1992

Improvement of the Data Analysis Algorithm by Applying the Decision Tree Method.

Won Chan Jung
Louisiana State University and Agricultural & Mechanical College

Order Number 9301065

Improvement of the data analysis algorithm by applying the decision tree method

Jung, Won Chan, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

IMPROVEMENT OF THE DATA ANALYSIS ALGORITHM BY APPLYING THE DECISION TREE METHOD

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirement for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Won Chan Jung
B.S., Henderson State University, 1986
May, 1992

TABLE OF CONTENTS

ABSTRACT

Data analysis (reconstructablity analysis) is an area used on a data set which has several variables and a function value to find the most important factor that causes the function values to fall within a desired range. Normal data analysis algorithm [1] finds the most important factor in $O(2^n)$ time.

This dissertation introduces a newly developed system of algorithms called the Data Analyzing Tree (DAT) which is designed to either reduce the time complexity or produce more accurate result. DAT-1 uses $O(n^2)$ time to produce the same results as the normal data analysis method, and DAT-2 produces the result with a higher fall-into-the-range rate while using the same time complexity as the normal data analysis. Therefore, DAT-1 is suitable to get quick results, and DAT-2 or a higher numbered DAT is suitable to get more accurate results. DATs give more choices of the algorithm, so the users can chosse the appropriate algorithm depending on the circustances.

CHAPTER 1. INTRODUCTION

The values of a function are determined by the values of the variables used in the function. It is not too difficult determine the function value if the function's behavior is written as an expression and the values of the variables are known. If the function's behavior is not easily expressible, there may be some ways to figure out the function's behavior with a set of the variable values and the resulting function values. However, those methods can only be applied to the mathematical functions which have exact behavior on the values of the variables. This means that a given set of variable values always results in one specific function value. However, in the real world there are many cases where the values of the variables influence the function values but do not always produce the same value. In this case, the function cannot be written down by a simple mathematical expression. For example a student's grades have something to do with the student's studying method, the amount of time he spends studying, and his intelligence. But, the relation between the grades and the values of those variables cannot be written down as a mathematical function because the same studying method and the same amount of time spent studying by the same student will not always result in the same grade. In this case, only the approximate behavior of the function can be obtained.

One way to solve problems like the above example is by data analysis. It collects as much data as possible and comes up with the values of variables to make the function value fall within a certain range. For the above example,

certain studying methods, various amounts of time spent studying, and certain ranges of intelligence which give the highest probability of getting a certain grade will be produced as an answer by the data analysis algorithm.

This method has been developed by Dr. Bush Jones [1] and has been used for several years, but, there is room for improvement. A new method to improve this approach is introduced in this dissertation. The idea for this improvement came from the decision tree which is a method used in machine learning.

The decision tree algorithm is used to teach the computer how to determine certain rules by testing a set of data items. Each data items has several attributes with values and a class value. The value of the class is determined by the values of the attributes. The computer tests the data set and builds a decision tree. Once the tree is built, it can give the class value for any combination of attribute values.

The background work is introduced for both data analysis and the decision tree algorithm in chapter 2; the newly developed method to improve the performance of data analysis, called the data analyzing tree is introduced in chapter 3. The detailed algorithm and the tracing of the algorithm is illustrated in chapter 4. Experimental results are in chapter 5, and the conclusion is in chapter 6.

CHAPTER 2. BACKGROUND WORK

2.1 Data Analysis

2.1.1 Introduction

Data Analysis determines the behavior of a unknown function's values. In a data set that consists of several variables and a function value, the function values are determined by the variable values. The data analysis algorithm can find the most important group of variables that affects the function values the most.

This algorithm takes a data set as a table such as table 2.1 [1]. Table 2.1 represents data on the life of a battery for various conditions on the battery ; variables include material, temperature, and casing. These variables are assumed to have some effect on the F-value (Battery Life). We are interested in a certain range of the F-value. For example, if we are trying to make the battery life last longer than 130 hours, we are interested in the range (F-value > 130). Obviously, we cannot control the battery life directly, but we can control the variables. Therefore, we need to determine the variable values that causes F-value to fall within the desired range.

The data analysis algorithm can find the combination of variable values that most affect the F-value to fall within the range, so the users can control the variables found by this algorithm to get a desired result of F-value.

Table 2.1 Data Set for Battery Life

Material	Temperature	Casing	Life
1	53	2	130
2	51	1	150
2	67	2	174
3	53	1	168
1	80	1	58
3	65	2	153
1	82	1	71
2	66	2	106
3	87	2	60
1	63	1	42
1	61	1	75
3	54	2	158
2	86	1	47
3	50	1	139
2	64	1	128

2.1.2 Definitions of Problem and Terminologies [1]

In the language of reconstructability analysis, we are concerned with a system associated with a finite set of variables $\{ V_i \}$. A system also has a behavior which is described by a real valued system function f (.). Each

nonempty subset of variables identifies one subsystem of the system, and states $\{\alpha\}$ and substates $\{\beta\}$ of the system are determined by particular value assignments to the variables (states and substates are referred to as factors in the program).

For example, the factor of (material = 1 and temperature = 52.3) is a substate of the system which consists of all the data observations that satisfy the conditions of the factor.

The algorithm can be employed to provide information on "factors", which are specific sets of values for the variables in various combinations. For example, we can determine the most important factors and how these factors influence battery life, or we can predict the influence of an unknown factor on battery life, or we can obtain a variety of other information about the system.

2.1.3 Clustering

There are two types of variable values : discrete values and continuous values [1]. For a data set to be used in the data analysis algorithm, the variables in the set must have discrete values. Variables with continuous values must be clustered to form categories of the values. The clustering algorithm employs a hierarchical clustering technique of using a minimum distance criteria to form centroids and then looking for a natural breakpoint in the centroids. The following categories or clusters for temperature were formed by the clustering algorithm. [1]

Cluster 1. Temperatures in (50.0, 58.3) represented by 52.3

Cluster 2. Temperatures in (58.3, 74.0) represented by 64.3

Cluster 3. Temperatures in (74.0, 87.0) represented by 83.8

Throughout the remaining section of the algorithm, the temperature values will be represented by one of these three cluster values.

2.1.4 Important Value Finding

After the clustering is performed, the most important value for each variable must be found. In data analysis, we consider only the most important value for each variable. The important value for a variable is the value that causes the highest fall-into-the-range rate for F-value.

For the example in table 2.1, the following three important values are found for the variables :

For Material :

if the value is 1, the rate is 0/5 = 0.0

if the value is 2, the rate is 3/5 = 0.6

if the value is 3, the rate is 4/5 = 0.8

For Temperature :

if the value is 52.3, the rate is 4/5 = 0.8

if the value is 64.3, the rate is 3/6 = 0.5

if the value is 83.8, the rate is 0/4 = 0.0

For Casing :

if the value is 1, the rate is 4/9 = 0.44

if the value is 2, the rate is 3/6 = 0.5

Therefore, the vector of the most important variable values is (3, 52.3, 2).

2.1.5 Important Substates (Factors)

Once all the important variable values are found, we can proceed checking all the important factors (combinations of variable values). To find the most important factors, we compare the fall-into-the-range rate for every possible factor.

Since we have three variables, there are seven non-empty substates (factors). They are :

material = 3

temperature = 52.3

casing = 2

material = 3 and temperature = 52.3

material = 3 and casing = 2

temperature = 52.3 and casing = 2

material = 3, temperature = 52.3, and casing = 2

For the desired F-value range of 'over 130', the factor of (material = 3 and temperature = 52.3) is found as the most important factor because it has the highest fall-into-the-range rate of 100%.

2.1.6 Interactions and Predictions [1], [5]

The options offered after finding the important factors include the ability to compute interactions and the ability to predict the effects of factors not present in the original data. First, we need the notion of a factor effect. An effect is a change in system behavior (system function value) expressed relative to a flat system (or equivalent to the mean system behavior). An interaction is the contribution to a factor effect that is due to the components of a factor acting all in unison.

Let V={ $v_1 = c_1, v_2 = c_2, \ldots, v_n = c_n$ } be a set of n variables with a particular value assignment (a factor or substate), let V_m be the set of all subsets of V, and let V_p denote the set of all "proper" subsets of V. We compute the unbiased reconstructions $\mu(\ V_m\ )$ and $\mu(\ V_p)$. For any factor which embodies V, we then consider the difference in its effect as computed from these two unbiased reconstructions. The difference represents the contribution to a factor effect that is due to the components acting in unison.

Predicting the effect of a factor that is not present in the original data requires a process similar to the computation of an interaction. Although the factor effect is unknown, we can often form accurate estimates of some of its subfactors' effects. The unbiased reconstruction is computed from such subfactors, and the resulting distribution provides a prediction of the effect for the overall factor. The goodness of such a prediction will depend on the importance of any missing subfactor interactions and the missing overall

interaction. However, such a prediction does make use of all the known relevant information.

It is important to know how widespread the effect is that is associated with a factor which is referred to as the factor region size. When a variable is clustered, many values become represented by a single value. This single value is said to represent an interval or a simple region of variable values. A factor is a combination of such single values, hence it actually represents many factors which we refer to as a region. The number of factors which a factor actually represents relative to the total number of values available to be represented by factors is called the factor's region size.

2.1.7 Summary

This algorithm leads us to a combination of variables with values which affects the function values the most. Therefore, we can concentrate on those variables trying to keep the values of the variables within that range so we can get the desired range of function values. This approach is useful, especially in a real life, when trying to improve the function value without knowing which variables to work on.

To determine the set of factors with values which affect the function values the most, we must follow these steps :

A) Cluster values of continuous variables.

B) Find the significant values for each variable.

C) Find the most significant combination of variable with these values.

D) Output the necessary information

To get the most significant combination of variable with values, we have to consider all the combinations of variables and compare them. Considering all possibilities requires checking all 2^n combinations where n is the number of variables. Clearly, the number of combinations increases exponentially as n increases.

Another rare problem occurs when v_i has a significant value in x_i and v_j has a significant value in x_j, but the unchecked factor ($v_i = y_i$ and $v_j = y_j$) is more significant in affecting the f-value to fall within the desired range. Since we take the most significant value for each variable, the values y_i and y_j are discarded, so we never check the case when $v_i = y_i$ and $v_j = y_j$. Methods for recovering from these problems are discussed in Chapter 3.

2.2 Decision Tree

2.2.1 Introduction

The computer is not just a calculating machine but is becoming a machine that can also help people make decisions. A computer can make a decision based on rules given by people which limits its ability to the ability of the person who provides the rules. Another way the computer can make a decision is by learning on its own from examples. The area that attempts to make the computer learn from examples is called machine learning.

Machine learning is an area of artificial intelligence which allows the computer to learn the pattern of the behavior of the data. It learns totally from examples without any help from a human. The computer does not have to follow rules given by people, but it can make up the rules on its own. Those rules that the computer generate may not be perfect, but after gaining enough experience, those rules will be closer and closer to the perfect ones.

A set of data items, where each data items consists of several attributes and a classification that the data falls into, is given for a computer to learn the pattern to decide the classification of certain values of attributes. There are also many ways for a computer to learn the pattern of the data behavior. One of those methods is the decision tree algorithm. It is a method that allows the computer to design a tree-structured pattern from the given data of attributes and the classifications. Once a tree-structured pattern (called a decision tree) is built, it can follow the path of the tree to determine the classification of a given

data set of attributes with values.

A decision tree has the following structure :

-. It has a general tree structure;

-. Each leaf of the tree indicates a class;

-. Each intermediate node contains the attribute to be tested;

-. Each intermediate node may have as many branches as the number of possible values of the attribute it contains.

2.2.2 Data Sets for the Decision Tree Algorithm

In the decision tree algorithm we deal with a data set. Each data item consists of several attributes which are variables that can hold a value and a classification value which classifies the data. There is a function that is applied to all the data which takes the values of attributes and comes up with a classification value. The algorithm is designed to derive the function from testing the data set.

Consider a data set of three attributes a_1, a_2, and a_3, which are binary digits and one classification C which is also a binary digit. Let the value of the classification C be the value of the following expression.

$$C = 1 \text{ if } (a_3 = 1) \text{ AND } ((a_1 = 1) \text{ OR } (a_2 = 1))$$

$$0 \text{ otherwise}$$

Of course the computer does not know that the value is decided by the above rule, and it has to find the rule by testing the data set like the table 2.2.

Table 2.2 Data Set for the Given Example

a_1	a_2	a_3	C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

2.2.3 Building a Decision Tree

The main idea of building a decision tree is dividing the data set into several equivalence classes according to an attribute that is tested at that level. The procedure then goes on to the descendants of the current node until all the data in the subset have the same classification value.

The following algorithm builds a decision tree (starting with the root node) :

1) If the current node is the root, take the entire data set as a test set T and put all attributes in the attributes list L; otherwise, take T and L from the immediate ancestor.
2) If all the classification values for T are the same, put the classification value in the node, declaring this is the leaf of the tree, and return to the ancestor by terminating further processing at the current node.
   If not, proceed to (3).
3) Select the most significant attribute (A_s) from L and create a descendant d_i for each possible value (v_i) of A_s.
4) Apply the algorithm recursively to each descendant d_i created in (3) by sending the subset of the T that satisfies $A_s = v_i$ and the list $\{L - A_s\}$.

It is important to decide which attribute to test first because testing the attributes that affect the class value more significantly at the higher level (closest to the root) can reduce the number of nodes in the decision tree. For the example in table 2.2, it is reasonable to have a tree like figure 2.1. This tree has seven nodes which is the minimum number of nodes needed for this example, so it is called the optimal decision tree. If a_1 is placed at the root instead of a_3, the tree will look like figure 2.2.

