and x is the threshold value for B. Add a new branch below the root each subsets.

Let S_{jk} be the subset of S that $a_j < v$ & $b_k < x$

If S_{vi} is empty

{Then add a leaf node below this branch with label = most frequently occurring class in set S}

else

{Add a new sub tree below this branch and repeat the procedure for other subsets}

end

end

Figure 3.4: JE Algorithm

3.3 Linear Attribute Combination algorithm 1-LAC1

The Linear attribute combination algorithm is capable of generating non-orthogonal decision boundaries and produces a simpler decision tree relative to that of JE algorithm, which is limited to orthogonal decision boundaries [10]. Given a pair of attributes *XY*, representing the axes of a two dimensional space the LAC1 algorithm first determines the bounding rectangles of the two classes by finding minimum x_{min}, y_{min} and maximum x_{max}, y_{max} of each class in the feature space. The eight vertices of the bounding rectangles are then projected on to the *X* and *Y*-axes to find the intercept points of each class. Assuming the decision boundary to be straight lines in the *XY* feature space passing through each combination of intercept points projected on *X* and *Y*-axes, the associated entropy for each line that partitions the data in two subsets as $y - mx \leq c$ and $y - mx > c$ are computed. The line/decision boundary with minimum entropy is selected as the test for that particular attribute combination. Repeating the above procedure for each pair wise combination of attributes, the pair with minimum entropy is selected as the test at the decision node. The following example illustrates the above procedure for the same data set used to illustrate the JE algorithm in Table 3.1.

The bounding rectangle of two classes is obtained by finding the maximum and minimum values of each attribute *X* and *Y* for the two classes. Figure 3.5 shows the data distribution along with the bounding rectangles. The eight vertices of the bounding rectangles are projected on the *X*, *Y*-axes and the intercepts points are obtained. Table 3.2 gives the vertices of bounding rectangles and their respective projection on *X*, *Y*-axes. Decision boundaries are obtained by drawing lines through every pair of intercepts and the entropy for each pair is computed. The decision boundary with minimum entropy is assigned as

the root node. The process is repeated at each node until all the instances in the data are correctly classified. The decision boundary with $y+1.14x=4$ was found to have minimum entropy with the best classification performance for the data set under discussion. The instances that have $y+1.14x \leq 4$ are classified as class '0' and the instances that have $y+1.14x > 4$ are classified as class '1'.

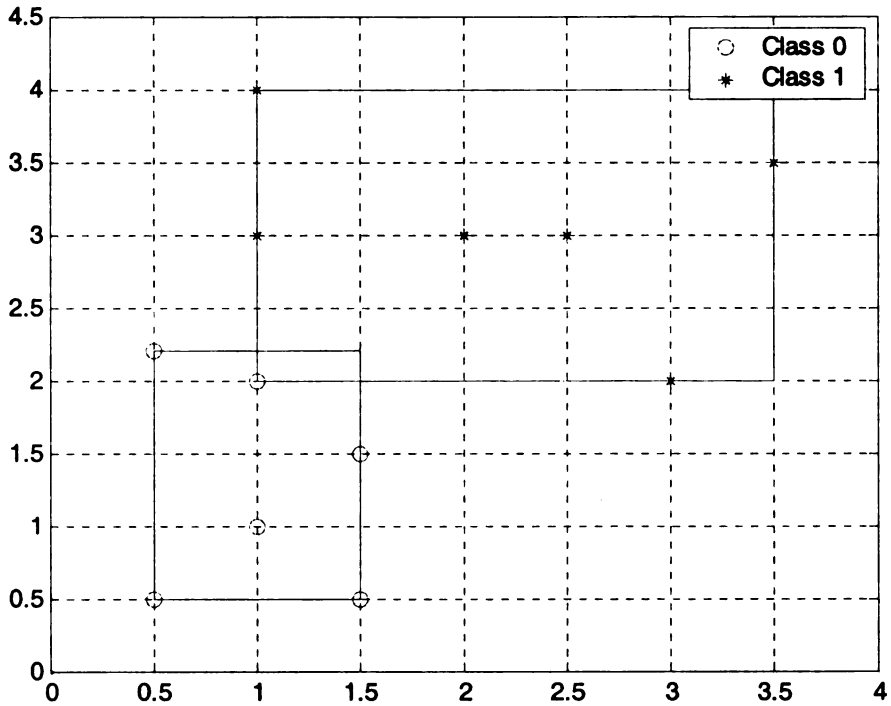

Figure 3.5: Data distribution of Table 3.1

Table 3.2: Intercept points and Vertices of bounding rectangles

Class	Vertices of bounding rectangle	Projections on X axis	Projections on Y axis
'0'	(.5,0.5);(1.5,.5);(1.5,2.2);(.5,2.2)	(.5,0);(1.5,0)	(0,0.5);(0,2.2)
'1'	(1,2); (3.5,2) ;(3.5,4); (1,4)	(1,0); (3.5,0)	(0,2); (0,4)

Figure 3.6 shows the decision boundary obtained using the LAC1 algorithm for the above data set. The decision boundary is non orthogonal in nature. Figure 3.7 shows the implemented decision tree for the data set. Unlike the JE algorithm we have only two subsets for each node. Each decision node is binary and a single decision node correctly classifies the data set into two leaf nodes for the data set under discussion. The ID3 algorithm generates a tree with two decision nodes and the JE algorithm decision tree needs 4 leaf nodes and only one decision node. Compared to both ID3 and JE algorithms non-orthogonal decision boundary obtained using the linear attribute combination algorithm LAC1 provides superior classification performance [10] as seen in the results chapter.

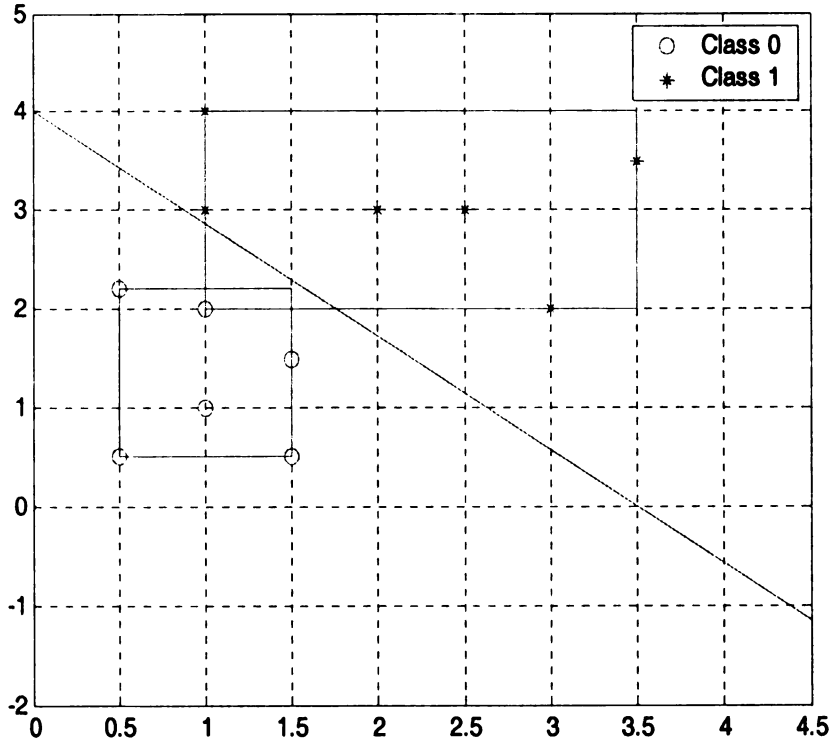

Figure 3.6: Decision boundary obtained using the LAC1 algorithm

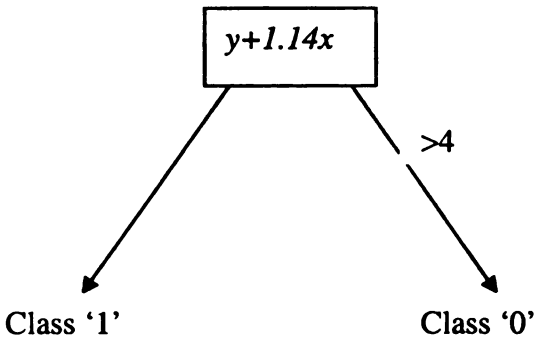

Figure 3.7: Decision Tree using LAC1

Total attribute combination set AB is determined by all possible pair wise combination of attributes. All the instances in training data set S is assigned a class.

Given a training data set S, classes set C and total attributes combination set A B and intercept points set P of the set as input to the Linear Combination algorithm (LC) it executes the following procedure.

If all entries in S are from the same class

{ Return a leaf node with that class name as label }

else if S is empty

{ Return a single node with value failure }

else

{Let the line L_q passing through points P_j, P_k be the line that has the minimum entropy, with best classification of training set S.

Let the points P_i & P_j be from the two dimensional space formed by using attributes

A_j & B_k .Hence let (A_j, B_k) be the attribute pair from AB whose distribution space contains the line L_q .

Then the decision attribute pair at the root node is (A_j, B_k) and the decision test at the root node is equation of line L_q passing through points P_i & P_j .For the subsets that have instances with and y>c add branch below the Root.

Let S_{jk} be the subset of S that satisfies<= y-mx

If S_{ji} is empty

{Then add a leaf node below this branch with label = most frequently occurring class in set S}

else

{Add a new sub tree below this branch and repeat the procedure for other subsets}

end

end

Figure 3.8: LAC1 algorithm

CHAPTER 4: MULTIVARIATE DECISION TREE USING LINEAR ATTRIBUTE COMBINATTION (LAC2) ALGORITHM

4.1 Linear Attribute Combination Algorithm 2 - LAC2

In the JE algorithm described in the last chapter, pair wise combinations of attributes are evaluated and the combination with minimum entropy is selected for the decision node. This makes the algorithm computationally intensive particularly when the continuous values that an attribute can take are large. Apart from the problem of determining the right threshold, it also has the disadvantage of producing complex decision trees associated with orthogonal decision boundaries. The LAC1 algorithm with non-orthogonal decision boundaries discussed in the previous chapter also has some drawbacks. In LAC1, the training data is correctly classified if and only if either one of the class is clustered about the origin. This is due to the fact that the line of separation is obtained by joining intercept points of the bounding rectangle vertices with the X and Y-axes. If the two classes are clustered far away from the origin, the line of separation, which is closer to origin, will lie below the two classes instead of separating them. LAC1 algorithm therefore needs additional preprocessing to overcome this drawback. A modified Linear attribute combination algorithm (LAC2) is proposed in this thesis to get a decision boundary, which separates the two classes no matter where they are clustered. In this algorithm, the line of separation of the form $y = mx + c$ is obtained by constructing lines through the point which lies approximately at center of the region that separates the two classes distinctly. This algorithm makes use of the fact that each class is mostly clustered around the mean of its distribution. Along with the minimum entropy criterion

the variance of two classes is used as an additional criterion to select the attribute combination and attribute test.

The two dimensional test, for the combination of features *(A, B)* is determined by identifying the decision boundary separating the two classes in the two dimensional A-B space.

Consider a training data *S* set with *n* instances described by attributes *A*, *B*, C with values $\{a_1, a_2, \ldots a_n\}$ $\{b_1, b_2, \ldots b_n\}$ and $\{c_1, c_2, \ldots c_n\}$ respectively at a decision node. The training set S can be represented as a set of pattern vectors

$$S=\{ (\underline{X}_i, Y_i) \}, i= 1..n \tag{4.1}$$

where $\underline{X}_i = \{a_i, b_i, c_i\}$ is the feature vector and Y_i = 0 or 1 represent the two classes.

With 3 attributes *ABC* the number of possible attribute combinations that a decision node can have is $3C_2$=3. Here the two dimensional test for each combination, partitions the data at a test node into two subsets. For example, consider the attribute combination *AB*. Assuming linear decision boundary in two dimensional space spanned by the *A* and *B* axes, represented by the function,

$$F(a,b) = b - ma + c \tag{4.2}$$

where m is the slope of the decision boundary and c is the intercept. The objective is to find an optimal value for the slope m and the constant c in the equation,

$$b = ma + c \tag{4.3}$$

that classifies the two classes at a decision node with minimum misclassification error.

The step by step implementation of the procedure for estimating m and c consists of 3 major modules:

I. Determine the pivotal point on the decision boundary,

II. Determine the $\{ l.d.b \}_{opt}$,

III. Constructing a decision tree with attribute combination that has maximum information gain

Details of each step are explained next.

I. Determine the pivotal point lying on the linear decision boundary

1. For two dimensional attribute space (A, B) compute the means of two classes in S (a_μ^1, b_μ^1) and (a_μ^0, b_μ^0) as

$$a_\mu^0 = \frac{1}{n_0}\sum_{i=1}^{n_0} a_i \;;\; b_\mu^0 = \frac{1}{n_0}\sum_{i=1}^{n_0} b_i$$

where n_0 is the number of patterns in class '0'

$$a_\mu^1 = \frac{1}{n_1}\sum_{i=1}^{n_1} a_i \;;\; b_\mu^1 = \frac{1}{n_1}\sum_{i=1}^{n_1} b_i \qquad (4.4)$$

where n_1 is the number of patterns in class '1'.

