This is to certify that the
thesis entitled

Generalization of ID3 Algorithm To Higher Dimensions

presented by

Savita S. Bhat

has been accepted towards fulfillment
of the requirements for the

Master of Science	degree in	Electrical and Computer Engineering

Lalita Udpa

Major Professor's Signature

Jan 6, 2004

Date

GENERALIZATION OF ID3 ALGORITHM TO HIGHER DIMENSIONS

By

Savita S. Bhat

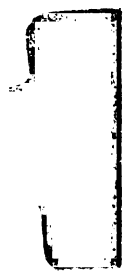
A THESIS

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

Generalization of ID3 algorithm to Higher Dimensions

BY

Savita S. Bhat

Quinlan's ID3 algorithm is used to obtain an efficient classifier in the form of a decision tree from a given training database. Each data sample in the training data set consists of several attributes and belongs to a specific class. A leaf node in the resulting decision tree from the algorithm contains the class name while the non-leaf nodes of the tree are the decision nodes, deciding the attribute and outgoing branches at that particular node. The ID3 algorithm uses a criterion based on information gain to help it decide which attribute should be chosen at the decision node.

This thesis presents a modification of the basic ID3 algorithm for building the decision tree. In contrast to the single feature nodes used in the original ID3 algorithm, this thesis investigates the feasibility of using two attributes at each node. Two approaches are studied. The first method uses joint entropy with two attributes at a time in the decision node while the second method uses a linear combination of two attributes to generate the optimal partition. Both of these methods use the basic principle of minimum entropy or maximum information gain as the criteria. The proposed methods offer considerable scope for generating higher order decision trees, with their results offering the same accuracy and similar computational time as the ID3 algorithm.

ACKNOWLEDGMENTS

Many people have been a part of my thesis work, as friends, teachers and colleagues. I would like to express my sincere gratitude towards my thesis advisor, Dr. Lalita Udpa, for her continuous encouragement in my thesis work as well as expert guidance. Working with her was a joyful experience.

I would also like to thank my thesis committee members, Dr. Satish Udpa, Dr. George Stockman and Dr.Pradeep Ramuhalli. Their expert opinion and comments helped me in correcting this thesis report and understanding the different perspectives of the topic.

I would like to thank my lab mates for their useful inputs during our discussions on this topic. I am forever indebted to my family, who in spite of being far away were one of the most contributing factors in my work. This work would not have been possible without their support. Last but not the least, I am grateful to all my friends who were always there for me and encouraged me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 Introduction

After its recognition in the mid 1950's, the subject of artificial intelligence as a discipline has come a long way and so has machine learning as a major research area. Research on learning has resulted in diverse systems such as the ones that adapt their own performance by monitoring and then adjusting parameters and others that see learning as acquisition of knowledge in the form of concepts. The practical importance of machine learning of the latter kind has been underlined by the advent of knowledge-based expert systems [1]. The traditional approach of such systems is the interview approach involving a domain expert and knowledge expert. However the traditional technique fails to keep pace with increasing demands and complexity of the problem. Feigenbaum [2] calls this as the bottleneck problem and a major obstacle to the use of expert systems. This has resulted in further investigation of different machine learning methods as a means of explicating the knowledge.

There are various approaches to machine learning such as decision tree learning, learning using artificial neural networks, learning based on Bayesian perspective etc. Decision tree learning is a method for approximating discrete valued target functions, in which the learned function is represented by a decision tree [3]. Biological learning systems form the inspiration for the use of artificial neural networks for learning. Artificial neural networks are built using densely interconnected set of simple computational units, similar to neurons in the brain. Each unit takes a number of real-

valued inputs and produces a single real-valued output. These inputs may be outputs of other network units and along the same lines, the output of a node serves as an input to other units. While artificial neural networks take a neurobiological approach to machine learning, Bayesian learning algorithms rely on a probabilistic quantitative approach. These algorithms are based on the assumption that the quantities of interest are governed by statistical distributions and that optimal decisions can be made on the basis of probabilities together with the observed data [3]. Thus it can be seen that machine-learning algorithms are based on diverse areas, such as probability, information theory, neurobiology, mathematics etc.

This thesis focuses on decision tree algorithms for machine learning. The basic decision tree learning algorithms employ a top-town greedy search through the space of possible decision trees. Quinlan's ID3 algorithm and its successor C4.5 are basically built on this approach [1]. This thesis uses the ID3 algorithm as the basis and focuses on modifying this algorithm to two-dimensional features.

1.2 Decision Tree Learning

Decision Tree learning is one of the widely used methods for inductive inference. Members of TDIDT (Top Down Induction Decision Trees) family are characterized by their representation of acquired knowledge as decision trees. These systems use relatively simple language for knowledge formulation and as a consequence the learning methodologies used are considerably less complex [1]. Decision tree learning is basically an inductive learning method.

Top Down Induction of Decision Trees is an approach used for solving the task of supervised learning. In inductive learning, the learner must acquire knowledge from the given training set having limited number of instances and generalize correctly to the unseen instances of the given problem domain. Decision trees classify the training instances by sorting them from the root to some leaf node. Each decision node in the tree represents one of the attributes in the given data set and branches from this node are various values of that particular attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the branch corresponding to the value of the attribute in the given example. The process is repeated for each sub tree rooted at the new node [3]. The ID3 algorithm for decision trees is an example of inductive learning [1].

The ID3 algorithm was introduced by Quinlan to obtain classification models in the form of decision trees from a given data set. The underlying strategy is an example of non-incremental learning, in that it derives its classes from a fixed set of training instances. An incremental learning algorithm revises the current classification model, if necessary, with a new instance. The ID3 algorithm is based on the Concept Learning System (CLS) algorithm. Basically a divide and conquer method, the CLS algorithm constructs a decision tree that attempts to minimize the cost of classifying an object. The CLS algorithm considers two types of costs: the measurement cost of determining the value of property A exhibited by the object and the misclassification cost if the object is misclassified. CLS chooses the division that will minimize the cost and then recursively divides each of the partitioned subsets. In this approach, the expert decides which feature

to select. The ID3 algorithm is an improvement over CLS algorithm, in that it replaces the cost driven approach with an information-driven evaluation function. The ID3 algorithm uses Shannon's entropy function to automatically determine the attribute with the most amount of discriminating information. The method searches through the training set, obtains the best separation and recursively goes on partitioning the subsets based on the "best" attribute. ID3 uses a greedy search algorithm, that is, after picking the "best" feature it never reconsiders the previous actions. Quinlan also developed an extension of the ID3 algorithm to address some issues related to the handling of continuous attributes, training data with missing attribute value etc. in the C4.5 algorithm. The procedure for building a decision tree, ID3 algorithm, C4.5 and comparison between these two algorithms is explained in the following chapter.

CHAPTER 2. BACKGROUND

2.1 Decision Tree Classifiers

A decision tree is a graphical approach of classifying a certain dataset. Decision trees clearly show how to reach from an arbitrary starting node to a particular leaf node or equivalently the decisions, evaluating different tests along the way. A decision tree is typically constructed by dividing a complex data set into various simpler partitions based on the outcome of simple tests conducted on one attribute at a time. At each branch node or decision node a test is constructed in such a manner that it defines the next partition of the data set. This process is repeated until each leaf node represents a set with all samples belonging to one particular class.

In order to construct a decision tree that correctly classifies the given training data set, has a high classification performance on the test data and yet is simple, we need to look for attributes that are most informative. Such attributes help us find a decision tree having minimum nodes and yet give maximum information. To illustrate this, let's take a simple data set given in Table 2.1.

X	Y	F
0	0	0
1	0	1
0	1	0
1	1	1

Table 2.1: Example of training data set [5]

In this example, there are two features/attributes: X and Y, two classes: 1 and 0 and four instances described in terms of features and classified into two classes. We can build more than one decision tree from this data. Two of them are illustrated in Figure 2.1

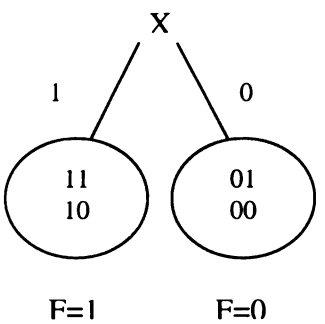
In Figure 2.1 (a), the test on feature X is used first to obtain the decision tree. The dataset is branched depending on the value of X into two subsets. Since the instances from each of these subsets are from the same class, we don't need any further tests and we have the decision tree classifier. On the other hand, consider Figure 2.1 (b). If we start with testing Y first, we have two different subsets according to values of Y but here we need to have one more test to obtain the leaf-nodes. This is because after the test on feature Y, the instances in each of the two subsets are not from the same class. Consequently a single test based on Y is not sufficient and a further test on X is required to construct the decision tree as shown in Figure 2.1 (b).

2.2 Information Theory Concepts

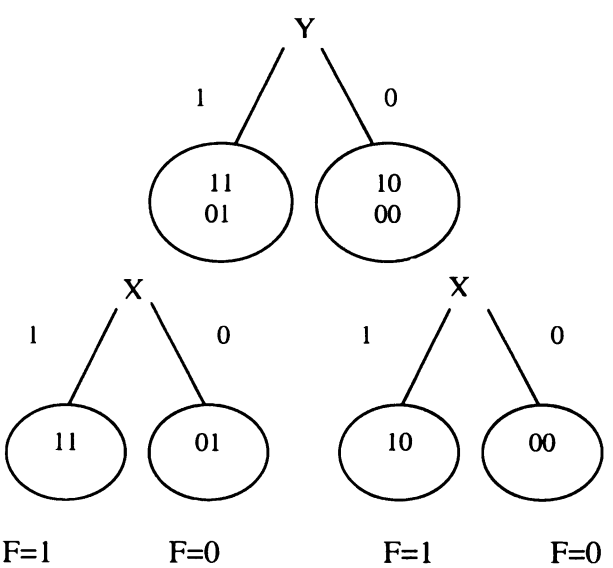
In the decision tree, each node is a non-categorical attribute and each branch corresponds to a possible value of the attribute. A leaf of the tree gives the expected class for instances fulfilling the path from root node of the decision tree to that leaf. To obtain the tests that give maximum information in classification, we use Shannon's information theory concept as the basis [5]. A pioneer of information theory, Shannon developed a measure of information contained in a message. It is now widely used in computer science, communications, information processing and in data compression and storage.

The information conveyed by a message is related to the "surprise" value

associated with the message [6]. For example, assume we have two cricket teams playing in a championship. One of them (Team A) is really good and is the defending champion while other one (Team B) is an inexperienced new team. Now it will not be surprising to hear that team B loses to Team A. But if the result is the contrary i.e. Team A loses to Team B,

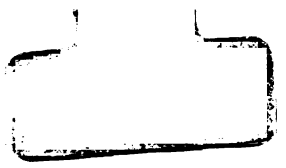


(a)



(b)

Figure 2.1: Two possible decision trees generated for the example in Table 2.1[5].



then that is certainly a surprising result. Now these two results carry similar words and in terms of message transmission, they are equivalent. Yet the surprise values associated with the two messages are different because one of the messages is more probable while the other is not. So the information contained in the message depends on *a priori* expectations. Hence smaller the prior probability of the message, more surprising is the message. In other words, the more probable the message, the less information it conveys [6].

Shannon [5] defines the information content of a message as a function of the probability of occurrence of the message. Given a set of n permissible symbols in a message, each having probability of occurrence $p_1, p_2, p_3, \ldots, p_n$, the information contained in a message n_i of probability p_i is

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

For example, information content conveyed by the outcome of a fair six-sided dice roll is $-\log_2 \frac{1}{6} = 2.5850$ bits, which is the maximum amount of information in a dice roll. Hence the information in the message comprising above n symbols, is $I(p_1, p_2, p_3, \ldots, p_n)$

$$I(p_1, p_2, p_3, \ldots, p_n) = -\sum_{i=1}^{n} \log_2(p_i) \text{ bits}$$

Also, the average or expected value of the information can be defined as $H(p_1, p_2, p_3, \ldots, p_n)$

$$H(p_1, p_2, p_3, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

This can also be recognized as the entropy of the message. For example, if we imagine a randomly selected data vector S_i from a set S belonging to say class C_i, then this assumption has probability equal to

$$\frac{freq(C_i, S)}{|S|}$$

where $|S|$ is the cardinality of the set S.

Correspondingly the information conveyed is [7]

$$I(S_i) = -\log_2\left(\frac{freq(C_i, S)}{|S|}\right) \text{ bits.}$$

The expected value of information is given by summing over k classes weighted by their frequencies in S, which is

$$H(S) = -\sum_{i=1}^{k} \frac{freq(C_i, S)}{|S|} \times -\log_2\left(\frac{freq(C_i, S)}{|S|}\right) \text{bits.}$$

The quantity computed in above equation is called as *Entropy* of the set S and it measures the average amount of information needed to identify the class of an instance in set S [7]. Now if we consider a test X on set S with n outcomes such that S is partitioned into n subsets, the expected information is the weighted sum over the subsets, as

$$H_X(S) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} \times I(S_i)$$

and the information gain by applying test X on set S is

$$gain(X) = H(S) - H_X(S)$$

2.3 ID3 Algorithm

This concept of information theory is used in the ID3 algorithm to build a classifier from an arbitrary collection of objects. The ID3 family of algorithms infers the decision tree by growing them from top to bottom greedily selecting the next best attribute for each decision branch added to the tree. The letters ID stands for 'Iterative dichotomizer' indicating the fact that the algorithm builds a binary tree iteratively. Let S represent an arbitrary collection of objects. If S is empty or contains objects of only one class, then the simplest decision tree is to mark the set with its class. Otherwise let T be the test on an attribute O with possible outcomes $O_1, O_2, O_3, ..., O_w$. Equivalently, T produces a partition $\{ S_1, S_2, S_3, ..., S_w \}$ of S with S_i containing those objects having outcome O_i. Figure 2.2 shows the graphical representation of this scenario.

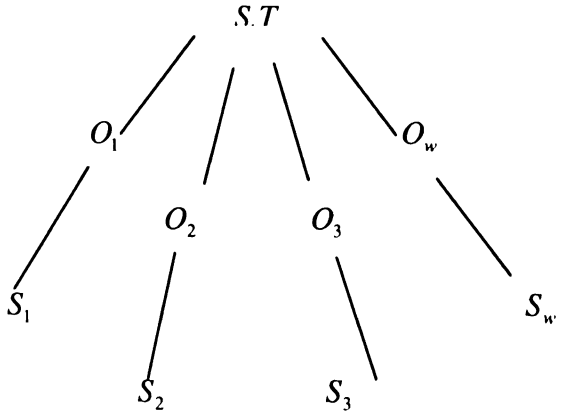


Figure 2.2: A tree structuring of the objects in S [1].