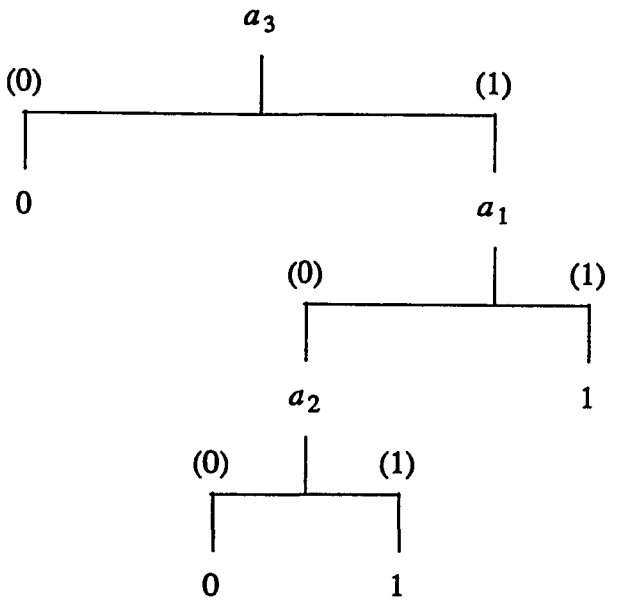

Figure 2.1 Optimized Decision Tree for the Data in Table 2.2

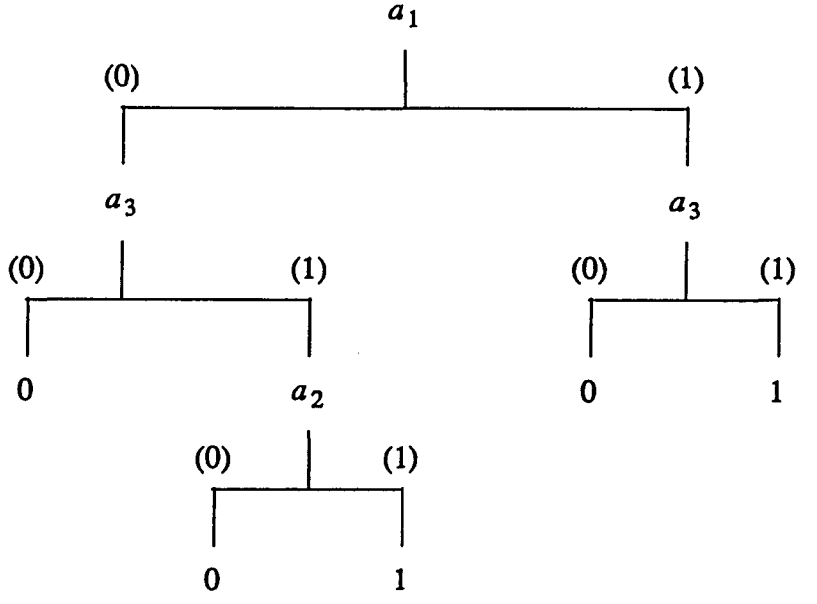

Figure 2.2 Not Optimized Decision Tree for the Data in Table 2.2

The tree in figure 2.2 has **nine** nodes which is two more than the previous tree. Even though this tree generates correct answers, it will take more comparisons to get the answers which is time consuming. For this reason, the more significantly effecting attribute should be checked at the higher level.

J.R. Quinlan [10] suggests a way of finding the most significant attribute. The method checks the effectiveness of each attribute by calculating the probability that each attribute's value will decide the value of classification and takes the most effective attribute.

Applying Quinlan's method at the root will be :

$$A_1 : P(A_1=0) \times P(C=0|A_1=0) + P(A_1=1) \times P(C=1|A_1=1) = 0.625$$

$$A_2 : P(A_2=0) \times P(C=0|A_2=0) + P(A_2=1) \times P(C=1|A_2=1) = 0.625$$

$$A_3 : P(A_3=0) \times P(C=0|A_3=0) + P(A_3=1) \times P(C=1|A_3=1) = 0.875$$

It shows that the knowledge of the value of A_3 can determine the class value 87.5% of the time while the knowledge of the values of either A_1 or A_2 can determine the class value only 62.5% of the time. So Attribute A_3 is selected at the top node.

Once a tree is built, we can apply this tree to any set of attribute values to find its classification. Starting from the root, we simply check the value of the attribute that the node contains and follow the path to the appropriate descendant. We keep following the path until a leaf is found and then report the

classification value that the leaf has which is the classification value for the values entered.

2.2.4 F-family data set

Quinlan's method usually finds the most significant attribute, but there are several cases in which his method cannot find the most significant attribute. This is due to the necessity for the method to decide which attribute to check before it builds a tree. Therefore, an unexpected result can occur at any time. In those cases, we must apply one of the optimization methods to the tree to get a decision tree with fewer nodes.

The F-family is a family of test data sets, F_n, such that $n = k + 2^k$, where k is a positive integer. F_n has n attributes and a class value, where each attribute and class value can have a value of either **0** or **1**. The first k attributes work like address bits, and others work like data bits. Address bits point to a data bit which decides the class value. Therefore, to determine the class value for an observation, we check the address bits first to see which data bit they point to and take the value of that data bit as a class value. For example, F_6 contains 6 attributes, a_1, a_2, a_3, a_4, a_5, and a_6. The first two attributes are the address bits, and they point to exactly one data bit among a_3, a_4, a_5, and a_6, and the class value C in F_6 is the value of the data bit that the first two bits point to. The rules to decide the class value are as followings :

If $a_1 = 0$ and $a_2 = 0$, $C = a_3$.

If $a_1 = 0$ and $a_2 = 1$, $C = a_4$.

If $a_1 = 1$ and $a_2 = 0$, $C = a_5$.

If $a_1 = 1$ and $a_2 = 1$, $C = a_6$.

As mentioned before, one must check the first two attributes to determine the value of C. Therefore, the value of C can be found in three steps. Obviously the decision tree has to have a_1 or a_2 at the top node to be optimal.

When Quinlan's method is applied, however, it looks for the most significant attribute by calculating the effectiveness of the attributes to the class value. Thus, it never considers a_1 or a_2 as the most significant attribute because of the following result given in [11] :

$$P\ (C = 1 \mid a_1 = 1) = 0.5$$

$$P\ (C = 1 \mid a_2 = 1) = 0.5$$

$$P\ (C = 1 \mid a_3 = 1)$$

$$= P\ (a_1=0 \text{ and } a_2=0) + P\ (a_1<>0 \text{ or } a_2<>0)\ \ 0.5$$

$$= 0.25 + 0.75\ \ 0.5 > 0.5$$

For the same reason,

$$P\ (C = 1 \mid a_4 = 1) > 0.5$$

$$P\ (C = 1 \mid a_5 = 1) > 0.5$$

$$P\ (C = 1 \mid a_6 = 1) > 0.5$$

Therefore, the decision tree that is produced by Quinlan's method has a_3, a_4, a_5, or a_6 at the root of the tree, which makes the tree suboptimal.

2.2.5. Optimization

Jung, Jones, and Chen [12] developed an optimization method using a "LOOK AHEAD" technique, (i.e., it searches from top-down to find a more significant attribute than the root attribute). Then, it pulls the more significant attribute up to the root to obtain a tree with fewer nodes. Quinlan's method finds the most significant attribute only, but this optimization method takes not only the attribute that Quinlan's method finds but also those attributes found in the next level. It compares these attributes to find the most significant attribute and place the newly found most significant attribute at the current node. The attribute to be placed at any node is not found by the estimation but by actual comparison to make less-noded tree.

In this algorithm it is assumed that each attribute has two possible values for the sake of simplicity. The algorithm can be generalized with more than two possible values in each attribute. Each inner node in the decision tree is labeled by the attribute a_i, which is used to partition the data set at that node. The leaves in the tree are labeled by their class value.

< Algorithm for Optimization >

(1) Build a decision tree by Quinlan's method, name the tree T_1, and count the number of descendants. Name the root of T_1 as p_1, the left child of p_1 as p_2 and the right child of p_1 as p_3. (The number of descendants can be counted while building a tree.)

(2) Build a new decision tree T_2 by using the attribute p_2 as the initial attribute at the root node of T_2 to partion the data set and also to count the number of descendants. (The subsequent steps in building T_2 use Quinlan's method.)

(3) Repeat step (2) for p_3 and name that tree T_3.

(4) Compare the numbers of descendants of T_1, T_2, and T_3, and take the tree T^* which has the least number of nodes among those three trees.

(5) Apply this algorithm to the immediate descendants the root of T^*. (The algorithm will be applied throughout the tree recursively.)

2.2.6 Result

Figure 2.3 shows a decision tree for F_6 built by Quinlan's method. It has a_3 at the root as expected and starts generating the tree from there. Of course, it is not an optimal decision tree for F_6. Figure 2.4 shows a decision tree for F_6 after applying the optimization algorithm to the tree in figure 2.3.

The lower branch means that the attribute's value is **0**, and the upper branch means the value is **1**. For example in figure 2.3, if a_3=**0**, one takes the lower branch and checks a_1; if a_3=**1**, one checks a_4.

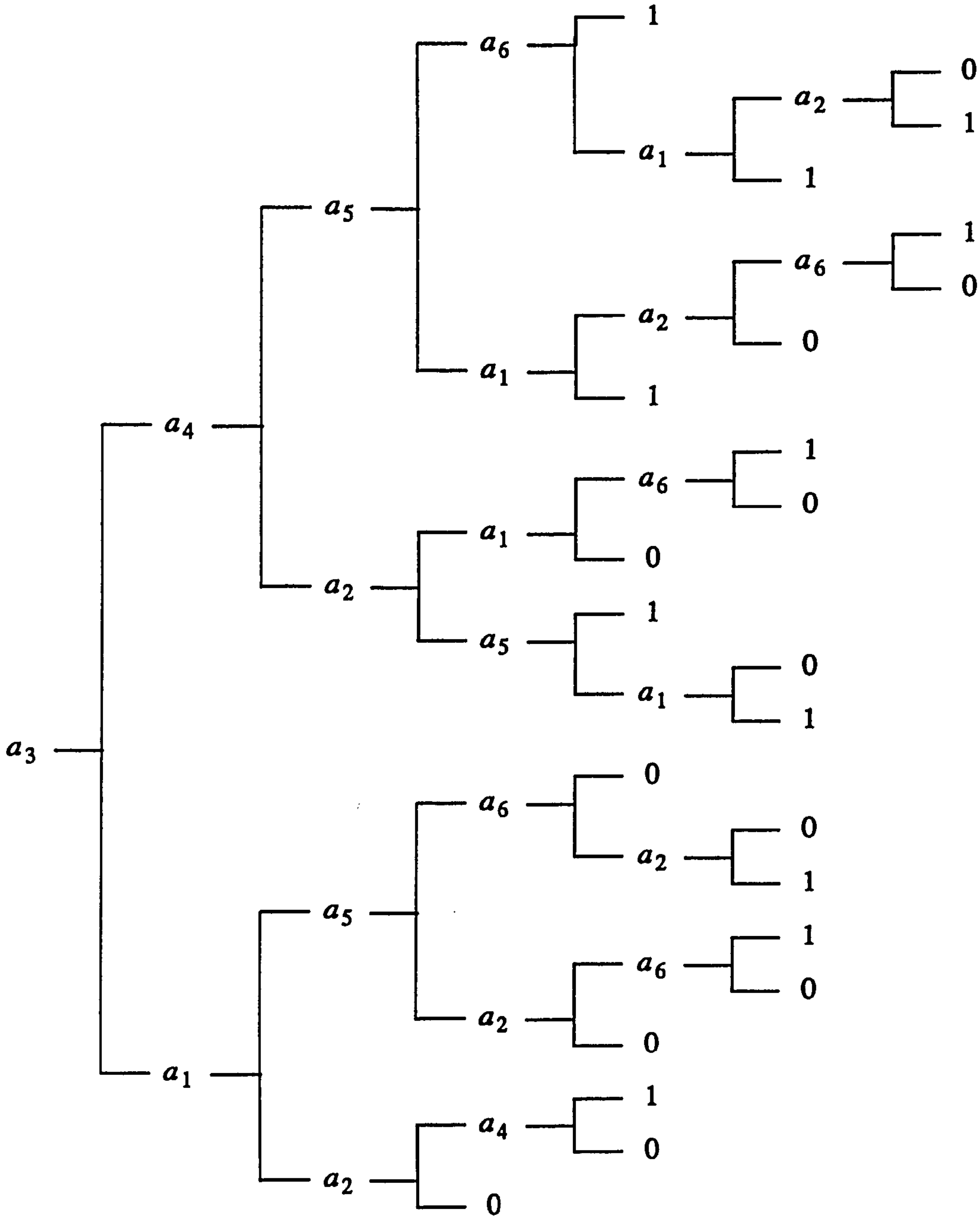

Figure 2.3 Decision Tree by Quinlan's Method

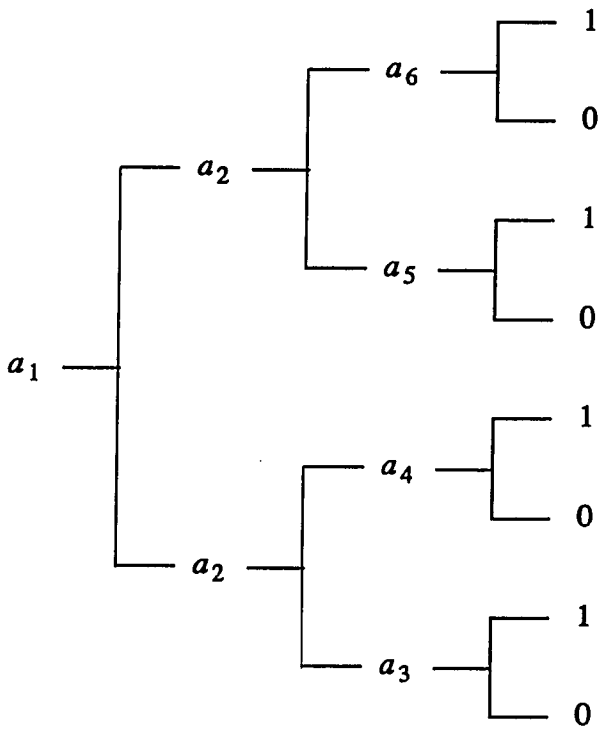

Figure 2.4 Optimized Decision Tree

It is easy to see that the number of nodes is reduced from 45 to 15. And it took just one optimization pass to generate the tree in figure 2.4. Even the worst possible decision tree for F_6, which has 91 nodes, can be reduced to the optimal tree by applying this algorithm just twice. Figure 2.5 is the worst possible decision tree, which is reduced to the tree in figure 2.6 after one execution of the algorithm. Figure 2.7 is the optimal tree that is obtained by applying the algorithm to the tree in figure 2.6, which is identical tree as the tree in figure 2.4.