2. Determine the bounding rectangles R^0 and R^1 for two classes, in the two dimensional spaces by determining the minimum and maximum values of the two attributes as

$$(a_{\min}^0, b_{\min}^0) = \min\{a_i, b_i\}_{i=1}^{n_0} \;;\; (a_{\max}^0, b_{\max}^0) = \max\{a_i, b_i\}_{i=1}^{n_0}$$

where instance $i \in$ class '0'

$$(a_{\min}^1, b_{\min}^1) = \min\{a_i, b_i\}_{i=1}^{n_1} \;;\; (a_{\max}^1, b_{\max}^1) = \max\{a_i, b_i\}_{i=1}^{n_1} \qquad (4.5)$$

where instance $i \in$ class '1'

3. Let *l (a, b)* represent the line joining mean (a_μ^1, b_μ^1) and (a_μ^0, b_μ^0).

4. Determine the intersection points of line *l* with R^0 and R^1.

Let $l \cap R^i = \{\underline{I_{inner}^i}, \underline{I_{outer}^i}\}$, $i=0,1$

where $\underline{I_p^i} = (a_p^i, b_p^i)$, p = *inner, outer* and the $\underline{I_p^i}$ intersection points satisfy the condition

$$d\{(a_\mu^0, b_\mu^0), \underline{I_{inner}^1}\} < d\{(a_\mu^0, b_\mu^0), \underline{I_{outer}^1}\}$$

$$d\{(a_\mu^1, b_\mu^1), \underline{I_{inner}^0}\} < d\{(a_\mu^1, b_\mu^1), \underline{I_{outer}^0}\}$$

where *d(. , .)* is the Euclidean distance

5. The pivotal point (a_{db}, b_{db}) lying on the decision boundary is then chosen as the midpoint of the line segment defined by $\underline{I_{inner}^o}$ and $\underline{I_{inner}^1}$ as below.

$$a_{db} = \frac{a_{inner}^0 + a_{inner}^1}{2} \quad ; \quad b_{db} = \frac{b_{inner}^0 + b_{inner}^1}{2} \tag{4.6}$$

II. Determine the linear decision boundary (l.d.b) $b = ma + c$ *that partitions the training set with maximum information gain*

This step can be performed in two possible ways. The first approach is a brute force search for optimal values of the parameters *m* and *c* and it is done by evaluating the information gain for a set of *l.d.bs* systematically at increasing angles. Assuming θ_r is the angle made by the *l.d.b* with *A* axis, a set of lines are constructed at increasing angles θ_r according to equation 4.7.

$$\theta_1 = 0,\ \theta_r = r\Delta\theta\ ;\ r = 1,2\ldots k$$

where $k = (180/\Delta\theta)$ (4.7)

For instances, choosing $\Delta\theta$=15, $\{\theta_r, r=1,2\ldots,k\}$={ 0,15,30,....180}

This will result in set of *l.d.bs* as in equation 4.8, generated using equation 4.9.

$$b = m_r\ a + c_r \tag{4.8}$$

$$m_r = tan(\theta_r)$$

$$c_r = b_{db} - m_r\ a_{db} \tag{4.9}$$

where (a_{db}, b_{db}) is defined in equation 4.6.The decision boundary b=*ma*+*c* partitions the training data *S* into two subsets as

$$S_1 = \{ (\underline{X}_i, Y_i) \},\ i = 1\ldots\ s_1$$

where s_1 is the number of instances that satisfies $b_i - m\ a_i \leq c$

$$S_2 = \{ (\underline{X}_i, Y_i) \},\ i = 1\ldots\ s_2 \tag{4.10}$$

where s_2 is the number of instances that satisfies $b_i - m\ a_i > c$

The entropy for the above partition is computed for each *l.d.bs* and the resulting information gain is determined. The optimal decision boundary is then selected as follows.

$$\{\ l.d.b\ \}_{opt} = max\ (\ Gain\ \{\{\ l.d.b\ \}_r\}\ );\ r=1,2\ldots k \tag{4.11}$$

The $\{\ l.d.b\ \}_{opt}$ given by equation 4.12 is the optimal line of separation of two classes corresponding to maximum information gain.

The $\{ l.d.b \}_{opt}$ is chosen as the test for the two dimensional linear attribute combination.

$$b = m_{opt}\ a + c_{opt} \tag{4.12}$$

where m_{opt} and c_{opt} are the corresponding optimal parameters of m and c.

In the second approach the search for optimal values m and c is done more intelligently than in the first approach. The *l.d.bs* are constructed only at 6 angles given by

$$\theta_r = \{ \theta^1_{\min}, \theta^1_{\max}, \theta^0_{\min}, \theta^0_{\max}, \theta_{bis1}, \theta_{bis2} \} \tag{4.13}$$

where the angles θ_r are determined from the data distribution. First the origin of the two dimensional space is shifted to (a_{db}, b_{db}). The shifted attribute values (a_i', b_i') $i=1..n$ is given by

$$\begin{aligned} a_i' &= a_i - a_{db}; \\ b_i' &= b_i - b_{db}; \quad i=1,2...n \end{aligned} \tag{4.14}$$

The rectangular to polar coordinate transformation in the shifted coordinates will result in θ_i given by equation 4.15.

$$\theta_i = \tan^{-1}\left(\frac{b_i'}{a_i'}\right); \quad i=1,2...n \tag{4.15}$$

Using the polar coordinates of the shifted data points the six angles are computed by equation 4.16 for constructing *l.d.bs*.

$$\theta^0_{\min} = \min\{\theta_i\}_{i=1}^{n_0}$$

$$\theta^0_{\max} = \max\{\theta_i\}_{i=1}^{n_0}$$

where instance $i \in$ class '0' and n_0 is the number of patterns in class '0'

$$\theta^1_{\min} = \min\{\theta_i\}_{i=1}^{n_0} \; ; \qquad (4.16)$$

$$\theta^1_{\max} = \max\{\theta_i\}_{i=1}^{n_0}$$

where instance $i \in$ class '1'and n_1 is the number of patterns in class "1".

$$\theta_{bis1} = bisector(\theta^0_{\min}, \theta^1_{\max})$$

$$\theta_{bis2} = bisector(\theta^1_{\min}, \theta^0_{\max})$$

Shifted attribute values are used only for angle computation alone. The parameters m_r and c_r in the decision boundary are determined using true attribute values (a_i, b_i). The parameter m_r and c_r are computed for each *l.d.b* at the computed angles $\theta_r = \{\theta^1_{\min}, \theta^1_{\max}, \theta^0_{\min}, \theta^0_{\max}, \theta_{bis1}, \theta_{bis2}\}$ and their respective partitions of the data are determined using equation 4.10. The entropy is computed for each *l.d.b* and the optimal boundary with maximum information gain is selected for the attribute combination by implementing 4.10 and 4.11.

III. Constructing a decision tree with attribute combination test that has maximum information gain.

Compute the optimal decision boundary $\{ l.d.b \}_{opt}$ for each attribute combination by implementing step *I*, and second approach in *II* and determine the attribute combination with maximum information gain as shown in equation 4.17.

$$Node\ attribute = max\ \{\ gain\ \{\ l.d.b\ \}_{opt}^{u,v}\ \}$$

where $gain\ \{\ l.d.b\ \}_{opt}^{u,v}$ is the maximum information gain for attribute combination u, v (4.17)

The decision boundary corresponding to attribute combination with maximum information gain is selected as the test at the decision node and the outgoing branches are obtained. If more than two attribute combination test has maximum information gain, then the attribute combination with maximum interclass distance given by equation 4.18 between $\underline{I_{inner}^{o}}$ and $\underline{I_{inner}^{1}}$ is selected at the test node.

$$Maxdist^{u,v} = max\ \{\ d(\ \underline{I_{inner}^{0}}, \underline{I_{inner}^{1}}\ )_{u,v}\ \} \tag{4.18}$$

Using the $\{\ l.d.b\ \}_{opt}^{u,v}$ corresponding to maximum information gain the outgoing branches are determined. Each branch is assigned as leaf node if the subset in that branch satisfies the condition that all patterns in the subset belong to same class. The tree is built recursively from the branch if the subset in the branch contains a mixture of patterns from both classes, repeating the steps *I, II, III* until all the instances are correctly classified.

4.2 Illustration of LAC2 with examples

4.2.1 Example 1

The following example illustrates the LAC2 algorithm. Consider the data set in Table 3.1. The training set has two continuous valued attributes X and Y belonging to 'class 0' and 'class 1'. Here we have only two attributes and therefore only one combination of two attributes is possible. Our aim is to determine the best linear decision boundary with maximum information gain in two dimensional XY space. The decision tree with twodimensional test $y=mx+c$ is built by implementing steps I, II, III steps in section 4.1. The vertices of the bounding rectangles, the mean of two classes, and the innermost intersection points are given in the Table 4.1. Figure 4.1 shows the data distribution of the two classes with bounding rectangles, the line joining the means and the two innermost intersection points obtained by implementing the step I in section 4.1.

Table 4.1: Computed coordinates by LAC2 for Data in Table 3.1

Class	Vertices of bounding rectangle	Mean	Innermost intersection points
'0'	(.5,0.5); (1.5,5) (1.5,2.2); (.5,2.2)	(1.0,1.28)	(1.5, 2.05)
'1'	(1,2); (3.5,2) (3.5,4); (1,4)	(2.16, 3.1)	(1.46, 2.0)

Table 4.2 shows the computed minimum, maximum and bisector angles

$\theta^1_{\min}, \theta^1_{\max}, \theta^0_{\min}, \theta^0_{\max}, \theta_{bis1}, \theta_{bis2}$ using the second approach in step II.

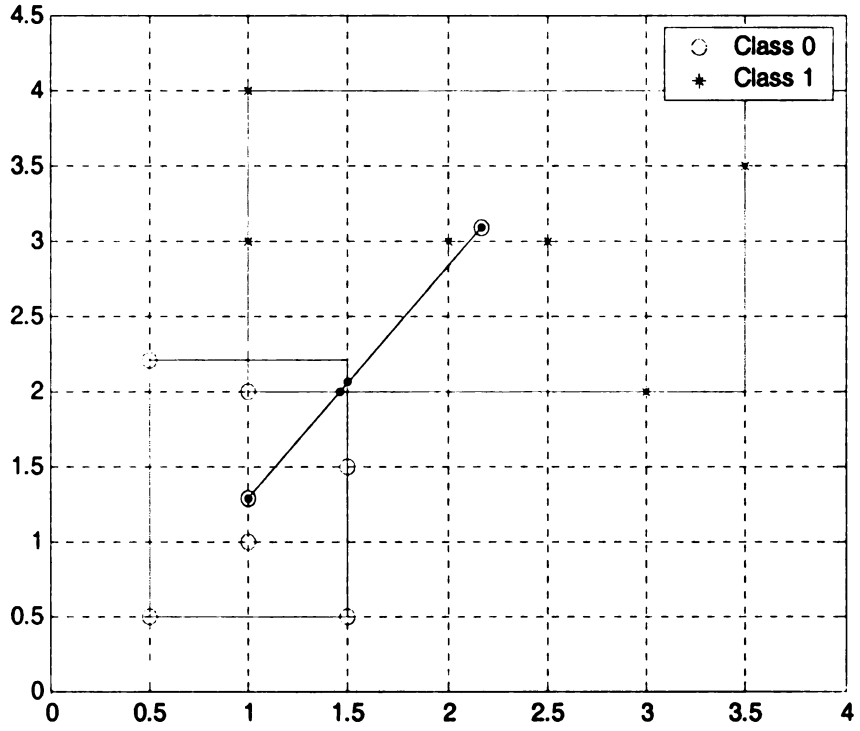

Figure 4.1: Data distribution with computed coordinates

Table 4.2: Angle in Degrees

θ_{bis1}	θ_{bis2}	θ^0_{max}	θ^0_{min}	θ^1_{min}	θ^1_{max}
84.5	334	270	170	36	358

The decision boundary at angle θ_{bis2} partitions the training data with maximum information gain and the two dimensional test $y = m_{opt}\ x + c_{opt}$ with parameters $m_{opt} = -0.49$ and $c_{opt} = 2.7$ are selected as the test at the decision node. The parameters m_{opt} and c_{opt} are computed using equation 4.7 and 4.8.

$m_{opt} = tan\ (334\ (3.14/180)) = -0.49$

$a_{db} = (1.5+1.46)/2 = 1.48$

$b_{db} = (2.05+2.0)/2 = 2.03$

$c_{opt} = b_{db} - m_{opt} = 2.7$

The data under discussion is linearly separable and all the instances of two classes are separated by the decision boundary $y+0.49x=2.7$. Implementing step *III* results in a simple decision tree with single decision node classifying all instances correctly. Figure 4.2 shows the final decision boundary with minimum entropy or maximum information gain. This decision boundary classifies all the instances correctly.

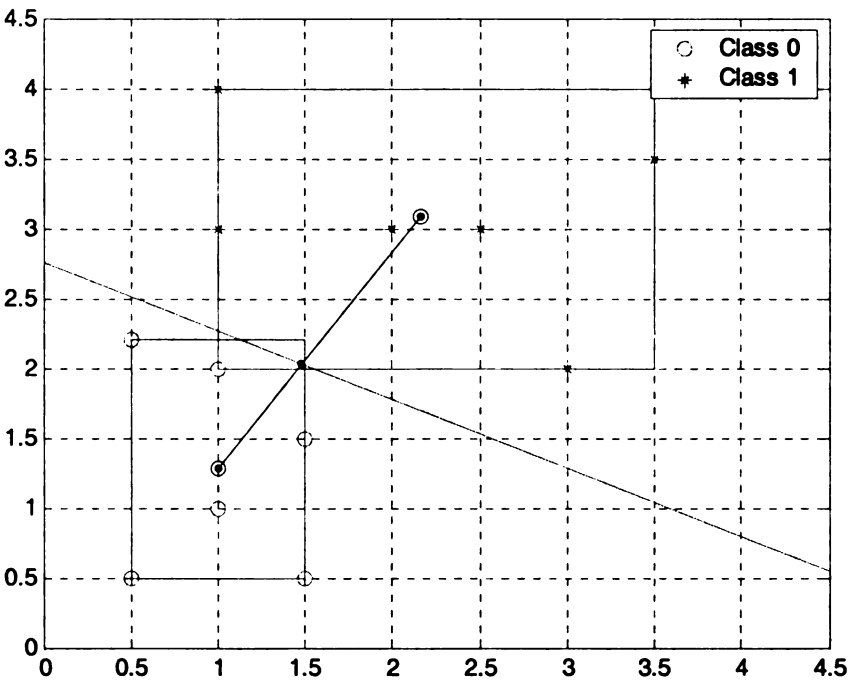

Figure 4.2: Final Decision boundary Obtained by LAC2 algorithm

In the above data set only one combination of attribute is possible. For data sets, having more than two attributes the number of attribute combinations will be more than three. In such cases the minimum entropy of the selected test equation for each attribute combination is compared and finally the attribute combination with minimum entropy is selected as the test equation at the node.

In general for a pattern vector with g attributes $\{Y_1, Y_2 ... Y_g\}$ there are $h = {}^{n}C_2$ possible pair wise combination of attributes. For each attribute combination (Y_i, Y_j) $i \neq j$ the LAC2 algorithm gives the best decision boundary function $f_{i,j}$ in the $Y_i - Y_j$ space. The final bivariate test is selected using the maximum information gain criterion similar to that of in ID3 algorithm.

4.2.2 Example 2

Consider another example of training data set shown in Table 4.3. This training data set has total of 12 instances and 3 attributes *X, Y, Z*. The data distribution for different attribute combination in a two dimensional space and the decision boundary with maximum information gain is shown in the Figure 4.3. The attribute combination *XY* is selected at root node as it has minimum entropy. Implementing the steps *I, II, III* in section 4.1 results in a decision tree with two test nodes as shown in the Figure 4.4.

The training data is linearly non separable, therefore the decision tree by LAC2 algorithm needs more than one decision node to classify all instances correctly. The decision tree using ID3 algorithm for same data needs three decision nodes and 4 leaf nodes to classify all the instances correctly. Figure 4.5 shows the decision tree obtained using ID3 algorithm. The decision tree obtained using LAC2 algorithm is simpler than that obtained using ID3 algorithm.