If we could replace each subset S_i in this Figure 2.2 by a decision tree for S_i, then the result would be a decision tree for all of S. Every time set S is partitioned, we get a

number of S_i s, each smaller than *S*. This divide-and-conquer strategy finally gives a decision tree, that partitions S into exclusive sub trees. The choice of test T is crucial to make the decision tree minimal. The first induction programs in the ID series used an intuitive evaluation function [1]. However the ID3 algorithm adopted an information theory based and more methodical approach that depends on two assumptions. Let *S* contain *p* objects of class *P* and *n* objects of class *N*. Based on this set *S*, Quinlan made two assumptions as follows:

1. Any correct decision tree will classify objects in set *S* in the same proportions as their representation in *S*. The probability of an arbitrary object belonging to class *P* is $\frac{p}{(p+n)}$ and to class N is $\frac{n}{(p+n)}$.

2. When a decision tree is used to classify an object, it returns a class. A decision tree can be regarded as a source of a message '*P*' or '*N*', with the expected value of information needed to generate this message given by,

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

In general, if attribute A with values $\{ A_1, A_2, A_3, \ldots, A_v \}$ is used as a test, set *S* is partitioned into $\{ S_1, S_2, S_3, \ldots, S_v \}$ where S_i contains those objects in *S* that have value A_i of *A*. Let S_i contain p_i objects of class *P* and n_i objects of class *N*. Then the expected value of information of class S_i is $H(p_i, n_i)$ and the expected value of information of the tree with attribute A as root is then obtained as the weighted average

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

where the weight for the i^{th} branch is the proportion of the objects in S that belong to S_i [1]. And the information gain in this partition is,

$$gain(A) = H(p,n) - H(A)$$

The gain criterion is then used to select an attribute that maximizes the mutual information gain between test A and the class i.e. ID3 tests all the possible partitions with all candidate attributes and chooses the attribute X to maximize *gain(X).* The attribute X that maximizes the gain is used to expand the node. This procedure is repeated recursively to form the decision trees for each of the subsets formed due to that partition.

This is illustrated with the help of training data set in Table 2.2 [1]. The Training set deals with the instances of weather conditions for playing golf. There are 14 instances in the training data set in Table 2.2 with four attributes/features represented by

1. Outlook={sunny, overcast, rain},
2. Temperature={hot, mild, cool},
3. Humidity={high, normal},
4. Windy={true, false}.

and two classes P for 'Play' and N for 'Don't Play'. Of these 14 instances, 9 belong to class P and 5 belong to class N. Hence the average information content of this data set S is

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

The information gain of an attribute/test at the root of current tree is the total information of the tree minus the amount of information needed to complete the classification after performing the test.

No	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

Table 2.2: Training Data Set [1].

The information needed to complete the classification can be computed as the weighted

average information in all the partitions, i.e. if we first partition S on the basis of one of the attributes into sets $\{S_1, S_2, S_3, \ldots, S_n\}$, then the information needed to classify an element of S becomes the weighted sum over all the partitions $\{S_1, S_2, S_3, \ldots, S_n\}$ is

$$H_X(S) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} \times H(S_i)$$

where X is the attribute chosen to partition S . In the golfing example, for the attribute 'Outlook' there are three values {sunny, overcast, rain}. These divide the training set into three subsets; five of the 14 objects having value for attribute 'Outlook' as 'Sunny' in first subset, five having value 'Rain' in second subset and four having value 'Overcast' in third subset. The graphical depiction of this partitioning is shown in Figure 2.3. For the first branch we can see that

$$p_1 = 2,\ n_1 = 3,\ I(p_1, n_1) = 0.971$$

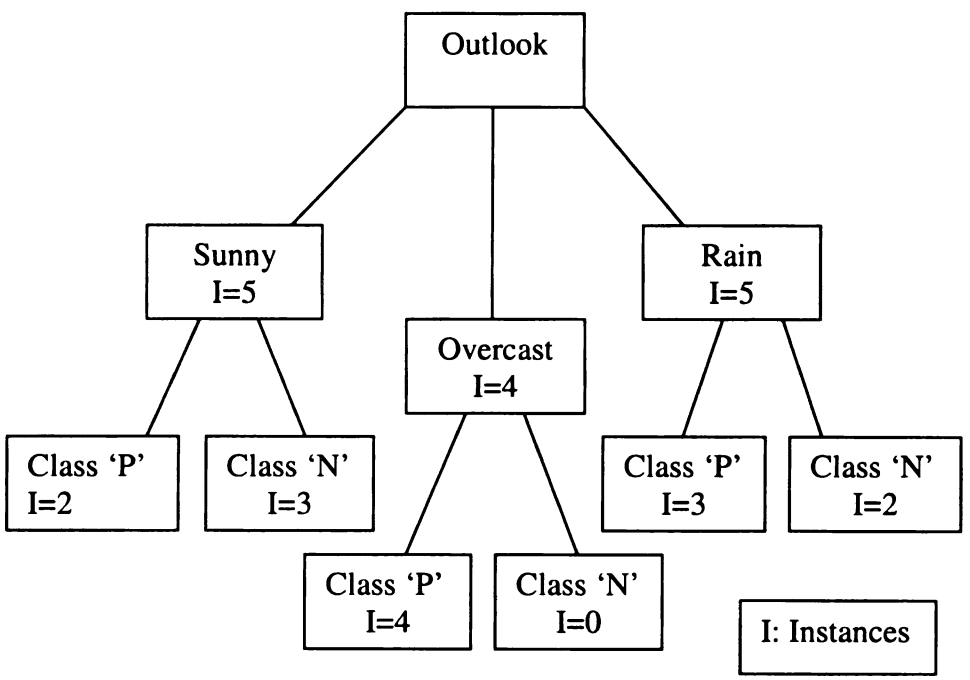


Figure 2.3: Distribution of attribute 'outlook'

and similarly for second and third branches,

$$p_2 = 4,\ n_2 = 0,\ I(p_2, n_2) = 0$$

$$p_3 = 3,\ n_3 = 2,\ I(p_3, n_3) = 0.971$$

The information content using 'Outlook' as the root attribute is

$$H(outlook, S) = \frac{5}{14} \times \left(-\frac{2}{5} \times \log_2\left(\frac{2}{5}\right) - \frac{3}{5} \times \log_2\left(\frac{3}{5}\right) \right) +$$

$$\frac{4}{14} \times \left(-\frac{4}{4} \times \log_2\left(\frac{4}{4}\right) - \frac{0}{4} \times \log_2\left(\frac{0}{4}\right) \right) +$$

$$\frac{5}{14} \times \left(-\frac{2}{5} \times \log_2\left(\frac{3}{5}\right) - \frac{2}{5} \times \log_2\left(\frac{2}{5}\right) \right)$$

$$= 0.694 \ bits$$

The information gain for attribute 'Outlook' is

$$Gain(outlook) = H(S) - H(outlook, S)$$

$$= 0.940\text{-}0.694$$

$$= 0.246.$$

Since *H(S)* is a constant, the information gain for each attribute is equivalent to minimizing the mutual information of attributes and the classes.

Similarly for other attributes, we can calculate the average information and gain.

Temperature:

$$H(temperature, S) = \frac{4}{14} \times \left(-\frac{2}{4} \times \log_2\left(\frac{2}{4}\right) - \frac{2}{4} \times \log_2\left(\frac{2}{4}\right) \right) +$$

$$\frac{6}{14}\times\left(-\frac{4}{6}\times\log_2\left(\frac{4}{6}\right)-\frac{2}{6}\times\log_2\left(\frac{2}{6}\right)\right)+\frac{4}{14}\times\left(-\frac{3}{4}\times\log_2\left(\frac{3}{4}\right)-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)\right)$$

$$= 0.911 \; bits$$

and

$$Gain(temperature) = H(S) - H(temperature, S)$$

$$= 0.940 - 0.911$$

$$= 0.029.$$

Humidity:

$$H(humidity, S) = \frac{7}{14}\times\left(-\frac{3}{7}\times\log_2\left(\frac{3}{7}\right)-\frac{4}{7}\times\log_2\left(\frac{4}{7}\right)\right)+$$

$$\frac{7}{14}\times\left(-\frac{6}{7}\times\log_2\left(\frac{6}{7}\right)-\frac{1}{7}\times\log_2\left(\frac{1}{7}\right)\right)$$

$$= 0.79 \; bits$$

and

$$Gain(humidity) = H(S) - H(humidity, S)$$

$$= 0.940 - 0.789$$

$$= 0.151.$$

Windy:

$$H(windy, S) = \frac{6}{14}\times\left(-\frac{3}{6}\times\log_2\left(\frac{3}{6}\right)-\frac{3}{6}\times\log_2\left(\frac{3}{6}\right)\right)+$$

$$\frac{8}{14}\times\left(-\frac{6}{8}\times\log_2\left(\frac{6}{8}\right)-\frac{2}{8}\times\log_2\left(\frac{2}{8}\right)\right)$$

$$= 0.892 \; bits$$

and

$$Gain(windy) = H(S) - H(windy, S)$$

$$= 0.940 - 0.892$$

$$= 0.048.$$

The tree forming method in ID3 chooses the attribute with highest information gain or lowest entropy as the test attribute for that particular node of the decision tree. In the example under consideration, attribute 'Outlook' would be chosen as the test at that node. In this case it is the root node of the entire decision tree. Then the training set *S* would be divided into subsets according to the values of attribute 'Outlook' and a decision tree for each subset would be found recursively using the same approach.

The information gain approach has natural bias for the attributes having many values over those with few values. It has a strong bias in favor of tests with many outcomes [7]. Assume that we add an attribute; let's say Date (e.g. 29 December 1994) that has a large number of possible values in the training data *S* in Table 2.2. It would have the highest information gain since every value of the 'Date' attribute perfectly predicts the target class value over the training data. If we apply the information gain approach then 'Date' attribute would be selected as the attribute for the root node and we would get a broader tree with depth one i.e. we would get a tree having large number of child nodes as the leaf nodes with no further branches. From the point of view of Test Data, however, such a decision tree would be useless despite being the perfect classifier for training data.

To compensate for this, Quinlan [7] suggested a gain ratio criterion. The gain

ratio normalizes the information of the attribute having many values by using split information, which is sensitive to how broadly, and uniformly the attribute splits the data [3]:

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

where A is the attribute with c different values and S is the data set with $\{ S_1, S_2, S_3, \ldots, S_c \}$ are c subsets due to partitioning by attribute A. This represents the potential information generated by dividing S into c subsets whereas the information gain measures the information relevant to classification that arises from the same division. The GainRatio definition uses the gain and split information as follows

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

GainRatio expresses the proportion of information generated by the split that is useful for classification. The gain ratio criterion selects the attribute whose gain ratio is highest. This criterion discourages selection of attributes with many values. For example, consider two attributes A and B where A has c different values while B is a boolean attribute splitting the training set exactly in half. The split information for A will be $\log_2 c$ while split information for B will be 1. If A and B have same information gain, then the GainRatio criterion will ensure that B is chosen and the bias associated with multi-valued attributes is removed.

Now in our example data set, the split informations and gain ratios for all the attributes are as follows:

1) **Outlook:**

$$SplitInformation(S, outlook) = -\frac{5}{14}\log_2\frac{5}{14} - \frac{4}{14}\log_2\frac{4}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

$$= 1.5774 \; bits.$$

Hence the gain ratio is

$$GainRatio(S, outlook) = \frac{Gain(S, outlook)}{SplitInformation(S, outlook)}$$

$$= \frac{0.246}{1.5774}$$

$$= 0.156.$$

2) **Temperature:**

$$SplitInformation(S, temperature) = -\frac{4}{14}\log_2\frac{4}{14} - \frac{6}{14}\log_2\frac{6}{14} - \frac{4}{14}\log_2\frac{4}{14}$$

$$= 1.5567 \; bits.$$

Hence the gain ratio is

$$GainRatio(S, temperature) = \frac{Gain(S, temperature)}{SplitInformation(S, temperature)}$$

$$= \frac{0.029}{1.5567}$$

$$= 0.0186.$$

3) **Humidity:**

$$SplitInformation(S, humidity) = -\frac{7}{14}\log_2\frac{7}{14} - \frac{7}{14}\log_2\frac{7}{14}$$

$$= 1 \; bits.$$

Hence the gain ratio is

$$GainRatio(S, humidity) = \frac{Gain(S, humidity)}{SplitInformation(S, humidity)}$$

$$= \frac{0.151}{1}$$

$$= 0.151.$$

4) Windy:

$$SplitInformation(S, windy) = -\frac{6}{14}\log_2\frac{6}{14} - \frac{8}{14}\log_2\frac{8}{14}$$

$$= 0.9852 \; bits.$$

Hence the gain ratio is

$$GainRatio(S, windy) = \frac{Gain(S, windy)}{SplitInformation(S, windy)}$$

$$= \frac{0.048}{0.9852}$$

$$= 0.0487.$$

It is seen from these numbers that once again the 'Outlook' would be chosen as the root node.

Using information theory as the basis, the ID3 algorithm constructs a decision tree in a top-down fashion. Based on the chosen attribute as test, the ID3 algorithm partitions the training set into disjoint subsets where all the examples in a subset have the same value for that attribute. The ID3 algorithm then selects the next attribute as a test at the current node and uses the test to partition that subset. Thus it recursively constructs a sub-tree for each subset and continues to divide until all the members of each leaf belongs to

the same class. The implementation of ID3 algorithm on the golfing training set given in Table 2.2 yields the decision tree as shown in Figure 2.4. The procedure and steps of ID3 algorithm are summarized in Figure 2.5.