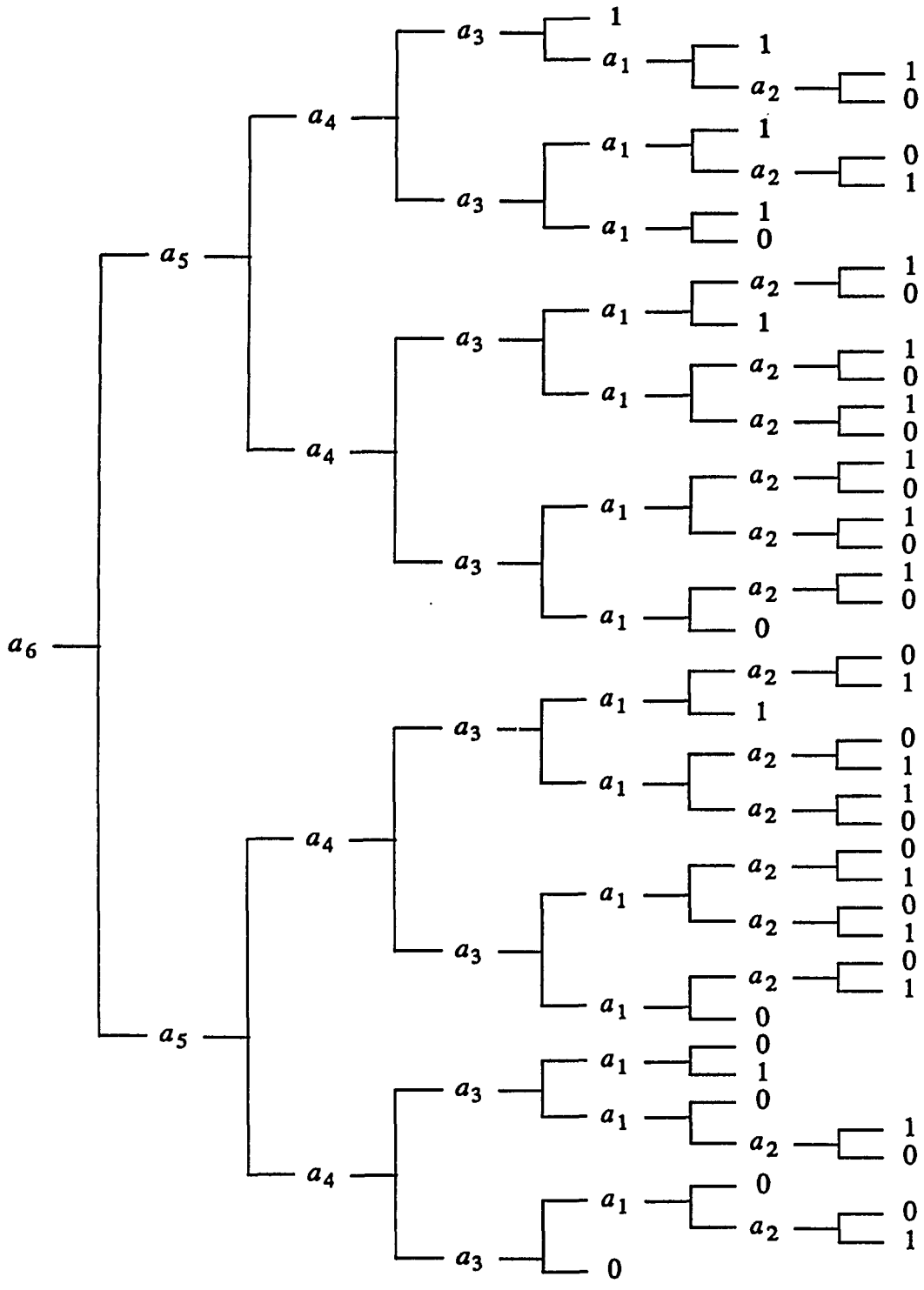

Figure 2.5 The Worst Possible Decision Tree for F_6

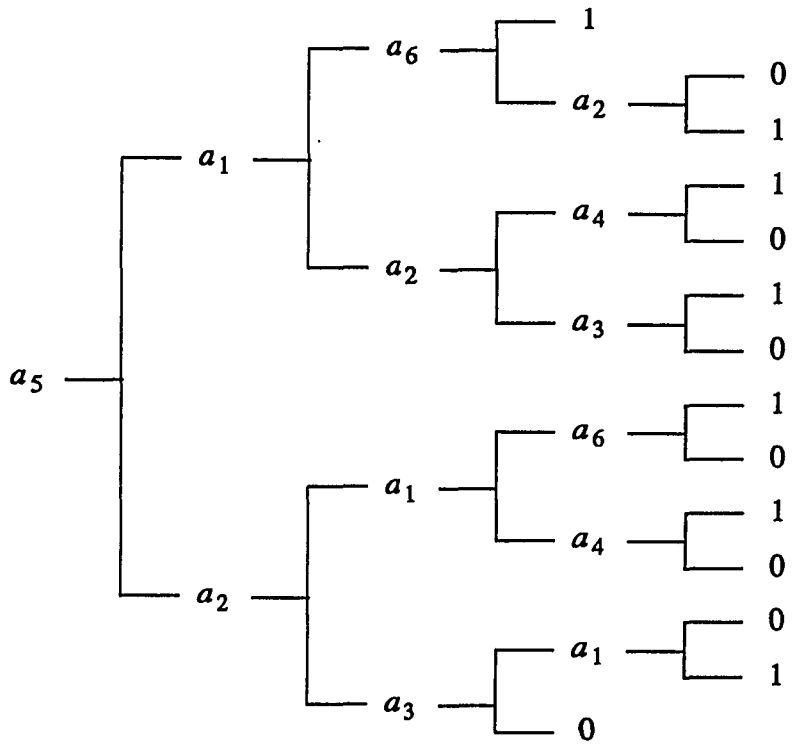

Figure 2.6 The Tree Generated from Figure 2.5

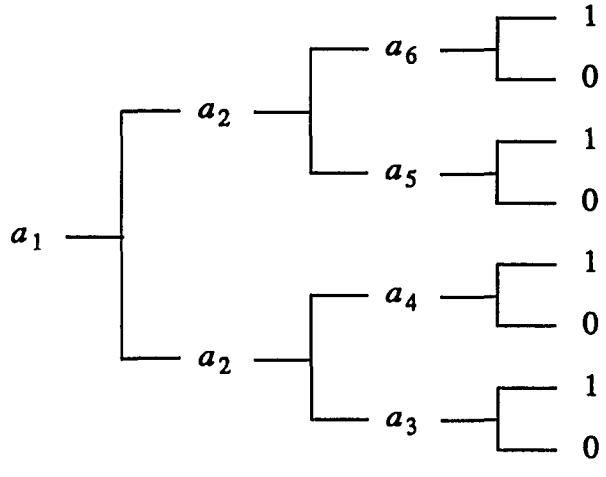

Figure 2.7 Optimized Decision Tree

CHAPTER 3. DATA ANALYZING TREES

3.1 Time Complexity of the Data Analysis Algorithm

Data analysis is the method of analyzing the data table to produce an important factor that can improve the function value. The biggest problem with this method is that it must check all the combinations of the variables. If 3 variables, a_1, a_2, and a_3 are used in the data table having the values v_1, v_2, and v_3 as the important values, this method has to check the following factors.

$a_1 = v_1$

$a_2 = v_2$

$a_3 = v_3$

$a_1 = v_1$ and $a_2 = v_2$

$a_1 = v_1$ and $a_3 = v_3$

$a_2 = v_2$ and $a_3 = v_3$

$a_1 = v_1$, $a_2 = v_2$, and $a_3 = v_3$

It takes on the order of $(2^n - 1)$ time assuming that n is the number of the variables. Therefore, the time complexity of the data analysis is $O(2^n)$. When this method is used, a data table with the maximum number of variables is desired because it results in a better solution, but it is unfeasible to include too many variables because of the time complexity shown above.

The alternate choice to solve this problem is the data analyzing tree which is the method that applies the decision tree algorithm to data analysis.

Instead of taking all the combinations of the variables into consideration, it takes one variable at a time to check and decides whether the variable should be included. This method is used in the decision tree algorithm, so it is appropriate to compare and discuss data analysis and the decision tree algorithms.

3.2 Comparison between Data Analysis and Decision Tree

There are several similarities, but also some differences, between the data analysis and the decision tree algorithms. Even though the purpose of each is different, it is possible to use those similarities to improve the performance of data analysis.

3.2.1 Similarities

-. Both the data analysis and the decision tree algorithms deal with a data set consisting of a set of variables (attributes) and the function value (class value).

-. The combinations of the variables (attributes) effect the function values (class values).

-. The relation between the combinations of the variables (attributes) and the function values (class values) is unknown.

-. The algorithm determines the relation between the combinations of the variables (attributes) and the function values (class values).

3.2.2 Differences

-. A combination of the attributes in the decision tree always produces a certain class value while a combination of the variables in data analysis does not always produces a given function value.

-. The purpose of the decision tree is to determine the relation between the combinations of the attributes and the class values, so the tree can predict the class value for any given combination of the attributes. The purpose of data analysis is to determine the relation between the combinations of the variables and the function values so the method can predict the combination of the variables with the best fall-into-the-range rate.

3.3 Data Analyzing Trees

3.3.1 Introduction

The previously mentioned problems of the normal data analysis algorithm leads to the new development of the data analyzing trees (DAT's). DAT-family is a set of data analyzing tree algorithms which perform the same tasks as data analysis method with a smaller time complexity or with more accurate results. DAT-family are DAT-1, DAT-2, DAT-3, ... ,DAT-k, ... and so on, where k means the number of branches that the tree makes at each node. That number can be increased up to the maximum of the number of clusters that each variable in the data set takes. For the data set in table 2.1 which takes

a maximum of three values for each variable, DAT-1, DAT-2, or DAT-3 can be used. A higher numbered DAT will take more time to complete the tree while it gives more information than a lower numbered DAT.

3.3.2 Characteristics of DAT

The characteristics of DAT are quite similar to those of the decision tree, but there are some differences. The data set that DAT deals with is different from the data set of the Decision Tree. The following are the characteristics of DAT :

-. The shape of the DAT is exactly the same as the decision tree. The difference is the information contained in each node.

-. In DAT, the inner nodes have the names of the variables to be checked and the fall-into-the-range rates while the leaves have the fall-into-the-range rates only.

-. Each node of DAT-k checks and finds the best variable to make the highest fall-into-the-range rate, and makes k branches according to the values of the variable.

-. Each branch from node i to node j in DAT is based on a condition that the variable in node i has certain value.

-. The fall-into-the-range rate of any given node i is the rate that the function value is in the desired range when the conditions of all the branches from the root to node i.

The special characteristics of DAT-1, DAT-2, and DAT-3 will be discussed in section 3.4, 3.5, and 3.6 respectively. Each variable a_i in the data set here is assumed to have the maximum of three values or clusters.

3.4 DAT-1

Each node in this DAT has either one descendant or none. The node without any descendants is the unique leaf of the tree. Therefore, the tree has the shape of a list. Each node has to decide which variable to take and make a branch based on a certain value of the variable. This tree has only two conditions for each variable : either the variable has a certain value, or we don't care what the value is.

The following is an example of DAT-1. This DAT-1 is produced after checking the data set in table 2.1 which has three variables where each variable has a maximum of three values.

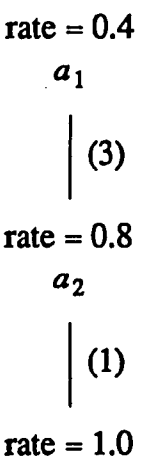

Figure 3.1 An Example of DAT-1

This tree contains the information of fall-into-the-range rates for three different conditions. Those rates for the conditions are :

the over all fall-into-the-range rate is 0.4.

if $a_1 = 3$, then the fall-into-the-range rate is 0.8.

if $a_1 = 3$ and $a_2 = 1$, then the fall-into-the-range rate is 1.0.

