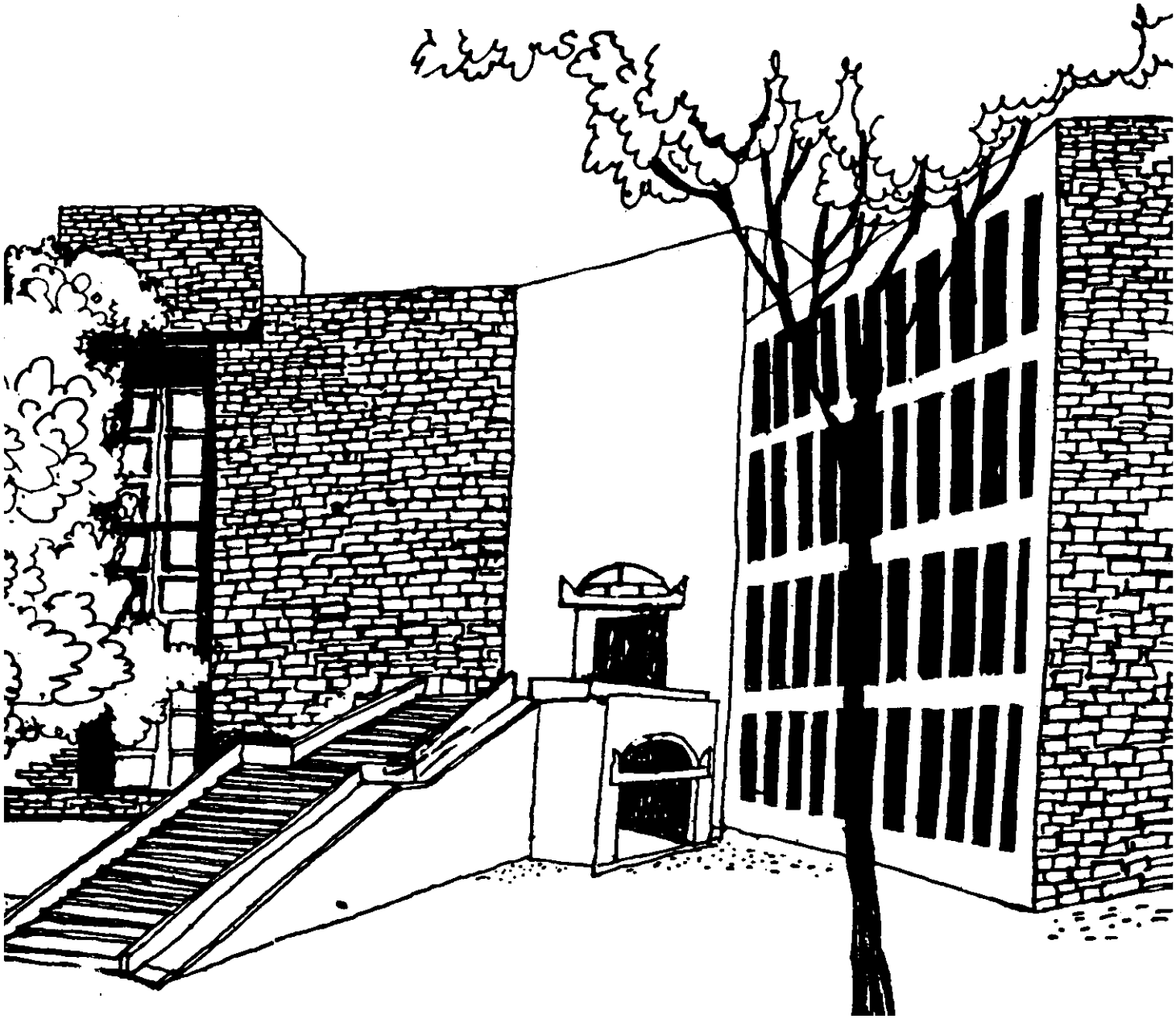
विद्याविनियोगाधिकारः

IIM

AHMEDABAD

# Working Paper



# KNOWLEDGE ACQUISITION FROM EXAMPLES USING A REFERENCE CLASS

By

S. Yegneshwar

&

S. Arunkumar



W P No. 1017
April 1992



INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

# Knowledge Acquisition from Examples using a Reference Class

S.Yegneshwar* S.Arunkumar†

**Abstract**

Acquiring knowledge from examples is frequently used in expert systems. A common model is building of a decision tree which discriminates each class from every other class. Though such a model performs well as far as classification accuracy is concerned, the resultant knowledge is opaque to the user. In this paper, we propose a new model of acquiring knowledge from examples. In this model, a reference class description is first learnt from which each class description is learnt. Each of these class descriptions is used to classify test examples. The proposed model has been tested on two applications. The results of these experiments suggest that it is possible to learn a knowledge base which not only performs well but that is also intelligible.

*Computer and information Systems Group, Indian Institute of Management, Vastrapur, Ahmedabad 38 015, India. Tel: 0272-407241. email : yegnesh@iimahd.ernet.in

†Dept. of Computer Science and Engineering, Indian Institute of Technology, Bombay 400 076, India. Tel 022-5782545. email : sak@dhruv.ernet.in

# Contents


**1 Introduction** 1

**2 Learning Reference Class Description** 3

2.1 Generation Tree for Binary Valued Attributes . . . . . 4

2.1.1 Algorithm for Construction of GT . . . . . 4

2.1.2 Drawback of this Algorithm . . . . . 6

2.1.3 Modified Algorithm for Construction of GT . . . . . 7

2.2 Algorithm for Construction of GT for Real and Integer Valued Attributes . . . 8

2.3 Algorithm for Construction of GT for Nominal Valued Attributes . . . . . 11

2.4 Learning GT from Examples with Missing Attribute Values . . . . . 13

**3 Learning Elementary Class Description** 17

3.1 Algorithm for Constructing GT of Elementary Class . . . . . 17

3.2 Redundancy of an Attribute at a Node of a GT . . . . . 21

3.3 Importance of Attributes . . . . . 26

**4 The Inference Process** 30

**5 Practical Applications of the proposed system** 32

5.1 Classification of different kinds of glasses . . . . . 32

5.2 Classification of cars based on risk factor . . . . . 33

**6 Conclusion** 36

# 1 Introduction

A knowledge based system is a computer system which uses the knowledge of an expert stored in a suitable form to solve problems requiring significant expertise. The need for such systems is felt because experts are few in number and their knowledge is not available to many. The chief bottleneck in the development of knowledge based systems is the acquisition of knowledge from the experts [Duda 83]. One of the popular methods of knowledge acquisition is from preclassified examples. The examples are past cases where the attributes (either symptoms or tests) and the corresponding values are recorded. All such examples with the right classification (i.e., the correct class or diagnosis) are used for knowledge acquisition. In this paradigm of knowledge acquisition, the description of each class (which is the required knowledge) is learnt by finding patterns in the given examples. This mode of knowledge acquisition is also referred to as learning from examples or instance based learning. [Bund 85]. The pioneering domain independent learning systems are ID3 [Quin 79, Quin 86] and INDUCE [Mich 80, Mich 83]. Another well tested system is CART [Brie 84, Craw 89].

ID3 and CART generate a decision tree which helps classify a test example. A decision tree is a tree where a node represents an attribute and an arc represents a value assumed by the attribute at the node from which the arc originates. At each node the set of examples at that node is split into subsets based on the value of the attribute at that node. At the root the set of examples is all the given examples. This process of progressively splitting a set of examples into subsets based on an attribute value is continued till all the lowermost nodes have examples belonging to only one class. It is clear that the emphasis of all these systems is to generate a description which separates each class from every other class. Most of the extensions to ID3 [Nort 89, Mant 91] deal with alternative ways of selecting attributes at a node. A better attribute selection would lead to the generation of a smaller decision tree. The aim here is to generate the smallest decision tree which separates examples of each class from examples of every other class. INDUCE finds the most general description of a class that is consistent with examples of all the other classes, i.e., a description generated for a class should not cover an example of any other class. The emphasis of finding a description which covers all the examples of a given class and examples of no other class is the same as finding a discriminating description.

The most important attribute in these learning systems are those which are most discrimi-

nating. Attributes whose values are common to examples of all the classes would be considered irrelevant. These attributes would not aid in discriminating among a given set of classes. However, in practice there could be test examples about which nothing can be said or which may not belong to any of the classes whose description is learnt. In such cases, a test example may wrongly get classified. For instance, given a set of examples (equal proportion of which falls into one of the three cases listed below) for the class **call** (for promotion) and the class **do-not-call** (for promotion) : ((no-of-years-since-last-promotion, 3), (perf-in-the-last-year, good) ,(recommendation, good) **call**), ((no-of-years-since-last-promotion, 3), (perf-in-the-last-year, bad), (recommendation, poor) **do-not-call**) and ((no-of-years-since-last-promotion, 3), (perf-in-the-last-year, bad), (recommendation, good) **do-not-call**) ID3 would come up with a decision tree equivalent to : if perf-in-the-last-year = good then class = **call** else class = **do-not-call**. Suppose now a test example with the following (attribute,value) pairs is given : ((no-of-years-since-last-promotion, 1),(perf-in-the-last-year, good), recommendation, good)). This would be classified as **call** by the decision tree generated, though it should have been classified into neither of the classes since it is clear from the learning examples that the decision tree is applicable only to persons with 3 years experience after the last promotion. If the attribute no-of-years-since-last-promotion had been used while generating the description of **call** and **do-not-call** from the learning examples, the test example would not have satisfied either of the descriptions.

This shortcoming led us to design a learning system that first learns the description of a reference class, where the reference class is the union of the learning examples of all the classes. The learning of this description is such that an attribute with a value common to most of the examples of the reference class is considered more important than an attribute with diverse values among the examples. The former kind of attributes help characterise this group of classes and the latter kind of attributes help discriminate each class from the other classes in this group. In the **call** and **do-not-call** case, the attribute with a common value for most of the examples is: no-of-years-since-last-promotion (= 3). The attribute which helps discriminate examples of one class from the others is perf-in-the-last-year. Individual class description (i.e., that of **call** and **do-not-call**) is learnt using the reference class. The description of the class **call** that would be generated by this learning system would be : A person with no-of-years-since-last-promotion = 3 and perf-since-the-last-promotion = good and recommendatio

= good. Similarly, the description of a **do-not-call** would be : A person with no-of-years-since-last-promotion = 3 and perf-since-the-last-promotion = bad. For this class the attribute recommendation is redundant. This is because, given that no-of-years-since-last-promotion = 3 and perf-since-the-last-promotion = bad the attribute recommendation could take any of the domain values, viz., good or bad. In other words, given the value of the other two attribute values, it does not matter what value the attribute recommendation assumes for this class. Unlike in ID3, an attribute could be redundant for one class but relevant for another class. When the earlier mentioned test example, viz., ((no-of-years-since-last-promotion, 1),(perf-in-the-last-year, good), recommendation, good)) is presented to this learning system, it would not be classified into either of the classes.

In section 2 an algorithm to learn the description of the reference class is described. Section 3 deals with learning of the description of an elementary class (a class for which learning examples are given) using its examples and the reference class description. In this section a definition of redundant attributes and a methodology to determine the importance of an attribute for a class is described. Section 4 describes an inference process which uses the descriptions generated. In section 5 the results of applying the learning system to two commonly used machine learning databases is discussed. Section 6 concludes the work with some directions for future research.

## 2 Learning Reference Class Description

The description of a given set of classes is facilitated by learning about the class of classes, referred to herein as the reference class. Learning of the reference class description is particularly useful in identifying attributes which take similar values across classes. This is necessary, since it is more natural to first state the features a class shares with other related classes before stating the discriminating features. The learning of the reference class description facilitates generation of such descriptions since it first identifies common features among the given set of classes before identifying the discriminating features.