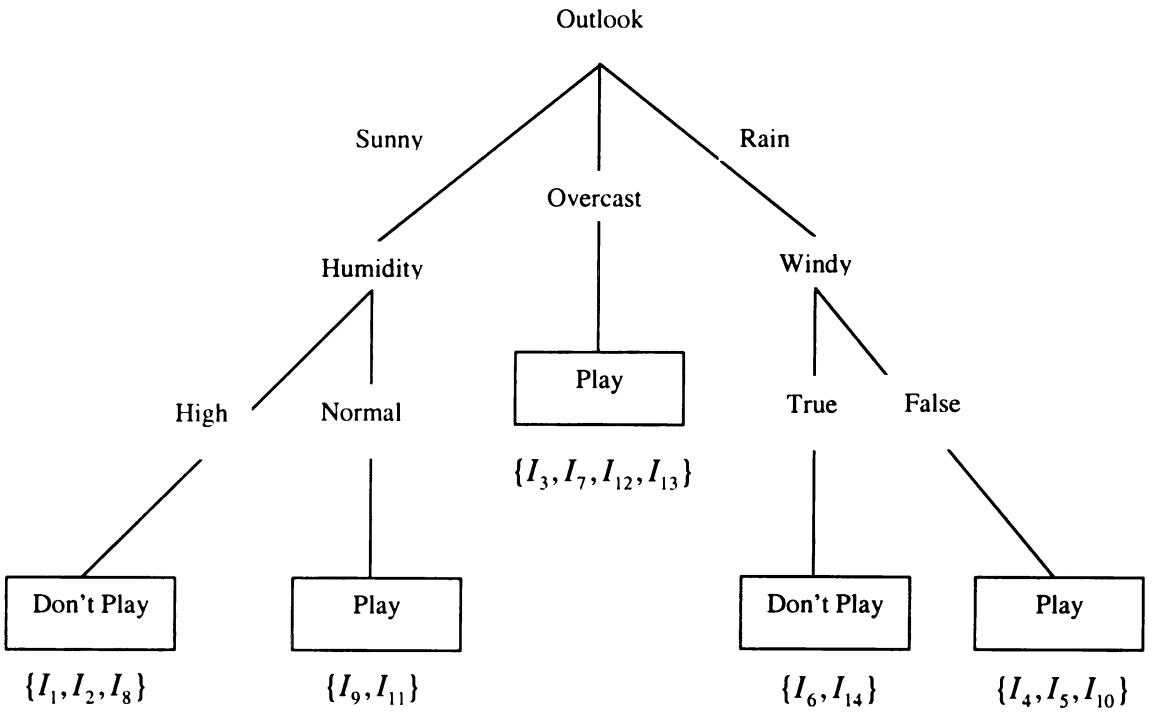


Figure 2.4: The Decision Tree for example in Table 2.2 [1]

A training data set S, attributes set A and classes' set C is input to the ID3 algorithm.

ID3 (S, A, C)
begin

If all entries in S are from the same class {

return a leaf node with that class name as a label.}

else if S is empty {
return a single node with value failure.}

else if A is empty {
return a single node with value as the most frequently occurring class in that training set.}

else{

let A_j be the attribute from A that has the highest Gain and best classifies the 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 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 as

ID3 (S_{vi}, A- { A_j }, C)}

end
return Root

Figure 2.5: ID3 Algorithm

2.4 C4.5

The basic ID3 Algorithm has many limitations. ID3 maintains only a single current hypothesis and it does not search for other possible trees once it has chosen the test attribute and does not perform any backtracking. The ID3 algorithm also cannot handle continuous data or data with missing attribute values. Quinlan [7] introduced the C4.5 as an extension to the original ID3 Algorithm to address some of the above issues.

C4.5 uses threshold values while handling continuous attributes. The training data is first sorted according to values of the continuous attribute, which is being considered. Assume we have a training data set *S* and attribute *A* having finite number of continuous values sorted as $\{ A_1, A_2, A_3, \ldots, A_n \}$. Any threshold value lying in between A_i and A_{i+1} will have the same effect as dividing the instances between two parts one having values for that attribute in between $\{ A_1, A_2, A_3, \ldots, A_i \}$ and other part having values as $\{ A_{i+1}, A_{i+2}, A_{i+3}, \ldots, A_n \}$. So only (n-1) possible partitions have to be examined. Generally the midpoint of any interval is chosen as the representative threshold i.e.

$$\text{midpoint} = \frac{A_i + A_{i+1}}{2}$$

C4.5 chooses the largest value of *A* in the entire training set that is less than the midpoint given above, rather than the midpoint itself so that the threshold values found in the decision tree actually occur in the training set [7]. 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} [7], the information gain for each partition is calculated resulting in the best partition at 'Humidity'=75 and the test for 'Humidity' becomes

'Humidity'<= 75 and >75.

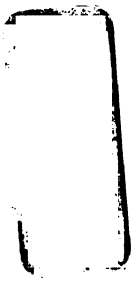
In the case of missing or unknown attribute values, C4.5 builds the decision tree by evaluating the gain or gain ratio, for that attribute using only the records having defined values for that attribute. Then the record with missing attribute value can be classified by estimating the probability of various possible results. For example, in the golfing training data set, an instance with 'Outlook' attribute as 'Sunny' and 'Humidity' as 'Unknown' follows the decision tree branch of 'Sunny' from the root node 'Outlook'. At this node, there are two branches with 'Humidity' as {High, Normal}. The branch with 'Humidity' = 'Normal' has two examples with leaf node classification as 'Play' and the branch 'Humidity' = 'High' has three examples with classification as 'Don't Play'. Thus the algorithm gives a decision for the instance with unknown humidity as 'Play' with probability 0.4 and 'Don't Play' with probability 0.6.

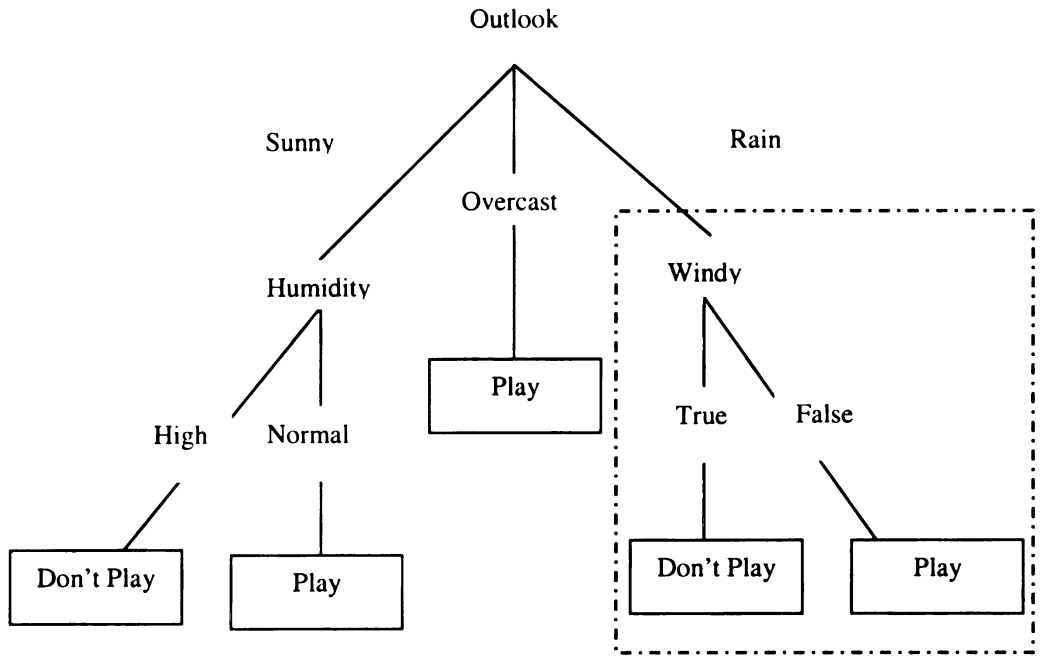
Another feature of C4.5 is pruning. A decision tree is pruned by replacing a whole sub-tree by a leaf node. This replacement is done when a decision rule identifies greater error rate in a sub-tree than in a single leaf. Pruning also helps in reducing noise effects and the complexity of decision trees. For example, we have the decision as shown in Figure 2.3 based on the training data from Table 2.2. Now consider the test data as shown in Table 2.3. Note that the last three records contradict the classifier in Figure 2.3, that is, if we follow the decision tree in Figure 2.3, these three instances would be misclassified as 'Don't Play'. Hence we have an error in 3 instances considering test and training data together. Now if we prune the tree and have the sub-tree for attribute 'Windy' changed to a leaf node 'Play', only two instances in training data which are classified as 'Don't Play'

would be misclassified and the error would be reduced to 2 instances considering test and training data together. The pruned decision tree is shown in Figure 2.6. As explained earlier, in the pruned decision tree a leaf node of 'Play' has replaced the sub tree for attribute 'Windy'.

No	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

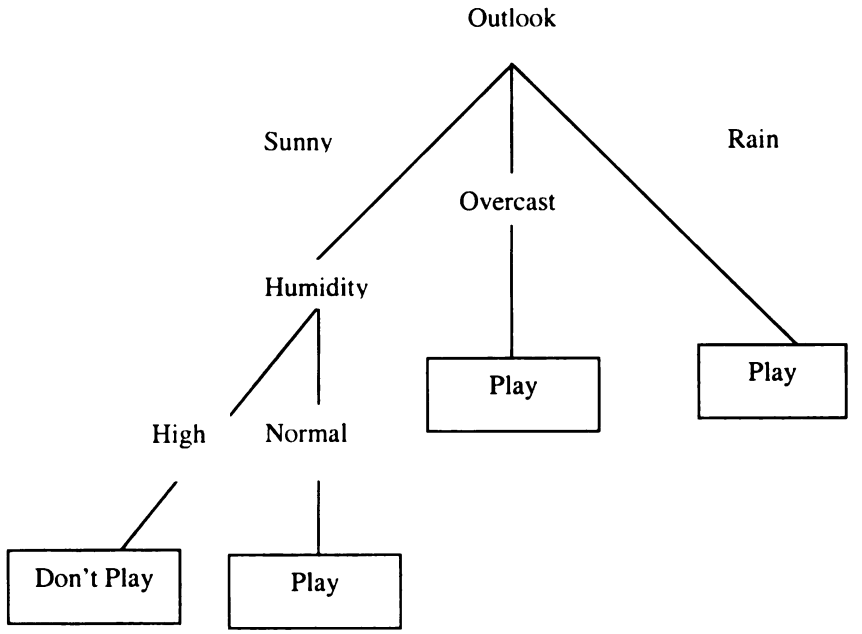
Table 2.3: Test Data





(a) Original decision tree

(b)



(b) Pruned decision tree

Figure 2.6: Pruned Decision Tree

CHAPTER 3. PROPOSED 2-D ID3 ALGORITHMS

3.1 Induction

The ID3 algorithm explained in Chapter 2 uses one attribute for each test, i.e. a univariate test, at a particular tree node to determine the sample space partition. Since the ID3 algorithm uses univariate tests, it can only have orthogonal decision boundaries. Consequently the correlation between any two attributes is not considered while building the decision tree. Thus in case of data distributions where the method using the correlation between attributes would have resulted in better classifier, the ID3 algorithm tends to build complicated and over-fitted decision boundaries to partition the input space.

To address this issue, the thesis proposes two 2 dimensional ID3 algorithms which use the combination of two attributes to decide a test at any particular node. The original 1-D ID3 algorithm can be extended to use two attributes for each test so as to take into account the correlation between attributes. This extension of the ID3 algorithm is the basis of this thesis. Two approaches for modifying the ID3 algorithm are proposed and explained in this chapter. The first approach uses two attributes together at a time. This approach is developed by simply modifying the 1-D ID3 algorithm to consider two attributes and their joint entropy while deciding a test. The second approach uses linear combination of two attributes to compute all possible decision boundaries and corresponding entropies. In both of these approaches, the bivariate test with minimum entropy is chosen at a particular node.

The basic ID3 algorithm uses entropies of attributes to choose the most optimal test attribute at a node. The entropy of a random variable is a measure of uncertainty related to that variable. The *entropy* of a random variable X with pdf $p(x)$ is defined by,

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

where logarithm to the base 2 is used to express entropy in units of bits[8]. For example, the entropy of a fair coin toss is 1 bit.

The definition of entropy can be extended to define the joint entropy of a pair of two random variables. The *joint entropy* of a pair of discrete random variables (X,Y) with a joint density function $p(x, y)$ is defined as

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y),$$

which can also be expressed as $H(X,Y) = -E\{\log_2(X,Y)\}$ where E represents the expectation operator [8]. The joint entropy is used as the cost function in the modified algorithm.

The properties of entropy are enlisted below:

1. $H(X) \geq 0$
2. $H_b(X) = (\log_b a) H_a(X)$ where a and b are different logarithmic bases. This property shows that the entropy can be changed from one base to another by multiplying by an appropriate factor.
3. $H(X,Y) = H(X) + H(Y)$, if random variables X and Y are statistically independent.

4. $H(X,Y) < H(X) + H(Y)$, if random variables X and Y are statistically dependent.
5. $H(X,Y) = H(X) + H(Y \mid X)$ where $H(Y \mid X)$ is the conditional entropy of Y with respect to X.

3.2 Joint Entropy based ID3 Algorithm

Assume an arbitrary sample set S with k attributes A, B, C, D, ..., K , each attribute having different values. For example, let attribute A take v different values $\{ A_1, A_2, A_3, \ldots, A_v \}$, attribute B take w different values $\{ B_1, B_2, B_3, \ldots, B_w \}$ and so on. The samples space consists of two classes P and N with p elements in class P and n elements in class N. Therefore the average information in the entire set is expressed as

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

In accordance with the 1-D ID3 algorithm, if attribute A is used to partition the set S into $\{ S_1, S_2, S_3, \ldots, S_v \}$ where S_i contains p_i objects of class P and n_i objects of class N, the average information in S_i is $H(p_i, n_i)$ and the average information in the tree with attribute A as root is then obtained as the weighted average or entropy

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

where the weight for the i^{th} branch is the proportion of the objects in S that belong to S_i [1]. The corresponding information gain is

$$gain(A) = H(p,n) - H(A)$$

The attribute having maximum information gain or minimum entropy is chosen as a test attribute. This is the original ID3 algorithm.

Case 1: Dependent attributes

In the proposed extension, we consider two attributes A and B with values $\{A_1, A_2, A_3, \ldots, A_v\}$ and $\{B_1, B_2, B_3, \ldots, B_w\}$ respectively at each node. So, the total number of combinations of their values is $\{v \times w\} = vw$. The sample set S is partitioned using this pair of attributes into $\{S_1, S_2, S_3, \ldots, S_{vw}\}$ where S_{ij} contains those objects from S having values A_i for attribute A and value B_j for attribute B. S_{ij} contains p_{ij} objects of class P and n_{ij} objects of class N. Therefore the information contained in the sub tree S_{ij} is $H(p_{ij}, n_{ij})$ and the average information in pair AB is the weighted average which is also the joint entropy of the pair (A,B).

$$H(A,B) = \sum_{i=1}^{v} \sum_{j=1}^{w} \frac{p_{ij} + n_{ij}}{p+n} I(p_{ij}, n_{ij}),$$

where the weight for the ij^{th} branch is the proportion of objects in S that belong to S_{ij}. The attribute pair having minimum joint entropy is then chosen as the test pair at that particular node.