Table 4.3: Training Data with 3 Attributes

Instance No	Attribute X	Attribute Y	Attribute Z	Class
1	1	1.3	1	0
2	1.5	0.8	3	0
3	0.8	2.5	2.5	0
4	0.5	0.5	1	0
5	2	1.5	0.5	0
6	1	2	2	0
7	1.5	4	2.5	1
8	2	3	3	1
9	2.5	3	1.5	1
10	3	2	1.5	1
11	3.5	3.5	2	1
12	1	1.6	5	1

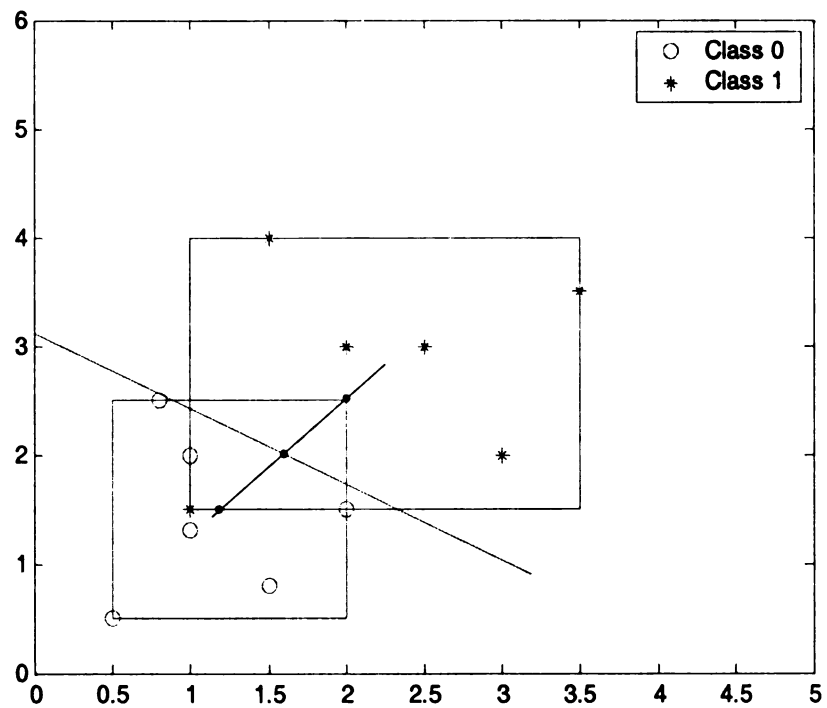

Figure 4.3 (a): Data Distribution for attributes combination X and Y at Root node

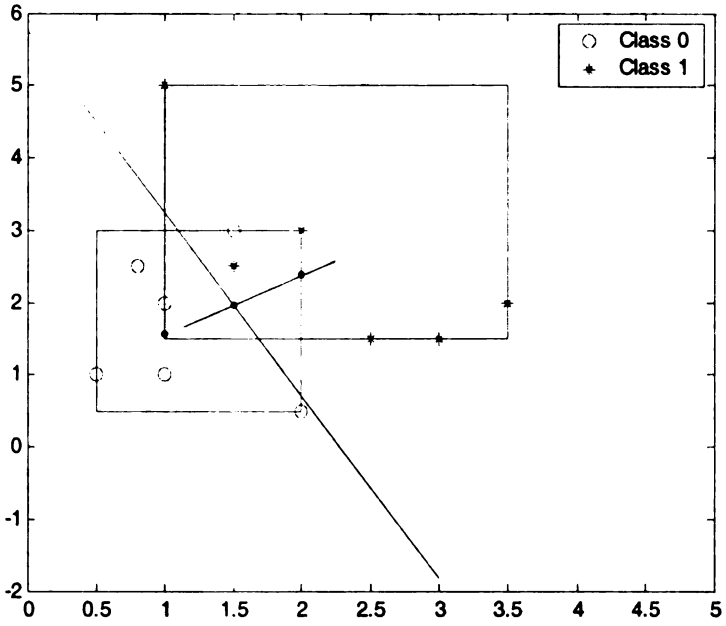

Figure 4.3 (b): Data Distribution for attributes combination X and Z at Root node

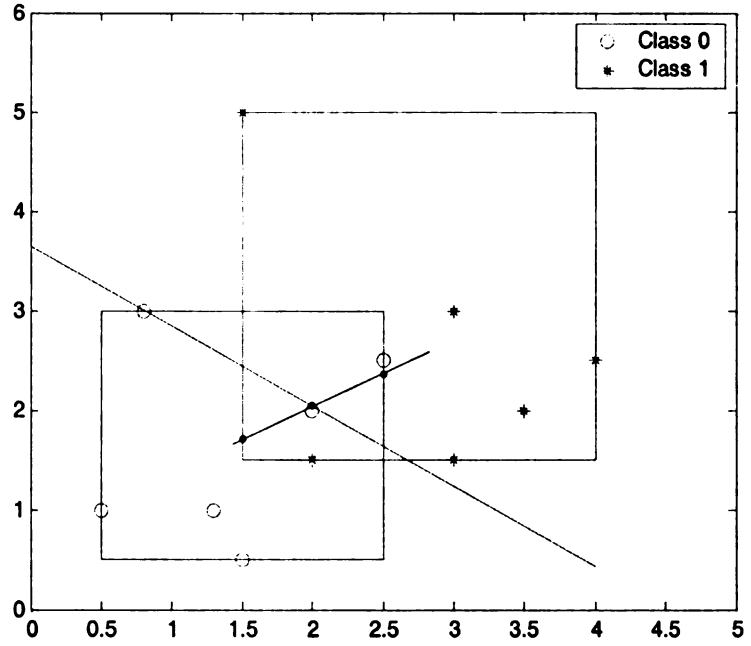

Figure 4.3 (c): Data Distribution for attributes combination Y and Z at Root node

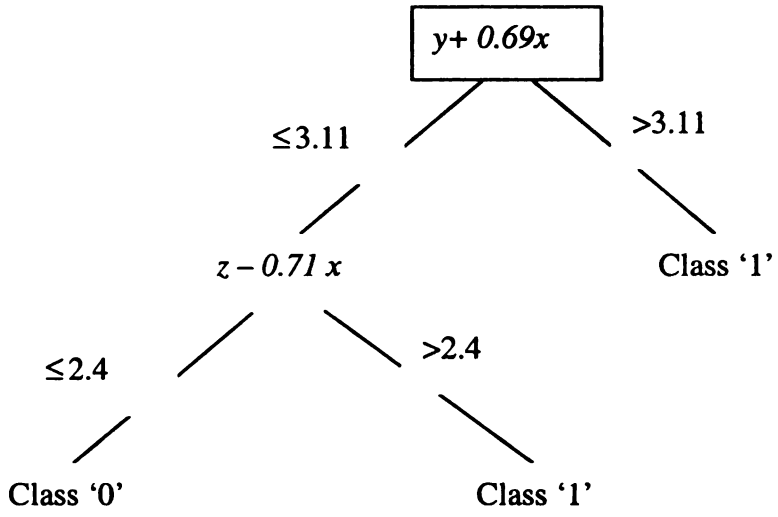

Figure 4.4: Decision Tree for Data set in Table 4.3 using LAC2

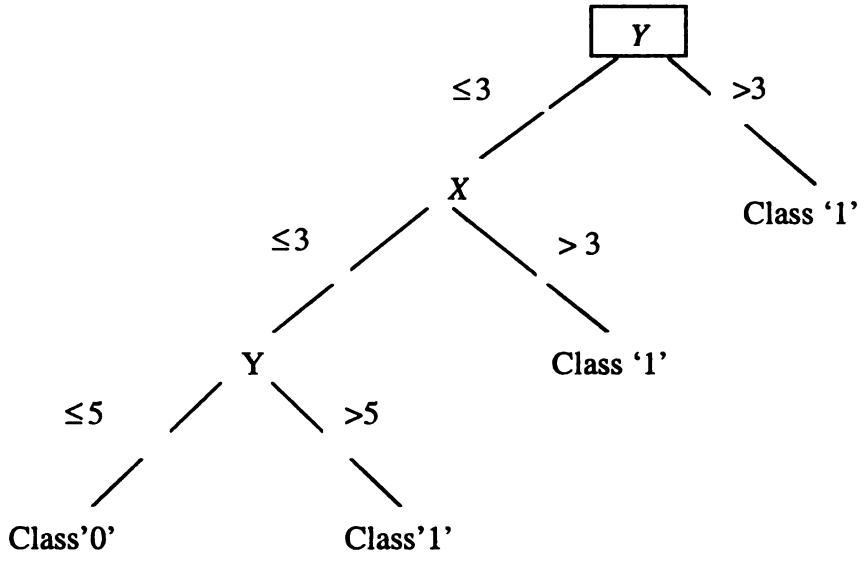

Figure 4.5: Decision Tree for Data set in Table 4.3 using ID3

4.3 Distance Criterion

If two or more attribute combinations have the same minimum entropy then the distance criterion is used in LAC2 algorithm for selecting the winning attribute combination. The distance between the innermost intersection points of two classes is a direct measure of the class separation and hence the classification performance for test data. This is because of the intuitive fact that for any classification model built using training data truly representative of the test data, the unknown instance of a particular class is clustered around the mean of the class. The examples shown in Figures 4.6 and 4.7 illustrate the above phenomena. Figure 4.6(a) shows the training data distribution and the decision boundary for attribute combination *AB* of the two classes. The distance between the innermost intersections (black dots) that separates two classes is very small. Figure 4.6(b) shows the test data for two classes and the misclassification of the test data by decision boundary. Though the training data is correctly classified misclassification in test data is typical when the interclass distance is small. Now consider Figure 4.7, which shows the training data distribution in the *AC* space for attribute combination *AC*. Compared to attribute combination *AB*, the weighted interclass distance of two classes for the same training data in Figure 4.7(a) is larger in the *AC* space. Figure 4.7(b) shows the test data for two classes, in which all instances are correctly classified by the decision boundary using attribute combination *AC*. Whenever more than one attribute combinations has minimum entropy, the attribute combination with maximum interclass "distance" is selected at the decision node.

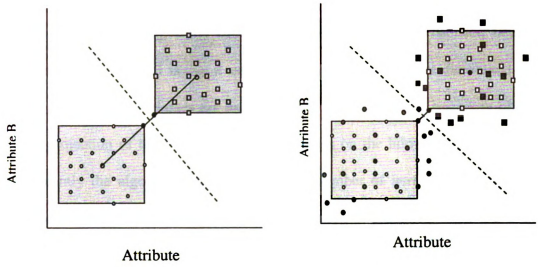

Figure 4.6: a) Distribution of Training data. b) Distribution of training and test data

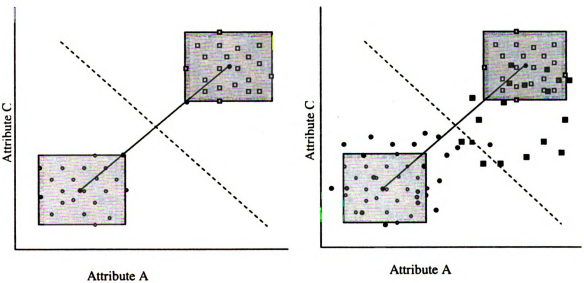

Figure 4.7: a) Distribution of Training data. b) Distribution of training and test data

4.4 Comparison with LAC1 algorithm

The examples of data distribution when LAC1 fails are shown in Figures 4.8 and 4.9. In both cases the LAC2 algorithm correctly identifies the optimal decision boundary. A major drawback of LAC1 is that it assumes that the two classes are distributed about the origin, so that the two classes are on either sides of the decision boundary obtained by joining projection of the patterns from two classes on the horizontal and vertical axes. Consequently the algorithm requires excessive number of iterations or equivalent tree expansion to converge to the true partition between two classes when the underlying assumption is not valid. The LAC2 algorithm overcomes this problem by constructing the decision boundary through center (a_{db} , b_{db}) of region between the two classes in the step *II.*

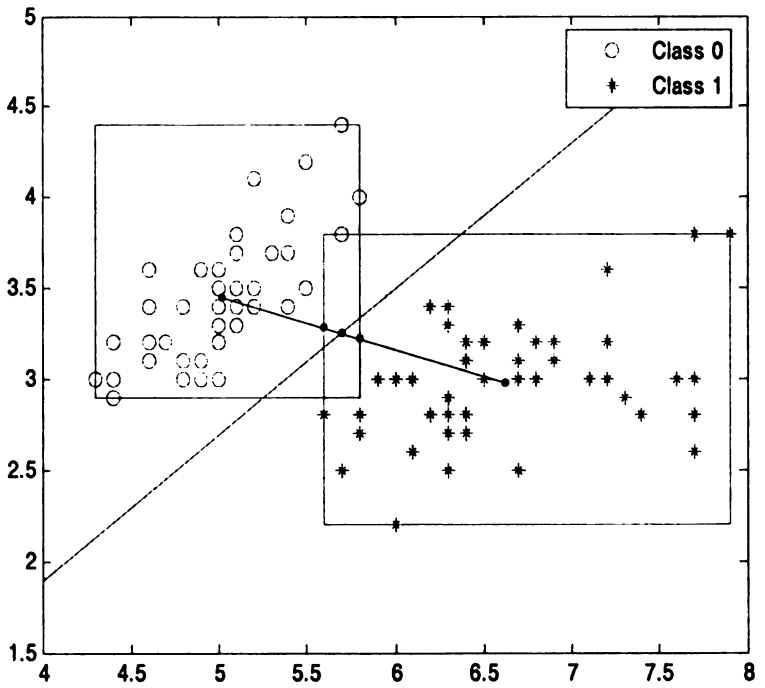

Figure 4.8: (a) Decision Boundary Obtained by LAC2 algorithm for separable classes in IRIS data set

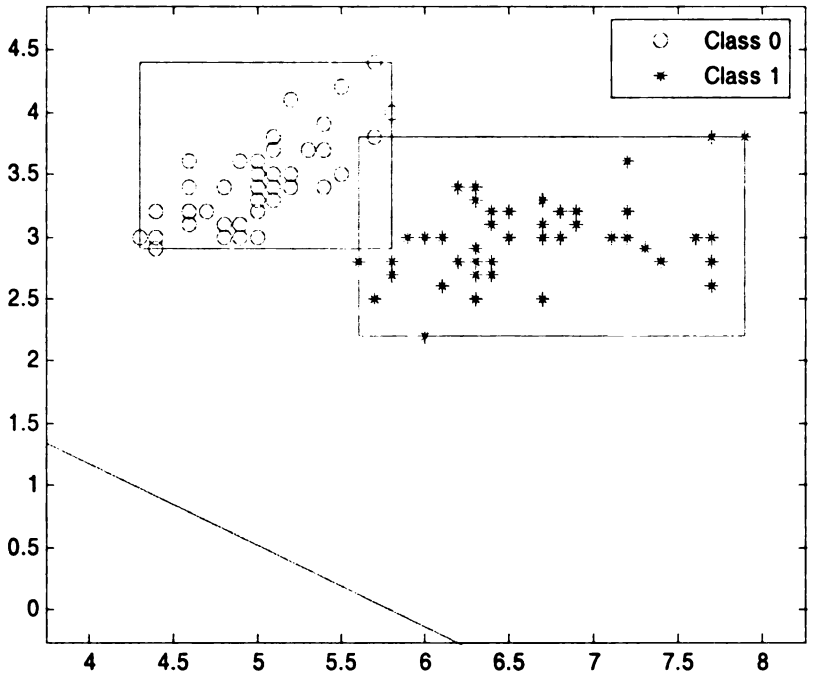

Figure 4.8: (b) Decision Boundary Obtained by LAC1 algorithm for separable classes in IRIS data set