The learning of the reference class description is enabled by the *maximum representation criterion* defined as one which maximises the number of examples (of the specified class) taking a particular value for an attribute (corresponding to the specified class). We define the

Generation Tree below as the tree representation of a class each of whose nodes represents the attribute with respect to which the branching is done and whose directed arcs from root to leaves carry the attribute value along with the proportionate number of examples taking this value. The node attributes are selected using the maximum representation criterion at each node. The structure of the reference tree is a binary tree. Note that in this tree, relatively more important attributes are selected higher up in the tree.

In this section, we give a procedure to learn the reference GT by pooling the examples of all the classes. We note that comparison across classes would be facilitated by constructing elementary class descriptions relative to this reference GT. We show that our approach is provably convergent. The reference GT is constructed for attributes taking binary, real/integer and nominal values. An approach for the case with missing attribute values is also described.

## 2.1 Generation Tree for Binary Valued Attributes

The concept of Generation Tree was proposed by S.Arunkumar and first reported in [Doct 85]. In this proposal, a methodology to generate elementary class description is described for binary valued attributes. However, this proposal has a major flaw which is explained below. We have used a modified version of this methodology (initially used for learning elementary class description and first reported in [Arun90]) for generating the reference class description for attributes taking binary, continuous values (integer/real) and nominal values [Yeg90]. The process of learning the elementary class description is described in the next section. The algorithm for the construction of the GT for binary valued attributes as given in [Doct 85] is described below.

**Remark 1** The GT can be interpreted as a collection of rules representing the same class with each path of the GT representing one rule, the antecedent of which is the conjunction of each arc of the path and the consequent is the class name. In a Bayesian framework, we can say that the frequency at the leaf of each path corresponds to the probability of the evidence given the class.

### 2.1.1 Algorithm for Construction of GT

The GT is built using the criterion of maximal representation at each node.

1. Consider the given set of examples at the root node.

2. Select the most representative attribute for the examples at the current node. This is done as follows:

   (a) If both the binary values occur for the given attribute among the set of examples at the current node, then find the cardinality of both the sets, where all the examples in the first set have one value (say 0) for this attribute and all the examples in the other set have the complementary value for this attribute. Else, find the cardinality of the given set.

   (b) Find the maximum of the cardinality of the resulting sets (or set).

   Repeat steps (a) and (b) for all the attributes and select that attribute as most representative which has maximum cardinality as determined from step (b). In case of ties, the criterion is: select that attribute which comes earliest in the attribute sequence.

3. For both the values, branch off from the current node. At the left child node, consider all those examples having value 0 for further branching and at the right child consider all those examples having value 1 for further branching.

4. *Termination Condition :* If all the attributes have been selected along all the paths of the tree, or if the frequency along all the leaves is less than a specified threshold, called *expansion threshold*, or if the best attribute at all the nodes are such that the frequency of the left and right child are equal, then stop. Else, go to step 2 and proceed in a depth first fashion.

**Note** In this context, the *true description* of the class is the binary stochastic description based on the maximum representation criterion.

**Example 1** *Suppose we have the following set of examples with four attributes.*

$$\begin{aligned} S = \{&(1011),(0110),(1011),(1110),(1011),\\ &(0110),(1110),(1011),(1110),(1011)\} \end{aligned}$$

*We show how the GT is built for this example set. We have,*

$$\begin{aligned} \{&(|a_1 = 1| = 8),(|a_1 = 0| = 2),(|a_2 = 1| = 5),\\ &(|a_2 = 0| = 5),(|a_3 = 1| = 10),(|a_3 = 0| = 0), \end{aligned}$$

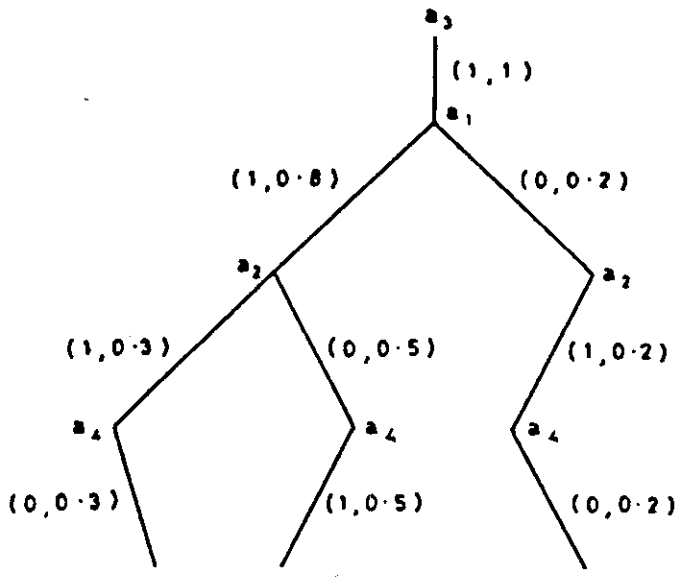



Figure 1: GT for binary valued attributes

$(|a_4 = 1| = 5), (|a_4 = 0| = 5)\}$

*Here |.| stands for cardinality.*

*Therefore, at the root node we select $a_3$. Then we select $a_1$ at the next node and proceeding similarly, we get the GT described in Figure 1.*

### 2.1.2 Drawback of this Algorithm



The termination condition used by this algorithm could lead to ignoring patterns in the data when they exists. Also, this termination condition gives rise to a non-intuitive definition of attribute redundancy. These points are best explained with the help of a simple example.

**Example 2** *Consider the following set of examples*

```
S = {(1 1 1),(1 1 1),(1 1 1),(1 1 1),(1 1 1),
     (0 0 0),(0 0 0),(0 0 0),(0 0 0),(0 0 0)}
```

*The algorithm above does not generate any tree. This is because cardinality of $a_1$=1 and $a_1$=0 are equal and similarly, for $a_2$ and $a_3$. Therefore, at the root the algorithm does not select any attribute and terminates. This tree of a single node corresponds to the* null class.

It is evident from the above set of examples that there are two disjoint patterns (or sub-descriptions) in this set. The only condition that stalls learning of these sub-descriptions is the termination condition, which states that the generation process should be stopped when the frequencies of the left and right child of the best attribute are equal. Also, in this example the by-product is that all the attributes are redundant for the class, whereas there is a distinct correlation of the attributes. The termination condition should be such as to define a class as null only if it assumes all the domain values with equal frequency.

### 2.1.3 Modified Algorithm for Construction of GT

We have proposed a modified algorithm for the construction of the GT for binary valued attributes [Arun 90] in which the termination condition is made more intuitive.

The new termination condition is as follows : If all the attributes have been selected along all the paths of the tree or if the frequency along all the leaves is less than called expansion threshold, then stop. Else, go to step 2 and proceed in a depth first fashion.

**Note** In the modified methodology, redundancy of attributes and the resultant reduction in number of sub-descriptions are determined for elementary classes after the GT is built completely (the process is described in Section 3.2).

**Example 3** *The modified GT corresponding to example 2 is constructed as follows: At the root, we have,*

$$\{(|a_1 = 1| = 5),(|a_1 = 0| = 5),(|a_2 = 1| = 5),(|a_2 = 0| = 5),$$
$$(|a_3 = 1| = 5),(|a_3 = 0| = 5)\}$$

*Thus, $a_1$, $a_2$ and $a_3$ qualify for the most representative attribute. Using the rule for tie (as given above), we select $a_1$. At the left child, we select $a_2$ using the same argument and proceeding in the same manner, we have the GT described in Figure 2.*

**Theorem 1** *The GT for binary valued attributes generated by the modified algorithm converges to the true description in the distribution sense.*

**Proof** *By Glivenko-Cantelli Theorem [DeGr 87], $| F_n(x) - F(x) | \longrightarrow 0$, with probability 1 uniformly in $x$, where $F_n$ is the empirical distribution and $F$ is the true distribution. This implies that for a given $\epsilon > 0, \exists n(\epsilon)$ such that $|F_n(x) - F(x)| < \epsilon$, with probability 1, $\forall n > n(\epsilon)$*

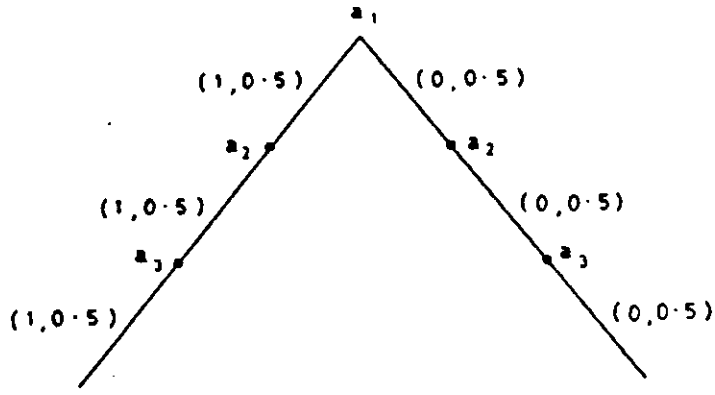



Figure 2: GT depicting correlation of attributes

*If we assume that the joint frequencies are such that in the true description, there are no ties in the selection of attributes at all the nodes of the GT, we can choose $\epsilon$ such that for all $n > n(\epsilon)$, chosen suitably, the structure and the attribute at each node of the tree remains the same. Also, when this is true, the joint frequencies at each node stabilise.* □

## 2.2 Algorithm for Construction of GT for Real and Integer Valued Attributes

The GT created is a binary tree. At each node, the attribute, value range and the joint frequency of both the left and right child are stored.

The GT is built using the same criterion of maximal representation at each node as in the binary valued attributes case.

1. Consider the given set of examples at the root node.
2. Select the most representative attribute among the examples at the current node. This is done as follows:
   (a) Start with the first attribute $a_1$.
   (b) For each value assumed by at least one example, find the number at the current node. Order this (value, cardinality) in ascending sequence of the value.
   (c) Start with the second value in the above list.

(d) Place the boundary at this chosen value and evaluate the centre of left and right partitions.

(e) Find the distance between these partitions (called inter-partition distance).

(f) If the current boundary is at the last value, then go to step (g), else place the partition boundary at the next value and go to step (d).

(g) Find the boundary for which the inter-partition distance is maximum.

(h) For the current attribute $a_u$ at node r, if the maximum inter-partition distance is greater than a specified threshold (equal to $\alpha$*least count, where $\alpha$ is specified and least count is determined from the learning examples), two resultant sets may be considered, viz., $S^1_{ru}$ and $S^2_{ru}$. The first set is the one where each example has a value for $a_u$ that is less than the boundary value. The other set is the one where each example has a value which is greater than the boundary value. The lower limit of the range of $S^1_{ru}$ is the least value of $a_u$ in $S^1_{ru}$ and the upper limit is one least count less than the boundary value. The lower limit of the range of $S^2_{ru}$ is the boundary value and the upper limit is the largest value of $a_u$ in $S^2_{ru}$. Select the larger of the two sets $S^1_{ru}$ and $S^2_{ru}$. Otherwise, the given set is retained as a single partition with the least value in the set being the lower limit and the highest value being the upper limit.

Repeat steps (b) to (h) for all the attributes, $a_u$, u = 2,..,n.

(a) Find the largest set as determined from step (h) considering only those sets different from the ones produced along this path. The corresponding attribute is $a_m$. This is the most representative attribute. In case of tie, choose the attribute which occurs earlier in the input list of attributes.

3. Split the set as determined from step 2h. If it is split into two, then the first set comprises all those examples which have a value less than the best partition's boundary value $a_m$ and the second set comprises all those examples whose value for $a_m$ is greater than or equal to this value. If it is to be retained as a single set, then the range is the range of values for all the examples at this node.