Case 2: Independent attributes

In case 1, the attributes were considered to be dependent. If the samples set has k independent attributes A, B, C, D, ..., K; the joint entropy of any two attributes will be

sum of individual attributes. Foe example, consider attribute A and B then the joint entropy for this pair of attributes $H(A,B)$ is,

$$H(A,B) = H(A) + H(B),$$

where $H(A)$ and $H(B)$ are entropies of attribute A and B respectively. The attribute pair having lowest joint entropy should be chosen as the test pair at a node. In this case the joint entropies are sums of pair of individual attribute entropies. So logically, the attribute pair having the lowest joint entropy will have the two attributes with two of the lowest individual entropies. This case is similar to the 1-D ID3 algorithm with the only difference being the number of attributes chosen at each test. In the 1-D ID3 algorithm, the attribute having lowest individual entropy is chosen. In this case, two attributes with lowest two individual entropies are chosen. Two examples are used to illustrate this new algorithm.

Example 1:

Consider a simple training set of the XOR gate. The dataset S is given in Table 3.1.

X	Y	Class
0	0	0
0	1	1
1	0	1
1	1	0

Table 3.1: Example dataset XOR gate

The sample space has two attributes X and Y with discrete values {0,1} and {0,1} respectively. The set is divided into two classes, class 0 and class 1. The decision tree generated using all the three algorithms i.e. ID3 algorithm, C4.5 and the modified ID3 algorithm are discussed below.

The average information for this dataset is

$$H(T) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1bit$$

In the 1-D single attribute ID3 algorithm,

1) For attribute X,

$$H(X,S) = \frac{2}{4}\times\left(-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)\right)+\frac{2}{4}\times\left(-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)\right)$$

$$= 1\ bit.$$

And

$$Gain(X) = 0$$

2) For attribute Y

$$H(Y,S) = \frac{2}{4}\times\left(-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)\right)+\frac{2}{4}\times\left(-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)-\frac{1}{4}\times\log_2\left(\frac{1}{4}\right)\right)$$

$$=1\ bit$$

And

$Gain(Y) = 0$

Since the gain is zero for both attributes, any one attribute can be used as a test attribute at the root node. The decision tree using the 1-D ID3 algorithm is shown in Figure 3.1.

Now using the modified ID3 algorithm the results are as follows,

$$H(X,Y) = \frac{1}{4}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) + \frac{1}{4}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) + \frac{1}{4}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) + \frac{1}{4}\left(-\frac{1}{1}\log_2\frac{1}{1}\right)$$

$=0$ *bit.*

The corresponding tree using the two-attribute ID3 algorithm is shown in Figure 3.2. The steps involved in the procedure are explained in Figure 3.7.

Decision trees can be characterized by the number of features they use for test at a node. Decision trees that are limited to testing a single feature at a node are usually much larger than trees that allow multiple-feature testing at a node. The univariate test can split the sample space with only orthogonal splits. The joint entropy algorithm explained above also results in orthogonal splits. However the depth of the tree is considerably smaller than that of the tree using single attribute ID3 algorithm.

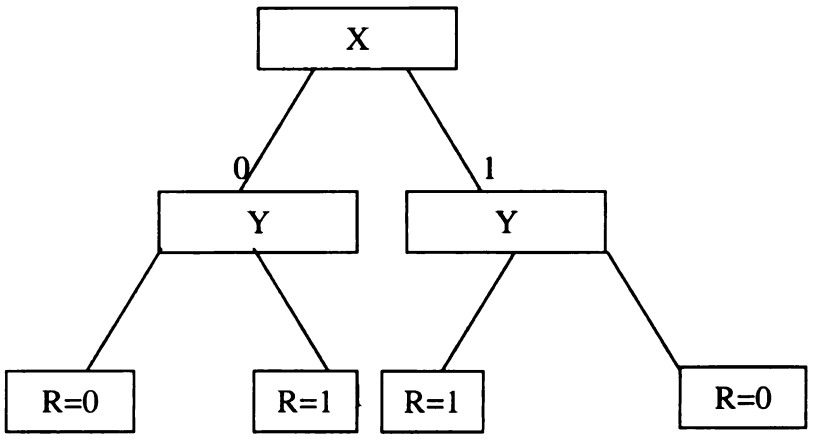


Figure 3.1: Decision Tree for dataset S using ID3 algorithm

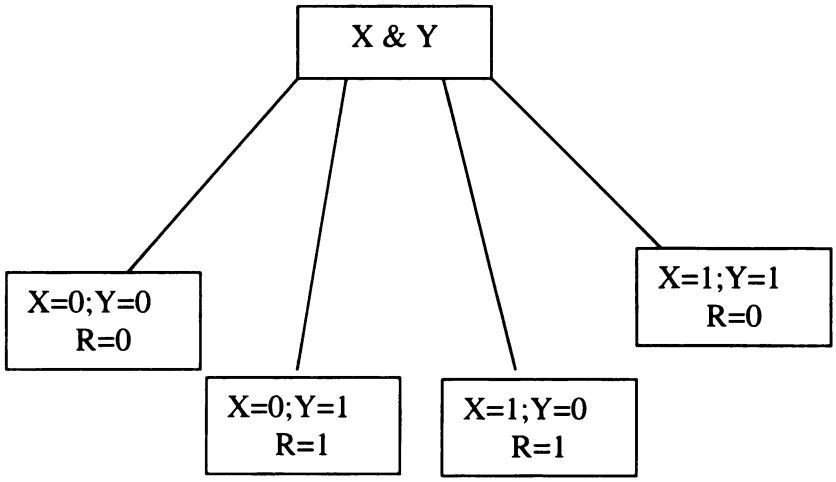


Figure 3.2: Decision Tree for Dataset S using modified ID3 algorithm

X	Y	Class
1	1	0
1.5	0.5	0
0.5	2.5	0
0.5	0.5	0
1.5	1.5	0
1	2	0
1	4	1
2	3	1
2.5	3	1
3	2	1
3.5	3.5	1
1	3	1

Table 3.2: Example data set approximating the hyper plane boundary x+y <=3.5

Example 2:

Consider the two-dimensional data set shown in Table 3.2. Figure 3.3 shows the distribution of data in the corresponding two-dimensional sample space along with the decision boundaries. The optimal decision boundary is the dash-dot line described by the equation $y + x = 3.5$, where the samples with $y + x \leq 3.5$ are classified as class '0' and samples with $y + x > 3.5$ are classified as class '1'. Solid lines parallel to X and Y axes show the decision boundary generated by single attribute ID3 algorithm. The ID3 algorithm results in two orthogonal splits on X and Y axes giving the decision tree as shown in Figure 3.4.

The joint entropy algorithm also uses orthogonal splits as shown in Figure 3.5 but it divides the sample space into four different sub spaces while the univariate ID3 divides the sample space into two subspaces as shown in Figure 3.3.

The dataset in Example 2 given in Table 3.2 illustrates the well-known problem that a univariate test can split the sample space using a boundary that is orthogonal to feature axes. The multivariate test used in joint entropy algorithm simply uses two features and considers the joint entropy as a test. Intuitively this result is not expected to be much different from the result obtained using the 1-D ID3 algorithm as shown in Figures 3.5 and 3.6. However, one can envision several data distributions where a 2-D ID3 algorithm can outperform a 1-D ID3. To illustrate this, consider the training dataset and test dataset with data distribution shown in Figure 3.8 (a), where the two attributes are highly correlated.

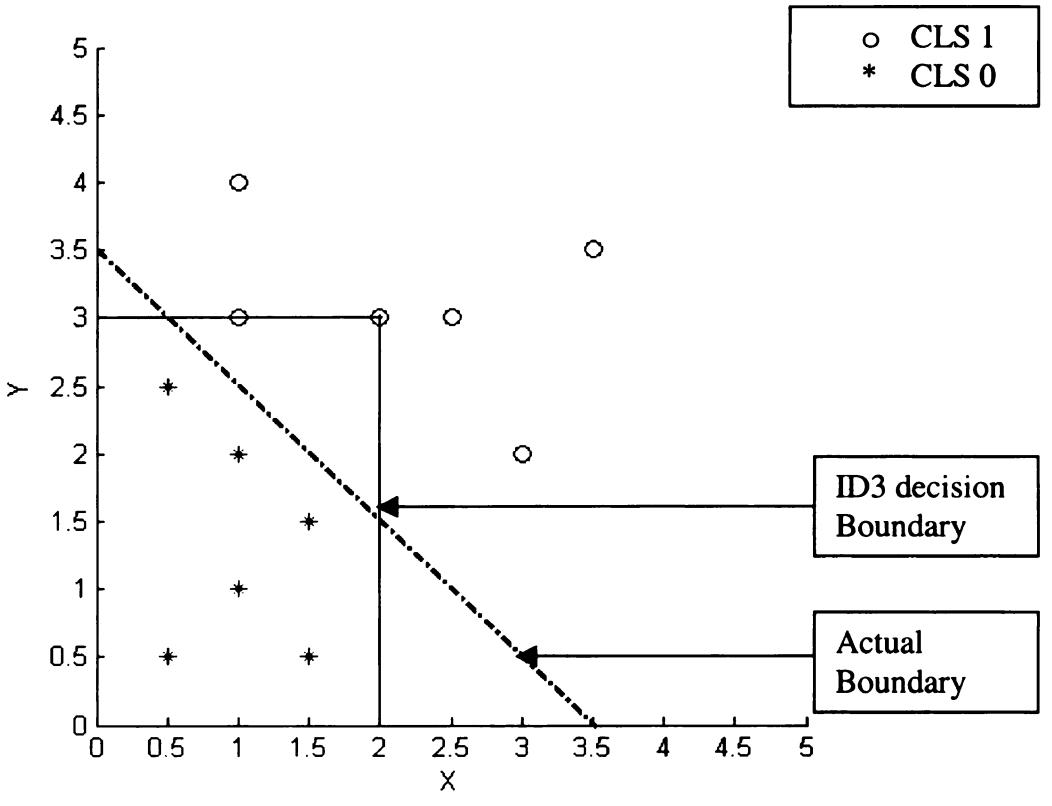


Figure 3.3: The example sample space; the actual decision boundary and ID3 decision boundary.

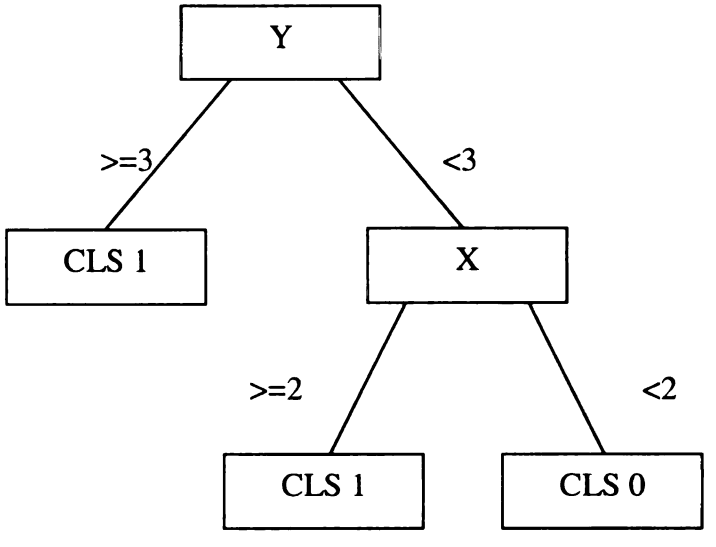


Figure 3.4: Decision Tree using ID3 algorithm.

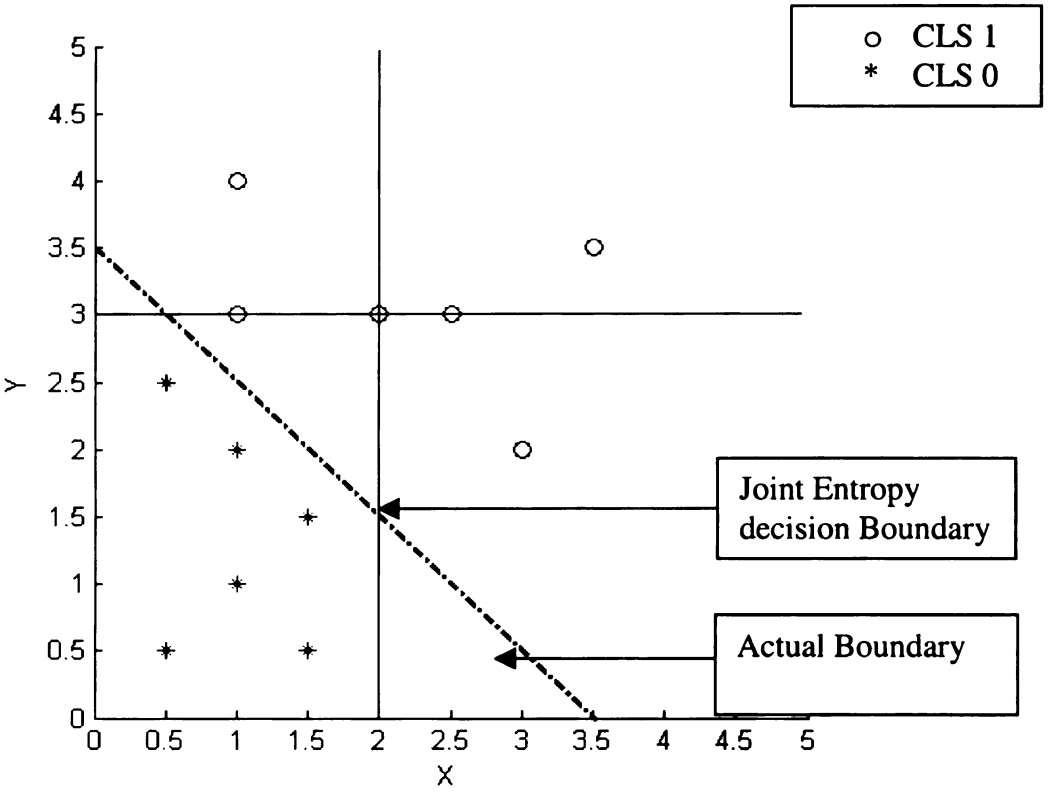


Figure 3.5: Sample space; actual decision boundary and JE algorithm decision boundary

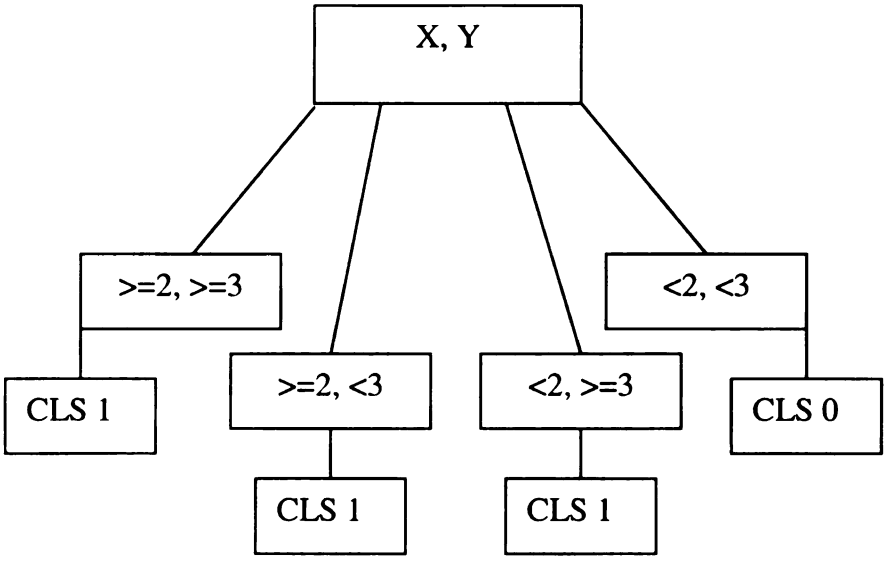


Figure 3.6: Decision Tree obtained using Joint Entropy algorithm

Total attribute combination set AB is determined by all possible pair wise combinations of attributes.