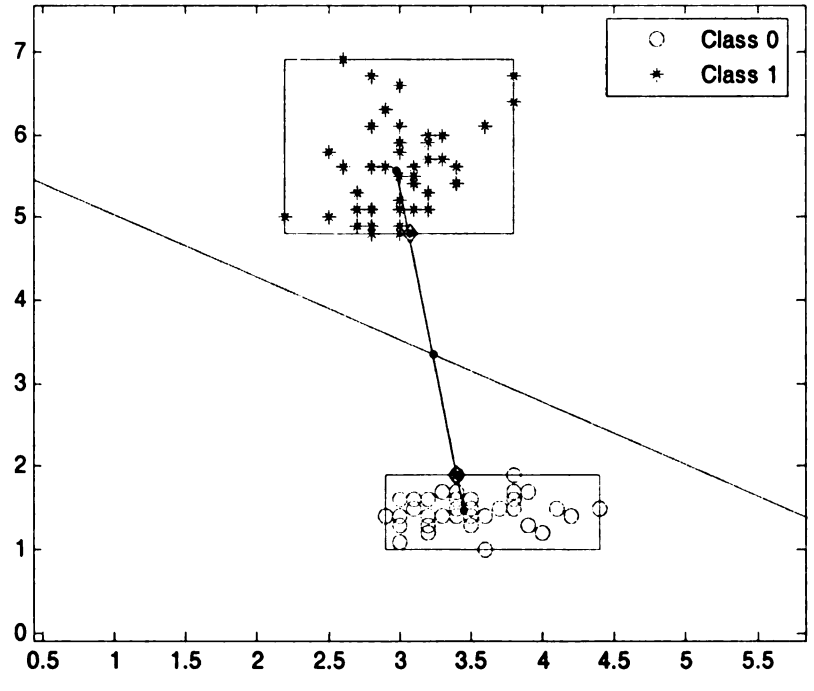

Figure 4.9: (a) Decision Boundary Obtained by LAC2 algorithm for separable classes in IRIS data set

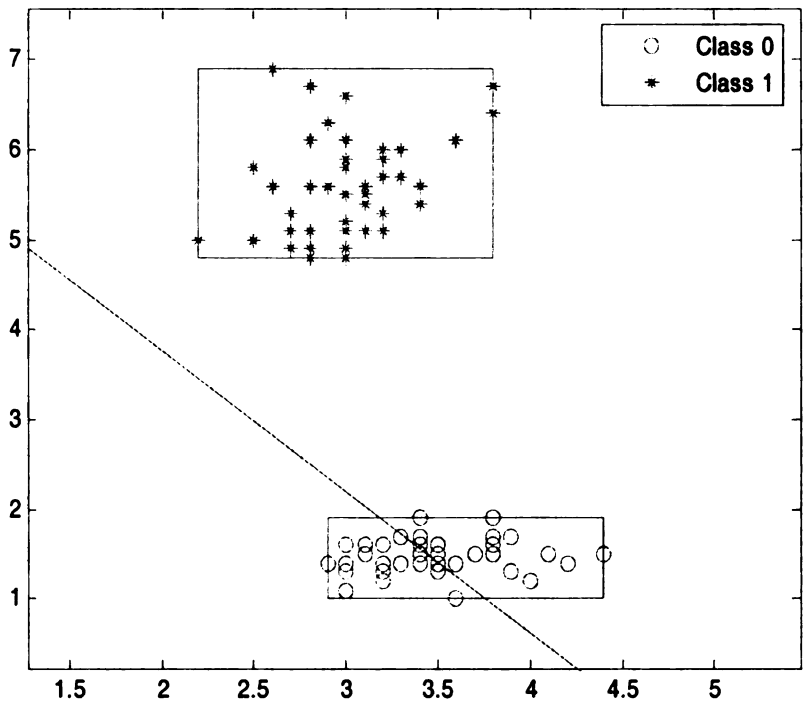

Figure 4.9: (b) Decision Boundary Obtained by LAC1 algorithm for separable classes in IRIS data set

The decision tree algorithms that we have discussed so far result in linear decision boundaries using one or more attributes at a decision node. For nonlinearly separable data, nonlinear decision boundaries will give much simpler tree with better classification performance than the decision trees with linear test. The following chapter explains an algorithm in which a nonlinear test is used at every decision node for partitioning the data for classification.

Total attribute combination set AB is determined by all possible pair wise combination of attributes. All the instances in training data set S is assigned a class.

Given a training data set S, classes set C and total attributes combination set A Bas input to the Linear Attribute Combination algorithm2 (LAC2) it executes the following procedure. If all entries in S are from the same class

{ Return a leaf node with that class name as label }

else if S is empty

{ Return a single node with value failure }

else

{Let approximate midpoint of separation region of two class be x0, y0 and m=tan {mx1, mx2, mn1, mn2, θ1, θ2} be the slope of line passing through x0, y0. Let the line L_q *with slope* m_q *passing through points x0, y0 be the line that has the minimum entropy, with best classification of training set S.*

Let the points x0, y0 be from the two dimensional space formed by using attributes A_j *&* B_k *.Hence let (* A_j, B_k *) be the attribute pair from AB whose distribution space contains the line* L_q.

Then the decision attribute pair at the root node is (A_j, B_k *) and the decision test at the root node is equation of line* L_q *passing through points x0, y0 with slope* m_q *.For the subsets that have instances with y<=mx+c and y>=mx+c add branch below the Root.*

Let S_{jk} *be the subset of S that satisfies y<=mx+c*

If S_{ji} *is empty*

{Then add a leaf node below this branch with label = most frequently occurring class in set S}

else

{Add a new sub tree below this branch and repeat the procedure for other subsets}

end

end

Figure 4.10: LAC2 Algorithm

CHAPTER 5: MULTIVARIATE DECISION TREE USING NONLINEAR ATTRIBUTE COMBINATION (NLAC) ALGORITHM

5.1 Nonlinear Attribute Combination Algorithm - NLAC

In this chapter we propose a decision tree algorithm with non- linear test at each node, replacing the linear univariate and bivariate tests. The algorithm with bivariate attribute tests discussed so far gives linear decision boundary in the two dimensional feature space. The algorithm can be made significantly more powerful by formulating rules that allow nonlinear decision boundaries. The number of decision nodes can be drastically reduced if we use a nonlinear decision node instead of a linear decision node. This is particularly true for data distributions in which two classes are not linearly separable. The nonlinear decision tree proposed in this thesis uses a nonlinear test with two attributes at every decision node. With the help of two attributes the coefficients of a second order polynomial equation is estimated so that maximum information gain or minimum misclassification error is obtained at each node. For a set of attributes the attribute combination with minimum entropy test is then selected for the node under consideration.

Consider a training data set S with n instances $\{in_1, in_2, ...in_n\}$. Each pattern in_n is three dimensional with features *X, Y, Z* feature space, that takes on $\{x_1, x_2, ...x_n\}$, $\{y_1, y_2, ...y_n\}$, $\{z_1, z_2, ...z_n\}$. Each pattern belongs to either class '0' or class'1'. In the above case we have three attribute combinations *XY, XZ,* and *YZ.* The nonlinear attribute combination algorithm fits a second order polynomial equation for each

attribute combination. The two classes are separated by second order nonlinear decision boundary estimated in the two dimensional feature space.

I. Determining the coefficients of second order non linear decision boundary

1. Given a two dimensional attribute space *X, Y* the second order nonlinear equation, can be represented as

$$ax_i^2 + by_i^2 + cx_i y_i + dx_i + ey_i + k = v_i \; ; i=1..n \tag{5.1}$$

The class v_i takes either 'class 0' or 'class 1' depending on the x_i, y_i values of attributes *X, Y* for a particular instance.

2. The six coefficients {*a, b, c, d, e, k*} of second order polynomial equation *a, b, c, d, e,* and *k* are estimated using Gauss Newton nonlinear least-square method as follows [11].

a) The nonlinear equation for all the instances in the training data set can be written in matrix form as [11] in equation 5.2. Prior to fitting a second order nonlinear equation, each attribute values were normalized by its maximum value in the training set and the same normalization is applied to the test data before fitting the nonlinear equation.

$$\begin{bmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ .. & .. & .. & .. & .. & .. \\ .. & & & & & \\ .. & & & & & \\ .. & & & & & \\ .. & & & & & \\ x_n^2 & y_n^2 & x_ny_n & x_n & y_n & 1 \end{bmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ k \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ .. \\ .. \\ .. \\ v_{n-1} \\ v_n \end{pmatrix} \tag{5.2}$$

In matrix form, equation (5.2) can be written as,

$$D\underline{c} = \hat{\underline{\gamma}} \tag{5.2}$$

where the matrix D has the attribute values for all the instances and the vector $\underline{c}$ contains the values of the coefficients and the vector $\hat{\underline{\gamma}}$ has the estimates of the actual class $\underline{\gamma}$.

b) The coefficients are determined by minimizing the square error between the estimated class and actual class. The cost function $\xi(c)$, of this optimization problem is expressed as the sum of squared classification errors as in equation 5.3.

$$\xi(c) = \frac{1}{2}\sum_{i=1}^{N}\left(\gamma(i) - \hat{\gamma}(i)\right)^2 \tag{5.3}$$

The misclassification error $e(i) = \gamma(i) - \hat{\gamma}(i)$ is a function of the coefficient vector $\underline{c}$. The error is minimized with respect to $\underline{c}$ and the coefficient update is expressed as

$$\underline{c}(k+1) = \underline{c}(k) - \left(J_k^T J_k\right)^{-1} J^T e^k \tag{5.4}$$

where J_k the Jacobian matrix of the error vector e^k at the k^{th} iteration is expressed as,

$$J_k = \begin{bmatrix} \frac{\partial e^k(1)}{\partial a} & \frac{\partial e^k(1)}{\partial b} & \frac{\partial e^k(1)}{\partial c} & \frac{\partial e^k(1)}{\partial d} & \frac{\partial e^k(1)}{\partial e} & \frac{\partial e^k(1)}{\partial k} \\ .. & .. & & & & \\ .. & & & & & \\ \frac{\partial e^k(N)}{\partial a} & .. & .. & & & \frac{\partial e^k(N)}{\partial k} \end{bmatrix} \tag{5.5}$$

In order to solve for the coefficient vector $\underline{c}$, we need the matrix $J_k^T J_k$ to be non-singular [11]. The problem of non-singularity problem is addressed by the modified Gauss Newton method where the equation for updating the coefficients is given by

$$\underline{c}(k+1) = \underline{c}(k) - \left(J_k^T J_k + \partial I\right)^{-1} J^T e^k \tag{5.6}$$

where ∂I is positive definite matrix for all instances. The updated coefficients at the k^{th} iteration are used in the $k+1$ iteration and the error vector e^{k+1} is estimated. The process is repeated until the estimated error becomes negligible and the estimated class for all the instances becomes reasonably close enough to the actual class information provided.

3. Once the coefficients are estimated using the modified Gauss-Newton method, the instances at the node are classified using equation 5.7. The node has two branches representing the outcomes of the test at the node. Equations 5.8a and 5.8b denote the classification rule of patterns into class '1' and class '0'.

$$(X_i, Y_i) \in \text{ class '0' if } ax_i^2 + by_i^2 + cx_i y_i + dx_i + ey_i + k \le 0.5 \tag{5.8a}$$

$$(X_i, Y_i) \in \text{ class '1' if } ax_i^2 + by_i^2 + cx_i y_i + dx_i + ey_i + k > 0.5 \tag{5.8b}$$

where a, b, c, e, d, k are the coefficients of nonlinear equation calculated using the modified Gauss-Newton method and x_i, y_i are the attribute values for the i^{th} instance.

4. The entropy for the above partition is computed and the resulting information gain is evaluated.

II. Constructing a decision tree with attribute combination test that has maximum information gain.

Let *{gain (X, Y), gain (X, Z), gain (Y, Z)}* be the information gain of the non linear decision boundary estimated for each two dimensional attribute combination by implementing step *I*.

$$Maxgain_{comb} = max\{gain(X, Y), gain(X, Z), gain(Y, Z)\} \quad (5.8)$$

The decision boundary corresponding to attribute combination with maximum information gain $Maxgain_{comb}$ is selected at the decision node and the outgoing branches are obtained. The instances in the two outgoing branches or subsets are examined and the leaf nodes are assigned if all the instances in a subset belong to same class. If the instances of a branch or subset contains pattern from different classes the tree is built recursively repeating the steps *I, II*.

5.2 Illustration using synthetic data

Figure 5.1 shows the data distribution of 37 instances belonging to class '1' and class '0' for the three-attribute combinations *XY*, *XZ* and *YZ*. The patterns are seen to be nonlinearly separable in all attribute combinations. The nonlinear equation for each combination is estimated by implementing step *I* in section 5.1. For the data distribution shown in the Figure 5.1 the attribute combination *XY* has the lowest entropy and hence the *XY* combination is selected at the root node in step *II*. The estimated coefficients *a, b, c, d, e, k* of nonlinear equation in two dimensional space *XY* are -2.97, -4.16, 3.52, 3.63, 2.50, and -1.72 respectively, which gives bivariate, nonlinear test with two outcomes as

$$-2.97x_i^2 - 4.16y_i^2 + 3.52x_i y_i + 3.63x_i + 2.5y_i - 1.72 \leq 0.5$$

$$-2.97x_i^2 - 4.16y_i^2 + 3.52x_i y_i + 3.63x_i + 2.5y_i - 1.72 > 0.5$$

Table 5.1 shows the attribute values for some instances in the above data distribution after normalizing each attribute with respect to its maximum value. Substituting the *X*, *Y* attributes values and the corresponding coefficients in the nonlinear equation $ax_i^2 + by_i^2 + cx_i y_i + dx_i + ey_i + k = v_i$ the values of v_i are determined as shown in Table 5.1. The values of v_i are then thresholded to get binary valued class information as seen in the last two columns of Table 5.1. The predicted class of instance 1 is same as true class. Similarly for all instances the predicted class in both subsets is the same as true class for the selected nonlinear test at root node.