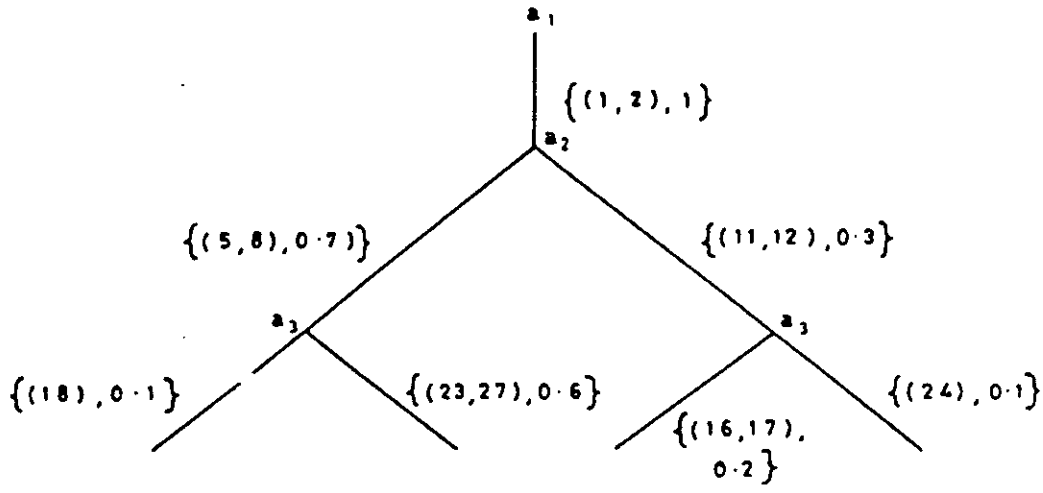



Figure 3: GT for integer valued attributes

4. If there is no new (attribute,value range) that can be considered along all the paths of the GT, or if the frequency at all the leaves are less than the expansion threshold, then stop. Else, go to step 2 and proceed in a depth first manner.

**Example 4** *Suppose we are given a set of examples specified as tuples of three attributes* $a_1$, $a_2$ *and* $a_3$. *If all the attributes are of type integer, then let*

```
S = { (1 5 18),(1 6 23),(1 7 24),(1 8 25),(1 6 26),
      (1 7 27),(1 11 24),(1 8 23),(2 12 16),(2 12 17)}
```

*and* $\alpha = 5$.

*At the root node the first attribute assumes two values, viz., 1 and 2. Placing the partition at the second value we have,*

*inter-partition distance = (centre2 - centre1) = (2 - 1) = 1.*

*Since this is less than (α*least count), the given set is not partitioned. The other two attributes partition the set. Therefore,* $a_1$ *is selected at the root. At the next node, the cardinality of the best partition for both* $a_2$ *and* $a_3$ *are the same. Using the rule for ties, we select* $a_2$. *The GT obtained by this procedure is as shown in Figure 3.*

**Theorem 2** *The GT for real/integer valued attributes generated by the above algorithm converges to the true description.*

**Proof** *Assuming that the joint distribution of (attribute, value) tuple is such that there are no ties in the selection of attributes at the nodes of the GT when the joint distribution is known, we can choose $\epsilon$ such that, for all $n > n(\epsilon)$, chosen suitably, the best attribute is chosen and the corresponding partition remains the same. This ensures that the tree structure stabilises* $\forall n > n(\epsilon)$. □

## 2.3 Algorithm for Construction of GT for Nominal Valued Attributes

The GT created is a binary tree. At each node we store one value for the left child and one or more values for the right child (if it exists), and also the attribute, and frequency of the left and right child.

The GT is built using the same maximum representation learning criterion. Let the examples be given as (attribute, value) pairs where the $j^{th}$ value of the $i^{th}$ attribute is denoted by $v_{ij}$.

1. Let the given set of examples represent the root.
2. Select the most representative attribute among the examples at the current node as follows:
   (a) For each attribute $a_i$, find a value $v_{ir}$ among the examples which has the maximum cardinality.
   (b) Select attribute $a_t$ which has the maximum cardinality among all the attributes as determined in step (a). In case of a tie, choose the attribute which comes earlier in the attribute list.
3. Split the set of examples at this node into two sets. The first set has all the examples which have value $v_{tr}$ for $a_t$, and the other set has all the examples with any value other than $v_{tr}$ among the examples.
4. If there is no new (attribute, value range) pair that can be considered along all the paths of the GT, or if the frequency at all the leaves are less than the expansion threshold, then stop. Else, go to step 2 and proceed in a depth first manner.

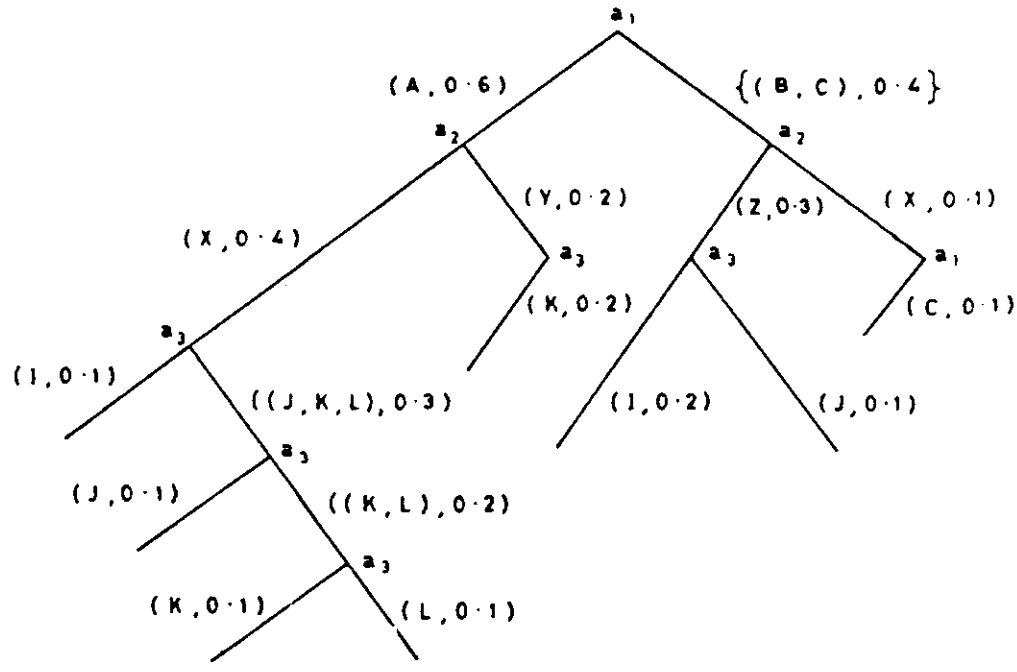



Figure 4: GT for nominal valued attributes

**Example 5** *Suppose we have the following set S of examples specified by three nominal attributes $a_1$, $a_2$ and $a_3$ whose ranges are {A,B,C,D}, {X,Y,Z} and {I,J,K,L}, respectively:*

```
S = { (A X I),(A X J),(A Y K),(A Y K),(A X K),
      (A X L),(B Z L),(B Z J),(B Z I),(C X I) }
```

*At the root node we select attribute $a_1$, since the cardinality of $a_1$=A is the highest among the three attributes. We branch off at the root node into two children : the examples at the left child having value A for $a_1$ and examples at right child having any other value (B or C) for $a_1$. Proceeding similarly, we generate the GT shown in Figure 4.*

By an argument similar to theorem 1, we get the following:

**Theorem 3** *The GT for nominal valued attributes generated by the above algorithm converges to the true description.*

**Complexity Analysis** The maximum number of nodes that can be generated for a sample size of n and m attributes is when the leaf frequency of each path is equal to 1/n and the

number of nodes is equal to $(2^x - 1) + (m - x) * n$, where x is the smallest integer such that $2^x \geq n$ for $n < 2^m$ and is equal to m for $n \geq 2^m$. The first term in the expression, i.e., $(2^x - 1)$, is the number of nodes at the top x levels. All nodes at depth x have frequency equal to 1/n, for x < m. Since the example set at each of these nodes can no longer be partitioned and (m-x) attributes remain along each of these n paths, the remaining number of nodes is equal to (m-x)*n. When x=m (i.e., when $n \geq 2^m$), the number of nodes generated is equal to $(2^m - 1)$. The bound for any n and m is obtained as follows:

$$
\begin{aligned}
& (2^x - 1) + (m - x) * n \\
& \leq 2 * n + (m - x) * n \qquad \{\text{since } 2 * n \geq 2^x\} \\
& \leq n * (m + 2) - n * x \\
& \leq n * (m + 2) - n * log_2 n \\
& \leq n * (m + 2 - 1) \qquad \{\text{if } n > 1\} \\
& \leq n * (m + 1)
\end{aligned}
$$

Therefore, the number of comparison of attributes at all the nodes is bounded by

$$n * (m + 1) * m$$

Note This is a loose bound since in practice there would be some duplicates and the expansion threshold used would be non-zero.

## 2.4 Learning GT from Examples with Missing Attribute Values

Consider the case where examples may have one or more missing values for the given attributes. The GT is built by applying the following steps iteratively.

1. Start at the root of the GT to be built. Let this be the current node. The set of examples at the current node is S, the given set of examples of sample size N.

2. Construct GT $T_0$ of a single level with examples having no missing attribute values (sample size = $N_0$), $T_1$ of a single level with examples not having value for only attribute $a_1$ (sample size = $N_1$), $T_2$ of a single level with examples not having value only for $a_2$ (sample size = $N_2$),... and $T_n$ of a single level with examples not having value only for $a_n$ (sample size = $N_n$) among the given set of examples S at the current node. If the proportion of examples of these trees $(\sum_{i=1}^{n}(N_i/N))$ is greater than a specified threshold, then attribute selection is made with S=$\{T_0,T_1,T_2,...,T_r\}$ using the procedure in step 3.

Otherwise, GTs $T_{12}$, $T_{12}$,..., $T_{23}$, $T_{24}$,..., $T_{(n-1)n}$,..., $T_{123...(k-1)k}$ are constructed (where $T_{ij}$ is a single level GT constructed using examples with only attributes $a_i$ and $a_j$ missing), till the threshold is satisfied, for some k. Then attribute selection is made in step 3 with S=$\{T_0,T_1,T_2,...,T_r,T_{12},T_{13},...,T_{(n-1)n},...,T_{1,2,...,(k-1)k}\}$.

3. Let n be the total number of attributes and $|s|$ the number of missing attributes in GT $T_s$. The weighted frequency is evaluated for each attribute $a_i$ as follows:

$$\sum_s \delta(T_s) * \frac{N_s}{N} * \frac{n - |s|}{n}$$

where 's' ranges over all the trees and $\delta(T_s) = 1$ if attribute $a_i$ is at the current node in $T_s$ and $\delta(T_s) = 0$ otherwise. The attribute $a_k$ for which this value is maximum is selected at the current node. In case of ties, the first attribute in the sequence is chosen.

4. Find the left and right partitions using $a_k$ and the examples at the current node. Note that only those examples for which attribute value is present for $a_k$ are used in the construction. The method of finding partitions for different types of attributes is the same as explained in Sections 2.1.3, 2.2 and 2.3.

5. If no new attributes can be considered along all the paths, then exit. Else, repeat steps 2 - 4 for the two child nodes with S=$S_l$ for the left child and S=$S_r$ for the right child. $S_l$ is the set of examples satisfying the left arc value for $a_k$ and $S_r$ is the set of examples satisfying the right arc value for $a_k$. So, at every node, only examples which satisfy path traversal are used in the construction.