The training data set S, attributes set A, classes C and total attributes combinations set AB are used as input to the Joint Entropy algorithm (JE).

JE (S, A, C, AB)
begin

If all entries in S are from the same class {
return a leaf node with that class name as a label.}

else if S is empty {
return a single node with value failure.}

else if A is empty {
return a single node with value as the most frequently occurring class in that training set.}

else{

let (A_i, B_j) *be the attribute pair from AB that has the lowest joint entropy and best classifies the training set S.*

Then the decision attribute pair at the Root Node is (A_i, B_j). *For each combination of values of* A_i *and* B_j, $A_i = v_i$, $B_j = w_j$ *as one of the combination, add a new branch below Root corresponding to the test* $A_i = v_i$, $B_j = w_j$.

Let S_{viwj} *be the subset of S that has value* v_i *for* A_i *and* w_j *for* B_j
If S_{viwj} *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 as
JE (S_{viwj}, *A, C, AB-* $\{(A_i, B_j)\}$*)}*

End
return Root

Figure 3.7: Joint Entropy ID3 Algorithm

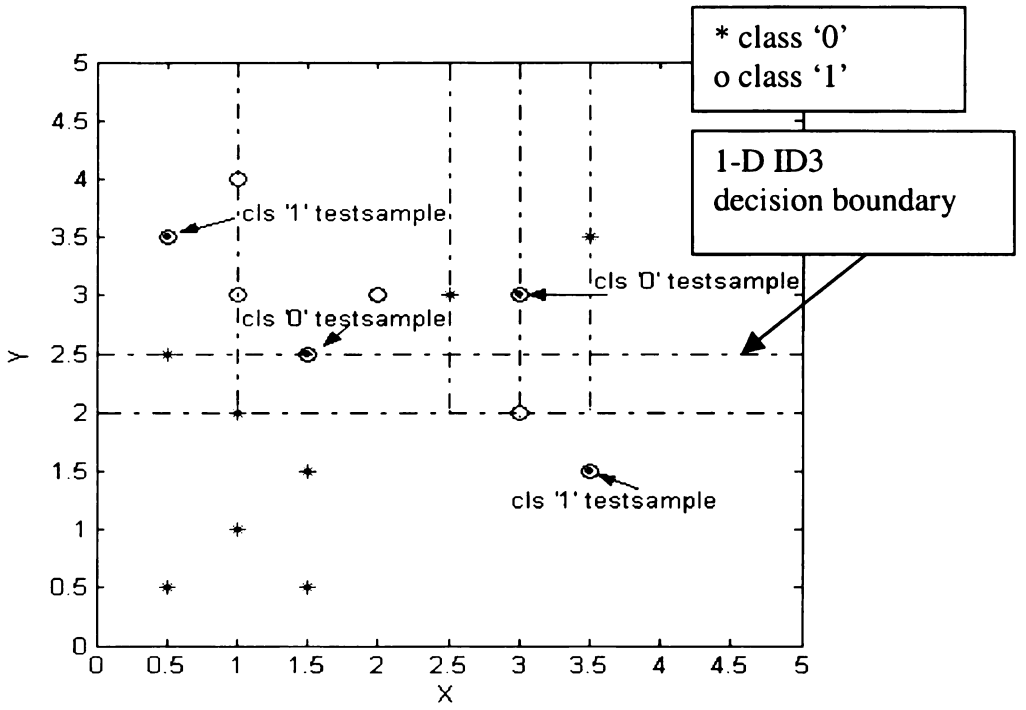


Figure 3.8 (a): Training and test dataset for illustration

The 1-D ID3 algorithm gives a complicated and over-fitted decision boundary for this data whereas the 2-D ID3 algorithm gives much simpler results with no over fitting. On applying these rules to the test data, 1-D ID3 rules result in misclassification of all the four test samples while 2-D ID3 algorithm properly classifies all the samples. Hence, it can be concluded that the 2-D ID3 algorithm is more powerful than 1-D ID3 in terms of complexity and decision boundaries it can generate and accuracy of classification. The decision trees using 1-D ID3 algorithm and 2-D ID3 algorithm are presented in Figure 3.8 (b) and (c).

Our second approach proposed in this thesis considers pair-wise linear combinations of features while looking for the test at a node and consequently generates non-orthogonal linear decision boundaries. The next section explains the Linear

Combination algorithm in detail.

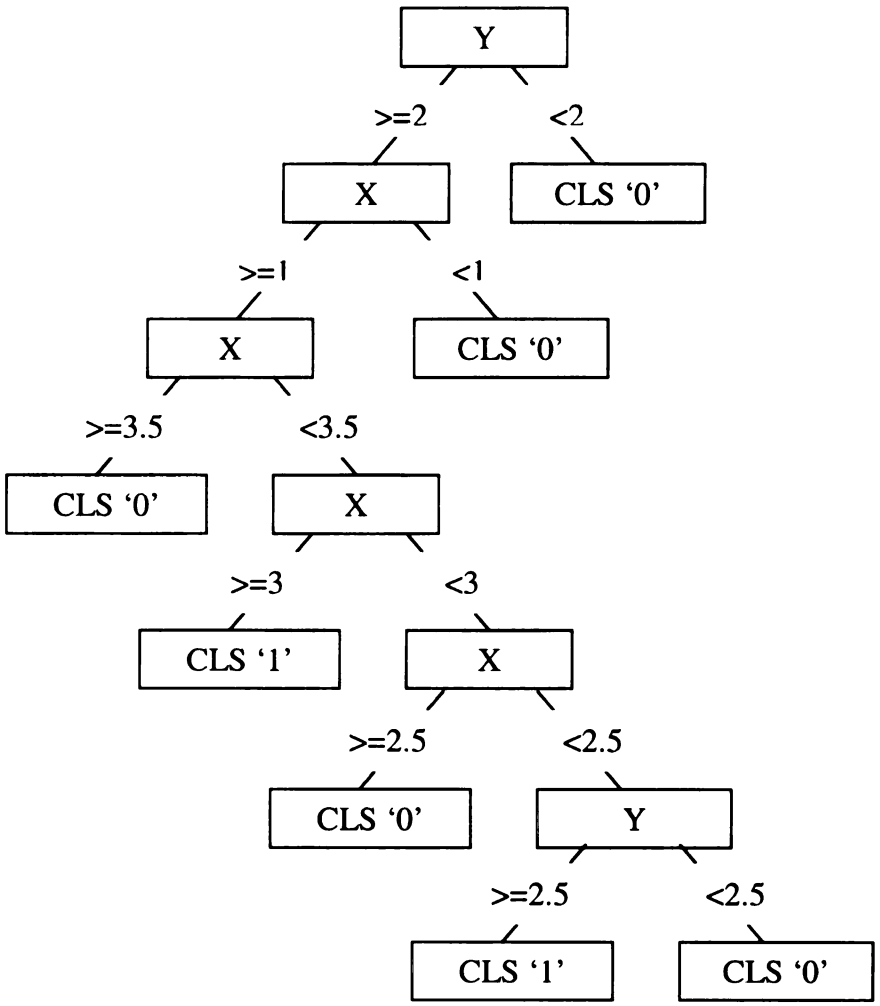


Figure 3.8 (b): Decision Tree using 1-D ID3 algorithm.

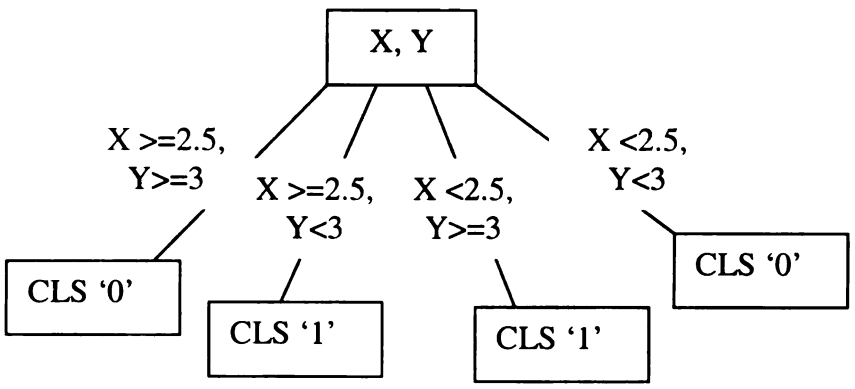


Figure 3.8 (c): Decision Trees using 2-D ID3 algorithm

3.3 Linear Combination Algorithm

The second approach, which has been developed to address the issue of poor performance of the ID3 algorithm due to the orthogonal splits, also uses information gain or entropy as the criterion for each test. The proposed algorithm is explained for the two-class problem, combines pairs of features similar to those considered in the Joint entropy algorithm. This algorithm looks for linear combination of the features giving rise to partitions similar to the actual boundary shown in Figures 3.3 and 3.5.

For each two-dimensional sample space corresponding to each pair of features, the intercept points used to estimate the boundary are computed. To get these intercept points, the bounding rectangle enclosing all the data points in each class is determined by computing the minimum and maximum values for the two features considered. Then the intercept points defining the linear boundary are obtained from the vertices of the bounding rectangles and their corresponding projection on both the feature axes. This is illustrated next using two cases,

1. Datasets having two attributes
2. Datasets having more than two attributes

Case 1: Data vectors with 2 attributes (2-D sample space)

Consider the same training sample set given in Table 3.2. There are two attributes, two classes, and a total of 12 samples, 6 from each class. Considering the samples from each class, the bounding rectangle is determined by computing the minimum and maximum values of each feature. Figure 3.9 shows the bounding rectangle and the

sample space.

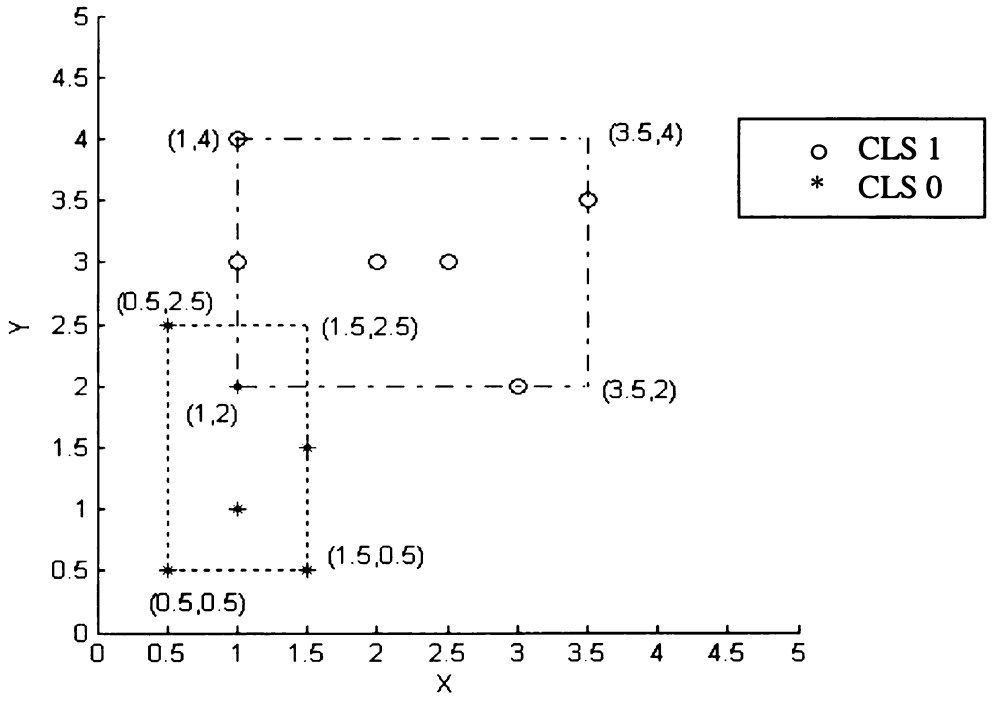


Figure 3.9: Training set; bounding rectangles for two classes; vertices of bounding rectangles.

These eight vertices and their projections on both the feature axes, X and Y, give the intercept points to be used in the next stage of the algorithm. The minimum and maximum values for feature X and Y in both the classes are given in Table 3.3. Hence, the intercept points defining the partition for this particular example are given in Table 3.3 and Table 3.4.

Candidate decision boundaries are obtained by drawing a line through every pair of points and the entropy for each partition is computed. This is repeated for all the pairs of features and the partition giving the lowest entropy is chosen to be the test at a node.

The above process at each node is repeated until the data is properly classified. For example, for the above dataset the line which best classified the data is X + Y = 3. The samples that have (X+Y) value less than or equal to 3 are classified as class '0' and samples having value greater than 3 are classified as class '1'. The decision tree is given in Figure 3.10 and the distribution of sample space with respect to this decision boundary is shown in Figure 3.11.

Values	CLASS 0	CLASS 1
Minimum value for feature X	0.5	1
Minimum value for feature Y	0.5	2
Maximum value for feature X	1.5	3.5
Maximum value for feature Y	2.5	4

Table 3.3: Minimum and Maximum values for both the features.