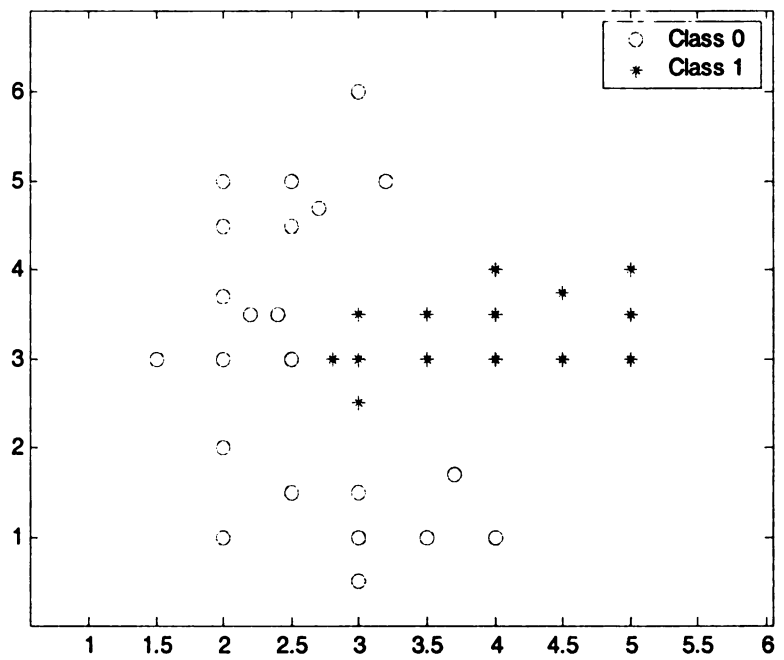

Figure 5.1: (a) Data Distribution of attributes combination X and Y at root node

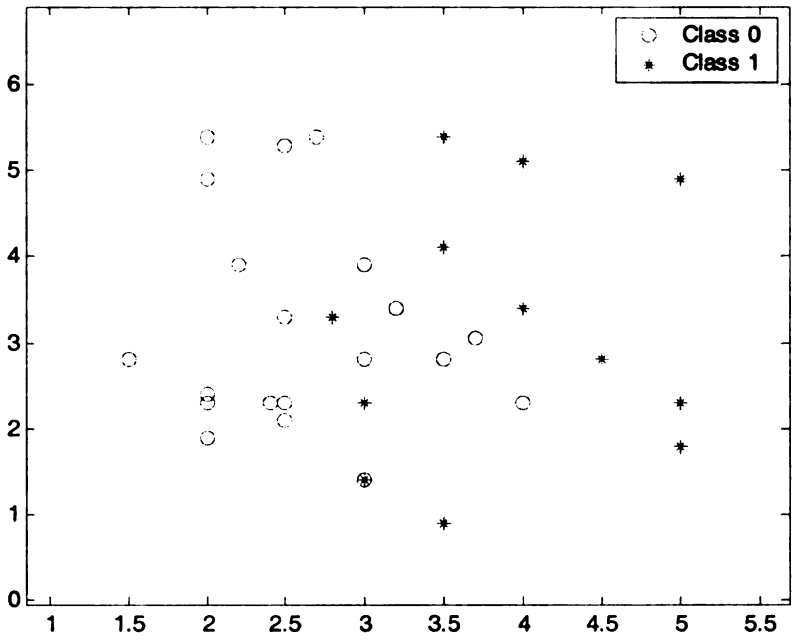

Figure 5.1: (b) Data Distribution of attributes combination X and Z at root node

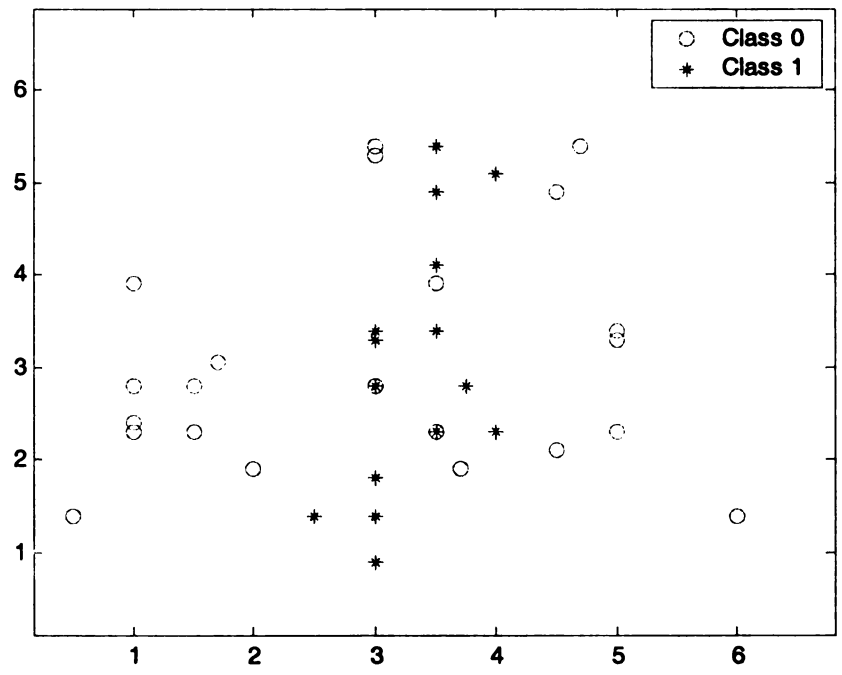

Figure 5.1: (c) Data Distribution of attributes combination Y and Z at root node

Table 5.1: Instances from data Distribution in Figure 5.1

Instances	Attribute X	Attribute Y	Attribute Z	Actual Class	v_i	Predicted Class
1	1	0.58	0.91	1	1.2	1
2	0.8	0.67	0.94	1	1.09	1
3	0.6	0.17	0.72	0	0.11	0
4	0.5	0.83	0.61	0	0.06	0

Therefore both subsets obtained after the root node are pure, in other words all the instances in a subset belongs to class'0' or class'1' and they are assigned as leaf nodes with respective class name. The Figure 5.2 shows the decision tree for the data distribution discussed above. The Figure 5.3 shows the plot of nonlinear decision boundary obtained by NLAC for data shown in Figure 5.1.

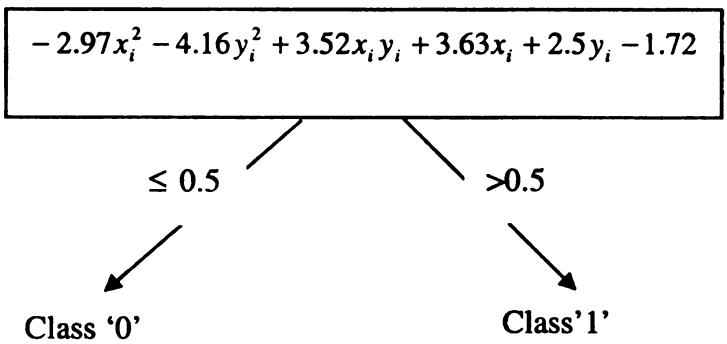

Figure 5.2: Decision Tree using NLAC algorithm for data shown in Figure 5.1

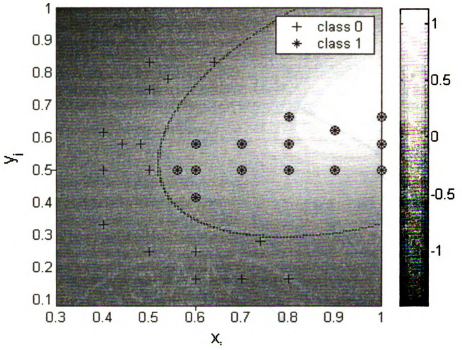

Figure 5.3: Nonlinear decision boundary obtained using NLAC on data shown in Figure 5.1

The decision boundary for the data distribution discussed is highly nonlinear in nature. The LAC2 algorithm with linear decision boundaries results in a complex decision tree as shown in Figure 5.4. The decision tree generated by LAC2 algorithm requires more decision nodes as classes are not linearly separable. The LAC2 algorithm performs well as long as the data is evenly distributed in and around their class mean and bounding rectangle of one class is not entirely contained in another class. For the same nonlinearly separable data in Figure 5.1 the LAC2 algorithm generated a decision tree with more than 4 decision nodes. When the classes are nonlinearly separable, decision tree obtained using nonlinear test nodes are relatively simpler than the decision tree obtained using linear tests at decision nodes. The results obtained using the LAC2 decision tree algorithm and nonlinear decision tree

algorithms are contrasted more elaborately with real world application data in the following chapter.

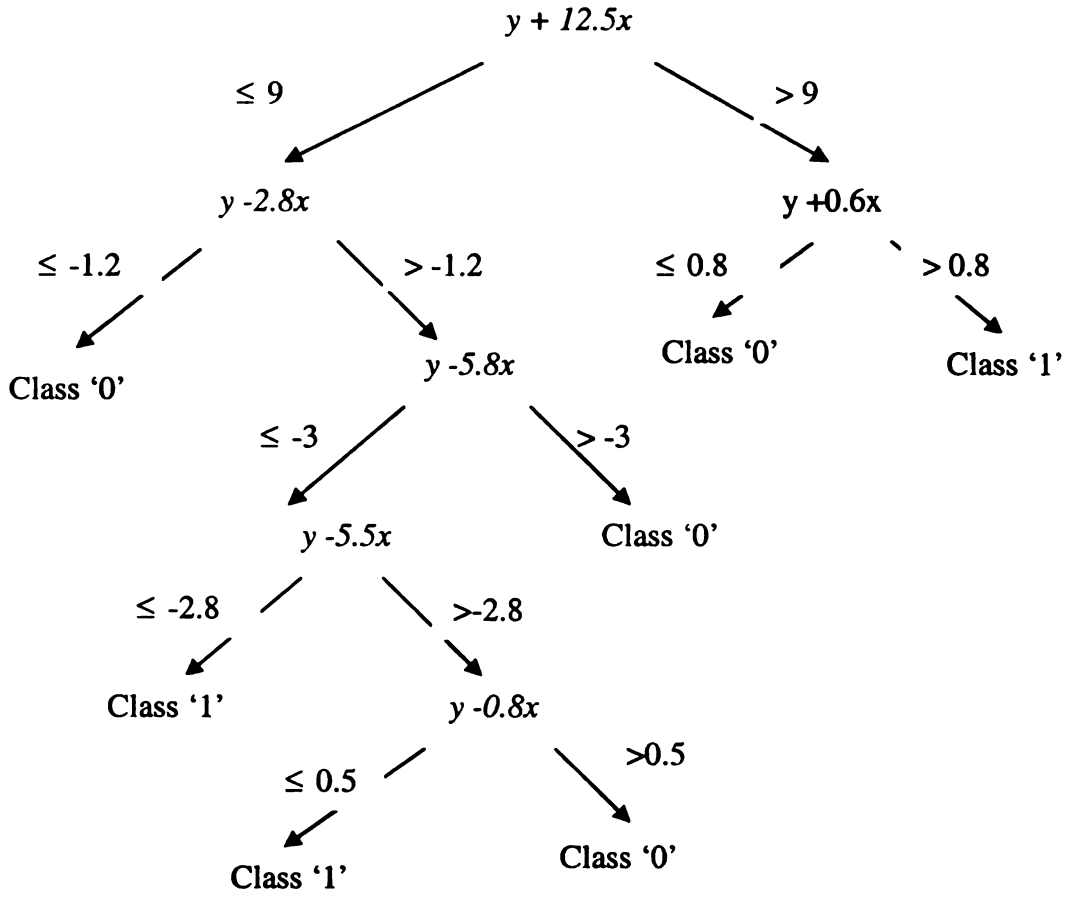

Figure 5.4: Decision Tree by LAC2 algorithm for data shown in Figure 5.1

Total attribute combination set AB is determined by all possible pair wise combination of attributes. All the instances in training data set S is assigned a class.

Given a training data set S, classes set C and total attributes combination set A Bas input to the Nonlinear decision tree algorithm it executes the following procedure.

If all entries in S are from the same class

{ Return a leaf node with that class name as label }

else if S is empty

{ Return a single node with value failure }

else

{Let $ax_i^2 + by_i^2 + cx_i y_i + dx_i + ey_i + k = v_i$ *be the non linear equation that has the minimum entropy, with best classification of training set S.*

Let the points x, y be from the two dimensional space formed by using attributes A_j & B_k *.Hence let* (A_j, B_k) *be the attribute pair from AB whose distribution space contains the line the non linear equation* $ax_i^2 + by_i^2 + cx_i y_i + dx_i + ey_i + k = v_i$.

Then the decision attribute pair at the root node is (A_j, B_k) *and the decision test at the root node is non linear equation with minimum entropy .For the subsets that have instances with* $v_i = 0$ *and* $v_i = 1$ *add branch below the Root.*

Let S_{jk} *be the subset of S that satisfies* $v_i = 0$

If S_{ji} *is empty*

{Then add a leaf node below this branch with label = most frequently occurring class in set S}

else

{Add a new sub tree below this branch and repeat the procedure for other subsets}

end

Figure 5.5: NLAC decision tree algorithm

CHAPTER 6: RESULTS AND DISCUSSION

6.1 Database description

The JE, LAC1, LAC2, and NLAC decision tree algorithms offers significantly improved results in terms of classification accuracy and simplicity of decision tree largely due to the use of bivariate tests at each decision node. In this chapter the performance of ID3 and JE orthogonal decision trees, LAC1 and LAC2 linear decision trees, and finally NLAC decision tree are evaluated for 10 different training and testing data pairs on three databases. The Iris benchmark data and two additional databases of signals from eddy current Array probe and Rotating Probe coil obtained during inspection of steam generator tubes are used for evaluating the different algorithms.

6.1.1 Iris Database

The Iris database is often used as a benchmark database in pattern recognition field. This dataset contain three types of Iris plants 'Setosa', 'Versicolor', and 'Virginica' each having 50 instances. Each pattern vector has four attributes namely sepal length, sepal width, petal length and petal width. Two of the three classes 'Versicolor' and 'Virginica' are nonlinearly separable from each other. The third class 'Setosa' is linearly separable from the other two classes. For testing the performance of different algorithms, the data from two nonlinearly separable classes are used. All attributes values are continuous valued and the class 'Versicolor' is assigned as class'0' and class 'Virginica' is assigned as class'1'.

6.1.2 Eddy Current database

a) Array Probe database

This data is basically eddy current data collected from an Array probe used during the inspection of heat exchange tubes in steam generator units from nuclear power plants. Array probe uses the eddy current principle and collects data from the heat exchange tubes for Non-Destructive evaluation [14]. More details of Eddy current and NDE methods can be found in EPRI Tech Report [14]. The data is collected at different excitation frequencies for detecting different types of defects such as cracks, MBMs, dents, corrosion etc. Given the data collected from a tube the data is preprocessed for noise removal and the region of interest (ROI) containing potential defect signals are identified. The next step extracts the features from the data in region of interest for classification. A training database with the ground truth is used for building the tree. Each ROI is an instance described by the features extracted from it. Using the ground truth the ROI's are assigned as class'1' for defects or class'0' for non-defects. The above database describing each instance with attributes and classes is divided into training and test data set. The database used for evaluating the performance of ID3, JE, LAC1, LAC2 and NLAC decision tree algorithm has two classes namely MBM defects labeled as class'1' and non-defects labeled as class'0'. This particular Array probe data was collected at four different frequencies, namely 90,170,380 and 750 KHz. All features used for classification are extracted from the data collected at excitation frequency 380 KHz. The database has 4 features or attributes namely, vertical peak-to-peak voltage, phase angle, ratio of vertical and horizontal component, and magnitude of the signal. Figure 6.1 shows

the images of Array probe data before noise removal, after noise removal and the identified ROI. Figure 6.2 shows the line scan of the data within the identified ROI.

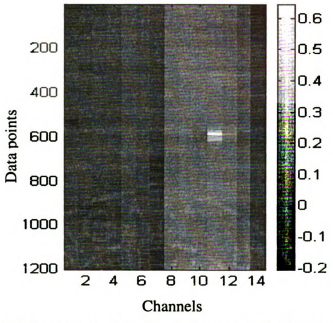

Figure 6.1: (a) Array probe Data Vertical component - Data before preprocessing.

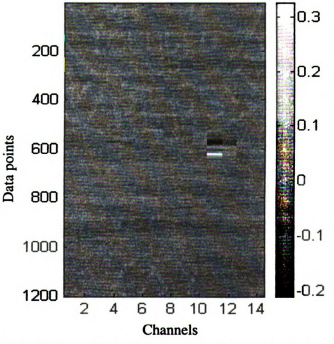

Figure 6.1: (b) Array probe Data Vertical component - Data after preprocessing.