Since the third condition has the highest fall-into-the-range rate, it will be reported as the most important factor to make the function value to fall into the range. Therefore, the answer should be the factor ($a_1 = 3$ and $a_2 = 1$) which is exactly the same as the one produced by the normal data analysis method.

3.5 DAT-2

Each node in this DAT has either one, two or no descendants. Each inner node selects the best variable and usually makes two branches, but if the sub data set that has to be passed to a descendant is empty, no branch will be made. This DAT has the shape of binary tree. This tree can check three conditions for each variable : if the variable has certain value, does not have a certain value, or doesn't care what value it has.

The following is an example of DAT-2. Again, this DAT-2 is produced after checking the data set in table 2.1.

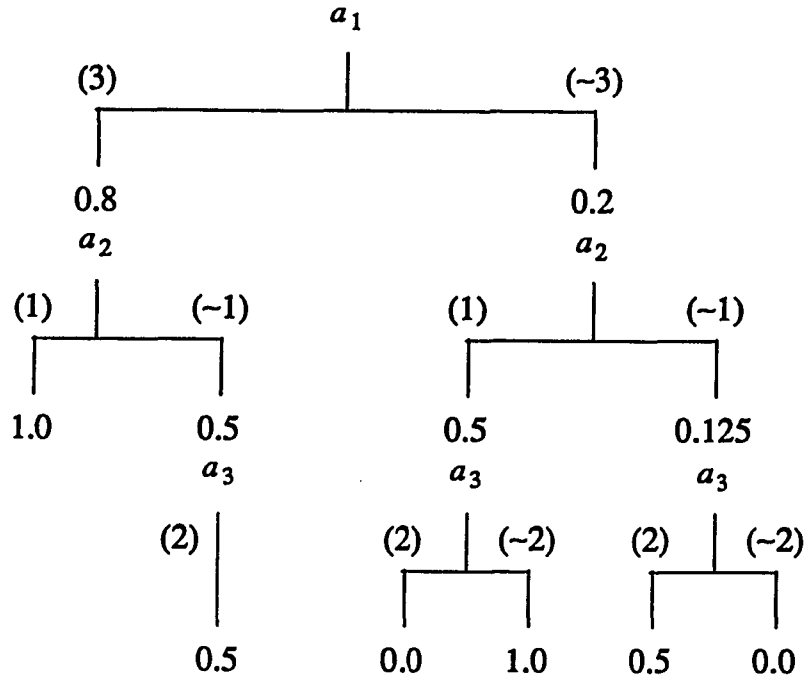

Figure 3.2 An Example of DAT-2

DAT-2 generates a tree in a similar way that DAT-1 does. Instead of making only one branch at a given node, it generates the maximum of two branches. DAT-2 stops generating the descendants at a given node when the fall-into-the-range rate at that node is either 1 or 0. That is why the left most node at the third level stops generating any more descendants. An interesting feature is that an inner node can generate one or two descendants. This feature is evident in the second left most node in the third level where no data for the case of right branch exists.

This tree produces two important factors for the highest fall-into-the-range rate. They are :

if $a_1 = 3$ and a_2=1, then the fall-into-the-range rate is 1.0.

if $a_1 \neq 3$, a_2=1, and $a_3 \neq 2$, then the rate is 1.0.

Unlike DAT-1 or normal data analysis algorithm which check the positive side of the variables, DAT-2 can generate factors with more information because it checks the negative side of the variables, too. This can be important because sometimes it is easier to control the variables not to have certain values than to have certain values.

3.6 DAT-3

Both DAT-1 and DAT-2 selected the most important value from each variable. DAT-1 checked the positive side of the variables' values only, and DAT-2 checked the negative side also. Since DAT-3 can make three branches at a given node, it doesn't have to select the most important value for each variable. It takes the data table as it is and starts generating trees. Each node in this DAT has either one, two, three or no descendants. Each inner node selects the best variable and should make three branches, but if the data subset that has to be passed to a descendant is empty, no branch will be made for that value.

The following is an example of DAT-3. Once again, this DAT-3 is produced after checking the data set in table 2.1.

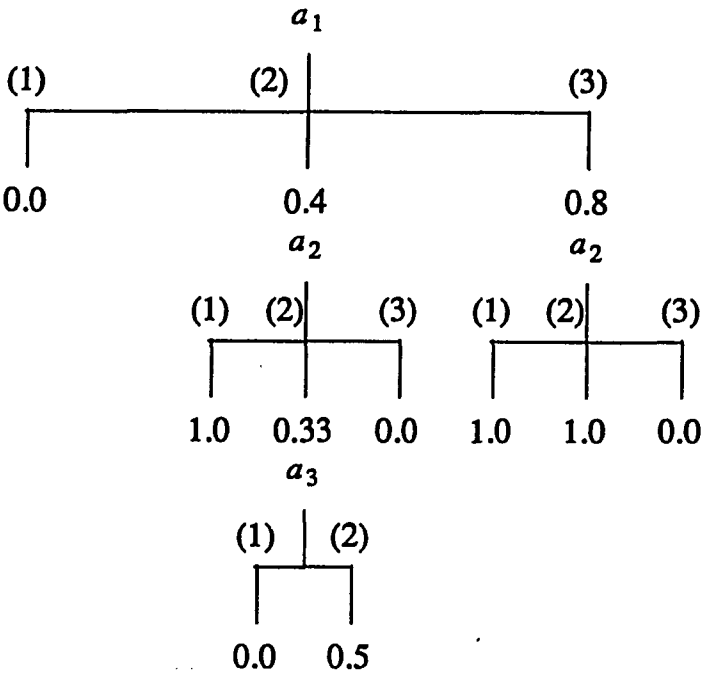

Figure 3.3 An Example of DAT-3

DAT-3 generates a tree with maximum of three branches for each inner node. DAT-3 also stops generating the descendants at a given node when the fall-into-the-range rate at that node is 1 or 0. This happens in the left most node of the second level.

This tree produces three important factors for the highest fall-into-the-range rate. They are :

if $a_1 = 2$ and a_2=1, then the fall-into-the-range rate is 1.0.

if $a_1 = 3$ and a_2=1, then the fall-into-the-range rate is 1.0.

if $a_1 = 3$ and a_2=2, then the fall-into-the-range rate is 1.0.

Unlike DAT-1, DAT-2, or normal data analysis algorithm which con-

sider only one value for a variable, DAT-3 takes all three values for each variable into consideration, so it generates the factors with more information.

CHAPTER 4. ALGORITHMS OF DAT'S

4.1 Input Data Table

All of the DAT-family algorithms are derived from the Decision Tree algorithm, so the algorithms for each DAT is very similar to each other. Because the number of branches that each node generates is different from each other, the algorithms also have some minor differences. The input data set is a table like in table 4.1 for all DAT algorithms. The values of the variables are 1, 2, or 3 which represent some specific categories or clusters.

Table 4.1 Data Set for DAT Algorithms

a_1	a_2	a_3	C
1	1	2	0
2	1	1	1
2	2	2	1
3	1	1	1
1	3	1	0
3	2	2	1
1	3	1	0
2	2	2	0
3	3	2	0
1	2	1	0
1	2	1	0
3	1	2	1
2	3	1	0
3	1	1	1
2	2	1	0

Table 4.1 came from the table 2.1 after direct conversion of the values of the variables. The values in a_1, a_2, and a_3 represent the specific categories or clusters. Then the table 4.1 is converted into the table 4.2 by collecting the same set of variable values and giving the fall-in-the-range rate. The number of observations that fall into the range are given in the column *Fall*, and the total number of observations are given in the column *Total*.

Table 4.2 Collected Data Set for DAT Algorithms

a_1	a_2	a_3	*Fall*	*Total*
1	1	2	0	1
1	2	1	0	2
1	3	1	0	2
2	1	1	1	1
2	2	1	1	1
2	2	2	1	2
2	3	1	0	1
3	1	1	2	2
3	1	2	1	1
3	2	2	1	1
3	3	2	0	1

Now, it is easy to see the fall-in-the-range rate for the existing observations.

4.2 DAT-1

4.2.1 The Significant Values

DAT-1 has to find the most significant value for each variable. To do so, it is necessary to compare the fall-into-the-range rate for the condition of each value.

For a_1 :

if $a_1 = 1$, then the rate is $0 / 5 = 0.0$

if $a_1 = 2$, then the rate is $3 / 5 = 0.6$

if $a_1 = 3$, then the rate is $4 / 5 = 0.8$

For a_2 :

if $a_2 = 1$, then the rate is $4 / 5 = 0.8$

if $a_2 = 2$, then the rate is $3 / 6 = 0.5$

if $a_2 = 3$, then the rate is $0 / 4 = 0.0$

For a_3 :

if $a_3 = 1$, then the rate is $4 / 9 = 0.44$

if $a_3 = 2$, then the rate is $3 / 6 = 0.5$

The factor with the most significant values is ($a_1 = 3$, $a_2 = 1$, $a_3 = 2$). The significant value for a_i is called as v_i. Thus, $v_1 = 3$, $v_2 = 1$, and $v_3 = 2$.

4.2.2 Algorithm

After the most significant values are found, DAT-1 follows the steps given in the next page.

< Initialization >

-. Let L be a list of variables a_i's along with their significant values v_i's, and initialize L to be the empty set.

-. Let D be a set of observations and initialize it as a set of all the observations.

-. Calculate the over all fall-into-the-range rate, and call it r.

< Algorithm >

1) Get L, D, and r from the predecessor.

2) If L contains all the variables in the original data set, report L and r to the predecessor and terminate.

3) Calculate the fall-into-the-range rate r_i for each and every variable a_i not in L having its significant value v_i, and select the variable a_k which has the largest fall-into-the-range rate. Call the largest rate r_{max}.

4) Let $L' = L \cup a_k$.

   If $r_{max} = 1.0$ then report r_{max} and L' to the predecessor and terminate.

   If $r_{max} = 0.0$ then report r and L to the predecessor and terminate.

5) Let D' be a subset of D where every $d \in D'$ satisfies the condition that $a_k = v_k$.

6) If D is not empty, call this algorithm recursively passing V', D', and r_{max} as new V and D for its descendant. Get the fall-into-range rate r'

and the list L' from the descendant.

7) Compare r and r'. If $r \geq r'$, report r and L; if $r < r'$, then report r' and L' to the predecessor.

4.2.3 Tracing the Algorithm with an Example

Table 4.2 which has 3 variables and 11 observation is used as an input data set in this example.

< At Node 1 >

$L = \{\ \}$

D = all the observations in the table

$r = 6 / 15\ (0.4)$

$r_1 = 4 / 5\ (0.8)$

$r_2 = 4 / 5\ (0.8)$

$r_3 = 3 / 6\ (0.5)$

$r_{max} = r_1$

$a_k = a_1$

$L' = \{\ a_1 = 3\ \}$

D' = all the observations that satisfies L'

Call node 2 passing L', D', and r_{max}.

< At Node 2 >

$L = \{\ a_1 = 3\ \}$

D = all the observations that satisfies L

$r = 4/5\ (0.8)$

$r_2 = 3/3\ (1.0)$

$r_3 = 2/3\ (0.66)$

$r_{max} = r_2$

$a_k = a_2$

$L' = \{ a_1 = 3, a_2 = 1 \}$

Since r_{max} = 1.0, report r_{max} and L' to node 1.

< At Node 1 >

$L = \{ \}$

$r = 0.4$

$L' = \{ a_1 = 3, a_2 = 1 \}$

$r' = 1.0$

Since $r' \geq r$, report L' and r'.

4.2.4 Review of the Algorithm

DAT-1 produced the result of (a_1 = 3, a_2 = 1). It produced the same result as the normal data analysis algorithm but in less time. It is certain that DAT-1 uses less time than Data Analysis method since DAT-1 checks only n combinations in worst case to produce the most important factor for the function value to fall into the range.

$a_1 = 3$

$a_1 = 3$ and $a_2 = 1$

Those two combinations are checked in the above example of DAT-1. Compared to 7 combinations in the normal Data Analysis method, DAT-1 produce the result very quickly, but can DAT-1 always produce the correct answer? Unfortunately, the answer is no which can easily be proved by Theorem 4.1

Theorem 4.1

The most important factor found by DAT-1 has a lower or equivalent fall-into-the-range rate than the factor found by the normal Data Analysis algorithm.

□ *Proof*

1) Let U be a set of all possible conditions, such that each condition contains any number between 1 and n number of distinct a_i's to have its significant value v_i, for any $1 \leq i \leq n$.

   U is the set of conditions that are checked by the Data Analysis algorithm.

2) Let $a_{k_1}, a_{k_2}, \ldots, a_{k_n}$ be the attributes which are checked in level 1, level 2, ... , level n respectively in DAT-1, along with their values $v_{k_1}, v_{k_2}, \ldots v_{k_n}$.

3) DAT-1 checks the following n conditions.

   $(a_{k_1} = v_{k_1})$

   $(a_{k_1} = v_{k_1}$ and $a_{k_2} = v_{k_2})$

.....

$(a_{k_1} = v_{k_1}, a_{k_2} = v_{k_2}, \ldots a_{k_1} = v_{k_1})$

Let L be a set of conditions that has the above n conditions as its elements.

3) For any condition c, if $c \in L$, then $c \in U$.

For all c which does not contain $a_{k_1} = v_{k_1}$, $c \notin L$, but $c \in U$.

That means $L \subset U$.

4) Since all the conditions in L are checked in Data Analysis algorithm, and not all the conditions in U are checked by DAT-1, it is clear that the highest fall-into-the-range rate found by Data Analysis is greater than or equals to the rate found by DAT-1.