**Note** The GT has been built only with known attribute values. However, the best possible GT with respect to maximum representation learning criterion has been learnt. Since the GT is built only with known attribute values, the algorithm for generation of elementary class GTs is the same as when all attribute values are present.

**Example 6** *Consider the following set of examples and threshold = 0.70.*

```
S = {(? ? 0 1),(? ? 1 1),(? 1 0 0),(? 1 1 0),(? 0 1 0),
     (1 ? ? 0),(0 ? ? 1),(0 ? 1 0),(0 ? 0 1),(1 ? 1 1),
     (0 0 ? ?),(0 1 ? ?),(0 0 ? 0),(1 0 ? 0),(0 1 ? 0),
     (? 0 0 ?),(? 1 0 ?),(0 1 1 ?),(1 0 0 ?),(1 0 1 ?)}
```

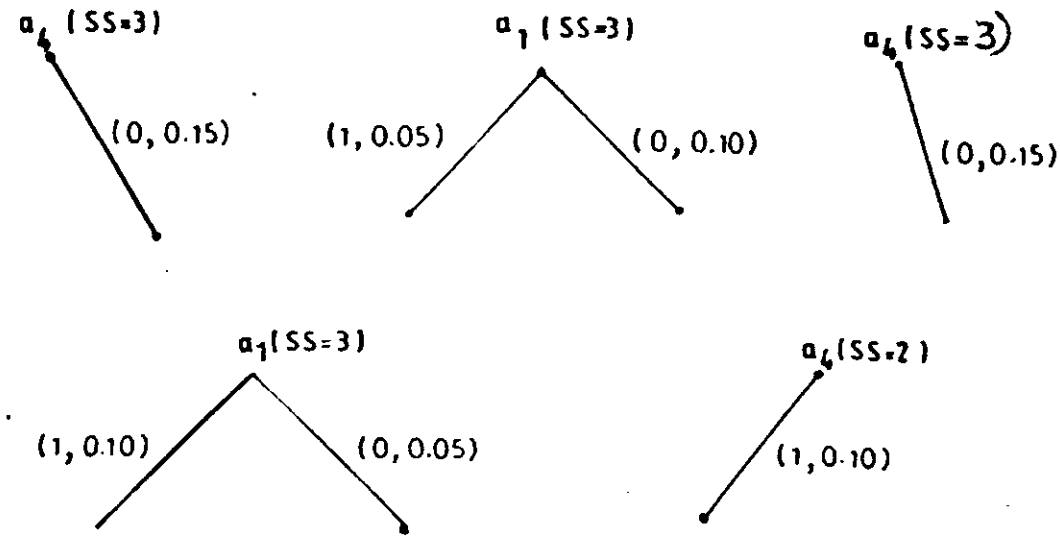



Figure 5: GTs $T_1$, $T_2$, $T_3$, $T_4$ and $T_{12}$ of one level

*The ?'s in the example set S represents missing values. The four GTs $T_1$, $T_2$, $T_3$ and $T_4$ corresponding to the four given attributes are constructed as given in Figure 5. The proportion of examples that these GTs cover is equal to (12/20 = 0.6) which is less than the given threshold. Hence, GT $T_{12}$ is constructed and is the last GT in Figure 5. The proportion now is equal to (14/20 = 0.7) which is equal to the given threshold. So, only $T_{12}$ is constructed. The best attribute at the root node is determined using the above five GTs as explained below:*

*Weighted frequency of* $a_1 = \frac{3*3/4+3*3/4}{20} = 0.225$

*Weighted frequency of* $a_2 = 0$

*Weighted frequency of* $a_3 = 0$

*Weighted frequency of* $a_4 = \frac{3*3/4+3*3/4+2*2/4}{20} = 0.275$

*So, $a_4$ is selected at the root in T. The GT T after one iteration is as given in Figure 6.*

*At the left child, the one level GTs under consideration are as given in Figure 7. The best attribute is $a_1$ at this node. Proceeding thus, we get the GT T as described in Figure 8.*

**Theorem 4** *The GT generated using the above algorithm converges to the true GT, provided that in the completely known case with no missing values for attributes, there are no ties in the selection of attributes.*

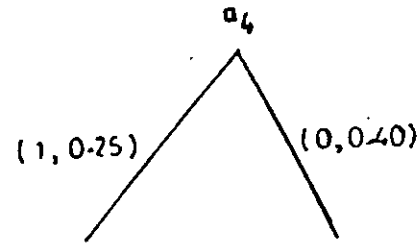



Figure 6: GT T of one level built with all the appropriate examples

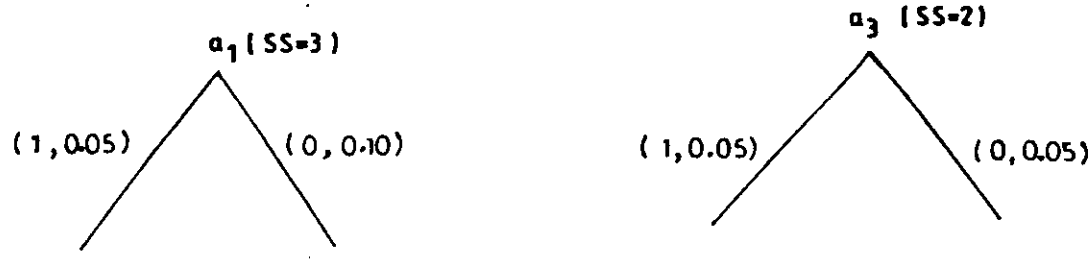



Figure 7: GTs $T_2$ and $T_{12}$ of one level

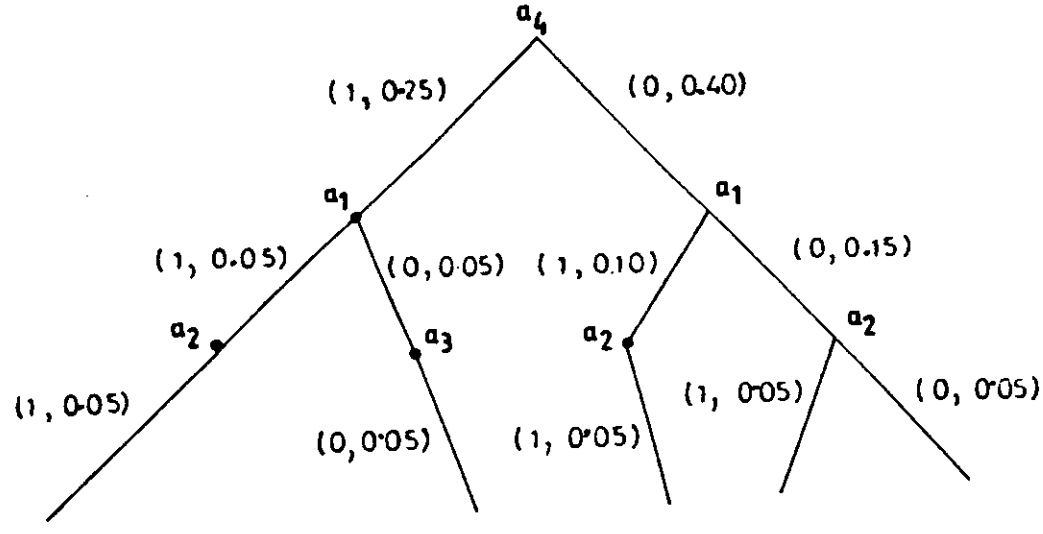



Figure 8: The Complete GT T

**Proof** *The uniqueness of the GT implies that in all the trees $T_1$, $T_2$,...,$T_r$ except one corresponding to the case for which the particular attribute is missing, $\exists N$ such that, the root attribute must necessarily occur at the root for $n > N$, by Glivenko-Cantelli theorem. Proceeding thus, it can be shown that the GT constructed will tend to the limiting GT as $n \longrightarrow \infty$.*
□

Note The GT corresponding to the reference class will hereafter be referred to as reference tree.

In this section, we have described an algorithm for constructing the GT for a reference class from examples of all the classes for binary, real/integer and nominal valued attributes. The algorithm has been proved to converge for all types of attributes. A provably convergent algorithm to deal with missing attributes was also described.

# 3 Learning Elementary Class Description

The description of an elementary class is learnt from its examples and the reference class description. This description represented as a binary tree has the same attribute sequence as the reference tree. The value ranges and the frequencies are determined from the statistics of its examples. This representation enables easy determination of redundancy and importance of attributes with respect to the given class.

In Section 3.1, the algorithm for construction of elementary class description is described. A methodology of determining redundancy of attributes is explained for binary, real/integer and nominal valued attributes. The construction of the tree description is intuitively based on selecting more important attributes higher in the GT. The concept of importance of an attribute for a class is formalised in Section 3.3.

## 3.1 Algorithm for Constructing GT of Elementary Class

1. Set the pointer to the root of the reference tree. The GT of current class $C_t$ has a corresponding node with the same attribute at its root.

2. Consider all the examples of $C_t$ at this node.

3. Label the current node of the tree corresponding to the class $C_t$ with attribute $a_r$ (note the initialisation corresponding to root in step 1).

4. If the attribute $a_r$ at the current node of the reference tree is of binary type, then find the frequency of those examples in $C_t$ at the current node having attribute value 1 and those having value 0. These frequencies are stored along the corresponding arcs to the child nodes.

   If the attribute $a_r$ at current node of the reference tree is of nominal type, then do the following : find the frequency of examples having the same value as the value in the left arc of the reference tree for $a_r$. Label the left arc at the current node with this value. All other values of $a_r$ which occur in the example set at the current node are the values of the right arc.

   If the attribute is of real/integer type, then do the following : if the attribute value of an example falls under the range of the left arc of the reference tree, it adds to the left arc frequency; else it adds to the frequency of the right arc at the current node.

   The attribute labels at the child nodes correspond to the label of the corresponding child nodes of the reference tree.

   **Note** The ranges on the left and right arcs of the current node are subsets of the ranges of the left and right arcs corresponding to the reference tree.

5. Repeat steps 3 and 4 corresponding to the examples of $C_t$ which satisfy the attribute restrictions obtained in the path traversal, until the GT is fully constructed in consonance with the reference tree.