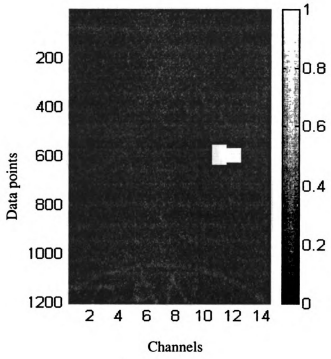

Figure 6.1: (c) Array probe data - Identified ROI

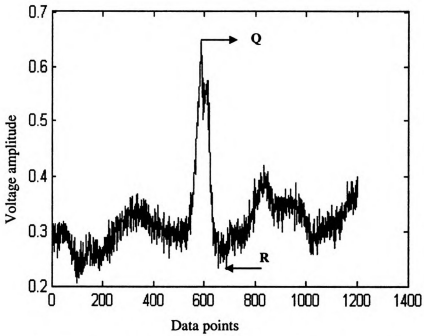

Figure 6.2: (a) Line scan of the ROI - Vertical Component.

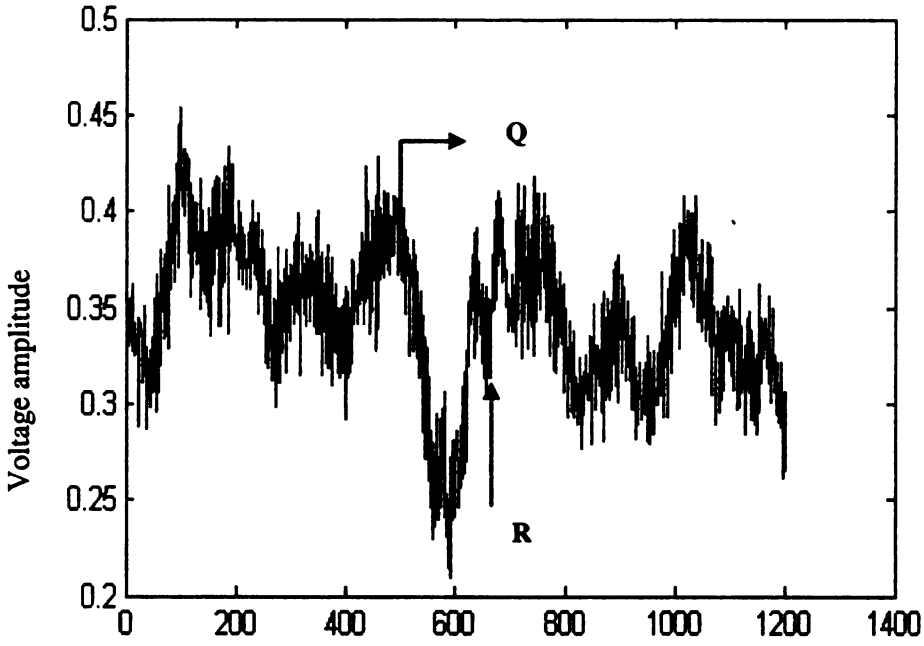

Figure 6.2: (b) Line scan of the ROI – Horizontal Component

The line scan through the peak value in the ROI is determined. For the ROI shown in Figure 6.1(c) the line scan through the peak value is in channel 11. Figures 6.2 (a) and 6.2 (b) show the vertical and horizontal component data from channel 11 in the array probe. After removing the mean in the ROI, the points Q and R in Figure 6.2(a) correspond to data points at which the vertical component signal has maximum v_q and minimum v_r voltage respectively. The corresponding horizontal voltage values at Q and R are determined as, h_q, and h_r. From these vertical and horizontal voltages the four features are computed by equation 6.1. The Table 6.1 shows the computed features. Similarly the features are extracted from each ROI and given as the input to decision tree algorithm for classification.

Vertical peak to peak = $v_q - v_r$ 6.1(a)

Phase angle = $\tan^{-1}\left(\frac{v_q}{h_q}\right)$ 6.1(b)

Ratio of vertical and horizontal = $absolute\left(\frac{v_q}{h_q}\right)$ 6.1(c)

Magnitude = $\sqrt{v_q^2 + h_q^2}$ 6.1(d)

Table 6.1: Features extracted from the ROI

Vertical peak to peak in volts	Phase angle in degrees	Ratio of vertical and horizontal in volts	Magnitude in volts
0.39	71.8	2.7	0.41

b) RPC Database

RPC is the acronym for Rotating probe coil. This probe also uses the eddy current principle for collecting inspection data similar to the Array probe. It also serves the same purpose of Non-destructive evaluation of heat exchange tubes, but the way the data is collected is different from that of the Array probe .The RPC probe moves in a helical fashion along the length of the tube collecting the data in circumferential direction as well as axial direction. The Array probe in contrast contains a circumferential array of sensors each collecting the data in axial direction. The resolution of data collected with the RPC is relatively higher than that of the Array probe, but the time required by the RPC for collecting the data is relatively longer than the time required by Array probe. The RPC is also excited at different frequencies for collecting the data. The data collected from the heat exchange units are pre-processed for removal of noise and the ROI's are identified.

Once the ROI's are identified features are extracted from the ROI's and labeled as defect or non-defect class using the ground truth. Data in each ROI is represented by four features or attributes namely maximum vertical voltage, maximum phase angle, maximum horizontal voltage and maximum magnitude of the signal. Features are extracted in a similar fashion as explained for Array probe.

6.2 Results

6.2.1 ID3 and JE Algorithm- Orthogonal decision boundaries

a) Iris database

The Iris database containing two nonlinearly separable classes with 50 instances each was divided randomly into training and testing data with 25 instances from each class. The process was repeated to obtain different sets of training and testing database pairs and all five algorithms were implemented. All four features were used for evaluating the performance. The results of performance of all algorithms are compared. The decision tree built using training data is tested with the corresponding test data from the dataset pair. Table 6.2 and Figure 6.3 shows the results obtained using ID3 and JE algorithm for the Iris database with four attributes. ID3 algorithm selects any one out of four attribute that has the minimum entropy at every decision node. With four attributes in each pattern vector, the JE, LAC1 and LAC2 algorithm has 6 combinations of two attributes. These algorithms select the combination with minimum entropy at every decision node. The misclassification rate in Table 6.2 for ID3 algorithm varies from 6 to 3 with average misclassification rate as 4.9. The misclassification rate for JE algorithm in Table 6.2 ranges from 2 to 4. For 7 out of 10 data sets JE algorithm consistently performs

better than ID3 algorithm. Besides classification accuracy JE algorithm also results in simpler decision tree with lesser number of decision nodes. But the number of leaf nodes is more in JE algorithm. This is because of the fact that for every decision node the JE algorithm partitions the data into four subsets whereas ID3 algorithm partitions the data into two subsets. Moreover although JE algorithm makes use of two attributes at every decision node, it generates orthogonal decision boundaries similar to ID3. The LAC1 algorithm overcomes the drawbacks of JE algorithm and builds a simple decision tree with non-orthogonal decision boundary with two attributes in a two dimensional feature space.

Table 6.2: Results of ID3 and JE algorithm on Iris database non-separable classes

# Sets	ID3		JE	
	# Misclassification	# Decision Nodes	# Misclassification	# Decision Nodes
1	4	3	4	2
2	5	4	3	2
3	6	3	4	3
4	5	4	3	2
5	4	3	4	2
6	5	4	3	2
7	6	3	4	2
8	5	4	3	2
9	2	3	2	2
10	5	4	3	2

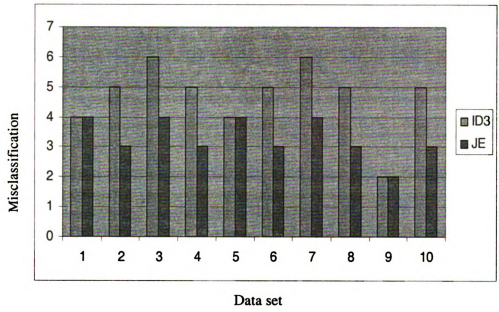

Figure 6.3: (a) Misclassifications with ID3 and JE on Iris database

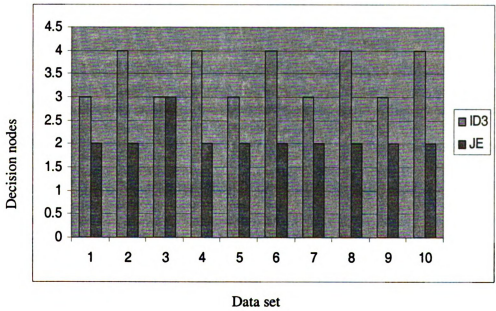

Figure 6.3: (b) Decision nodes with ID3 and JE on Iris database

b) Array probe database

The Array probe database comprising 27 instances belonging to defect class and 73 instances from non-defect class were split into training and test data sets. The training data consists of 18 instances from defect class and 47 instances from non-defect class. The test data was made of remaining 9 instances from defect class and 25 instances from non-defect class. Ten different data sets with training and testing pairs were generated randomly from the database for evaluating the performance. Table 6.3 and Figure 6.4 show the number of misclassifications and the decision nodes for Array probe data using ID3 and JE algorithm with orthogonal decision boundaries. The two classes 'defect' and 'non-defect' are nonlinearly separable in the Array probe data set. By choosing training data whose distribution is very close to test data distribution we can build a simple decision tree that classifies all instances correctly. However in practice this is seldom true. The multiple randomly selected training and test data pairs ensure that we have enough diversity in the choice of training and test data pairs. JE algorithm gives zero misclassification for data set four. On the ten data sets, the average misclassification by JE algorithm is 2.3 whereas for the ID3 algorithm the average misclassification is 2.5. Although the difference in misclassification rate is not too large, JE algorithm results in a simpler decision tree than the ID3 algorithm. The next section shows the results obtained using LAC1 and LAC2 algorithms with non-orthogonal decision boundary.

Table 6.3: Results of ID3 and JE algorithm on Array probe database

# Sets	ID3		JE	
	# Misclassification	# Decision Nodes	# Misclassification	# Decision Nodes
1	2	1	2	1
2	4	2	4	1
3	3	1	3	1
4	3	3	0	2
5	2	2	4	1
6	3	1	3	1
7	2	3	1	2
8	3	1	3	1
9	1	3	1	2
10	2	2	2	1

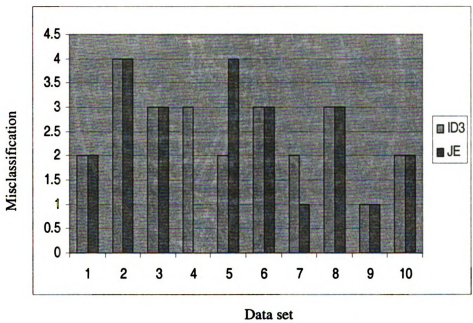

Figure 6.4: (a) Misclassifications with ID3 and JE on Array Probe database

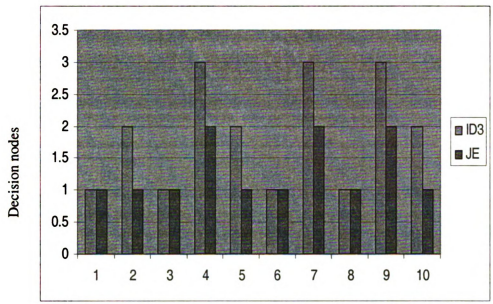

Figure 6.4: (b) Decision nodes with ID3 and JE on Array Probe database

c) RPC database

The RPC database used for evaluating the performance of the decision tree algorithms has 22 instances from 'defect' class and 115 instances from 'non-defect' class. The training data consists of 16 instances from 'defect' class and 81 instances from 'non-defect class. The test data comprises of remaining 6 instances from 'defect' class and 34 instances from 'non-defect' class. Randomly generated pairs of training and testing data sets were applied as input to the decision tree algorithm. The two classes defect and non-defect in RPC database are nonlinearly separable. Table 6.4 and Figure 6.5 summarize the results of performance with JE and ID3 decision tree algorithm. The misclassification rate for ID3 and JE algorithm is more or less in the same range.

Table 6.4: Results of ID3 and JE algorithm on RPC database

# Sets	ID3		JE	
	# Misclassification	# Decision Nodes	# Misclassification	# Decision Nodes
1	2	3	2	2
2	3	3	1	2
3	2	2	2	1
4	2	2	2	1
5	3	3	3	2
6	4	1	4	1
7	2	2	2	2
8	4	2	4	1
9	2	3	2	1
10	2	2	2	1

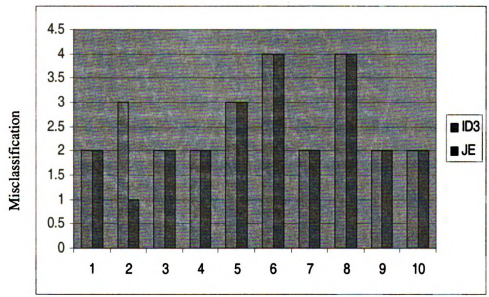

Figure 6.5: (a) Misclassifications with ID3 and JE on RPC database

Average misclassification for ID3 is 2.6 and for JE it is 2.4. However the JE algorithm results in a simpler decision tree with just 2 or 1 decision node whereas the ID3 algorithm uses upto 3 decision nodes for obtaining the same range of classification accuracy.

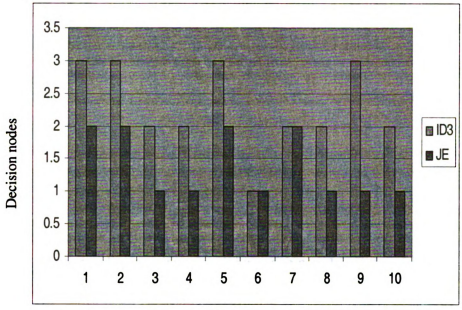

Figure 6.5: (b) Decision nodes with ID3 and JE on RPC database

d) Attributes Selection

Figure 6.6 and 6.7 shows the decision tree for data set number 1 obtained using ID3 algorithm and JE algorithm on Iris database shown in Table 6.2. Attribute '1, 2, 3, 4' corresponds to sepal length, sepal width, and petal length and petal width. The decision tree constructed by ID3 algorithm selects the attributes petal length, petal width and sepal width as the test attributes at three decision nodes. The decision tree constructed by JE algorithm consists of two decision nodes. The JE algorithm also uses the same three attributes used in ID3, but in two dimensional combination at each node, and hence lesser number of nodes than ID3 algorithm.