The data set which makes DAT-1 produce the wrong answer should have the following conditions. Only two groups of variables - a_i for one group, and a_j and a_k for another - are mentioned here, but it can be generalized into two groups of any numbers.

-. a_i is the most significant variable among a_i, a_j, and a_k.

-. The condition $a_i = v_i$ combined with any other condition reduces the fall-into-the-range rate.

-. The condition $a_j = v_j$ and $a_k = v_k$ produces the higher fall-into-the-range rate than the condition $a_i = v_i$ or combination of all three conditions.

The experiment shows that this kind of case does not happen very often, but since it can happen, it must be handled. Just consider the following data set table.

Table 4.3 Data Set for DAT-1

a_1	a_2	a_3	*Fall*	*Total*
1	1	2	4	5
1	2	1	2	5
1	2	3	1	5
1	3	1	5	5
1	3	3	2	5
2	1	2	2	5
2	1	3	2	5
2	2	1	2	5
2	2	3	4	5
2	3	1	1	5
3	1	1	1	5
3	1	2	2	5
3	2	3	4	5
3	3	1	1	5
3	3	2	2	5

Applying the algorithm to find the most significant value for each variable to the above data set will result the following :

For a_1 :

if $a_1 = 1$, then the rate is $14 / 25 = 0.56$

if $a_1 = 2$, then the rate is $11 / 25 = 0.44$

if $a_1 = 3$, then the rate is $10 / 25 = 0.4$

For a_2 :

if $a_2 = 1$, then the rate is $11 / 25 = 0.44$

if $a_2 = 2$, then the rate is $13 / 25 = 0.52$

if $a_2 = 3$, then the rate is $11 / 25 = 0.44$

For a_3 :

if $a_3 = 1$, then the rate is $10 / 25 = 0.40$

if $a_3 = 2$, then the rate is $12 / 25 = 0.48$

if $a_3 = 3$, then the rate is $13 / 25 = 0.533$

The factor with the most significant values is ($a_1 = 1, a_2 = 2, a_3 = 3$). The algorithm in section 4.2.2 is applied as following.

< At Node 1 >

$L = \{ \}$

D = all the observations in the table

$r = 35 / 75$ (0.47)

$r_1 = 14 / 25$ (0.56)

$r_2 = 13 / 25$ (0.52)

$r_3 = 13 / 25$ (0.52)

$r_{max} = r_1$

$a_k = a_1$

$L' = \{ a_1 = 1 \}$

D' = all the observations that satisfies L'

Call node 2 passing L', D', and r_{max}.

< At Node 2 >

$L = \{ a_1 = 1 \}$

D = all the observations that satisfies L

$r = 14 / 25$ (0.56)

$r_2 = 3 / 10$ (0.3)

$r_3 = 3 / 10$ (0.3)

$r_{max} = r_2$

$a_k = a_2$

$L' = \{ a_1 = 1, a_2 = 2 \}$

D' = all the observations that satisfies L'

Call node 3 passing L', D', and r_{max}.

< At Node 3 >

$L = \{ a_1 = 1, a_2 = 2 \}$

D = all the observations that satisfies L

$r = 3 / 10$ (0.3)

$r_3 = 1 / 5$ (0.2)

$r_{max} = r_3$

$a_k = a_3$

$L' = \{ a_1 = 1, a_2 = 2, a_3 = 3 \}$

D' = all the observations that satisfies L'

Call node 4 passing L', D', and r_{max}.

< At Node 4 >

$L = \{ a_1 = 1, a_2 = 2, a_3 = 3 \}$

D = all the observations that satisfies L

$r = 1 / 5$ (0.2)

Since L includes all the variables, report L and r.

< At Node 3 >

$L = \{ a_1 = 1, a_2 = 2 \}$

$L' = \{ a_1 = 1, a_2 = 2, a_3 = 3 \}$

$r = 0.3$ and $r' = 0.2$

Since $r \geq r'$, report L and r.

< At Node 2 >

$L = \{ a_1 = 1 \}$

$L' = \{ a_1 = 1, a_2 = 2 \}$

$r = 0.56$ and $r' = 0.3$

Since $r \geq r'$, report L and r.

< At Node 1 >

$L = \{ \}$

$L' = \{ a_1 = 1 \}$

$r = 0.47$ and $r' = 0.56$

Since $r' \geq r$, report L' and r'_{max}.

Therefore, DAT-1 reported $a_1 = 1$ as the most important factor for the function values to fall into the range. Now, we apply the normal Data Analysis method to the same data set. It selects the most significant value for each variable like DAT-1 and checks the following seven conditions.

if $a_1 = 1$, then $r = 14 / 15$ (0.56)

if $a_2 = 2$, then $r = 13 / 15$ (0.52)

if $a_3 = 3$, then $r = 13 / 15$ (0.52)

if $a_1 = 1$ and $a_2 = 2$, then $r = 3 / 10$ (0.3)

if $a_1 = 1$ and $a_3 = 3$, then $r = 3 / 10$ (0.3)

if $a_2 = 2$ and $a_3 = 3$, then $r = 9 / 15$ (0.6)

if $a_1 = 1$, $a_2 = 2$, and $a_3 = 3$, then $r = 1 / 5$ (0.2)

After checking all seven conditions, the Data Analysis algorithm reports the condition $a_2 = 2$ and $a_3 = 3$ as the most important factor for the function values to fall into the range. This problem happens because that DAT-1 checks only three conditions which include the condition of $a_1 = 1$ which was found to be the most significant node at the root while the true answer does not include the condition of $a_1 = 1$. This is a problem that can occur only when DAT-1 deals with the data set with the conditions given earlier, but it still proves that DAT-1 alone cannot get the job done correctly. For that reason, we need the back-tracking method to be introduced in the next section.

4.2.5 Back-Track Algorithm

This algorithm originated from the optimization method of the Decision Tree. In the Decision Tree algorithm this method is used to reduce the number of nodes in the tree, but here it is used to correct the wrong answer. This back-track algorithm can be applied directly to the tree, T, which has the over all best fall-into-the-range rate and which each node contains a variable a_k to check. This algorithm can be applied to T recursively starting from the root of T even though it generates correct answers for applying to the root most of the time.

< Algorithm >

1) Let a_k be the variable checked at the current node.
   Let T be the subtree of DAT-2 that has the current node at the root.
   Let r be the best fall-into-the-range rate for T.
2) Take the variable at the leaf of the tree, and call it a'_k.
3) Build a tree T' with a'_k at the current node.
4) Get the fall-into-the-range rate for T', and call it r'.
5) Compare r and r'.
   If $r' > r$ then take T' and a'_k as new T and a_k.
6) If the current node is not the leaf, apply the algorithm to the immediate descendant.

4.2.6 Tracing of the Back-Track Algorithm

< Back-Track At Node 1 >

$a_k = a_1$

$r = 0.56$

$a'_k = a_3$

Apply DAT-1 algorithm here by calling node 2.

< DAT-1 At Node 2 >

$L = \{ a_3 = 3 \}$

$r = 0.53$

$r_1 = 3 / 10 \ (0.3)$

$r_2 = 9 / 15 \ (0.6)$

$L' = \{ a_3 = 3 \text{ and } a_2 = 2 \}$

< DAT-1 At Node 3 >

$L = \{ a_3 = 3 \text{ and } a_2 = 2 \}$

$r = 0.6$

$r_1 = 1 / 5 \ (0.2)$

$L' = \{ a_3 = 3, a_2 = 2, \text{ and } a_1 = 1 \}$

Since L' contains all the variables report r and L.

< DAT-1 At Node 2 >

Since $r' > r$, report r' and L'.

< Back-Track At Node 1 >

Since $r' > r$, let $r = 0.6$, and $a_k = a_3$.

< Back-Track At Node 2 >

$a_k = a_1$

$r = 0.6$

$a'_k = a_2$

Apply DAT-1 algorithm here by calling node 2.

< DAT-1 At Node 3 >

$L = \{ a_3 = 3 \text{ and } a_1 = 1 \}$

$r = 0.3$

$r_2 = 1 / 5 \ (0.2)$

$L' = \{ a_2 = 2, a_1 = 1, \text{ and } a_3 = 3 \}$

Since L' contains all the variables and $r > r'$, report r and L.

< Back-Track At Node 2 >

Since $r > r'$, don't change anything and report r and a_k.

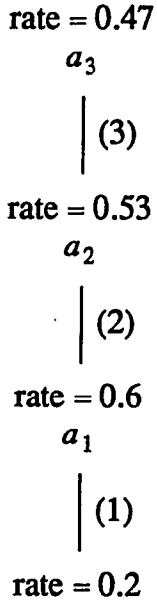

Figure 4.1 DAT-1 after Back-Tracking Algorithm

The figure 4.1 shows the tree produced by DAT-1 Back-Track algorithm for the data set in table 4.3. Notice that a_1 is at the leaf of the tree. That is because having a_2 at the top increases the significance of a_3, so a_3 is selected at level 2.

4.2.7 Time Complexity of DAT-1

Assume n is the number of variables in the data set. It takes n considerations for the DAT-1 to produce an answer without doing any back-tracking. To execute the back-tracking algorithm, it takes $n-1$ steps to consider the variable in the second level, $n-2$ steps to consider the variable in the third level, and so on. Therefore, the total time of a DAT-1 single pass and the back-tracking take $n + (n-1) + (n-2) + ... + 1$ considerations which is $n \times (n+1) / 2$. The time complexity for DAT-1 is, therefore, $O(n^2)$.

4.3 DAT-2

The main purpose of DAT-2 is to obtain more information than DAT-1 or normal data analysis can provide.

4.3.1 Algorithm

Like DAT-1, the DAT-2 algorithm also finds the most significant value from each and every variable. These are found exactly as they are found in the DAT-1 algorithm. Even though it has the same shape as binary decision tree, determining the most significant values is somewhat different than the decision tree method. The decision tree algorithm must locate the attribute (variable) along with its value that affects the value of the class value to be either 0 or 1. In DAT-2, the algorithm should find the variable along with its value that affects the value of the function value to fall into the range. That is, DAT-2 algorithm is interested in the function value being 1 not 0.

After the values of the most significant values are found, the DAT-2 algorithm follows the steps given below. The difference between the DAT-1 and the DAT-2 algorithms is that the DAT-2 makes two branches while the DAT-1 makes only one. One of two branches is the same as DAT-1, and another branch is under the condition that the variable does not have the significant value. Making two branches costs more time than the DAT-1 but allows the DAT-2 to have more information than the DAT-1 or the normal data analysis algorithms.

< Initialization >

-. Let L be a list of variables a_i's along with their significant values v_i's and initialize L to be the empty list.

-. Let D be a set of observations and initialize it as a set of all observations.

-. Calculate the over all fall-into-the-range rate, and call it r.

< Algorithm >

1) Get L, D, and r from the predecessor.

2) If L contains all the variables in the original the data set, report L and r to the predecessor and terminate.

3) Calculate the fall-into-the-range rate r_i for each and every variable a_i not in L having its significant value v_i, and select the variable a_k which has the largest fall-into-the-range rate. Call the largest rate r_L.

4) Let $L' = L \cup a_k = v_k$.
If $r_l = 1.0$ or $r_l = 0.0$ then let $r' = r_{max}$ and go to step (7).

5) Let D' be a subset of D where every $d \in D'$ satisfies the condition in L'.

6) If D' is not empty, call the algorithm recursively passing V', D', and r_L as the new V and D for its left descendant.
Get the fall-into-range rate r' and the list L' from the left descendant.

7) Let $L'' = L \cup a_k \neq v_k$.

Calculate the fall-into-the-range rate for the condition L'', and call the rate r_R.

If $r_R = 1.0$ or $r_{max} = 0.0$ then let $r'' = r_R$ and go to step (10).

8) Let D'' be a subset of D where every $d \in D'$ satisfies the condition in L''.

9) If D'' is not empty, call the algorithm recursively passing V'', D'', and r_R as new V and D for its right descendant.

Get the fall-into-range rate r'' and the list L'' from the right descendant.

10) Compare r, r', and r'' and report the largest rate and the corresponding list.

If any two or all three of the rates have the same value as the largest rate, report all corresponding lists.

4.3.2 DAT-2 as a Result of the Algorithm

The step-by-step tracing of DAT-2 is omitted because it generates so many nodes and operations. Instead, we will look at the data set and the resulting DAT-2. The table 4.2 which was used for the normal data analysis algorithm and the DAT-1 is used again as an input data set.

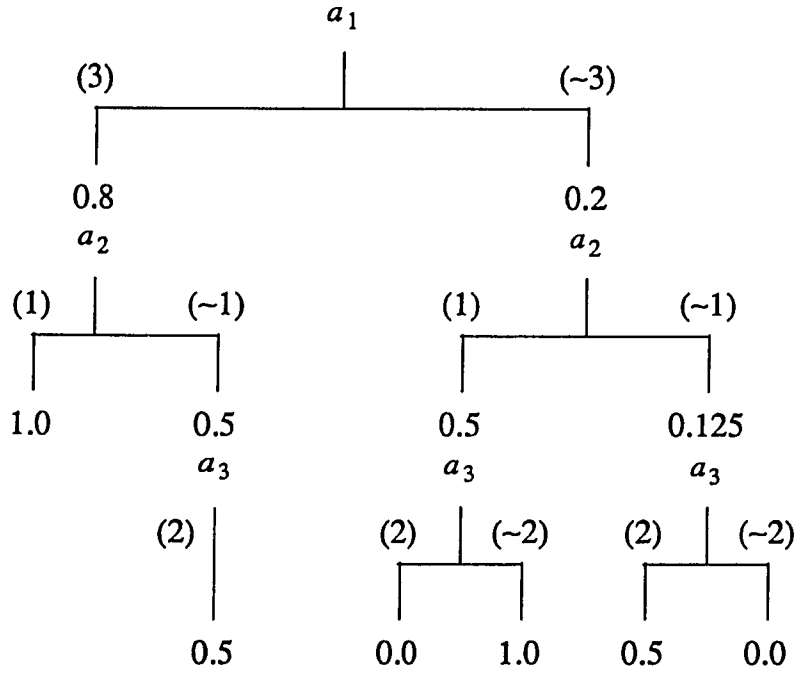

Figure 4.2 DAT-2 from the Data Set in Table 4.2

Note that the left-most node in each level is identical to the node in each level of DAT-1. That is because the left-most node in each level always checks the positive condition ($a_k = v_k$), and the most significant node at each level is selected as in DAT-1. Therefore, DAT-2 can be considered to be a tree which contains information about the data set including the information in DAT-1.