**Example 7** *Consider the following example for two classes with three attributes, the first binary, the second nominal and the third real. Suppose the expansion threshold is 0.3.*

$S_1 = \{(1\ a\ 0.5), (1\ a\ 0.6), (1\ a\ 0.7), (1\ b\ 0.9), (1\ b\ 1.0), (1\ b\ 1.5), (1\ b\ 1.7), (0\ a\ 2.0),$
$(0\ a\ 2.2), (0\ b\ 2.5)\}$

$S_2 = \{(1\ a\ 0.5), (1\ a\ 0.6), (1\ a\ 0.7), (1\ a\ 0.9), (0\ a\ 0.8), (0\ a\ 1.0), (0\ a\ 1.6), (0\ a\ 2.1),$
$(0\ a\ 2.2), (0\ b\ 2.4)\}$

*The reference tree generated using the maximum representation learning criterion is shown in Figure 9.*

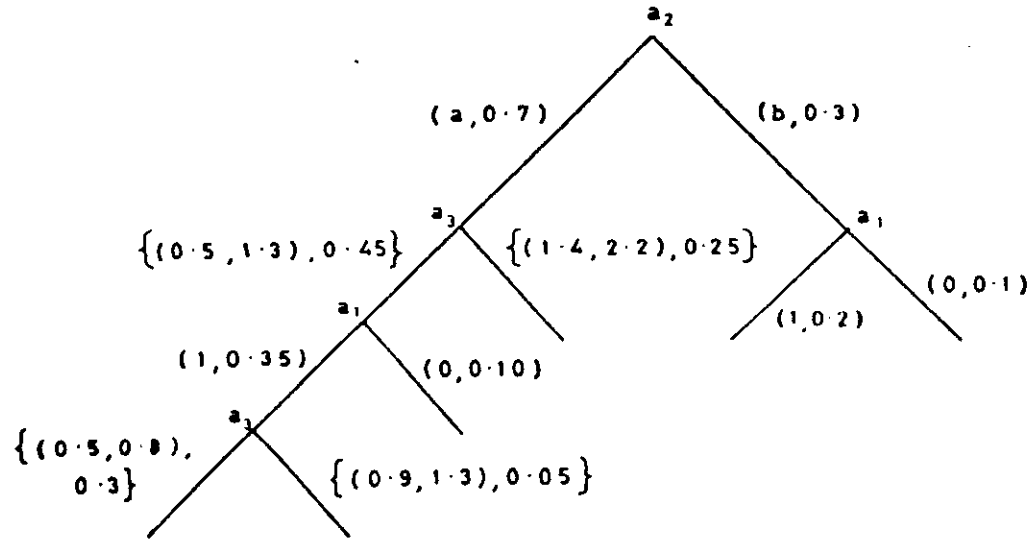



Figure 9: The Reference GT

*The GTs corresponding to the two classes are built using the attribute sequence in this reference tree as explained below : at the root node the attribute $a_2$ is selected and the frequency of examples assuming value 'a' and value 'b' is determined. In the set $S_1$, there are five examples assuming value 'a' and five examples assuming value 'b' for $a_2$. At the left child of the root, the attribute in the reference tree is $a_3$. All those examples at this node in class 1 whose value for $a_3$ falls within the range (0.5,1.3) represent its left child. Those examples whose value falls within the range (1.4,2.2) represent the right child of the current node. Proceeding thus, we have the GT for class 1 illustrated in Figure 10 and the GT corresponding to class 2 in Figure 11.*

Note The number of examples to be examined for construction of the GT of an elementary class is equal to n*m, where 'n' is the number of examples in the class and 'm' is the number of attributes.

By an argument along the lines of theorem 1, we get

**Theorem 5** *The GT for the elementary class converges to the true description of the class with respect to the reference tree.*

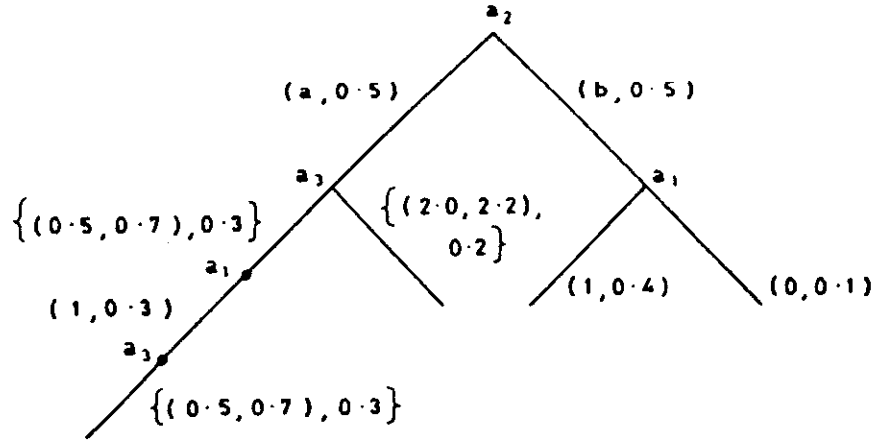



Figure 10: GT for class 1

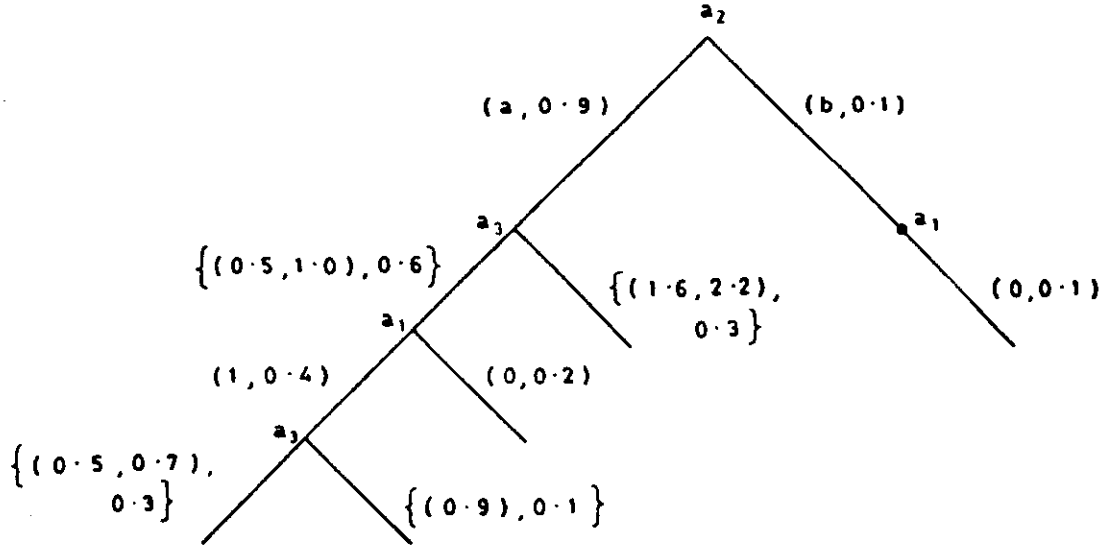



Figure 11: GT for class 2

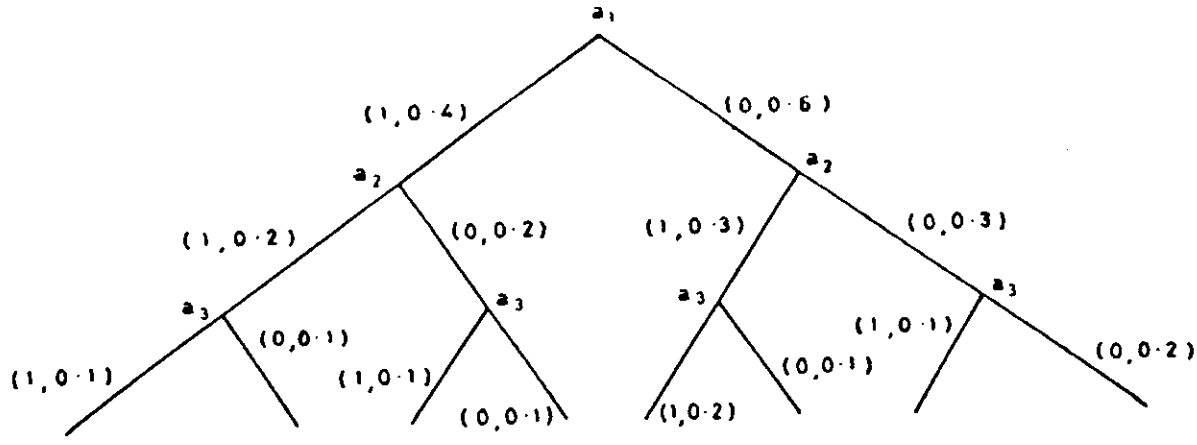



Figure 12: GT for binary valued attributes

## 3.2 Redundancy of an Attribute at a Node of a GT

Determining whether or not an attribute is redundant and the elimination of redundancy leads to simplification of class description. This is achieved relatively easily in our framework. While the GT is constructed top-down, redundancy is determined bottom-up eliminating nodes at which the weights are approximately equal. The reason that the complete GT has to be constructed prior to the elimination of redundancy is because of the existence of dependencies, one of which is illustrated in example 2.

**Definition 1** *In the binary valued and real/integer valued attribute case, an attribute is said to be redundant at a node if either both its descendants are leaf nodes or the attributes at all its descendant nodes are redundant and the frequency of its left and right arcs are equal.*

**Example 8** *Suppose the given set of examples is as follows:*

$S = \{(1\ 1\ 1),(1\ 1\ 0),(1\ 0\ 1),(1\ 0\ 0),(0\ 1\ 1),(0\ 1\ 1),(0\ 1\ 0),(0\ 0\ 1),(0\ 0\ 0),(0\ 0\ 0)\}$

*The corresponding GT is as shown in Figure 12. The GTs after removing the redundant nodes one step at a time are depicted in Figure 13.*

**Example 9** *Suppose the following set of examples is given:*

$S = \{(1\ 1\ 3),(1\ 1\ 5),(1\ 0\ 8),(1\ 0\ 10),(0\ 1\ 15),(0\ 1\ 16),(0\ 1\ 17),(0\ 1\ 18),$

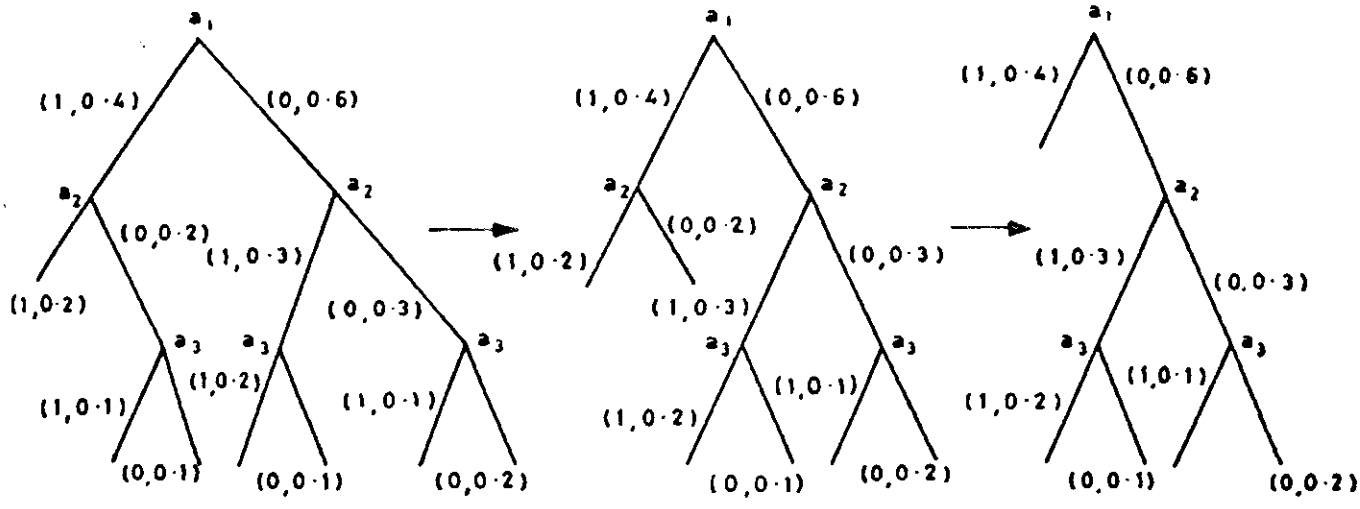



Figure 13: GTs depicting redundancy at a node

*(0 0 20),(0 0 21)}*

*The corresponding GT is given in Figure 14. The GTs after removing the redundant nodes one step at a time are shown in Figure 15.*

**Definition 2** *In the nominal valued attribute case, an attribute is said to be redundant at a node, if either both its descendants are leaf nodes or the attributes at all its descendant nodes are redundant and the frequency of the right arc is equal to the product of the cardinality of the values on the right arc and the frequency of the left arc.*

**Example 10** *Suppose the following set of examples is given:*

$S$ = *{(1 red),(1 red),(1 blue),(1 blue),(1 black),(1 black),(0 blue),*
*(0 blue),(0 blue),(0 black)}*

*The corresponding GT is as depicted in Figure 16. The GTs after removing the redundant nodes one step at a time are as shown in Figure 17.*

Note The equality is in the ideal case. In practice, however, for binary, real and integer valued attributes if the percentage difference in frequency is less than a specified threshold, then that node can be deleted. For nominal valued attributes if the percentage difference of cardinality * frequency of the left arc and frequency of the right arc is less than a specified threshold, then that node can be deleted.