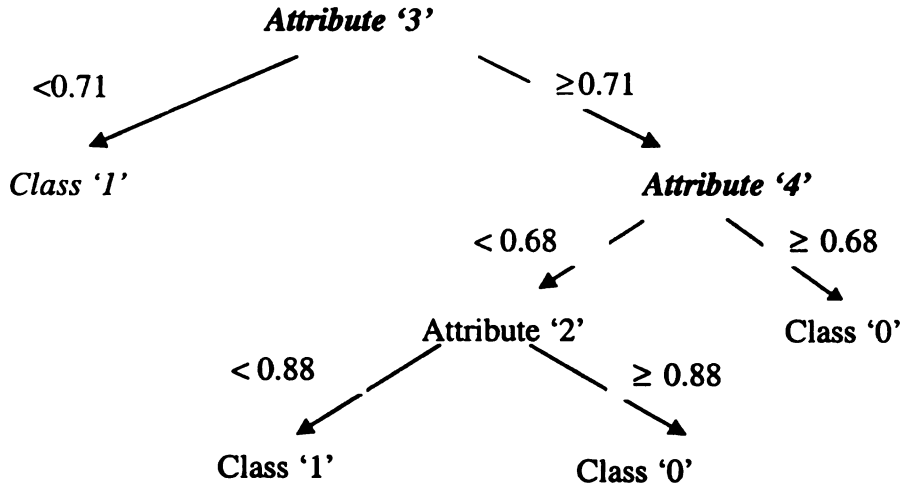

Figure 6.6: Decision Tree by ID3 algorithm on Iris Data set 1

Similarly for most of the data sets in the three databases discussed, JE algorithm selected same attributes as in ID3, but in two dimensional combinations with lesser number of nodes.

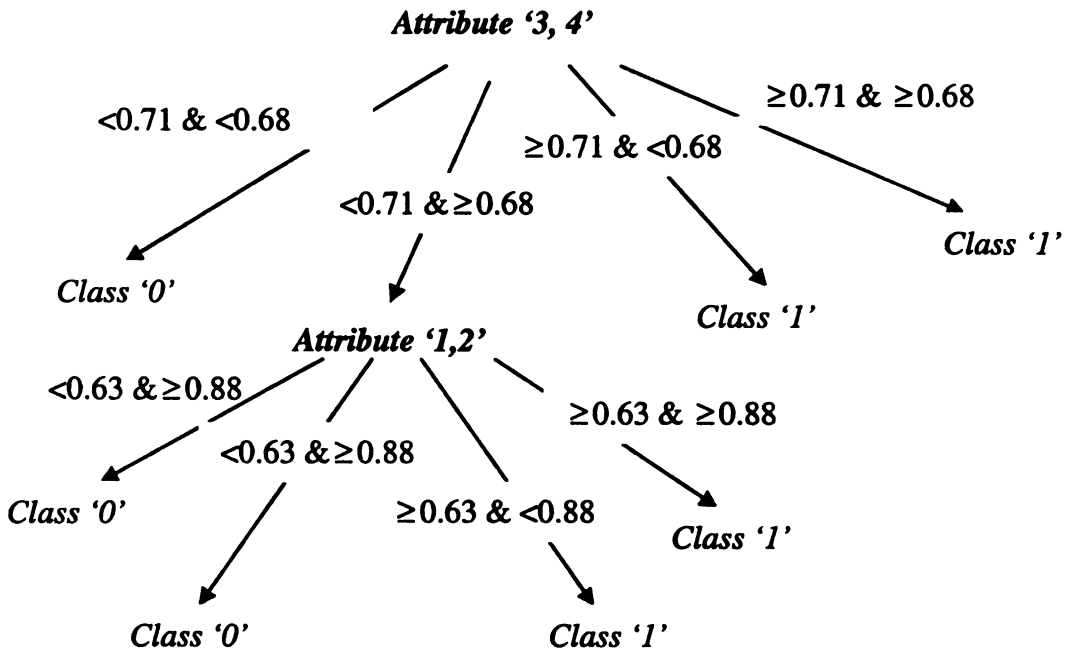

Figure 6.7: Decision Tree by JE algorithm on Iris Data set 1

6.2.2 LAC1 and LAC2 Algorithm–Linear, non-orthogonal decision boundaries

a) Iris database

The LAC1 and LAC2 algorithms generate linear but non-orthogonal decision boundaries in the two dimensional feature space. These algorithms are therefore more powerful than ID3 and JE algorithm. Table 6.5 and Figure 6.8 shows the results obtained using LAC1 and LAC2 algorithms on randomly generated training and testing data set pair used in section 6.2.1. The misclassification rate is seen to vary from 2 to 4 for LAC1 algorithm. On average the misclassification rate for LAC1 is 3.2 whereas for ID3 algorithm and JE algorithm the average misclassification rate is 4.7 and 3.3. Although the LAC1 algorithm outperforms ID3 algorithm in terms of classification accuracy the average level of complexity of the decision tree generated in LAC1 is the same as that of ID3 algorithm. This is because of the fact that the simplicity of the decision tree generated by LAC1 depends on the location of the instances with respect to origin. The LAC2 algorithm overcomes this drawback, as its performance is independent of the data location.

Table 6.5: Results of LAC1 and LAC2 algorithm on Iris database non-separable classes

# Sets	LAC1		LAC2	
	# Misclassification	# Decision Nodes	# Misclassification	# Decision Nodes
1	3	4	3	3
2	3	3	3	3
3	4	4	3	4
4	4	4	3	3
5	3	4	3	3
6	3	4	3	3
7	3	3	3	3
8	4	4	3	3
9	3	4	3	3
10	2	4	2	3

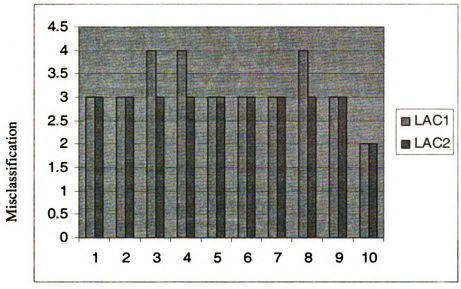

Figure 6.8: (a) Misclassifications with LAC1 and LAC2 on Iris database

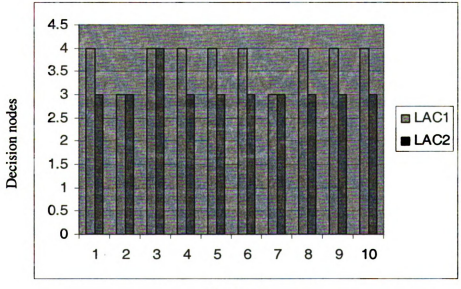

Figure 6.8: (b) Decision nodes with LAC1 and LAC2 on Iris database

The LAC2 algorithm gives better performance in terms of both classification accuracy and fewer decision nodes resulting in simpler decision tree relative to that obtained using

ID3, JE and LAC1 algorithms. The Iris database is non-separable in two dimensional feature spaces and hence the difference between the complexities of decision tree generated by LAC1 and LAC2 algorithms is not that obvious. For linearly separable and nonlinearly separable data in two dimensional feature spaces the LAC2 algorithm tends to give much simpler decision tree.

b) Array Probe database

The performance of linear decision tree algorithm LAC2 is more obvious in the case of Array probe database as it is nonlinearly separable in two dimensional feature spaces, unlike Iris database. Table 6.6 and Figure 6.9 shows the results obtained using LAC1 and LAC2 algorithms on Array probe database. The average misclassification rate with the LAC2 algorithm is 2 whereas with LAC1 algorithm it is 2.2. The LAC2 algorithm has number of misclassifications less than 3 in 7 out of 10 data sets making it more desirable than LAC1 algorithm.

Table 6.6: Results of LAC1 and LAC2 algorithm on Array Probe database

# Sets	LAC1		LAC2	
	# Misclassification	# Decision Nodes	# Misclassification	# Decision Nodes
1	3	2	2	1
2	3	2	2	2
3	3	2	3	2
4	4	4	0	2
5	2	2	2	1
6	3	2	3	1
7	1	3	1	2
8	3	2	3	1
9	4	3	1	2
10	0	2	3	1

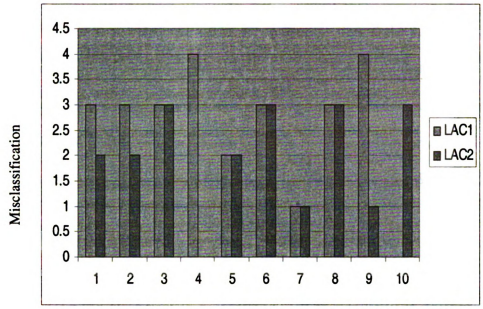

Figure 6.9: (a) Misclassifications with LAC1 and LAC2 on Array Probe database

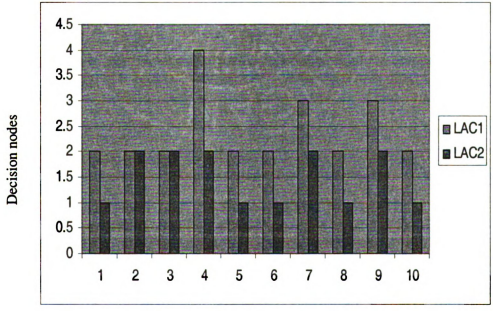

Figure 6.9: (b) Decision nodes with LAC1 and LAC2 on Array Probe database

Besides better classification accuracy, the LAC2 algorithm also provides much simpler decision tree with just 1 or 2-decision nodes in contrast to that of the LAC1 algorithm where the number of nodes ranges from 2 to 4. The LAC2 algorithm has better classification accuracy while yielding simpler decision trees relative to that obtained using the ID3 algorithm. The average number of decision nodes and the misclassification rate for LAC2 are 1.5 and 2 respectively whereas for ID3 the values are 1.9 and 2.5.

c) RPC database

The two classes in the RPC database are also nonlinearly separable similar to the Array probe database. Here the classification accuracy as well as simplicity of decision tree is better than non-orthogonal decision trees reported in section 6.2. Table 6.7 and Figure 6.10 shows the results obtained using LAC1 and LAC2 algorithms on the RPC database. The average misclassification classification rate for LAC1 and LAC2 are 1.8 and 1.7. The difference in misclassification rate is just 0.1. But the LAC2 algorithm has the advantage of lower computational complexity of the resulting decision tree in comparison to that of LAC1 algorithm.

Table 6.7: Results of LAC1 and LAC2 algorithm on RPC database

# Sets	LAC1		LAC2	
	# Misclassification	# Decision Nodes	# Misclassification	# Decision Nodes
1	0	3	2	2
2	1	2	1	2
3	2	2	2	1
4	1	2	1	1
5	2	2	0	3
6	3	2	3	1
7	2	2	2	2
8	3	2	2	1
9	2	2	2	2
10	2	2	2	2

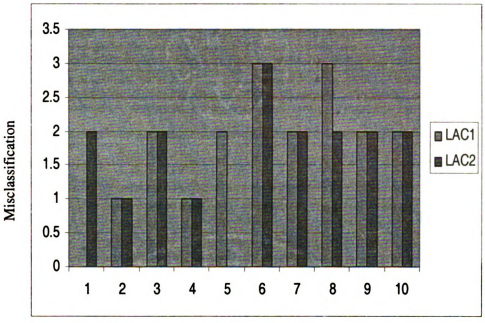

Figure 6.10: (a) Misclassifications with LAC1 and LAC2 on RPC database

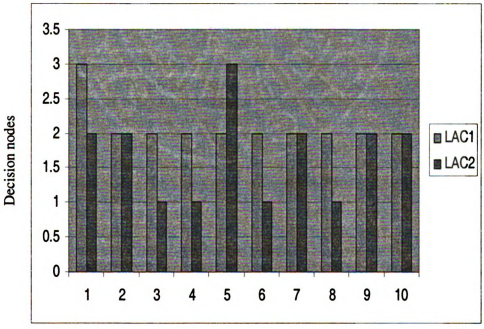

Figure 6.10: (b) Decision nodes with LAC1 and LAC2 on RPC database

Similar to results obtained for Array probe data we can see a significant difference in performance between LAC2 and ID3 algorithm. The average misclassification rate and average number of decision nodes for LAC2 are 1.7 and 1.7 respectively. For ID3 the corresponding values are 2.6 and 2.3.

b) Attributes selection

Figures 6.11 and 6.12 show the decision tree obtained using LAC1 and LAC2 algorithms on Array probe data set number 5 shown in Table 6.6. Attributes *'V, X, Y, Z'* correspond to vertical peak-to-peak voltage, phase angle, ratio of horizontal and vertical component, and magnitude of the signal. The decision tree constructed using LAC1 algorithm selects the attribute combination *V, X* and *V, Z* as the test attributes at two decision nodes. The decision tree generated by LAC2 algorithm consists of just one decision node with two dimensional attribute combinations *V, X*.

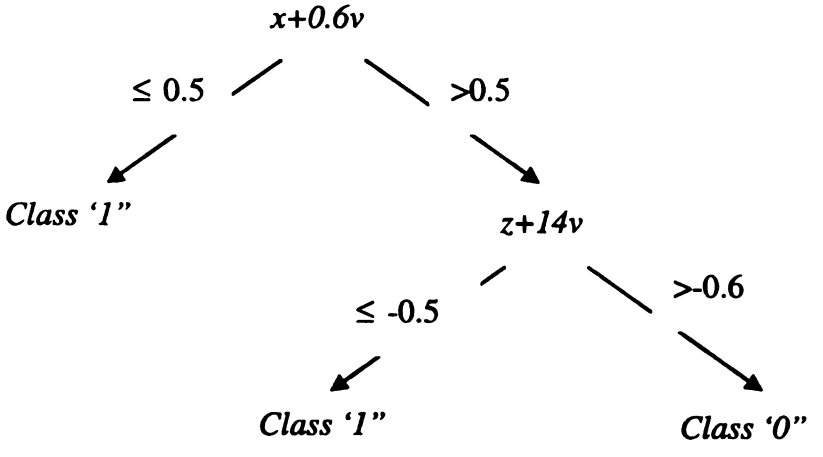

Figure 6.11: Decision Tree by LAC1 algorithm on Array Probe Data set 5

Although both LAC1 and LAC2 employ the same attribute combination at the root node, the LAC1 algorithm needs an additional decision node for classification due to its drawback discussed in chapter 4. Similarly for most of the data sets in the three databases discussed, the attributes combination in LAC2 algorithm were the same as the attribute combination used by LAC1 algorithm, but with lower number of decision nodes.

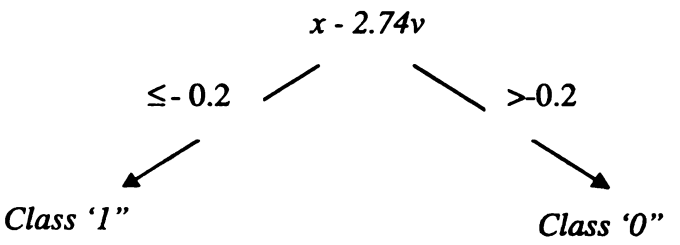

Figure 6.12: Decision Tree by LAC2 algorithm on Array Probe Data set 5

6.2.2 NLAC algorithm - Nonlinear decision boundaries

a) Iris database

The same training and testing data set pairs for Iris, Array probe, and RPC databases randomly generated in section 6.2.1 were used for evaluating the performance of NLAC algorithm. Table 6.8 and Figure 6.13 shows the results on the Iris database obtained using NLAC algorithm for different training and testing data pairs. The misclassification rate for NLAC algorithm is in the range of 2 to 3 similar to that of LAC2 algorithm. But NLAC decision tree is simpler than that of LAC2 algorithm. The average number of decision nodes with NLAC algorithm is 2.1. In contrast the LAC2 algorithm has a more complex decision tree structure with average number of decision nodes being 3.1.