As a matter of fact, the factor which is found by DAT-2 has the same or higher fall-into-the-range rate than those factors found by DAT-1 and normal data analysis which is proved in the next section. Even if the rates found in DAT-1 and DAT-2 are the same, DAT-2 can give more factors that result in

the rate. As seen in this case, DAT-2 gives two factors to have the rate 1.0 while DAT-1 and normal data analysis method gives just one factor.

4.3.3 Facts in DAT-2

There are some interesting facts in DAT-2. As mentioned earlier, DAT-2 is designed to produce more information than the DAT-1 or the data analysis algorithm. Theorem 4.2 and 4.3 proves that the factor found by DAT-2 has a fall-into-the-range rate higher than or equal to the rates found by either DAT-1 or data analysis without having to consider the back-tracking algorithm.

Theorem 4.2

The most important factor found by DAT-2 has a higher fall-into-the-range rate than the factor found by either DAT-1 or the normal data analysis algorithm.

☐ Definitions

-. Let V be a set of variable a_i, such that the condition $a_i = v_i$, where v_i is the most significant value for a_i, is included in the most important factor L found by data analysis.

-. Let V' be a set of variables a_j, such that $a_j \notin V$.

-. Let function $F(C)$ return the fall-into-the-range rate when a set of conditions C are true.

□ Proof

1) If V' is empty, it means L contains all the conditions for every a_i in the data set to have its significant value v_i.
   So, it is clear that the condition L is also checked by DAT-2.
2) If V' is not empty, there exists $a_{j_1}, a_{j_2}, \ldots a_{j_k} \in V'$, and $F(L) \leq$ F(L $\cup (a_{j_1} = v_{j_1}) \cup (a_{j2} = v_{j2}) \cup \ldots \cup (a_{j_k} = v_{j_k})$) since the condition on the right hand side of $\leq$ is checked in the data analysis algorithm and not taken as the most important factor.
3) There are 2^k conditions that include L as a sub-condition, and $F(L)$ is the weighted average of those 2^k conditions.
4) Therefore, there exists a condition L' among those 2^k conditions, such that $F(L') \geq F(L)$.
   Therefore, the claim is true.

In DAT-1, the order of checking nodes is important because once a variable is selected in a node, the condition of the variable having the significant value is included in the all the nodes which are descendants of the node. Because of this, the back-tracking algorithm is necessary.

In DAT-2, the order of the nodes to be checked is not as important as in DAT-1 because the order of the nodes to be checked does not affect the highest fall-into-the-range rate found in DAT-2.

Theorem 4.3

In DAT-2, the level in the tree at which a node is checked makes no difference in the answer.

☐ Definitions

-. Let n be an inner node in DAT-2.

-. Let V be a set of variables a_i, such that the condition $a_i = v_i$, where v_i is the most significant value for a_i is included in the list L at node n.

-. Let V' be a set of variables a_j, such that $a_j \notin V$.

-. Let function $F(C)$ return the fall-into-the-range rate when a set of conditions C are true.

☐ Proof

1) Since n is an inner node, V' is not empty, and there exists $a_j \in V'$. Since $F(L)$ is a weighted average of $F(L \cup (a_j = v_j))$ and $F(L \cup (a_j \neq v_j))$, one of the following three conditions is true.

$$F(L \cup (a_j = v_j)) > F(L) > F(L \cup (a_j \neq v_j))$$

$$F(L \cup (a_j = v_j)) = F(L) = F(L \cup (a_j \neq v_j))$$

$$F(L \cup (a_j = v_j)) < F(L) < F(L \cup (a_j \neq v_j))$$

2) If the first or second condition is true, the left descendant has the fall-into-the-range rate higher than or equal to the one in n.
   If the second or third condition is true, the left descendant has the fall-into-the-range rate higher than or equal to the one in n.

3) Therefore, n has at least one descendant which has a higher or equal rate than n.

4) Since n is an inner node, the highest fall-into-the-range rate can be found at the leaves.

5) All of the variables are checked at the leaves either having the significant value or not. Therefore, the order of variables doesn't make any difference in the results.

As stated in Theorem 4.3, it makes no difference which node is checked at any given inner node. This leads to a conclusion that the back-tracking algorithm is not necessary for DAT-2.

4.3.4 Time Complexity of the Algorithm

It takes 2^{n+1} considerations to produce the results in the worst case. The time complexity is $O(2^n)$ which is the same as the normal data analysis method while DAT-2 produces more information. Additionally, the average time complexity can be reduced for the case in which the branching is stopped because either D is empty or the rate is 1.0 or 0.0 at any given inner node.

4.4 DAT-p

Let DAT-p be the Data Analyzing Tree with p branches at any given inner node. The number of possible branches p is desirable to match with the number of possible values that each variable can take. Since three values for each variable is used in data analysis, DAT-3 is sufficient in normal cases. Therefore, we will concentrate on DAT-3 here, but the tree can have more than three branches if the number of values increases. DAT-3 takes more time than DAT-1 or DAT-2 but produces more information than either.

4.4.1 Algorithm

It is not necessary to find the most significant value for each variable because DAT-3 checks all the possibilities of the variables with their possible values.

< Initialization >

-. Let L be a list of variables a_i's along with their significant values v_i's, and initialize L to empty.

-. Let D be a set of observations and initialize it as a set of all observations.

-. Calculate the over all fall-into-the-range rate, and call it r.

< Algorithm >

1) Get L, D, and r from the predecessor.

2) If L contains all the variables in the original data set, report L and r to the predecessor and terminate.

3) Calculate the significance of each and every variable by weighted average of the fall-into-the-range rate for each value of the variable.
Select the variable a_k with the largest significance.

4) Calculate the fall-into-the-range rates for the conditions that a_k has v_1, v_2, and v_3, and call the rates r_1, r_2, and r_3.
For each i from 1 to 3, let $L_i = L \cup a_k = v_i$, and let D_i be all the observations that satisfies the condition L_i.

5) For each i from 1 to 3, do the following. If $r_i = 1.0$ or $r_i = 0.0$ or D_i is empty, let $r'_i = r_i$ and do not make any branch for v_i.
If not, make a branch passing L_i, D_i, and r_i to each descendant, and receive r'_i and L'_i from each descendant.

6) Compare r, r'_1, r'_2, and r'_3, and report the largest rate r'_i and the corresponding list L'_i.
If any two or all three of the rates have the same value as the largest rate, report all corresponding lists.

4.4.2 DAT-3 as a Result of the Algorithm

Table 4.2 is once again used here as an input data set, and figure 4.3 is the resulting DAT-3.

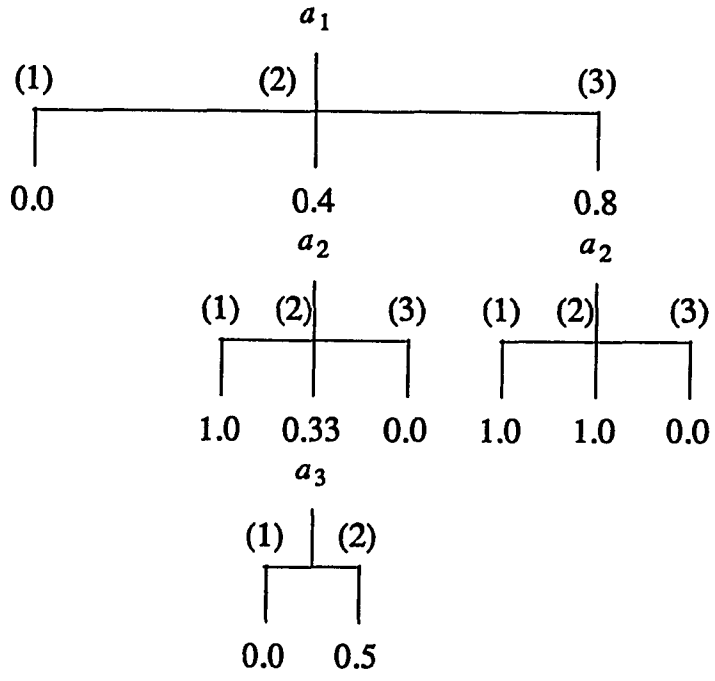

Figure 4.3 An Example of DAT-3

Since all the values of the variables in the data set are considered, many interesting things that could not be found from DAT-1 or DAT-2 are found here. The results are more specific than those from DAT-1 or DAT-2 which allows the users more information to control the variable values.

4.4.3 Facts of DAT-3

The fact that DAT-3 can find the factor which has the fall-into-the-range rate higher than or equal to the rate of the factor that DAT-1, DAT-2, or the normal data analysis algorithm is proved in this section.

Theorem 4.4

In DAT-3, the factor with the highest fall-into-the-range rate is found at the leaves. Therefore, the level in the tree at which a node is checked makes no difference in the answer.

☐ Definitions

-. Let n be an inner node in DAT-3.

-. Let V be a set of variables a_i, such that the condition $a_i = v_j$, where v_j is the value found to reach the current node n is included in the list L at node n.

-. Let V' be a set of variables a_j, such that $a_j \notin V$.

-. Let function $F(C)$ return the fall-into-the-range rate when a set of conditions C are true.

☐ Proof

1) Since n is an inner node, V' is not empty and there exists $a_j \in V'$. Since $F(L)$ is a weighted average of $F(L \cup (a_j = v_1))$, $F(L \cup (a_j = v_2))$, and $F(L \cup (a_j = v_3))$, there exist an integer k, $1 \le k \le 3$, such that $F(L \cup (a_j = v_k)) \ge F(L)$.

2) Therefore, n has at least one descendant which has a higher or equal rate than n.

3) Since it is true for all inner nodes, the highest fall-into-the-range rate can be found at the leaves.

4) All of the variables are checked at the leaves either having the significant value or not. Therefore, the order of variables doesn't make any difference in the results.

Theorem 4.5

The most important factor found by DAT-3 has a higher fall-into-the-range rate than the factor found by DAT-1, DAT-2, or the normal data analysis algorithm.

□ Definitions

-. Let V be a set of variables a_i, such that the condition $a_i = v_i$, where v_i is the most significant value for a_i or $a_i \neq v_i$ is included in the most important factor L found by the DAT-2.

-. Let V' be a set of variables a_j, such that $a_j \neq v_j$ is included in L.

-. Let function $F(C)$ return the fall-into-the-range rate when a set of conditions C are true.

□ Proof

1) If V' is empty, L contains all the conditions for every a_i in the data set to have its significant value v_i. Therefore, it is clear that the condition L is also checked by DAT-3.

2) If V' is not empty, there exists $a_j \in V'$. The condition $F(a_j \neq v_j)$ which is included in L is the weighted average of $F(a_j = v_{j_1})$ and $F(a_j = v_{j_2})$ where $v_j \neq v_{j_1} \neq v_{j_2}$. Therefore, there exists a value v'_{j_i}

that is either v_{j_1} or v_{j_2} for a_{j_i}, such that $F(a_j = v'_{j_i}) \geq F(a_j = v_j)$, that is $F((L - (a_j = v_j)) \cup (a_j = v'_{j_i})) \geq F(L)$.

3) Since (2) is true for all $a_{j_i} \in V'$, that is $F((L - (a_{j_1} = v_{j_1}) - (a_{j_2} = v_{j_2}) - ..... - - (a_{j_k} = v_{j_k})) \cup (a_{j_1} = v'_{j_1}) \cup (a_{j_1} = v'_{j_1}) \cup ..... \cup \cup (a_{j_k} = v'_{j_k})) \geq F(L)$.

4) The left hand side of (3) is the condition that is checked by DAT-3, so the claim is true.

4.4.5 Time Complexity of the Algorithm

It takes 3^n considerations to produce the answers in the worst case in DAT-3. The time complexity is $O(3^n)$ which is worse than the normal data analysis method or any other DAT algorithms, but DAT-3 produces the most accurate information. Its average time complexity can be reduced for the case that the branching is stopped when either D is empty or the rate is 1.0 or 0.0 at any given inner node.

DAT-3 is not suitable for a quick answer for the data set. It should be used for rather important and continuous usage of the data set to give a more accurate and complete answer for the data set.

4.5 Clustered DAT

When the maximum number of possible values for the variables in the data set is larger than three, clustered DAT may possibly reduce the number of iterations. The algorithm for this method is very similar to the DAT-*p*

algorithm except that this method takes several values that make the fall-into-the-range rate very close to each other to make just one branch for them.

< Algorithm >

1) Get L, D, and r from the predecessor.

2) If L contains all the variables in the original data set, report L and r to the predecessor and terminate.

3) Calculate the significance of each and every variable by the weighted average of the fall-into-the-range rate for each variable value.
Select the variable a_k with the largest significance.