	CLASS 0	CLASS 1
Vertices of bounding rectangle	(0.5,0.5); (1.5,0.5); (1.5,2.5); (0.5,2.5).	(1,2); (3.5,2); (3.5,4); (1,4).
Projections on X feature axis	(0.5,0); (1.5,0).	(1,0); (3.5,0).
Projections on Y feature axis	(0,0.5); (0,2.5).	(0,2); (0,4).

Table 3.4: Intercept points to be considered for next stage of Linear Combination algorithm.

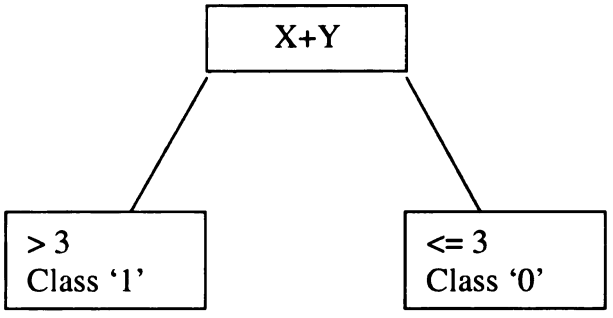


Figure 3.10: Decision Tree for the dataset in Table 3.2 using Linear Combination Algorithm

Case 2: data vectors with 3 attributes (3-D sample space)

In the dataset considered earlier, there are only two attributes. So, only one pair wise combination is possible. For the dataset having dimensionality or number of attributes higher than 2, we need to take all the possible pair wise combinations of all the attributes, treat each pair individually i.e. consider all the resulting two dimensional datasets and the corresponding potential partition boundaries in each of these samples spaces.

Consider the dataset given in Table 3.5 for illustration of this algorithm. The distribution of the sample space is given in Figure 3.12. This training dataset has two classes, 3 attributes and 12 samples, 6 from each class. There are 3 possible pair-wise combinations of attributes; attribute 1 and 2, attribute 1 and 3, attribute 2 and 3.

For each of these pairs, the bounding rectangles and hence the intercept points as explained earlier are calculated. The intercept points are given in Table 3.6. Lines passing through all pairs of intercept points are considered and the corresponding entropy is computed. The line having minimum entropy is then selected as the root node test and the process is repeated for each sub-tree. For this particular example the root node decision test uses attributes X and Z and the partition boundary in the 2-D subspace is given by the equation $3X + Z = 5$. The samples that have value (3X+Z) ≤ 5 are classified as class '0'. However subspace having value for (3X+Z) > 5 contains samples from class '0' and class '1' and hence further divided by another test with attribute Y.

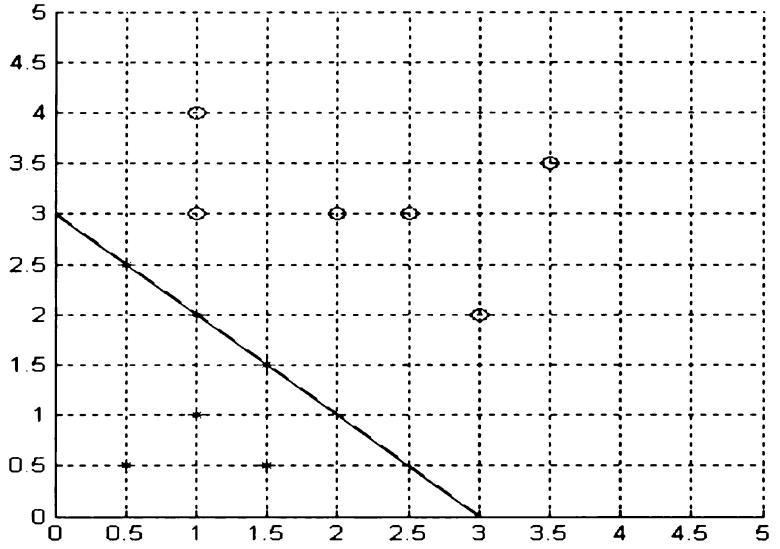


Figure 3.11: The decision boundary for the sample space using Linear Combination algorithm

X	Y	Z	Class
1	1	1	0
1.5	0.5	3.5	0
0.5	2.5	3	0
0.5	0.5	1	0
1.5	1.5	0.5	0
1	2	2	0
1	4	2.5	1
2	3	3	1
2.5	3	1.5	1
3	2	1.5	1
3.5	3.5	2	1
1	1.5	2.5	1

Table 3.5: Training data set with three features, 2 classes, 12 samples 6 from each class.

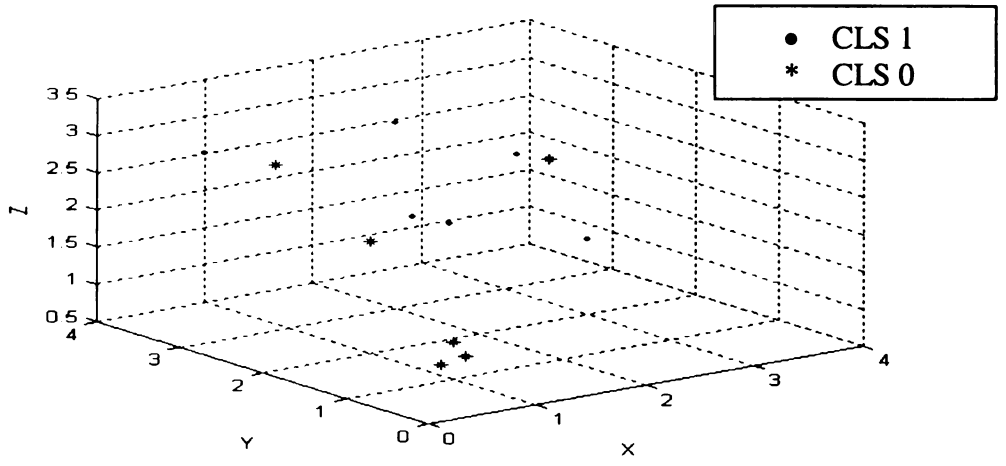


Figure 3.12: Distribution of sample data set from Table 3.5

The next test at this node is $Y \leq 0.5$. Samples having (Y) value ≤ 0.5 are classified as class '0' and samples having (Y) value > 0.5 are classified as class '1'. The decision tree for this sample dataset is given in Figure 3.14(a). The decision trees for this sample dataset using original 1-D ID3 algorithm and 2-D JE algorithm are presented in Figure

3.14 (b) and (c) respectively. The procedure and steps for the Linear Combination

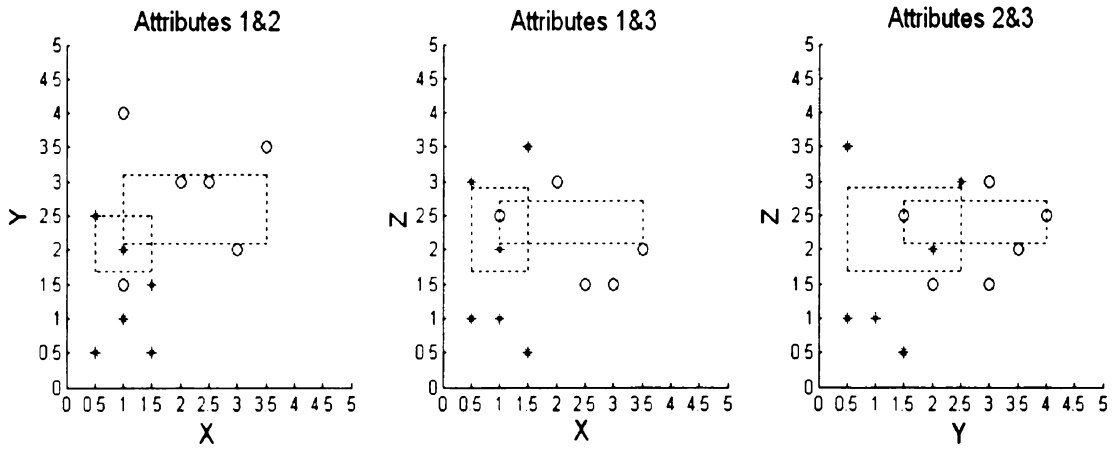


Figure 3.13: Different pairs of features and distribution of sample data set in those pairs.

algorithm are explained in Figure 3.15. The performance of the two algorithms is evaluated by analyzing the implementation results of classification using the two algorithms on different datasets and is discussed in the following chapter.

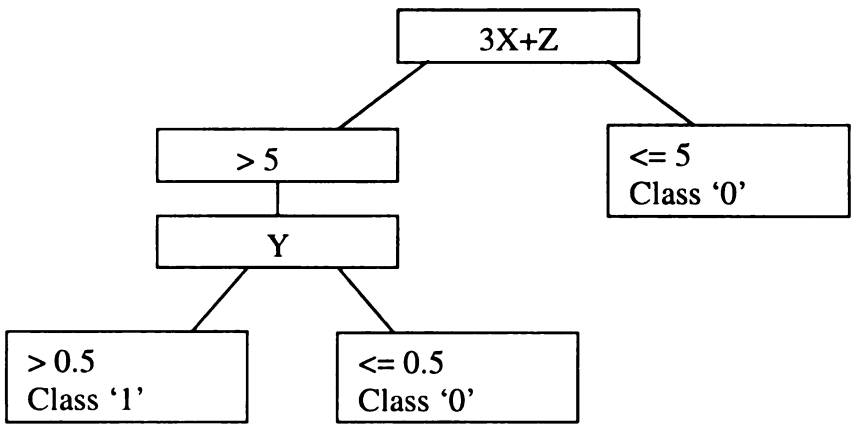


Figure 3.14 (a): Decision Tree for the dataset in Table 3. 5 using Linear Combination Algorithm.

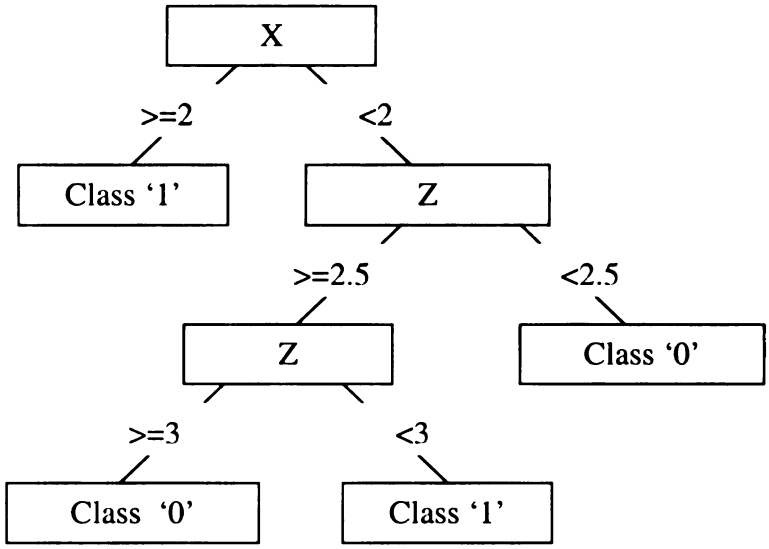


Figure 3.14 (b):Decision Tree for the dataset in Table 3.5 using ID3 algorithm

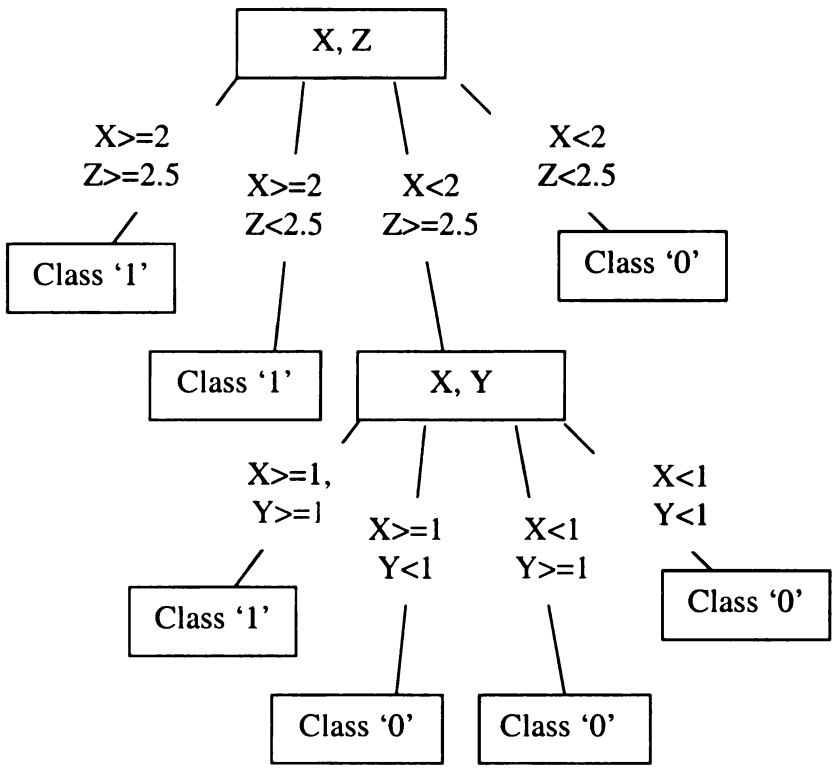


Figure 3.14 (c): Decision Tree for the dataset in Table 3.5 using 2-D ID3 algorithm (JE algorithm).

First Pair of attributes: **Attribute X and Attribute Y**		
	Class '0'	**Class '1'**
Vertices bounding the rectangle	(0.5,0.5);(1.5,0.5); (1.5,2.5);(0.5,2.5)	(1,1.5);(3.5,1.5); (3.5,4);(1,4)
Projections on the X feature axis	(0.5,0);(1.5,0)	(1,0); (3.5,0)
Projections on the Y feature axis	(0,0.5);(0,2.5)	(0,1.5);(0,4)
Second Pair of attributes: **Attribute X and Attribute Z**		
	Class '0'	**Class '1'**
Vertices bounding the rectangle	(0.5,0.5);(1.5,0.5); (1.5,3.5);(0.5,3.5)	(1,1.5); (3.5,1.5); (3.5,3); (1,3)
Projections on the X feature axis	(0.5,0); (1.5,0)	(1,0); (3.5,0)
Projections on the Y feature axis	(0,0.5);(0,3.5)	(0,1.5);(0,3)
Third Pair of attributes: **Attribute Y and Attribute Z**		
	Class '0'	**Class '1'**
Vertices bounding the rectangle	(0.5,0.5);(2.5,0.5); (2.5,3.5);(0.5,3.5)	(1.5,1.5);(4,1.5); (4,3);(1.5,3)
Projections on the X feature axis	(0.5,0);(2.5,0)	(1.5,0);(4,0)
Projections on the Y feature axis	(0,0.5);(0,3.5)	(0,1.5);0,3)

Table 3. 6: Intercept points to be considered for classification for the data set in Table 3.5

The total combination set AB is determined by all possible pair-wise combinations of attributes.