Table 6.8: Results of NLAC algorithm on Iris database

# Sets	NLAC	
	# Misclassification	# Decision Nodes
1	3	1
2	2	2
3	2	2
4	2	3
5	3	2
6	3	2
7	3	2
8	2	3
9	3	2
10	3	2

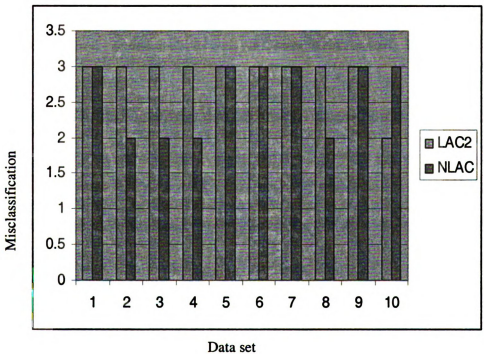

Figure 6.13: (a) Misclassifications with LAC2 and NLAC on Iris database

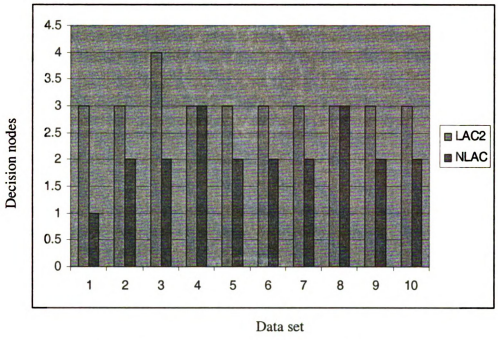

Figure 6.13: (b) Decision nodes with LAC2 and NLAC on Iris database

The average misclassification rate and number of decision nodes for LAC2 are 2.9 and 3.1 whereas for NLAC they are 2.6 and 2.1 respectively. The difference in the misclassification rate for LAC2 and NLAC is just 0.3. However the NLAC algorithm offers a decision tree with minimum number of decision nodes. NLAC algorithm uses two decision nodes for 8 out of ten data sets, whereas LAC2 algorithm uses more than two decision nodes for all ten data sets.

b) Array probe database

The results obtained using NLAC algorithm on Array probe database is summarized in Table 6.9 and Figure 6.14. In section 6.2.2 the decision trees generated by LAC1 and LAC2 have two decision nodes for the data set with zero misclassification. The LAC1 and LAC2 needs at least two decision nodes to classify all non linearly

separable classes, whereas the decision tree generated by NLAC algorithm needs just 1 decision node to classify all the instances correctly for the same data set .

Table 6.9: Results of NLAC algorithm on Array probe database

# Sets	NLAC	
	# Misclassification	# Decision Nodes
1	2	1
2	0	1
3	0	1
4	1	1
5	2	1
6	0	1
7	1	1
8	2	1
9	1	2
10	0	1

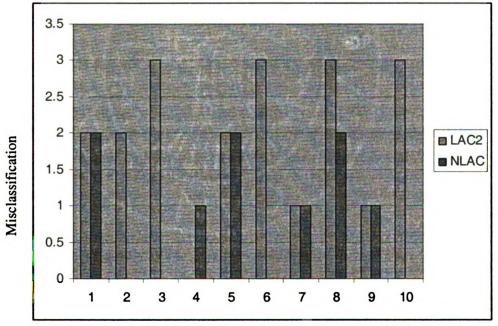

Figure 6.14: (a) Misclassifications with LAC2 and NLAC on Array probe database

In 9 out of 10 data sets the decision tree obtained by NLAC algorithm has just one node whereas LAC2 and LAC1 algorithms resulted in trees with 1 to 4 decision nodes. In terms of classification accuracy the NLAC algorithm has 0 or 1 misclassification for 7

out of 10 data sets. These results clearly show that the NLAC algorithm reduces the complexity of the decision tree without any compromise in classification accuracy.

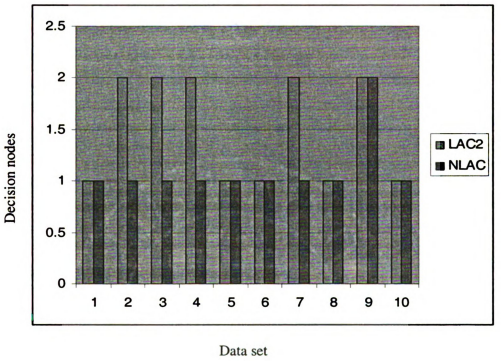

Figure 6.14: (b) Decision nodes with LAC2 and NLAC on Array probe database

c) RPC database

The result obtained using the NLAC algorithm on RPC database is shown in the Table 6.10 and Figure 6.15. In this database NLAC algorithm yields a simple decision tree with 2 decision nodes and zero misclassification for 3 data sets. The LAC2 and LAC1 algorithm generates complex decision tree with three decision nodes and zero misclassification on one data set The average number of decision nodes in the tree generated by NLAC algorithm is lesser than LAC2 and LAC1 algorithms. The NLAC decision tree clearly outperforms the LAC2 and LAC1 algorithms in terms of both

number of decision nodes and classification accuracy when the dataset has classes that are not linearly separable.

Table 6.10: Results of Non linear algorithm on RPC database

# Sets	NLAC	
	# Misclassification	# Decision Nodes
1	0	2
2	0	2
3	1	1
4	1	1
5	0	2
6	2	1
7	1	1
8	2	1
9	1	2
10	3	1

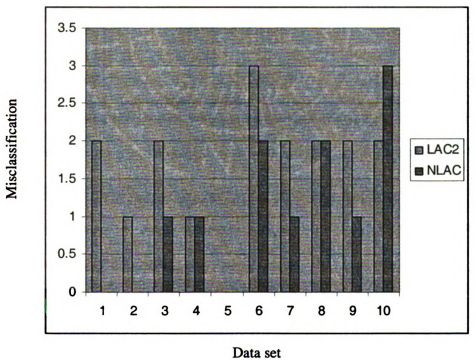

Figure 6.15: (a) Misclassifications with LAC2 and NLAC on RPC probe database

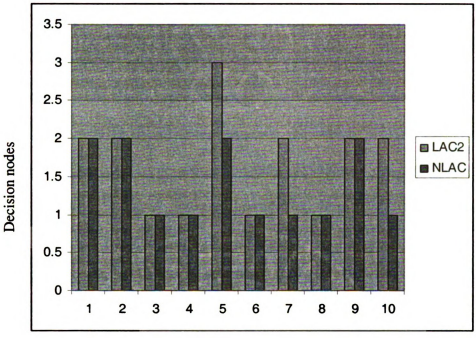

Figure 6.15: (b) Decision nodes with LAC2 and NLAC on RPC probe database

d) Attributes selection

The attributes combination chosen by NLAC algorithm is similar to the attributes combination selected by the LAC1 and LAC2 algorithms. Apart from the attribute combination selected by NLAC algorithm the LAC1 and LAC2 algorithms need additional attribute combination for classifying the nonlinearly separable data resulting in higher number of decision nodes.

6.3 Results summary

The Tables 6.11, 6.12 and 6.13 show the average number of misclassifications and decision nodes on all three databases for the five different decision tree algorithms. The performance of LAC2 algorithm is better than ID3 algorithm in terms of number of decision nodes as well as classification accuracy. The results demonstrate the fact that multivariate decision trees with non-orthogonal boundaries are more efficient than decision tree with orthogonal decision boundaries. Similarly the NLAC algorithm performs better than LAC2 algorithm and appears to be more promising for data distributions with classes that are non- linearly separable.

Table 6.11: Iris Database non-separable classes

Iris probe Database	Average Misclassification	Average Decision node
ID3	4.7	3.5
JE	3.3	2.1
LAC1	3.2	3.8
LAC2	2.9	3.1
NLAC	1.8	2.1

Table 6.12: Array probe Database nonlinearly separable classes

Array probe Database	Average Misclassification	Average Decision node
ID3	2.5	1.9
JE	2.3	1.3
LAC1	2.2	2.3
LAC2	2	1.5
NLAC	0.9	1.1

Table 6.13: RPC Database nonlinearly separable classes

RPC Database	Average Misclassification	Average Decision node
ID3	2.6	2.3
JE	2.4	1.4
LAC1	1.8	2.1
LAC2	1.7	1.7
NLAC	1.1	1.4

Average number of decision nodes for JE is smaller compared to ID3 but the number of leaf nodes for every decision node is higher than in other algorithms. Every algorithm performs well for a particular data distribution. The decision trees proposed in this thesis are suitable for two class problem with continuous valued attributes. The number of attributes should also be small. Large number of attributes leads to more attribute pair combinations in JE, LAC1, LAC2 and NLAC algorithm and hence increases the computation complexity. Of the five algorithms JE algorithm takes the most amount of computation time as it computes the entropy for all combinations of feature values taken by the two attribute combination to determine the appropriate thresholds for every attribute combination. The computation time required by five algorithms in decreasing order is JE, LAC1, LAC2, ID3 and NLAC algorithm. The computation time for Iris training data set is presented in Table 6.14.

Table 6.14: Computation time for Iris training data set

Iris data set	ID3	JE	LAC1	LAC2	NLAC
Computation time in seconds	0.4	4.2	0.7	0.53	0.24

CHAPTER 7: SUMMARY AND CONCLUSIONS

7.1 Summary and Conclusions

The objective of this thesis is to develop a decision tree based classification algorithm that generates non-orthogonal and nonlinear decision boundaries in two dimensional feature spaces. The basic ID3 uses one attribute at every decision node and uses the entropy concept for selecting the test at each node. The ID3 algorithm gives orthogonal decision boundaries and very often tends to over fit the data thereby increasing the size of the tree depth. The tree depth can be reduced considerably by making use of more than one attribute at each decision node. The JE and LAC1 algorithm uses two attribute combinations at every decision node and uses the entropy concept for selecting the test. The JE and LAC1 algorithm reduces the tree depth with comparable or better classification performance than ID3. However the JE algorithm generates orthogonal decision boundaries with increased number of leaf nodes and LAC1 generates complex decision tree when the data distribution is close to the origin.

Two algorithms based on linear and nonlinear attribute combinations referred to as LAC2 and NLAC are proposed in this thesis to overcome the drawbacks in ID3, JE and LAC1 decision tree algorithms. The LAC2 and NLAC algorithms use two attribute combinations and employ the entropy criterion for selecting the bivariate test at each decision node. The performance of LAC2 and NLAC decision tree algorithms were evaluated with three real world application data and contrasted with the corresponding performance of the ID3, JE, and LAC1 decision tree algorithms. Both LAC2 and NLAC decision tree algorithms produce trees with reduced tree depth when compared to ID3, JE and LAC1 algorithms. The LAC2 algorithm generates a decision tree with linear decision

boundaries and its performance is independent of data distribution unlike LAC1 algorithm. For nonlinearly separable data sets the tree depth obtained with linear decision tree can be reduced to almost half, by replacing the linear decision boundaries with nonlinear decision boundaries. For all three databases the tree depth obtained with NLAC algorithm is smaller than that obtained with the other four algorithms without any compromise in classification accuracy. The LAC2 algorithm is most optimal for two class problems with continuous valued attributes that are distributed uniformly in and around the mean of its distribution in the two dimensional feature space. Similarly the proposed NLAC decision tree algorithm is most optimal when the data is nonlinearly separable in two dimensional feature spaces. In JE, LAC1, LAC2 and NLAC algorithms, the instances that are not classified correctly in a particular two dimensional feature space are classified in next node with same or different two dimensional feature space.

The univariate ID3 and its extension C4.5 are extensively used in building classification trees and in data mining because of its robustness and consistency. The results on all three databases obtained using JE, LAC1, LAC2 and NLAC algorithms with multivariate tests are more promising for continuous valued attributes. For higher dimensional pattern vector, the proposed algorithms can be extended to higher dimensional multivariate test at each decision node. Secondly the order of nonlinearity (currently 2) can also be increased. A final extension of the algorithm is the extension to classification of more than two classes.

BIBILIOGRAPHY

[1] Richard O.Duda,Peter E.Hart and David G.Stork "Pattern Classification".New York;Wiley,2001.

[2] J.Quinlan. "Induction of Decision Trees." Machine Learning Vl.1:81-106,1986.

[3] Michie,D.(1983).Inductive rule generation in the context of the Fifth generation. Proceedings of the Second International Machine Learning Workshop. University of Illinois at Urbana-Champaign

[4] Richard O.Duda and Peter E.Hart. "Pattern Classification and scene analysis" New York, Wiley [c1973]

[5] Sammut, C.A. (1985). Concept development for expert system knowledge bases. Australian Computer Journal 17.

[6] C.Shanon. "A Mathematical Theory of Communication." Bell System Technical Journal,
Vol.27,p.379-423 and 623-656,1948.

[7] J.Quinlan. "C4.5:Programs for Machine Learning". Morgan Kaufmann 1993.

[8] T.Mitchell. "Machine Learning." McGraw-Hill, p.52-81,1997

[9] C.Brodley and P.Utgoff, "Multivariate decision Trees." COINS Technical Report 92-82,1992

[10] Savita S.Bhat. "Generalization of ID3 Algorithm to Higher Dimensions" M.S Thesis, Michigan State University.

[11] S.Haykins. "Artificial Neural Networks"

[12] M.Seo. "Automatic Ultrasound Signal Classification scheme".Mater's thesis.

[13] C.Brodley and P.Utgoff, "Multivariate Decision Trees." COINS Technical Report 92-82,1992

[14] EPRI Tech Report, 2003.

[15] E.Feignbaum,P.Mccorduck. "The Fifth Generation:Artificial Intelligence and Japans Computer Challenge to the world." Addison Wesley, Reading, MA 1983.

[16] A.Jessop. "Informed Assesments: An Introduction to Information, Entropy and Statistics." Ellis Horwood, 1995.

[17] R.Mantataras et al., "Comparing information-thoretic attribute selection measures:A statistical approach." *AI Communicatons* 11,1998

[18] M.Last A.Kandel, O.Maimon. "Information Theoretic Algorithm for Feature Selection." *Pattern Recognition Letters*, 2001.

[19] J.Quinlan. "Simplifying decision trees." *International Journal pf Man-Machine Studies*, 27, 1987.

[20] Yao,Wong,Butz. "On Information-Thoretic Measures of Attribute Importance." *Proceedings of Third Pacific-Asia on knowledge Discovery and Data Mining, 1999*

[21] A.Collin. "Building Decision Trees with the ID3 Algorithm." Dr.Dobb's Journal, p.107-109, 1996.

[22] J.Finaly and S.Dix. "An Introduction to Artificial Intelligence." UCL Press,Taylor and Francis Group,1996

[23] S.Russel,P.Norvig. "Artificial Intelligence-A Modern Approach." Pearson Education Asia, 2001.

[24] P.Winston. "Artificial Intelligence" Addision and Wesley Publishing Company, 1984.

[25] A.Collin. "Building Decision Trees with ID3 Algorithm.: Dr.Dobb's Journal,p.107-109,1996

[26] U.Fayyad et al(Ed). "Advances in Knowledge Discovery and Data Mining." AAAI Press, 1996.