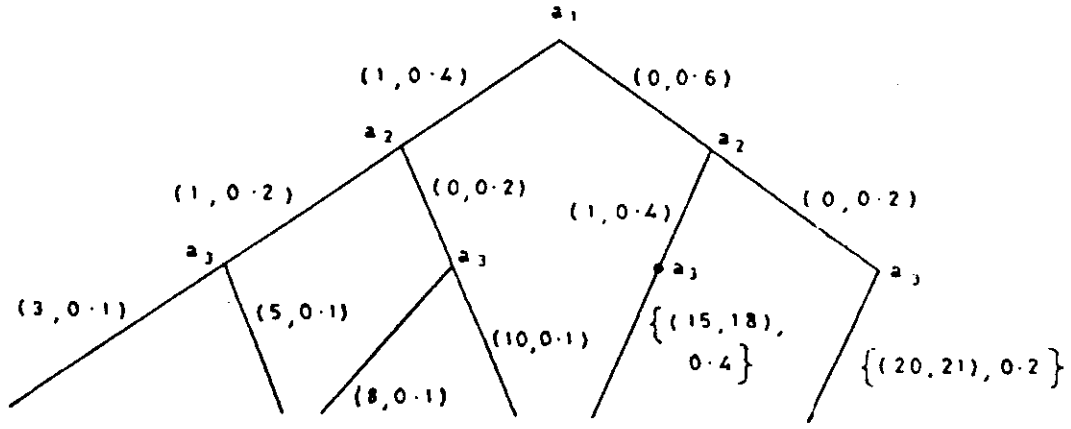



Figure 14: GT for binary and integer valued attributes

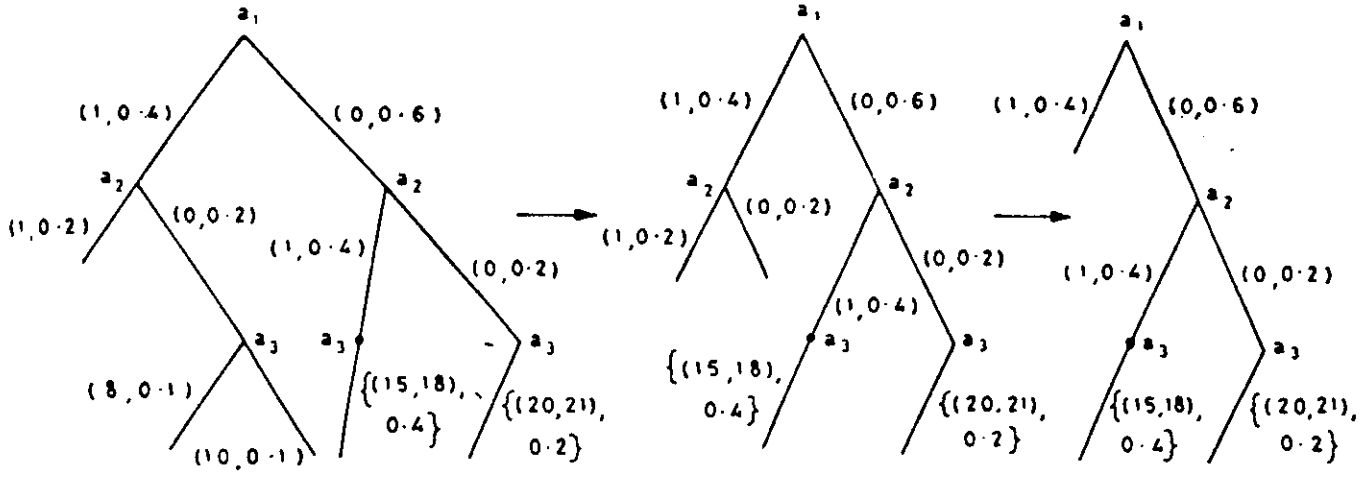



Figure 15: GTs depicting attribute redundancy at a node

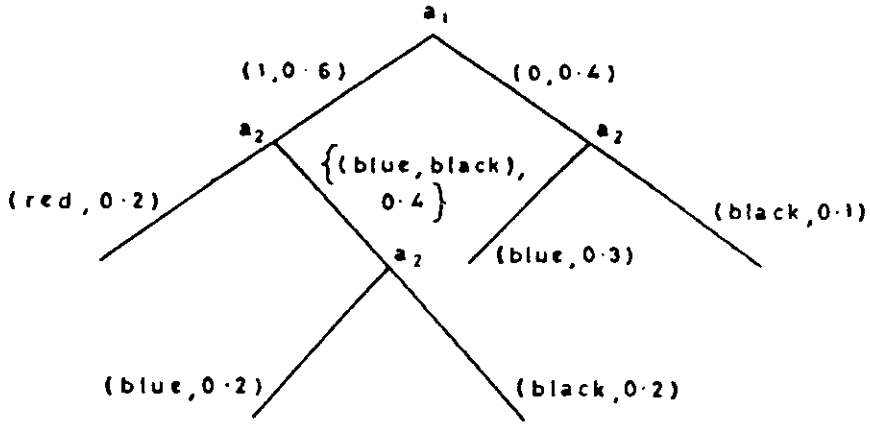



Figure 16: GT for binary and nominal valued attributes

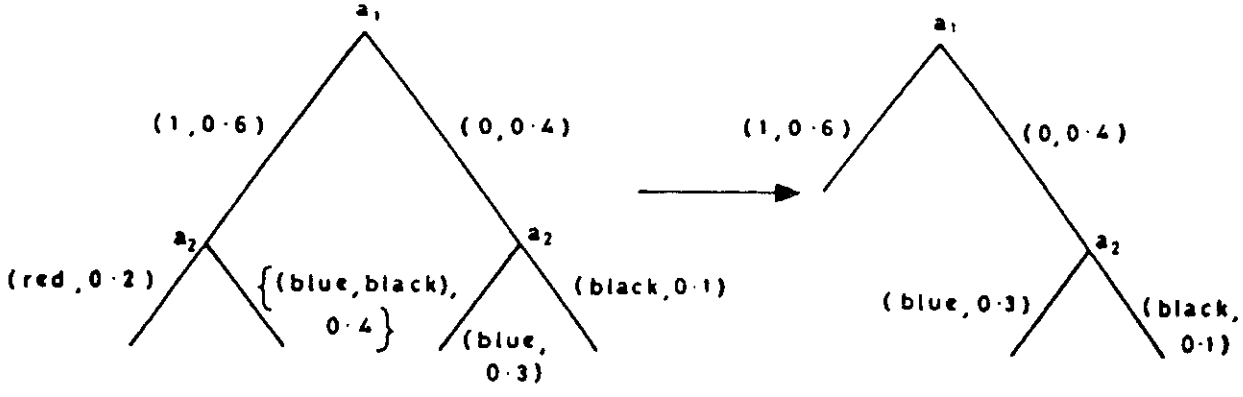



Figure 17: GTs depicting attribute redundancy at a node

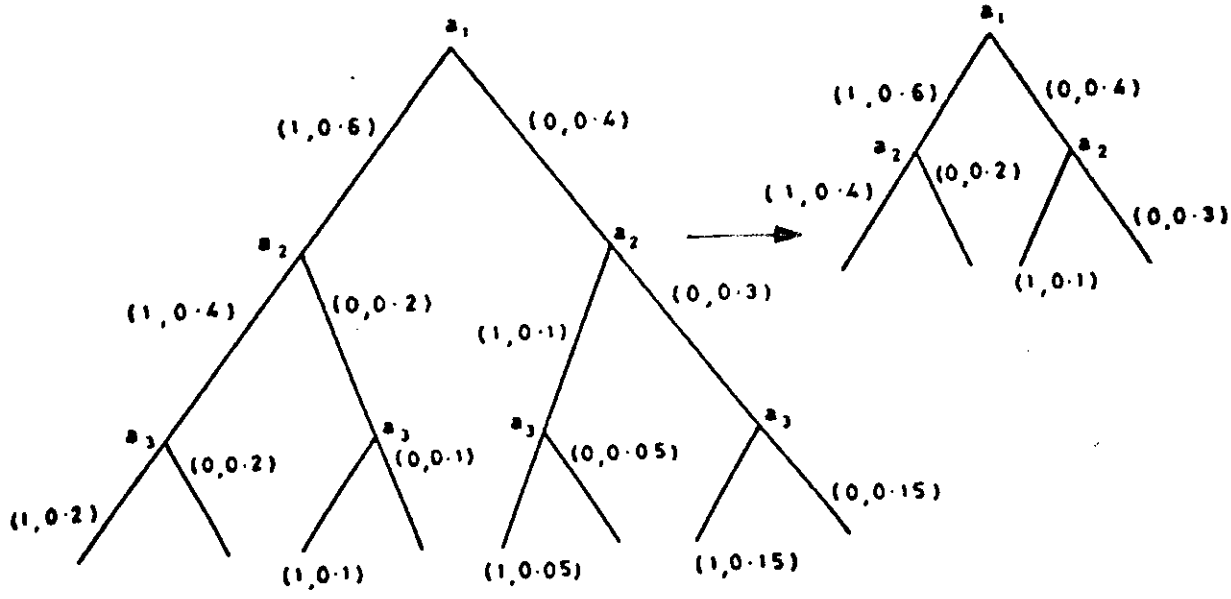



Figure 18: GTs depicting attribute redundancy for a class

**Definition 3** *An attribute is redundant for a class if it is redundant at all the nodes where it is chosen, i.e., it does not occur at any node of the GT after the redundancy check is applied.*

**Example 11** *Suppose the following set of examples is given:*

$S = \{(1\ 1\ 1),(1\ 1\ 1),(1\ 1\ 1),(1\ 1\ 1),(1\ 1\ 0),(1\ 1\ 0),(1\ 1\ 0),(1\ 0\ 1),(1\ 0\ 1),$
$(1\ 0\ 0),(1\ 0\ 0),(0\ 1\ 1),(0\ 0\ 1),(0\ 0\ 1),(0\ 0\ 1),(0\ 0\ 0),(0\ 0\ 0),(0\ 0\ 0),$
$(1\ 1\ 0),(0\ 1\ 0)\}$

*The corresponding GTs are as shown in Figure 18.*

**Definition 4** *A class is said to be a null class, if each element of the cartesian product of the domain of attributes occurs with equal frequency. In the GT framework, such a class should be represented by a single node.*

Note The definition of redundancy helps reduce the GT in such a case to a single node. This is illustrated by the following example.