The intercept points set P is computed by finding a bounding rectangle in every two-dimensional space corresponding to every attribute pair from combination set AB.

A training data set S, classes' set C, total attributes combinations set AB and the potential points set P containing points for each pair of attributes are used as input to the Linear Combination algorithm (LC).

LC (S, C, AB,P)
begin

If all entries in S are from the same class {

return a leaf node with that class name as a label.}

else if S is empty {

return a single node with value failure.}

else if A is empty {

return a single node with value as the most frequently occurring class in that training set.}

else{

let the line L_k passing through points P_i & P_j be the line that has the lowest joint entropy and best classifies the training set S.

let the points P_i & P_j be from the two dimensional space formed by using attributes A_i & B_j. Hence let (A_i, B_j) be the attribute pair from AB whose distribution space contains the line L_k.

Then the decision attribute pair at the Root Node is (A_i, B_j) and the decision test at the root node is the equation of line L_k passing through points P_i & P_j.

Figure 3.15.1: The Linear Combination Algorithm (continued to next page).

For the test to be less than zero or equal to zero and greater than zero, add one branch below Root.

Let S_{ABi} *be the subset that has value for decision boundary as either less than/ equal to zero or greater than zero.*

If S_{ABi} *is empty*

Then add a leaf node below this branch with the most frequently occurring class in S.

Else add a new sub-tree below this branch as $LC(S_{ABi}, C, AB-\{(A_i, B_j)\})$

End

Figure 3.15.2:The Linear Combination Algorithm(continued from previous page)

CHAPTER 4. RESULTS AND DISCUSSION

In the previous chapters, we have discussed the ID3 algorithm; it's extension C4.5 and the proposed 2-D ID3 algorithms based on 1) Joint Entropy (JE) and 2) Linear Combination (LC). The objective of these modifications is to enhance the capabilities of the basic ID3 algorithm. Although the Joint Entropy algorithm helps in reducing the depth and size of the tree and also increases the classification accuracy in some cases, the desired non-orthogonal boundary was not achieved. On the other hand the Linear Combination algorithm is potentially capable of generating a decision boundary that take into account the correlation between features, thereby resulting in non-orthogonal decision boundaries.

This chapter describes the databases used for evaluating the performance of these algorithms. The implementation and results of this analysis are presented and discussed.

4.1 Databases

Two databases were used to evaluate the performances of the proposed 2-D ID3 algorithm; the first is the IRIS database while the second consists of field data collected from a nuclear power plant steam generator tube inspection. This database is referred to as the Eddy Current (EC) bobbin probe data. Same databases were also used in the ID3 classification and these results were used to compare the performance of all the classification algorithms. Since the proposed algorithms work with two class problems, the databases were chosen so as to meet this two-class criterion.

4.1.1 The IRIS database

The IRIS database is perhaps the most frequently used and best-known benchmark database found in pattern recognition and data mining literature. This database was collected and created by Fisher. The dataset contains 3 classes, each having 50 instances. Each of the 3 classes represents a type of IRIS plant. Each data vector comprises four features, which define the type or class of the IRIS plant. The features are sepal length, sepal width, petal length and petal width. These features are numeric, continuous features. One class is linearly separable from other two while the remaining two are not linearly separable. The two linearly non-separable classes are used for the analysis. The database was taken from the UCI Machine Learning Repository of databases. In this study, the data is randomly split into training and test datasets.

4.1.2 The BOBBIN database

This dataset generated by the Electrical Power Research Institute (EPRI) is obtained from steam generator inspection in nuclear power plants. The data was collected by passing a bobbin coil eddy current probe through the heat exchange tubes in the steam generator unit. Eddy current inspection is generally carried out at different excitation frequencies for detecting cracks, corrosion or dents etc in the tubes. The probe simultaneously collects data at several excitation frequencies and hence the data analysis and classification results improve significantly by using information from signals at these different frequencies. The primary objective of this analysis is to detect, segment each indication in the signal and then classify each indication. The excitation frequencies used

are 35 KHz, 200 KHz, 400 KHz and 600 KHz. A MIX frequency channel is also created using 200 KHz and 400 KHz channels. More information on this data can be obtained in [10].

For our analysis, data from only one tube was considered. A simple two-class problem is used for classification, the classes being labeled as DEF for defect and NDD for non-defects. A typical defect signal is shown in Figure 4.1. 8 different features were computed using peak-to-peak voltage (VPP),and phase angles (PH) of the signals at different frequencies.

The following features were used:

1) VPP in 200 KHz channel,
2) VPP in 400 KHz channel,
1) VPP in 600 KHz channel,
2) VPP in MIX channel,
3) PH in 200 KHz channel,
4) PH in 400 KHz channel,
5) PH in 600 KHz channel,
6) PH in MIX channel.

4.2 Implementation and Results

Both of the above databases were used in the evaluation of JE and LC algorithms. Both algorithms were implemented using C programming language and rules for the decision trees were determined. These rules were then implemented in MATLAB for use

on test data.

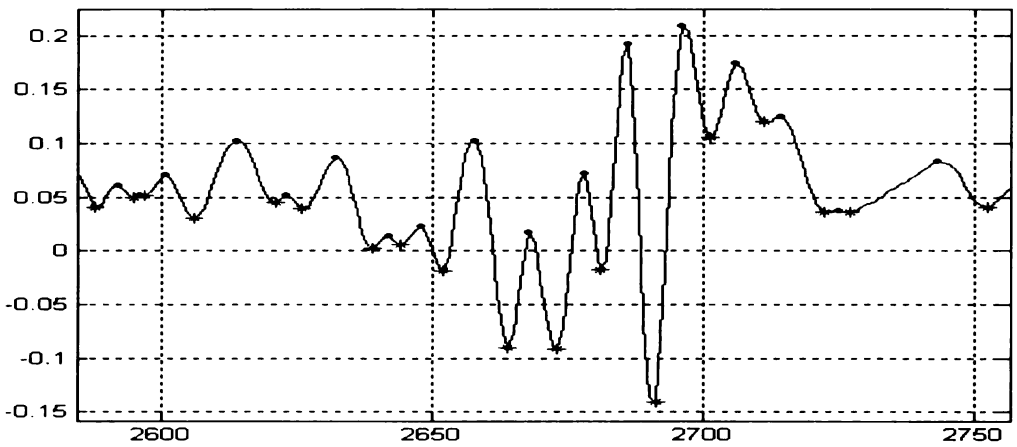


Figure 4.1: Typical Defect signal

The data samples were randomly divided into training and test databases. In case of IRIS database, the training database was computed using random selections of 50 data samples, 25 from each class and the remaining 50 samples (again 25 from each class) were treated as test database. This process was repeated to compute 10 different training and test database pairs. The analysis was carried out using different number of features, which decided the dimensionality of the signal space. In the first study all four features of the data vectors were used. In the second study only three features for each data sample were employed. The three features considered were: sepal length, petal length, petal width. In both the studies, 10 different training/test database pairs were randomly selected.

In the case of the BOBBIN database, only one tube with defect signals was considered. The multichannel data contains 200,400 and 600 KHz signals. A typical

signal at 400 KHz is shown in Figure 4.2. A moving average filter with 3 data point window is first used to filter high frequency noise. The tube section with defect and NDD signals was selected. For this section, the local minima and corresponding local maxima are identified to generate individual signals from potential defects. The minima-maxima points are shown in Figure 4.2. These minima and corresponding maxima are used for computing an individual signal and its features.

The potential defect signal consisting of minima-maxima pairs were marked as 'DEF' or 'NDD' as per the ground truth provided by EPRI. The training databases were calculated by randomly selecting 13 NDD data point signals along with 13 DEF signals and remaining NDD signals were included in the test database. 10 such databases pairs were computed. The analysis was carried out using 8 and 5 dimensional data vectors. In the 5-feature data vector, feature numbers 2,5,6,7 and 8 are used.

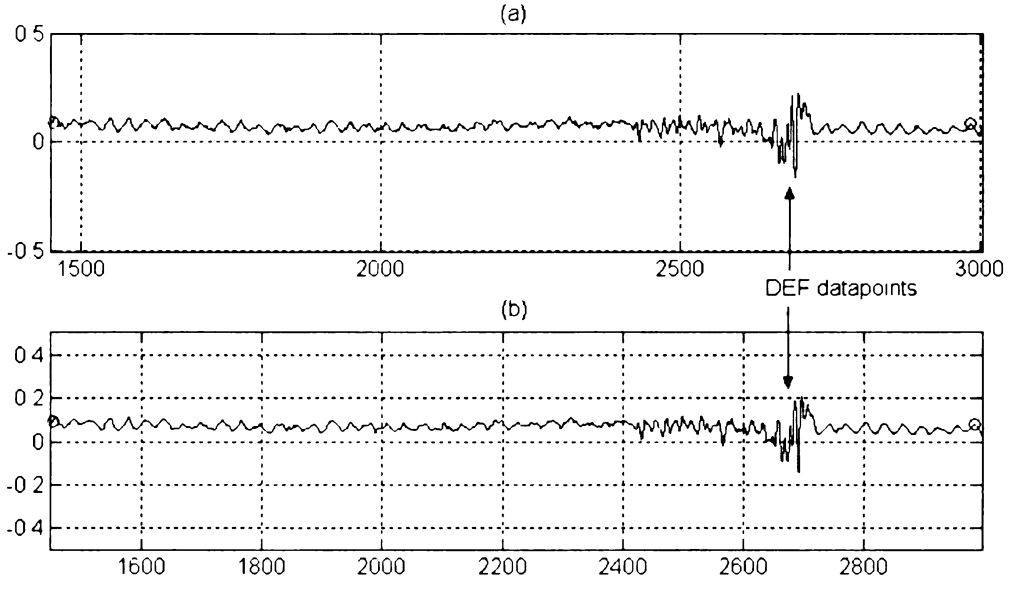


Figure 4.2: (a) Selected tube section before filtering (b) Selected tube section after filtering.

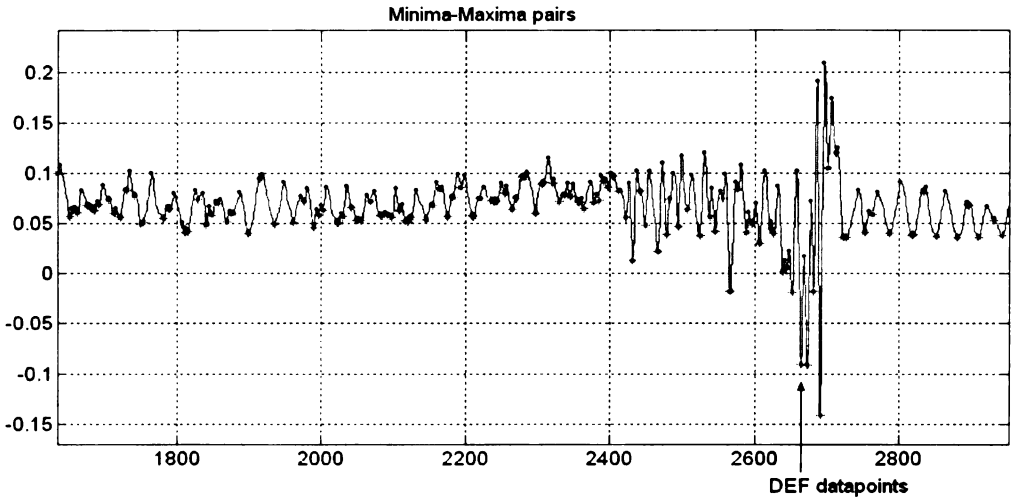


Figure 4.3: Minima-Maxima pairs

4.2.1 JE Algorithm

The decision trees and corresponding rules were computed using the ID3 algorithm and JE algorithm for all the training sets. These rules were implemented in MATLAB and used to classify the test data.

The results for ID3 and JE for all the test sets are given in Tables 4.1, 4.2, 4.4 and 4.5. Table 4.1 presents the results obtained using both ID3 and JE algorithms on the IRIS dataset, with three attribute data vectors. Table 4.2 presents similar results on the IRIS dataset using all four features of the data vector and both the algorithms. Table 4.3 presents the average error and corresponding variance of error in classification of the test data using both ID3 and JE algorithms.

Table 4.4 presents the results obtained using both the algorithms on the BOBBIN database using 5 attribute data vectors. Table 4.5 presents similar results obtained using 8

attribute data vectors. The average errors and corresponding variance of errors in classification of the test data set using both the algorithms are given in Table 4.6.

Set Number	ID3/JE algorithms	Misclassified as class 1	Misclassified as class 2	Total misclassification	% error
1	ID3	1	6	7	14
	JE	1	6	7	14
2	ID3	0	2	2	4
	JE	0	2	2	4
3	ID3	5	0	5	10
	JE	1	0	1	2
4	ID3	4	2	6	12
	JE	4	2	6	12
5	ID3	1	2	3	6
	JE	0	2	2	4
6	ID3	2	0	2	4
	JE	3	0	3	6
7	ID3	1	4	5	10
	JE	4	3	7	14
8	ID3	3	2	5	10
	JE	2	0	2	4
9	ID3	2	3	5	10
	JE	2	2	4	8
10	ID3	3	1	4	8
	JE	3	1	4	8

Table 4.1: IRIS dataset results, 50 samples in each test set, 2 classes, features used are sepal length, petal length, and petal width.

The results presented in Table 4.3 and Table 4.6, clearly show that the JE algorithm performs better than the ID3 algorithm. Also it can be seen that the increase in number of features affects the error variance. For lower number of features, error variance in 1-D ID3 algorithm is lower than that in JE algorithm and for higher number of features, error variance in ID3 algorithm error variance is higher than the JE algorithm

error variance. Also the decision trees and rules computed using the JE algorithm are much simpler than those with conventional 1-D ID3 algorithm. Figure 3.5 and Figure 3.6 gives details of decision tree structures and rules obtained using the two algorithms.