4) Calculate the fall-into-the-range rates for the conditions that a_k has v_1, v_2, ... ,v_p, and call the rates r_1, r_2, ... , v_p.
Make q intervals between 0 and 1, and call those intervals I_1, I_2, ... ,I_q.
For each I_i from 1 to q, let $L_i = L \cup (a_k$ has the value that results in a fall-into-the-range rate in I_i), and let D_i be all the observations that satisfies the condition L_i. Calculate the average of the fall-into-the-range rate, and call the average for the interval I_i, s_i.

5) For each i from 1 to q, do the following. If $s_i = 1.0$ or $s_i = 0.0$ or D_i is empty, let $s'_i = s_i$ and do not make any branch for I_i.
If not, make a branch passing L_i, D_i, and s_i to each descendant, and receive s'_i and L'_i from each descendant.

6) Compare s, s'_1, s'_2, ... , s'_q, and report the largest rate s'_i and the corresponding list L'_i.

7) If any two or all three of the rates have the same value as the largest rate, report all corresponding lists.

This method should be used when the number of possible values of the variables is large to reduce the number of iterations. This method cannot give a result as accurate as DAT-p, but it reduces the amount of time. If the average of q clusters are formed at each level ($q < p$), the time complexity will be $O(q^n)$.

4.6 Algorithms for Other Usages of DAT's

4.6.1 Cut-Off Method

Up to this point, the purpose of DAT was to find the factor that results in the maximum fall-into-the-range rate. However, it is quite possible that the user of DAT is interested in the minimum set of the variables that guarantees certain fall-into-the-range rate.

In this case, the desired rate can be set as a cut-off value, and the algorithm for DAT-2 and DAT-3 can be altered as follows to save time to produce the correct answer.

-. Let SL be an integer which keeps track of the level not to branch, and initialize it to n.

-. Apply the algorithm with one more rule that stops branching if the level is SL.

-. Whenever the fall-into-the-range rate reaches or exceeds the cut-off value, record the current level as SL, and stop branching at the current node.

This cut-off rule can reduce the search time because it reduces the number of nodes to be checked. Both DAT-2 and DAT-3 can guarantee the minimum set of variables for the cut-off value.

DAT-1 cannot guarantee that the answer is minimum. The reason is that DAT-1 branches only one way, so there is always the possibility that the factor with fewer conditions are not checked. Therefore, DAT-1 must go through the back-tracking algorithm to make sure the answer has a minimum of conditions, and the time saving cannot be expected. Let's check the following example.

$$P(a_1 = v_1, a_2 = v_2, a_3 = v_3) = 9/10 \ (0.9)$$

$$P(a_1 = v_1, a_2 = v_2, a_3 \neq v_3) = 8/10 \ (0.8)$$

$$P(a_1 = v_1, a_2 \neq v_2, a_3 = v_3) = 7/10 \ (0.7)$$

$$P(a_1 = v_1, a_2 \neq v_2, a_3 \neq v_3) = 8/10 \ (0.8)$$

$$P(a_1 \neq v_1, a_2 = v_2, a_3 = v_3) = 9/10 \ (0.9)$$

$$P(a_1 \neq v_1, a_2 = v_2, a_3 \neq v_3) = 4/10 \ (0.4)$$

$$P(a_1 \neq v_1, a_2 \neq v_2, a_3 = v_3) = 5/10 \ (0.5)$$

$$P(a_1 \neq v_1, a_2 \neq v_2, a_3 \neq v_3) = 1/10 \ (0.1)$$

Assuming 0.9 is the desired cut-off value, the DAT-1 algorithm will check the following three conditions :

$F(a_1 = v_1) = 32/40$ (0.8)

$F(a_1 = v_1$ and $a_2 = v_2) = 17/20$ (0.85)

$F(a_1 = v_1, a_2 = v_2,$ and $a_3 = v_3) = 9/10$ (0.9)

and report ($a_1 = v_1$, $a_2 = v_2$, and $a_3 = v_3$) as the answer which satisfies the cut-off condition. However, the actual minimum factor that satisfies the cut-off value is $F(a_2 = v_2$ and $a_3 = v_3) = 9/10$ (0.9), which has one less condition than the factor found by DAT-1. Therefore, the back-tracking method is again needed for the DAT-1 cut-off method.

4.6.2 Optimization

The main purpose of data analysis is to find the most important factor(s) for any given data set. However, keeping DAT for the later use can be as important as just finding the most important factor. It is possible that one variable is harder to control than other variables. For example we can assume that a variable temperature can be easily controlled, but another variable material is not as easy to control because the price of materials vary considerably. The user may want a different answer without a hard variable to control in the factor. In this case, DAT's can be used as a decision tree. DAT's can give the fall-into-the-range rate for the specific set of conditions of variables to have certain values. That makes the optimization method very important for efficient usage of DAT's.

In DAT-1, it is not possible to optimize the tree because it checks only one direction of the variables, so changing the order of the variables to be

checked makes difference in the results. However, theorems 4.3 and 4.4 show that the order of the variables to be checked in the algorithm does not effect the answer in DAT-2 and DAT-3. Even though changing the order of the variables does not effect the answer, it makes big difference in the number of nodes in DAT-2 or DAT-3. There is always a possibility that branching can be stopped by either reaching the point that the D is empty or the fall-into-the-range rate is 1.0 or 0.0. Therefore, finding the most significant node at a given level is very important for the sake of efficiency, and the optimization method that is used to optimize the decision tree can be applied here for DAT-2 and DAT-3 to reduce the number of nodes in the tree.

DAT is said to be optimal if it has the minimum number of nodes to generate the correct answer. The optimization algorithm used to make DAT-2 or DAT-3 optimal is quite similar to the decision tree optimization algorithm introduced in section 2.5 and the back-tracking algorithm mentioned in section 4.2.5. Again this algorithm is applied to the root of the DAT's which is applied to all the nodes recursively.

< Algorithm >

1) Let a_k be the variable checked at the current node.
   Let T be the subtree of DAT that has the current node at the root.
   Let d be the number of nodes of T.
2) Take the immediate left descendant's variable and call it a_L.
   Take the immediate right descendant's variable and call it a_R.

3) Build a tree T_L with a_L at the current node.

4) Get the number of nodes for T_L, and call it d_L.
Get the number of nodes for T_R, and call it d_R.

5) Find the minimum value among d, d_L, and d_R, and take the corresponding sub-tree and variable as new T and a_k.

6) If the current node is not a leaf, apply the algorithm to the both left and right descendants.

4.7 Advantages and Disadvantages of DAT's

4.7.1 DAT-1

< Advantages >

-. DAT-1 is the fastest method to find the most important factor for the function values.

-. DAT-1 is suitable when a quick answer is needed.

< Disadvantages >

-. DAT-1 needs a back tracking which makes the algorithm complicated.

-. DAT-1 is not suitable when an accurate and complete answer is needed.

4.7.2 DAT-2

< Advantages >

-. DAT-2 is as fast as the normal data analysis method while providing a more accurate and complete answer.

-. Back-tracking method is not necessary.

-. DAT-2 is suitable for the long time usage of the tree.

-. DAT-2 can save time for cut-off input.

-. Optimization is possible .

< Disadvantages >

-. DAT-2 takes exponential time to complete.

-. DAT-2 is not suitable when a quick answer is needed from the data set with many variables.

4.7.3 DAT-*p*

< Advantages >

-. DAT-*p* produces the most and complete answer.

-. Back-tracking method is not necessary.

-. DAT-*p* is suitable for the long time continuous usage of the tree.

-. DAT-*p* can save time for cut-off input.

-. Optimization is possible.

< Disadvantages >

-. DAT-p takes $O(p^n)$ to complete.

-. DAT-p is not suitable when a quick answer is needed from a data set with many variables.

CHAPTER 5. EXPERIMENTAL RESULTS

5.1 General Information of the Experiment

From the algorithms and examples, the performances of DAT's are clearly described, but to make sure that those algorithms work for any type of data set, the following experiment was performed. Since it is difficult to gather data sets which are used in real life, the data sets used in this experiment were generated by using a random number generator. The details of the data set generation, experiment, and the result are explained in this chapter.

5.2 Data Set Generator

The main purpose of this generator is to generate the data set in a compact form which is similar to a real data set. The data set that is used in the data analysis has following characteristics :

-. The number of variables n in the data set is given.

-. There are three possible values, 1, 2, and 3, for each variable, which is decided by the random number generator.

-. The probability that a variable can have any value is 1 / 3 because the usual data analysis uses clustering technique.

-. The number of observations is random between 2^{n-1} and 2^n.

-. There is an important value randomly generated u_i for each variable a_i, where u_i can be 0, 1, 2, or 3.

- No duplicate observations appear in the data set.
- An observation O is said to be important if each variable a_i in O has the value v_i, such that $v_i = u_i$ where $u_i > 0$.
- For each observation, there is a random non-negative integer f and a random positive integer t, which means that out of t tries, the function value falls into the desired range f times. Therefore, f / t is the fall-into-the-range rate for the observation, and the rate is higher for the important observations.

Therefore, when the number of variables n is given, the data set generator produces the data set which has the somewhat important factor that the fall-into-the-range rate becomes higher. The difference between the rate for the normal observations and the rate for the important factor is also randomly decided by the generator. This makes the data set quite similar to the real data set which makes the experiment more realistic.

5.3 Steps in the Experiment

To compare the performances of the algorithms, they must be applied to the same data set. After the data set is generated, the normal data analysis method, DAT-1, DAT-1 with back-tracking, DAT-2, and DAT-3 algorithms are applied to the same data set. The number of checkings that each method used and the resulting fall-into-the-range rates that are found by each algorithm are reported. One hundred data sets are generated with each of three, four, five, and six columns and used for each algorithm.

5.4 Result of the Experiment

Table 5.1 through table 5.4 are tables that contains the result that the algorithms produced for one hundred data sets with three, four, five, and six columns. As shown in table 5.1 which deals with the data set with three columns, DAT-1 without back-tracking produces exactly the same results as the normal data analysis method even though it uses less than a half the iterations that the normal data analysis method uses. Since all the answers produced by normal data analysis and DAT-1 are the same, the performance of DAT-1 with back-tracking cannot be analyzed here. DAT-2 increases the overall fall-into-the-range rate by 1.12% with the same time complexity as the normal data analysis, and DAT-3 increases 4.25% of the over-all rate. Also, DAT-2 and DAT-3 produce multiple answers several times which can allow the user to select the factor of the variables that are easier to control.

Somewhat interesting and important behavior of the DAT's are found in this experiment. The performances of DAT-2 and DAT-3 are very closely related to the number of variables in the important factor found by the normal data analysis method or DAT-1 algorithm. Of the six cases in which the normal data analysis produced important factors with one variable, DAT-2 and DAT-3 made the differences in the fall-into-the-range-rate each time. Out of the 38 cases in which the normal data analysis method produced important factors with two variables, DAT-2 and DAT-3 made the differences in the fall-into-the-range-rate for 35 and 36 times, respectively. However, out of the 56 cases that the normal data analysis produced important factors with three vari-

ables, DAT-2 and DAT-3 made the differences in the fall-into-the-range-rate for only 1 and 27 times, respectively.

DAT-3 increases the fall-into-the-range rate from those of DAT-1 and DAT-2. It is the most accurate data analyzer for the data set with three possible values for the variables. Of course it takes too much time to be compared to the other methods, but it can be considered as a method to analyze the data very deeply if time permits.

Table 5.2 through 5.4 shows clearly that DAT-1 without back-tracking sometimes produces the fall-into-the-range rate which is less than the rate that the normal data analysis produces. Out of 100 trials with the data sets of four, five, and six columns, just 6, 6, and 11 cases made the differences in answers of DAT-1 and normal Data Analysis respectively. All other cases produced exact same answers for both algorithms. Even for those cases that DAT-1 and the normal data analysis method produce two different answers, DAT-1 with back-tracking produces exactly the same results as the normal data analysis algorithm.

The most important fact illustrated by this experiment is that the performances of the DAT's get better as the number of variables in the data set increases. For DAT-2, only 42% of the total cases make differences in the fall-into-the-range rates when the number of variables is three, but the percentages of differences in the rates became 56%, 74% and 82% for the case of four, five, and six variables in the data set respectively. Also for DAT-3, the percentages of making differences in the fall-into-the-range rates grow from

69% to 79%, 96%, and 97% as the number of variables in the data set increases from three to four, five, and six respectively.

These differences can be explained by the percentage that all the variables are included in the most important factor found by DAT-1. It is already pointed out that the efficiencies of DAT-2 and DAT-3 have something to do with the number of variables found by the normal data analysis or DAT-1 with back-tracking. As the number of variables in the data set increases, the chance of having all the variables in the important factor found by DAT-1 decreases. With three variables in the data set, all three variables are included in the important factor found by DAT-1 for 56 percent of the time. It decreases to 40 percent when the number of variables is four, becomes 22 percent when the number of variables is five, and becomes only 20 percent when the number of variables is six. With the chance of having all the variables in the important factor by DAT-1 decreasing, it is very natural that the number of the cases that DAT-2 and DAT-3 produce more accurate answers increases.

Therefore, as the number of columns of the data set increases in the data set, not only does the difference between the normal data analysis and DAT-1 in the number of iterations gets bigger, but also the chance of making differences in the fall-into-the-range rate from the normal data analysis by using DAT-2 or DAT-3 becomes larger.