**Example 12** *Suppose the given set of examples is as follows:*

$S = \{(1\ 1\ 1),(1\ 1\ 0),(1\ 0\ 1),(1\ 0\ 0),(0\ 1\ 1),(0\ 1\ 0),(0\ 0\ 1),(0\ 0\ 0)\}$

*The corresponding GTs are as shown in Figure 19.*

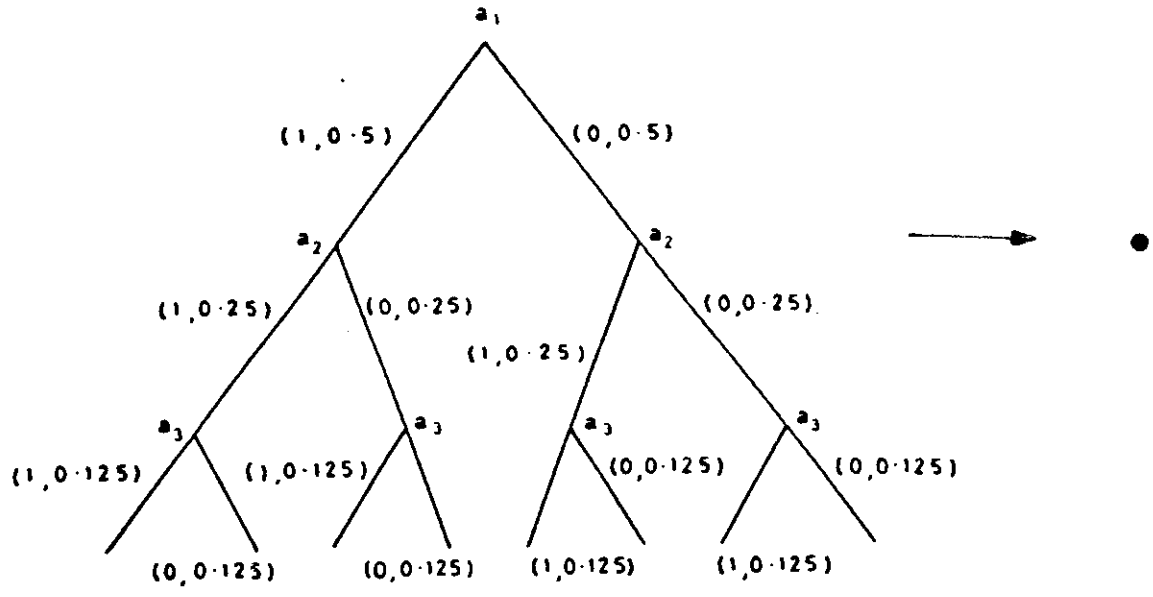



Figure 19: Reduction of null class GT to a single node

## 3.3 Importance of Attributes

The importance of an attribute is the relevance of the attribute for a particular class. It is denoted as $I(a_r \mid C_x)$, read as importance of an attribute $a_r$ given class $C_x$. This is relevant for inference of a new example to be classified at a later instance.

Since the importance of an attribute is judged by the representativeness of the attribute in the class, we consider the importance of an attribute given the class as directly proportional to the frequency at the node at which it occurs and inversely proportional to the distance of the node from the root [Pras 87].

$$I(a_r \mid C_x) \; \alpha \; f_r \tag{1}$$

$$I(a_r \mid C_x) \; \alpha \; 1/d_r \tag{2}$$

where

$f_r$ is the frequency at the node where $a_r$ occurs (this frequency is equal to the sum of the arc frequency at this node) and $d_r$ is the distance of the node from the root.

As a first approximation we have chosen the following form for $I(a_r \mid C_x)$ which satisfies eqns. (1) & (2).

$$I(a_r \mid C_x) = \sum_{p=1}^{t_r} A * (n - d_p) * f_p \tag{3}$$

and

$$\sum_{r=1}^{n} I(a_r \mid C_x) = 1$$

$$i.e., \quad \sum_{r=1}^{n} \sum_{p=1}^{t_r} A * (n - d_p) * f_p = 1 \tag{4}$$

where
n is the distance from leaf to the root node (i.e. equal to the number of attributes considered), A is the normalising factor, and the summation is over all the nodes where $a_r$ occurs.

**Normalising factor (for a specific tree)** The normalising factor 'A' of a GT where each attribute $a_r$ occurs at a distance (r - 1) from the root node along every path is equal to

$$\frac{2}{(n * (n+1))}$$

**Proof** Consider the GT in Figure 20 where $a_1$ is the most representative attribute at level 1 (distance 0 from root), $a_2$ for all nodes at level 2 (distance 1 from root node), etc.

Since every attribute occurs at the same depth along all the paths, we have $d_p = d_r$. Therefore,

$$\sum_{r=1}^{n} A * (n - d_r) * \sum_{p=1}^{t_r} f_p = 1$$

But every $\sum_{p=1}^{t_r} f_p$ for each level is equal to 1. Therefore,

$$A * \sum_{r=1}^{n} (n - d_r) = 1$$

Since $d_1 = 0, d_2 = 1, ....., d_n = n - 1$

$$A * n + (n-1) + (n-2) + ..... + 1 = 1$$

$$A = \frac{2}{(n * (n+1))} \quad \square$$

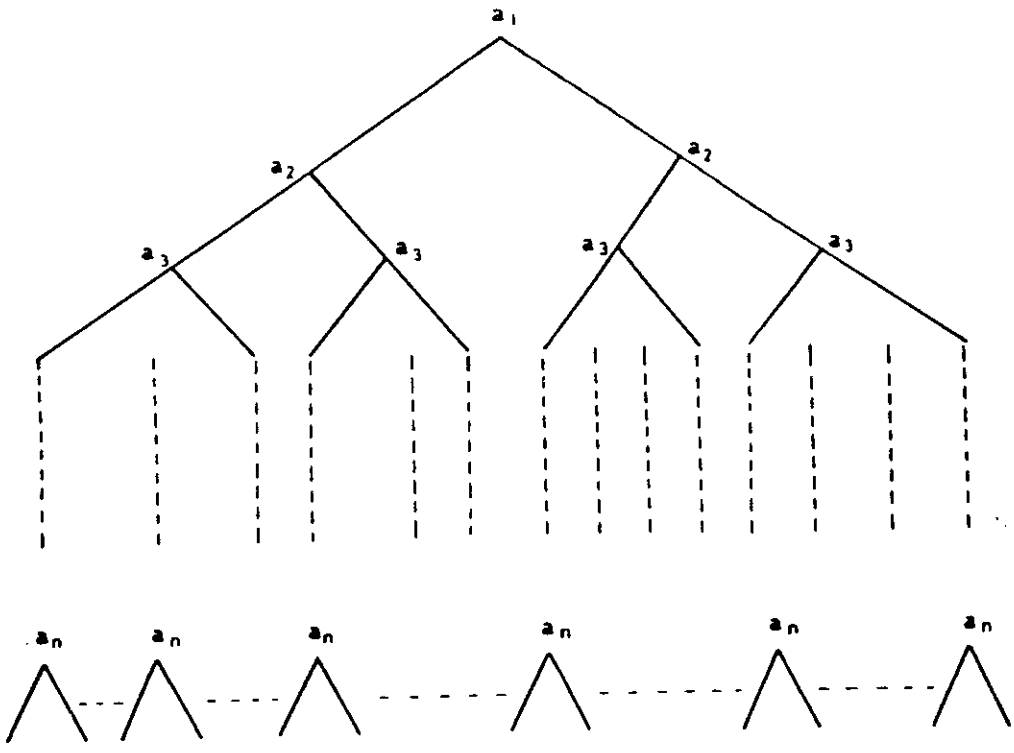



Figure 20: A GT of depth n

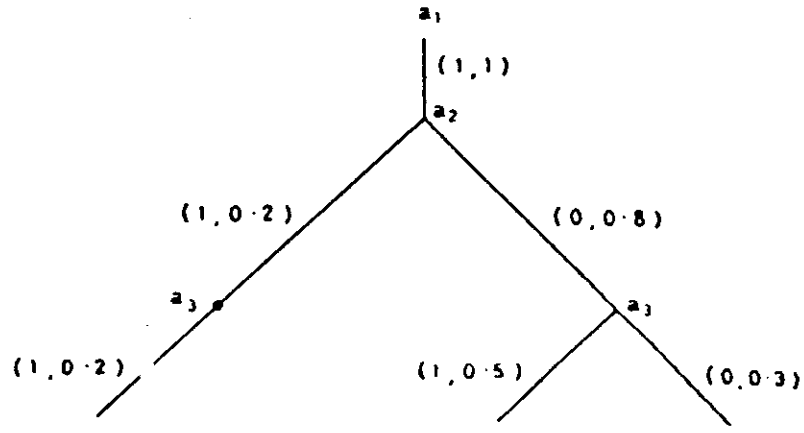



Figure 21: GT for binary valued attributes

Therefore,

$$I(a_r \mid C_x) = \frac{2}{(n*(n+1))} * \sum_{p=1}^{t_r}(n - d_p) * f_p \tag{5}$$

It is very easy to see why this value of A will be true for any GT.

**Normalising factor (for a general tree)** The normalising factor 'A' for any GT is equal to

$$\frac{2}{(n*(n+1))} \tag{6}$$

**Proof** We give a constructive proof for this claim. Consider the term within the summation in eqn.(4). For a given attribute $a_r$, this is the contribution from all the nodes where $a_r$ occurs. The sum of the importance for all the attributes is equivalent to summing up the contribution from each node of the tree. Consequently, the value of A as in (6) will be the normalising factor for any GT. □

**Example 13** *Consider the GT in Figure 21. The importance of the three attributes* $a_1$, $a_2$ *and* $a_3$ *are evaluated as shown below :*

$$I(a_1 \mid C_1) = \frac{2}{(3*(3+1))} * \{(3-0)*1\} = 0.5$$

$$I(a_2 \mid C_1) = \frac{2}{(3*(3+1))} * \{(3-1)*0.2 + (3-1)*0.8\} = 0.33$$

$$I(a_3 \mid C_1) = \frac{2}{(3*(3+1))} * \{(3-2)*0.2 + (3-2)*0.5 + (3-2)*0.3\} = 0.17$$

This section dealt with the construction of elementary class GT from its examples and the reference tree. An algorithm to find redundant attributes was proposed. Determination of the importance of an attribute for a class for a GT representation was also explained.

# 4 The Inference Process

The inference process finds the best match between the test example and the GTs of each class. This is done as follows:

1. Evaluate $f(e \mid C_i)$, the validity of class $C_i$ using the GT corresponding to this class as shown below:

$$\begin{aligned} f(e \mid C_i) &= \{f(e_{p1} \mid C_i) * \sum_{l=1}^{n_1} I(e_l) + f(e_{p2} \mid C_i) * \sum_{l=1}^{n_2} I(e_l) + \\ &\quad f(e_{p3} \mid C_i) * \sum_{l=1}^{n_3} I(e_l) + ..... + f(e_{pk} \mid C_i) * \sum_{l=1}^{n_1} I(e_l)\} \\ &= \sum_{j=1}^{k} (f(e_{pj} \mid C_i) * \sum_{l=1}^{n_j} I(e_l) \end{aligned}$$

where
$f(e \mid C_i)$ is the validity of class $C_i$ given example e, pj is the jth path which the example matches either completely or incompletely (incompletely if the example has missing values), $e_{pj}$ is the set of (attribute, value) pairs along path pj, $f(e_{pj} \mid C_i)$ the validity of class $C_i$ given $e_{pj}$ is the leaf frequency along path pj, nj is the number of attributes along path pj for which attribute value is available in example e.