Set Number	ID3/JE attributes	Misclassified as class 1	Misclassified as class 2	Total misclassification	% error
1	ID3	1	2	3	6
	JE	0	2	2	4
2	ID3	4	2	6	12
	JE	4	2	6	12
3	ID3	1	1	2	4
	JE	1	0	1	2
4	ID3	3	11	14	28
	JE	3	1	4	8
5	ID3	4	0	4	8
	JE	3	2	5	10
6	ID3	0	2	2	4
	JE	0	2	2	4
7	ID3	4	4	8	16
	JE	4	0	4	8
8	ID3	0	3	3	6
	JE	0	3	3	6
9	ID3	3	1	4	8
	JE	2	1	3	6
10	ID3	1	1	2	4
	JE	1	1	2	4

Table 4.2: IRIS dataset results, 50 samples in each dataset, 2 classes, features used: sepal length, sepal width, petal length, and petal width.

IRIS dataset	Average Error (%)		Variance	
	ID3	JE	ID3	JE
Three features	8.8	7.6	10.84	19.37
Four features	9.6	6.4	56.71	9.6

Table 4.3: Average results for both the algorithms using both the datasets.

Set Number	ID3/JE attributes	Misclassified as class 1	Misclassified as class 2	Total misclassification	% error
1	ID3	2	28	30	21.74
	JE	2	16	18	13.04
2	ID3	0	51	51	36.95
	JE	0	50	50	36.23
3	ID3	0	36	36	26.08
	JE	0	49	49	35.50
4	ID3	1	31	32	23.18
	JE	0	15	15	10.86
5	ID3	1	29	30	21.73
	JE	1	15	16	11.59
6	ID3	0	35	35	25.36
	JE	0	35	35	25.36
7	ID3	1	21	22	15.94
	JE	5	8	13	9.42
8	ID3	0	47	47	34.05
	JE	0	46	46	33.33
9	ID3	1	16	17	12.31
	JE	1	16	17	12.31
10	ID3	0	30	30	21.73
	JE	1	37	38	27.53

Table 4.4: BOBBIN dataset, 138 test samples, 2 classes, 5 features

Set Number	ID3/JE attributes	Misclassified as class 1	Misclassified as class 2	Total misclassification	% error
1	ID3	0	18	18	13.04
	JE	0	21	21	15.21
2	ID3	0	29	29	21.01
	JE	0	30	30	21.73
3	ID3	0	21	21	15.21
	JE	0	30	30	21.73
4	ID3	1	14	15	10.87
	JE	1	15	16	11.59
5	ID3	0	32	32	23.18
	JE	1	28	29	21.01
6	ID3	0	36	36	26.08
	JE	2	25	27	19.56
7	ID3	0	20	20	14.49
	JE	0	27	27	19.56
8	ID3	0	34	34	24.63
	JE	0	24	24	17.39
9	ID3	0	32	32	23.18
	JE	0	32	32	23.18
10	ID3	0	30	30	21.73
	JE	0	24	24	17.39

Table 4.5: BOBBIN dataset, 138 test samples, 2 classes, and 8 features.

BOBBIN dataset	Average Error (%)		Variance	
	ID3	JE	ID3	JE
Five features	23.9	21.5	54.72	124.17
Eight features	19.34	18.83	29.19	12.35

Table 4.6: Average error and variance for BOBBIN dataset

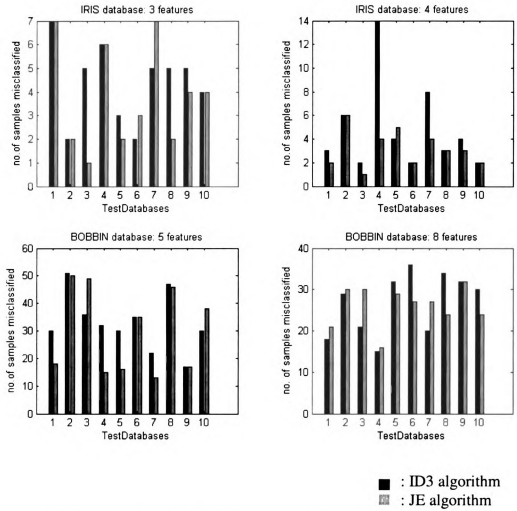


Figure 4.4: Bar charts for results using ID3 and JE algorithms

4.2.2 LC Algorithm

We used the same datasets as those used in JE to test the LC algorithm. The analysis was carried out for different number of features, which decided the dimensionality of the signal space.

The IRIS dataset consists of a total of 100 samples. Of these, the training set was

obtained by randomly selecting 50 samples, 25 from each class and the remaining 50 samples were treated as the test database. The process was repeated to obtain 10 training and test datasets. In this study all four features of the data vectors were used. The rules and decision tree were developed using the LC algorithm. The results of this analysis are presented in Table 4.7. Table 4.8 presents a comparison of the average errors and error variance obtained using the LC and ID3 algorithm. The error variance in 1-D ID3 algorithm results is higher than that in LC algorithm results.

For the BOBBIN dataset, the training set used comprised randomly selected 13 NDD signals and 13 DEF signals and the remaining samples were used as the test database. The process was repeated for 8 and 5 dimensional data vectors to obtain 2 sets of 10 training and test datasets pairs. The rules were computed for each of the training sets and were used for classifying the test data. The results of this analysis are presented in Table 4.9 and Table 4.10. Table 4.11 presents the average error and error variance obtained using both ID3 and LC algorithms for datasets having 5 features and 8 features.

The results clearly show that the performance of ID3 and that of LC algorithm are comparable. . For lower number of features, error variance in 1-D ID3 algorithm is higher than that in LC algorithm and for higher number of features, error variance in ID3 algorithm error variance is lower than the LC algorithm error variance. As the number of features i.e. the dimensionality of the sample space increases, the error variance in case of 1-D ID3 algorithm decreases while error variance in case of LC algorithm increases. Although the LC algorithm gives a slightly higher error ratio and higher error variance, the decision trees and partition boundaries generated by the LC method are much simpler

than those generated using the 1-D ID3 algorithm.

Set Number	ID3/LC attributes	Misclassified as class 1	Misclassified as class 2	Total misclassification	% error
1	ID3	1	2	3	6
	LC	1	5	6	12
2	ID3	4	2	6	12
	LC	4	2	6	12
3	ID3	1	1	2	4
	LC	3	1	4	8
4	ID3	3	11	14	28
	LC	1	3	4	8
5	ID3	4	0	4	8
	LC	3	3	6	12
6	ID3	0	2	2	4
	LC	0	3	3	6
7	ID3	4	4	8	16
	LC	3	0	3	6
8	ID3	0	3	3	6
	LC	6	0	6	12
9	ID3	3	1	4	8
	LC	1	5	6	12
10	ID3	1	1	2	4
	LC	5	0	5	10

Table 4.7: IRIS test datasets, 50 samples, 2 classes, features used: sepal length, sepal width, petal length, petal width.

IRIS dataset	Average Error (%)		Variance	
	ID3	LC	ID3	LC
Four features	9.6	9.8	56.71	6.62

Table 4.8: Average error and variance using ID3 and LC algorithms.

Set Number	ID3/LC attributes	Misclassified as class 1	Misclassified as class 2	Total misclassification	% error
1	ID3	0	18	18	13.04
	LC	0	31	31	22.46
2	ID3	0	29	29	21.01
	LC	0	29	29	21.01
3	ID3	0	21	21	15.21
	LC	1	25	25	18.11
4	ID3	1	14	15	10.86
	LC	1	15	15	10.86
5	ID3	0	32	32	23.18
	LC	0	16	16	11.59
6	ID3	0	36	36	26.08
	LC	0	43	43	31.15
7	ID3	0	20	20	14.49
	LC	0	22	22	15.94
8	ID3	0	34	34	24.63
	LC	1	34	34	24.63
9	ID3	0	32	32	23.18
	LC	0	39	39	28.26
10	ID3	0	30	30	21.73
	LC	0	34	34	24.63

Table 4.9: BOBBIN dataset;138 samples; 2 classes;8 features.

Set Number	LC/LC attributes	Misclassified as class 1	Misclassified as class 2	Total misclassification	% error
1	ID3	2	28	30	21.73
	LC	0	37	37	26.81
2	ID3	0	51	51	36.95
	LC	0	40	40	28.98
3	ID3	0	36	36	26.08
	LC	0	52	52	37.68
4	ID3	1	31	32	23.18
	LC	0	31	31	22.46
5	ID3	1	29	30	21.73
	LC	0	38	38	27.53
6	ID3	0	35	35	25.36
	LC	0	38	38	27.53
7	ID3	1	21	22	15.94
	LC	0	28	28	20.28
8	ID3	0	47	47	34.05
	LC	0	32	32	23.18
9	ID3	1	16	17	12.31
	LC	0	27	27	19.56
10	ID3	0	30	30	21.73
	LC	0	41	41	29.71

Table 4.10: BOBBIN dataset; 138 samples; 2 classes; 5 features.

BOBBIN dataset	Average Error (%)		Variance	
	ID3	LC	ID3	LC
Five features	23.9	26.37	54.72	28.63
Eight features	19.34	20.86	29.19	45.48

Table 4.11: Average error and variance for BOBBIN dataset using ID3 and LC algorithm

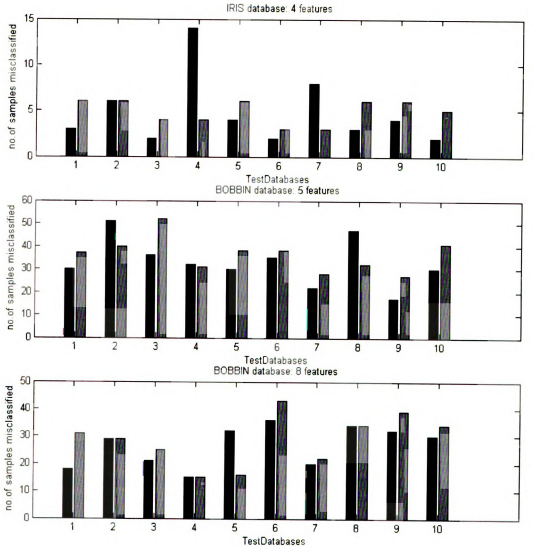


Figure 4.5: Bar charts for ID3 and LC algorithms

CHAPTER 5. CONCLUSION

The purpose of this study was to enhance the capabilities of the basic ID3 algorithm for linearly non-separable datasets by modifying the conventional ID3 classifier. The ID3 classifier works exceptionally well on linearly separable datasets. Using entropy as a performance measure, the ID3 algorithm chooses the attribute having lowest entropy in a rule at the root node and develops the remaining tree using a greedy approach without backtracking. This is a univariate classifier, which classifies the data by constructing the decision boundaries composed of orthogonal splits.

This method inherently treats each attribute as an independent entity and any correlation between features is ignored. In most of the cases, the tree generated by the ID3 algorithm is very large and over fits the data by increasing the size of the tree than is justified by training cases [7]. The need for simpler classifiers arises when the cost of performing a test is expensive as is the case in medical diagnosis. One method of devising minimal trees is by using pruning techniques to avoid over-fitting. In general, pruning of decision tree results in smaller tree having lees nodes but it may increase the classification error. Consequently, use of smaller trees mainly depends on our "bias". If the classification accuracy is to be maintained then complex and larger decision trees are chosen over smaller, simpler trees.

The two algorithms introduced in this thesis were developed to address these issues. The two approaches were built on the basic principle of ID3 i.e. entropy calculation and selecting an attribute with minimum entropy. The two methods attempt to

reduce the depth of the decision tree and simplify rules. The algorithms essentially consider two features at a time at each node. The Joint Entropy algorithm as the name suggests computes the joint entropy and generates a rule based on the minimization of joint entropy. Consequently this method also results in orthogonal splits but helps in reducing the depth of the tree. The performance of this method is better than the ID3 algorithm in almost all the cases. In most of the cases considered in the analysis, this method offers lesser or equal error than the ID3 algorithm.

An alternative method developed deals with the issue of correlation or interdependence between the attributes by considering linear combination of attributes. The 1-D ID3 algorithm builds complex decision trees when the features are highly correlated and the approach using linear combination of features results in simpler trees. This is mainly due to orthogonal splits used in 1-D ID3 algorithm. The Linear Combination algorithm looks for linear combinations of two features at a time each resulting in a linear partition on the 2-D space. The entropy of each partition is computed and the linear combination having the lowest entropy is selected as the test at the node. This method is helpful in datasets, which have linear decision boundaries. This algorithm also develops rules that are less complex than those obtained using the 1-D ID3 algorithm. Although the initial performance of the LC algorithm is not as promising as expected, there is a room for improvement by optimizing this algorithm. The current algorithm for deriving the rules is not optimized and alternate approaches for deriving the optimal decision boundary should be investigated.

The ID3 algorithm and its extension C4.5 are robust, consistent with the data and

give good results. Although their efficiency is hampered by the fact that they use a single feature, they are widely used as classifiers and in data mining. Results presented clearly demonstrate that the 2-D ID3 algorithm introduced in this thesis can enhance the capabilities of univariate ID3 algorithm for classifying non-separable datasets. These results therefore provide ample justification for future research in extending the algorithm to '*n*' dimensional nodes.

BIBLIOGRAPHY

[1] J. Quinlan. "Induction of Decision Trees." *Machine Learning* Vol. 1: p.81-106, 1986.

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

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

[4] M. Seo." Automatic Ultrasound Signal Classification scheme". Master's thesis.

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

[6] A. Jessop. "Informed Assessments: An introduction to Information, Entropy and Statistics." Ellis Horwood, 1995.

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

[8] T. Cover and J. Thomas. "Elements of Information Theory." John Wiley and Sons Inc, 1991.

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

[10] *EPRI Tech Report,* 2003.

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

[12] R. Mantaras et al., "Comparing information-theoretic attribute selection measures: a statistical approach." *AI Communications 11*, 1998

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

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

[15] Y. Horibe. "Entropy and Correlation." *IEEE Transactions on Systems, man and Cybernetics*, Vol. SMC-15, NO. 5, September/October 1985.

[16] T. Kvalseth. "Entropy and Correlation: Some Comments." *IEEE Transactions on*

Systems, man and Cybernetics,Vol. SMC-17, NO.3, May/June 1987.

[17] J. Finlay and A. Dix. " An Introduction to Artificial Intelligence." UCL Press, Taylor and Francis Group, 1996.

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

[19] R. Duda, P.Hart and D. Stork. "Pattern Classification." John Wiley and Sons, Inc, 2001.

[20] P. Winston. "Artificial Intelligence." Addison and Wesley Publishing Company, 1984.

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

[22] D.Michie (Ed.). " Expert systems in the micro-electronic age." Edinburgh University Press, 1979.

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