Table 5.1 Result of the Experiment for Column = 3

3 Columns		N.D.A	DAT-1	DAT-1(BT)	DAT-2	DAT-3
Averages		74.67%	74.67%	74.67%	75.79%	78.92%
Iterations		7.00	3.00	6.00	12.98	37.24
Multiple Answers	2-Answers	0	0	0	3	2
	3-Answers	0	0	0	2	1
	4-Answers	0	0	0	0	1
Cases of Different Answers from N.D.A	Total	-	0/100	0/100	42/100	69/100
	1-var	-	0/6	0/6	6/6	6/6
	2-var	-	0/38	0/38	35/38	36/38
	3-var	-	0/56	0/56	1/56	27/56
Averages of Different Answers from N.D.A	Total	-	0.0%	0.0%	2.67%	6.15%
	1-var	-	0.0%	0.0%	4.42%	13.54%
	2-var	-	0.0%	0.0%	2.40%	6.80%
	3-var	-	0.0%	0.0%	1.63%	3.64%

Table 5.2 Result of the Experiment for Column = 4

4 Columns		N.D.A	DAT-1	DAT-1(BT)	DAT-2	DAT-3
Averages		76.02%	75.91%	76.02%	77.56%	80.79%
Iterations		15.00	4.00	10.00	28.03	111.40
Multiple Answers	2-Answers	0	0	0	4	9
	3-Answers	0	0	0	1	1
Cases of Different Answers from N.D.A	Total	-	6/100	0/100	56/100	79/100
	1-var	-	0/0	0/0	0/0	0/0
	2-var	-	0/7	0/7	7/7	7/7
	3-var	-	6/53	0/53	47/53	52/53
	4-var	-	0/40	0/40	2/40	20/40
Averages of Different Answers from N.D.A	Total	-	-1.80%	0.0%	2.75%	6.05%
	2-var	-	0.0%	0.0%	3.01%	7.84%
	3-var	-	-1.80%	0.0%	2.77%	6.64%
	4-var	-	0.0%	0.0%	1.18%	3.87%

Table 5.3 Result of the Experiment for Column = 5

5 Columns		N.D.A	DAT-1	DAT-1(BT)	DAT-2	DAT-3
Averages		75.37%	75.30%	75.37%	76.52%	81.71%
Iterations		31.00	5.00	15.00	58.10	338.43
Multiple Answers	2-Answers	0	0	0	3	6
	3-Answers	0	0	0	0	1
	4-Answers	0	0	0	0	1
Cases of Different Answers from N.D.A	Total	-	6/100	0/100	74/100	96/100
	1-var	-	0/0	0/0	0/0	0/0
	2-var	-	0/2	0/2	2/2	2/2
	3-var	-	0/17	0/17	17/17	17/17
	4-var	-	6/59	0/59	55/59	59/59
	5-var	-	0/22	0/22	0/22	18/22
Averages of Different Answers from N.D.A	Total	-	-1.10%	0.0%	1.56%	6.61%
	2-var	-	0.0%	0.0%	5.51%	14.13%
	3-var	-	0.0%	0.0%	3.87%	10.22%
	4-var	-	-1.10%	0.0%	0.70%	6.01%
	5-var	-	0.0%	0.0%	0.0%	6.61%

Table 5.4 Result of the Experiment for Column = 6

6 Columns		N.D.A	DAT-1	DAT-1(BT)	DAT-2	DAT-3
Averages		74.78%	74.58%	74.78%	77.93%	82.14%
Iterations		63.00	6.00	21.00	113.05	1012.44
Multiple Answers	2-Answers	0	0	0	4	9
	3-Answers	0	0	0	0	4
Cases of Different Answers from N.D.A	Total	-	11/100	0/100	82/100	97/100
	1-var	-	0/0	0/0	0/0	0/0
	2-var	-	0/0	0/0	0/0	0/0
	3-var	-	0/2	0/2	2/2	2/2
	4-var	-	3/25	0/25	25/25	25/25
	5-var	-	8/53	0/53	50/53	53/53
	6-var	-	0/20	0/20	5/20	17/20
Averages of Different Answers from N.D.A	Total	-	-1.81%	0.0%	3.83%	7.51%
	3-var	-	0.0%	0.0%	6.33%	11.15%
	4-var	-	-1.99%	0.0%	4.38%	10.50%
	5-var	-	-1.74%	0.0%	3.78%	6.85%
	6-var	-	0.0%	0.0%	0.69%	5.17%

5.5 Summary

From the result of the experiment, it is clear that the performance of analyzing the data set is improved by DAT algorithms. DAT-1 reduced the time of analysis down to $O(n^2)$ while producing the same results as the normal data analysis method, and DAT-2 produces more accurate results while using the same time complexity as the normal data analysis. DAT-3 uses too much time, but it can produce the most accurate results that can be obtained from the data set.

There are cases that using DAT-2 or DAT-3 cannot help to get the better results from those produced by the normal data analysis method especially when the number of variables in the most important factor found by the normal data analysis method is close to the total number of variables in the data set. However, with more variables in the data set, the chances of getting a more accurate answer using DAT-2 or DAT-3 becomes higher. Also the differences in analyzing time between the normal data analysis and DAT-1 becomes larger as the number of variables in the data set increases. As the number of the variables in the data set increases, the performance of DAT-1 becomes faster and the accuracy of the answers that DAT-2 and DAT-3 produce become higher.

CHAPTER 6. CONCLUSIONS

6.1 Data Analysis

Data analysis is a method of analyzing a set of data with several variables and a function value. It produces a vector of important variables along with their values which most increases the fall-into-the-range rate of the function value, The users can then use these variabl values to achieve a desired function range. To achieve more accurate results, as many important variables as possible in the data set are needed, but the data analysis method takes $O(2^n)$ comparisons to generate the results where n is the number of variables in the data set. This method uses only the important values of the variables which makes it possible to produce the results with a small fall-into-the-range rate by ignoring the other values.

6.2 Data Analyzing Trees

Data Analyzing Tree (DAT) algorithms are designed to solve the problems mentioned in the previous section. DAT-n is the tree-shape data analyzer which contains all the necessary information of the data set in the nodes. The shape of the tree is decided by the positive integer n, where n is the number of maximum branches at any given node. So, DAT-1 has the shape of the list, DAT-2 has the shape of binary tree, and so on.

Applying DAT algorithms to the data sets results in either a better time complexity or a higher fall-into-the-range rate depending on which DAT is

used. DAT-1 with back-tracking produces the same results as the normal Data Analysis method produces only in $O(n^2)$ times, and DAT-2 uses the same time complexity of $O(2^n)$ but produces more detailed result than those produced by normal data analysis. If time permits, DAT-p is the method of choice to produce the most detailed result.

The differences in fall-into-the-range rates between the results of normal data analysis and DAT-2 or DAT-3 depends on the number of variables that are found by the normal data analysis method or DAT-1 with back-tracking. Therefore, DAT-1 should be applied first for a brief analysis of the data set, then DAT-2 and DAT-3 should be applied to find more specific results. Since the number of variables found in DAT-1 dictates the performances of DAT-2 and DAT-3, the feasibility of DAT-2 and DAT-3 is determined by the application of DAT-1.

DATs can reduce the search time if the cut-off value is given. Unlike the normal data analysis method which must search all cases, DAT algorithms except for DAT-1 can find the minimum vector that has the rate exceeding the cut-off value without checking all cases. Optimization can be applied to reduce the number of nodes which requires less search time. Therefore, DATs are not only faster and more accurate but also more versatile.

6.3 Usage the DATs

Since there can be several DATs applied to the data set depending on the number of possible variable values, the users have more choices for

different situations. If the data set must be analyzed in a short time period (e.g. a basketball coach should decide the best in-bound play for the last second shot), DAT-1 should be used. If more specific results for a higher fall-into-the-range rate is desired (e.g. the supervisor of the factory wants to improve product quality), DAT-2 or DAT-3 is used. If the number of possible variable values is too big for DAT-p to produce the answers as quickly as desired, the clustered DAT method can be applied to reduce the searching time. Therefore, the users can select appropriate DAT methods for different situations.

6.4 Future Research

Data analysis determines the behavior of the F-values by the variable values. The behavior of the F-value can be determined with more accuracy if more variables are used and if more variable values are considered in a data set. However, having more variables and more values increases the time complexity, so an efficient method for a data set using more variables and values should be found. Some solution ideas are introduced in the following sections.

6.4.1 Variable Classes

Having more variables in a data set increases the time complexity exponentially except for DAT-1. One way to reduce the time complexity is to divide the variables into several classes and apply the DAT algorithm separately on each class. We then produce a new data set taking only the

variables found in the important factors of each class and apply the DAT method to the newly formed data set. Of course this method (dividing the variables into several classes) can be applied recursively to the new data set if the number of variables is still too large.

This method cannot be guaranteed to produce the same results as the direct DAT method because of the possibility of two important variables that produce the highest fall-into-the-range rate being discarded. This can happen when variables are separated into different classes and never considered as important in each respective class. Therefore, a careful class dividing method must be introduced, and simulation should be performed.

6.4.2 Improvement on Clustering

When a variable has continuous values, clustering is performed before the algorithm, as introduced in section 2.3, to force it discrete behavior. The clustering method combines the similar values in the same category. However, similarity in numbers does not always mean similar effects on the fall-into-the-range rate of F-value. A method that partitions values with similar effects on the the fall-into-the-range rate of F-value into the same clusters is the clustered-DAT method mentioned in section 4.5. Therefore, the clustered-DAT method produces a better clustering of values than the normal clustering method.

Clustered-DAT cannot be applied directly to continuous values, so a certain form of clustering is inevitable to enforce discrete behavior. One

approach for this problem is to begin with many small clusters. For the example in table 2.1, we can have more than three clusters by making the intervals smaller. Clustered-DAT is then applied. This method gives more precise consideration to the continous values which leads to more accurate results. The simulation experiment must be performed to determine the improvement of the accuracy by applying this method and to decide the appropriate number of clusters before applying the clustered-DAT.

6.4.3 Summary

The ability to deal with large data sets is very important. A large data set can be a data set with more variables and more values of the variables which clearly takes more time to produce results. DAT deals with some of those problems, but still there is still plenty of room for improvement. Classes of variables and clustered-DAT for clustering are two methods of improveing the performance of DAT, but they still need more work and experiment.

Data analysis is an area that allows us to predict the future with the data of the past. If it is to be utilized effectively, data must be perpetually collected. If we have faster algorithms to analyze larger data sets, we have a more accurate prediction of the future.

BIBLIOGRAPHY

[1] B. Jones, "A Program For Reconstructability Analysis"; Int. J. General Systems, Vol. 15. pp 199-205

[2] B. Jones, "Reconstructability Considerations with Arbitrary Data"; Int. J. General Systems, 11, 1985, pp 143-151

[3] B. Jones, "A Greedy Algorithm for a Generalization of the Reconstruction Problem"; Int. J. General Systems, 1, 1985, pp 63-68

[4] B. Jones, "Reconstructability Analysis of General Functions"; Int. J. General Systems, 11, 1985, pp 133-142

[5] B. Jones, "Determination of Unbiased Reconstructions"; Int. J. General Systems, 10, 1985, pp. 169-176

[6] B. Jones, "K-systems Analysis Versus Classical Multivariate Analysis"; Int. J. General Systems, 12, 1986, pp. 1-6

[7] G.J. Klir, "Identification of Generative Structure in Empirical Data"; Int. J. General Systems, 3, 1976, pp. 89-104

[8] G.J. Klir and R. E. Cavallo, "Reconstructability Analysis : Evaluation of Reconstruction Hypotheses"; Int. J. General Systems, 7, 1981, pp. 7-31

[9] G.P. Shaffer and P. Cahoon, "Extracting Information from Ecological Data Containing High Spatial and Temporal Variability : Benthic Microfloral Production"; Int. J. General Systems, 13, 1987, pp. 107-123

[10] J.R. Quinlan, "Learning Efficient Classification Procedures and Their Application to Chess End Games"; In Michalski, R.S., Carbonell,J., and Mitchell,T.M. (eds), Machine Learning : An Artificial Intelligence Approach, Vol. I, Palo Alto, Tioga Press, pp. 463-482

[11] J.R. Quinlan, "An Empirical Comparison of Genetic and Decision-Tree Classifiers", Proceedings of the 5th International Conference on Machine Learning, San Mateo, CA, 1988, pp. 135-141

[12] W. Jung, B. Jones, and J. Chen, "Optimization of the Decision Tree"; Proceedings of the 3rd International Conference on Tools for Artificial Intelligence, San Jose, CA, 1991, pp. 522-523

[13] P.E. Utguff, "ID5: An Incremental ID3"; Proceedings of the 5th International Conference on Machine Learning, San Mateo, CA, 1988, pp.107-120

[14] W. Van de Velde, "Incremental Induction of Topologically Minimal Trees"; Proceedings of the 7th International Conference on Machine Learning, Austin, TX, 1990, pp.66-74

VITA

The author started his study in Computer Science in 1984 as an undergraduate student in Henderson State University. After three years, he came to Louisiana State University in 1987 for his master's degree in Systems Science, but after passing the qualifying examination while pursuing for the master's degree, he switched his program into the Ph.D. program in Computer Science.

The author concentrated on researching on the field of data analysis (reconstructability analysis) and machine learning. He published a research paper titled 'Optimization of the Decision Tree' in the proceedings of the Third International Conference on Tools of Artificial Intelligence.

After getting his Ph.D. degree, the author wishes to continue on researching in data analysis and machine learning.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Won Chan Jung

Major Field: Computer Science

Title of Dissertation: Improvement of the Data Analysis Algorithm by Applying the Decision Tree Method

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

Feb. 10th, 1992