The validity defined above is based on the category validity of Medin, et.al.[Medn 87] which is defined as the probablity that an entity (or example) has some feature given

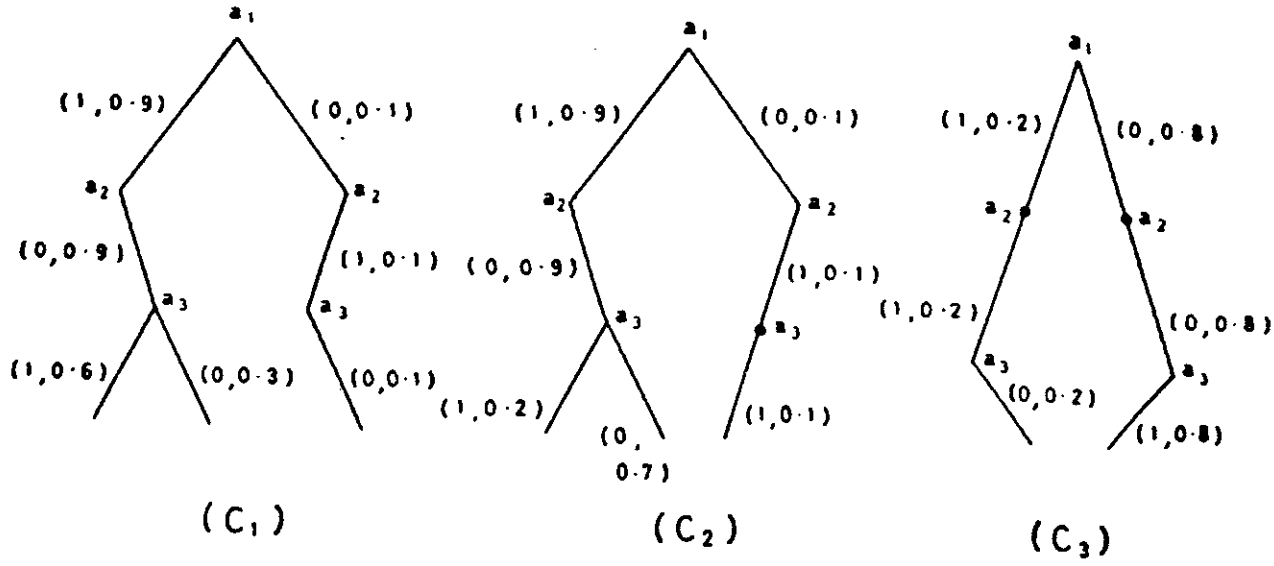



Figure 22: GTs for classes $C_1$, $C_2$, and $C_3$

that it belongs to a category (or class). We have used an extension of this which takes into account the absence of certain attributes. This has been done by weighting the category validity by the summation of the inportance of only those attributes that are present.

2. Select that $C_i$ for which $f(e \mid C_i)$ is the maximum.

**Note** It is quite straight forward to establish that the value of any $f(e \mid C_i)$ lies between 0 and 1.

**Example 14** *Suppose we have three classes for which the GTs are as given in figure 22. Suppose the test example is* $S = \{(a_1 = 1)(a_2 = ?)(a_3 = 0)\}$ *then, we have,*

$$f(S \mid C_1) = 0.6 * 2/3 = 0.4$$

$$f(S \mid C_2) = 0.1 * 2/3 + 0.2 * 2/3 = 0.2$$

$$f(S \mid C_3) = 0.2 * 2/3 = 0.133$$

Therefore, S is classified as $C_1$.

# 5 Practical Applications of the proposed system

The proposed system has been implemented and tested on two applications reported in the literature. In these applications, part of the example set was used for learning and the remaining for testing. It is not clear from the literature [Schl 87, Aha 88, Brie 84] as to how the example set was segmented into learning and testing sets. In certain of the following experiments the programs were run at least twice with different learning/test examples. A comparison of the proposed system is made with other learning systems in respect of classification accuracy (defined as the ratio of rightly classified to the total). Various runs of the proposed system for the same application are compared with respect to the classification accuracy, the number of sub-descriptions and the number of terms in a sub-description.

The inputs to the proposed system are the learning examples, expansion threshold, sample size of each class, attribute names, and the test examples. For each class, the frequency of rightly classified, frequency of not classified, frequency of wrongly classified and the number of sub-descriptions are output. The classification accuracy is evaluated from these frequencies. The results of the two applications are described in the following paragraphs.

## 5.1 Classification of different kinds of glasses

This application is concerned with classification of different types of glass required in some criminal investigations. At the scene of the crime, the glass left can be used as evidence if it is correctly identified. Each example is specified by 9 real valued attributes. There are 6 classes whose descriptions are to be learnt.

The first experiment was run with expansion threshold = 0.00. The classification accuracy is approximately 96%. The frequency details are as given in Table 1. The second experiment was run with expansion threshold equal to 0.10. The classification accuracy is approximately 78%. The frequency details are as given in Table 2.

Comparative figures for this application are available only for classes 1, 2 and 3 [Aha 88]. Classes 1 and 3 are windows that are float processed and class 2 is window that is not. The proposed system with expansion threshold = 0.10 was compared with three other systems, viz., Beagle (a rule based system), Nearest Neighbour and Discriminant Analysis. The error frequencies obtained in determining whether the glass was a type of float glass or not, for the

| Class | freq. of learning examples | number of sub-des | freq. of rightly classd. | freq. of not classd. | freq. of wrongly classd. |
|---|---|---|---|---|---|
| 1 | 70 | 31 | 65 | 2 | 3 |
| 2 | 76 | 42 | 73 | 1 | 2 |
| 3 | 17 | 14 | 16 | 0 | 1 |
| 4 | 13 | 8 | 13 | 0 | 0 |
| 5 | 9 | 9 | 9 | 0 | 0 |
| 6 | 29 | 15 | 29 | 0 | 0 |

Table 1: Freqs. when expansion threshold = 0.00

four learning systems is tabulated in Table 3. In this experiment, the proposed system has the least error.

## 5.2 Classification of cars based on risk factor

This application concerns classifying cars into different categories based on its risk factor. Every car is assigned a risk factor, an integral value varying from -3 to +3. The value -3 indicates that the car is very safe and +3 indicates that the car is very unsafe. There was no car with value -3. So, there were six classes involved. The number -2 was represented as 1, -1 as 2 and so on. Each example was represented by 25 attributes. The system was run with expansion threshold = 0.10. The classification accuracy achieved is 70.1%. The details of the experiment is as recorded in Table 4. In this experiment 14 out of the 25 attributes were either redundant or had very low importance value. The program was run with the same threshold after deleting these 14 attributes. There is a marginal increase in the classification accuracy as can be seen in Table 5.

| Class | freq. of learning examples | number of sub-des | freq. of rightly classd. | freq. of not classd. | freq. of wrongly classd. |
|---|---|---|---|---|---|
| 1 | 70 | 14 | 43 | 2 | 25 |
| 2 | 76 | 16 | 57 | 6 | 13 |
| 3 | 17 | 9 | 13 | 1 | 3 |
| 4 | 13 | 4 | 11 | 2 | 0 |
| 5 | 9 | 5 | 9 | 0 | 0 |
| 6 | 29 | 4 | 25 | 0 | 4 |

Table 2: Freqs. when expansion threshold = 0.10

| Type of Glass | sample size | Beagle | NN | DA | The proposed system |
|---|---|---|---|---|---|
| Float | 87 | 10 | 12 | 21 | 20 |
| Non-Float | 76 | 19 | 16 | 22 | 6 |

Table 3: Error frequencies

| Class | freq. of learning examples | number of sub-des | freq. of rightly classd. | freq. of not classd. | freq. of wrongly classd. |
|---|---|---|---|---|---|
| 1 | 3 | 1 | 3 | 0 | 0 |
| 2 | 20 | 6 | 22 | 0 | 0 |
| 3 | 48 | 17 | 44 | 6 | 17 |
| 4 | 46 | 14 | 40 | 2 | 12 |
| 5 | 29 | 16 | 31 | 0 | 1 |
| 6 | 13 | 6 | 5 | 14 | 8 |

Table 4: Freqs. when expansion threshold = 0.10

| Class | freq. of learning examples | number of sub-des | freq. of rightly classd. | freq. of not classd. | freq. of wrongly classd. |
|---|---|---|---|---|---|
| 1 | 3 | 1 | 3 | 0 | 0 |
| 2 | 20 | 6 | 22 | 0 | 0 |
| 3 | 48 | 14 | 46 | 6 | 15 |
| 4 | 46 | 14 | 40 | 2 | 12 |
| 5 | 29 | 15 | 31 | 0 | 1 |
| 6 | 13 | 6 | 5 | 14 | 8 |

Table 5: Freqs. when 14 attributes are deleted and expansion threshold = 0.10

# 6 Conclusion

A system that learns elementary class descriptions using a reference class was described in this paper. The need for a reference class was emphasised for a simple case involving call for promotion. The reference class description and the elementary class descriptions learnt have been shown to converge in the limit. The concepts of redundancy and importance of an attribute for a class was formally defined. Finally an inference process which uses the class descriptions learnt and the importance of an attribute was proposed. The performance of the system is comparable to some of the common learning systems reported in the literature.

The proposed learning system is non-incremental. A natural extension is development of an incremental version. The results of the two applications are encouraging, but some more empirical testing is necessary to establish its generality.

# References


[Aha 88] Aha D., "Documentation of the UCI Repository of Machine Learning Databases", University of California, Irvine, USA, 1988.

[Arun 90] Arunkumar S., and Yegneshwar S., "Knowledge Acquisition from Examples using Maximal Representation Learning", to appear in the *7th International Machine Learning Conference*, Texas, USA, 1990.

[Brie 84] Brieman L., Friedman J.H., Olshen R.A., and Stone C.J., "Classification and Regression Trees", Belmont, Wadsworth, 1984.

[Bund 85] Bundy A., Silver B., and Plummer D., "An Analytical Comparison of Some Rule Learning Programs", *Artificial Intelligence*, vol.27, 1985, pp.137-181.

[Craw 89] Crawford S.L., "Extensions to the CART algorithm", *Intl. Jnl. of Man-Machine Studies*, vol.31, 1989, pp.197-217.

[DeGr 87] DeGroot M.H., "Probability and Statistics", Addison-Wesley Publishing Co. 1987.

[Doct 85] Doctor M., "Knowledge Acquisition for Expert Systems", *BTech. Dissertation*,

Dept. of Computer Science and Engineering, Indian Institute of Technology, Bombay, India, 1985.

[Duda 83] Duda R.O. and Shortliffe E.H., "Expert Systems Research", *Science*, vol.220, 1983, pp.261-268.

[Mant 91] Mantaras De Lopez R., "A Distance Based Attribute Selection Measure for Decision Tree Induction", *Machine Learning*, vol.6, 1991, pp.81-92.

[Medn 87] Medin D.L., Wattenmaker D.W., and Michalski R.S., "Constraints and Preferences in Inductive Learning : An Experimental Study of Human and Machine Performances", *Cognitive Science*, vol.11, 1987, pp.299-339.

[Mich 80] Michalski R.S. and Chilauski R.L.,"Knowledge Acquisition by Encoding Expert Rules vs. Induction from Examples : A case study involving Soybean Pathology", *Intl. Jnl. of Man-Machine Studies*, vol.12, 1980, pp.63-87.

[Mich 83] Michalski R.S., "A Theory and Methodology of Inductive Learning", *Artificial Intelligence*, vol.20, 1983, pp.111-161.

[Nort 89] Norton S.W., "Generating Better Decision Trees", in Proc. of the Intl. Joint Conf. on Artificial Intelligence, Morgan Kaufmann Publishers, 1989, pp.800-805.

[Pras 87] Prasadram R., "Risk Analysis for Intelligent Systems : A Study", *BTech. Dissertation*, Dept. of Computer Science and Engineering, Indian Institute of Technology, Bombay, India, 1987.

[Quin,79] Quinlan J.R., "Discovering Rules by Induction from Large Collections of Examples", in *Expert Systems in the Micro-electronic age*, ed.Donald Michie, Edinburgh University Press, 1979, pp.168-201.

[Quin 86] Quinlan J.R., "Induction of Decision Trees", *Machine Learning*, vol.1, 1986, pp.81-106.

[Schl 87] Schlimmer J.C., "Concept Acquisition through Representational Adjustment", *Doctoral dissertation*, Dept. of Information and Computer Science, University of California, Irvine, U.S.A., 1987.

[Yeg 90] Yegneshwar S., "A Hierarchical Approach to Knowledge Acquisition from Examples", *Doctoral dissertation*, Dept. of Computer Science and Engineering, Indian Institute of Technology, Bombay, India, 1990.