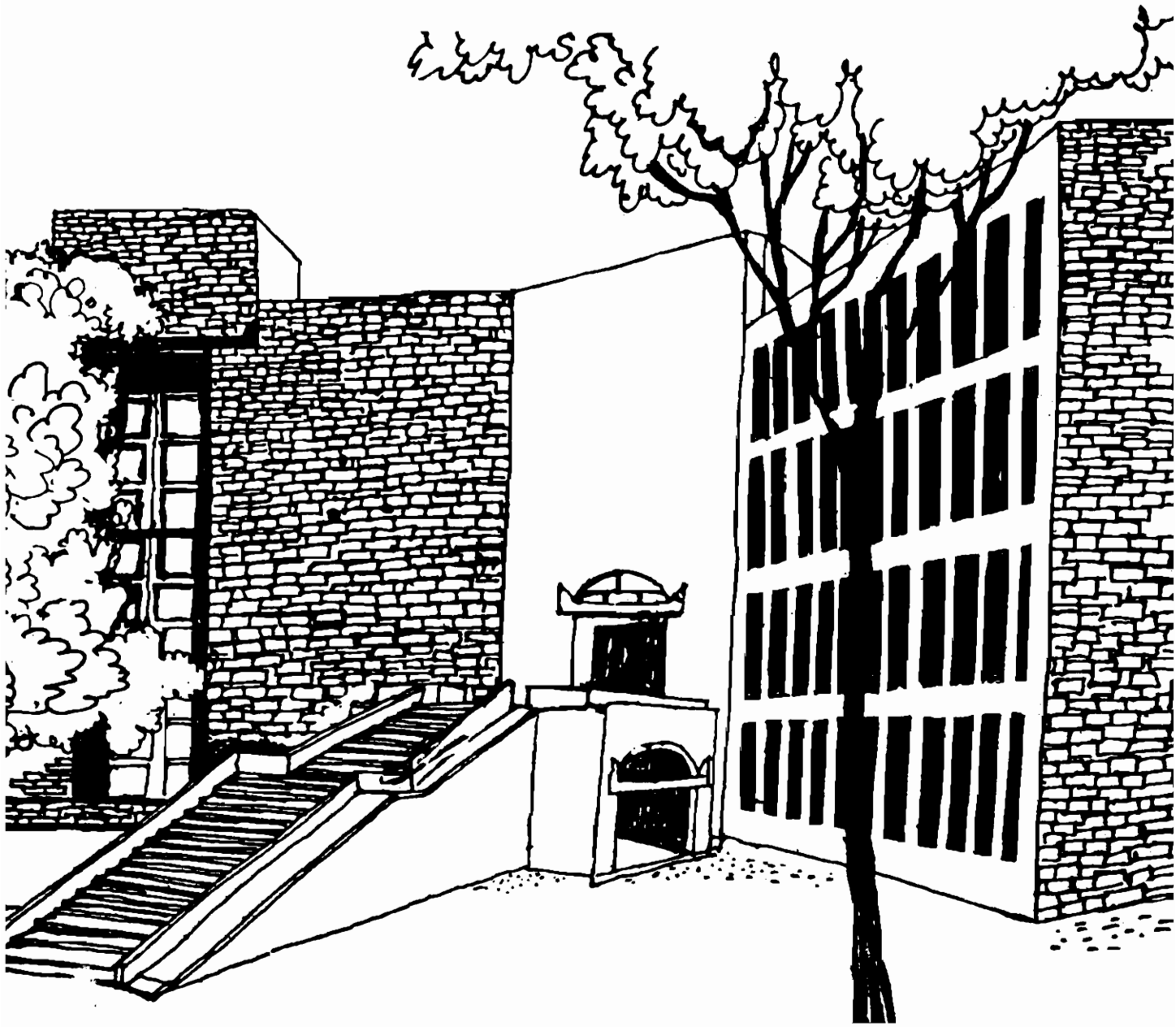
विद्याविनियोगाधिकारः
IIM
AHMEDABAD

Working Paper



LEARNING RELATIONSHIPS AMONG CLASSES SPECIFIED BY EXAMPLES

By

S. Yegneshwar

W P No. 999
January 1992

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

Learning Relationships Among Classes Specified by Examples

S.Yegneshwar

Abstract

Learning by examples is a commonly used method of knowledge acquisition in expert systems. The learning examples are past cases whose classification is known. When the number of classes is high, learning relationships amongst classes aids in sequential classification and therefore results in better explanations. A methodology to learn relationships amongst a given set of classes using a distance measure is described in this paper. This distance which is shown to be a metric is evaluated on the descriptions of the classes. The description of a class is learnt from its examples. The relationship learnt using this distance metric is shown to converge in the limit. The methodology described learns meaningful relationships for two applications commonly used by machine learning research groups.

Acknowledgements

I thank Mr. David Aha of the University of California, Irvine for sending me the automobiles and the thyroid disease databases which have been used to test the proposed system. The resource support given by the seed money project on "An expert system for trouble shooting of hardware failures in personal computers" is gratefully acknowledged.

Contents

Learning Relationships Among Classes Specified by Examples

1 Introduction

Learning class description from examples is an important problem in artificial intelligence. This method of learning is referred to as "instance based learning" or "learning from examples". This is a learning paradigm which is used for knowledge acqusition in expert systems for applications where past cases are available. Meta-DENDRAL [Buch 78] is one such pioneering learning system that induces rules of chemical structure from examples. However, this is a domain-dependent system. The initial domain-independent works which demonstrated the effectiveness are that of Michalski et. al. [Mich 80] and Quinlan [Quin 79]. Here, examples of each class are used to learn the class description. Michalski's INDUCE system demonstrated that for a particular case of plant diseases, the inductively acquired knowledge base performed better than the knowledge base acquired with the help of a knowledge engineer. ID3 [Quin 79] demonstrated good performance in chess end-games and thyroid diseases [Quin 87b]. Subsequently, IITN [Pao 82], Rule-Learning [Fu 85], PRG [Lee 86], Rough sets [Pawl 88], learning of causal relationships [Perl 88], CART [Craw 89], EG2 [Nunz 91], and a modified attribute selection for ID3 [Mant 91] followed.

In this learning paradigm, a teacher presents learning examples which are used to learn the class description. An example is specified as a set of (attribute,value) pairs - for instance, an example of the class "Default-Loans" could be ((turnover,50) (product,soap) (market-share,20)) and a description for this class could be "All examples for which turnover is between 25 and 100, product manufactured is soap, and market-share is less than 25 percent". The learning of the class description can be in one shot where all the examples are presented simultaneously, or it could be incremental. In the latter case, learning is invariably sequence

dependent. All the learning systems referred to above require all the learning examples to be presented at one time. Using the learning examples, these systems learn the description of the corresponding class. The class description learnt is represented either as a decision tree, or in first order predicate calculus, or as rules. The class description learnt can be used to describe the class represented by the given set of examples, or to classify a test example.

A major drawback of all the learning systems referred to above is that a description for each class is learnt but the inherent relationship among the classes is not learnt. Learning relationships among the classes helps learn intermediate classes with their class description and a hierarchy of classes. In this hierarchy, the root class is the most general and all nodes lower down are progressively more specific. The hierarchy aids not only in sequential classification but also results in better explanation during classification of test examples. Sequential classification is possible because at the higher level nodes the description is more general and hence the description consists of lesser number of attributes. As we go down the hierarchy, more attribute values are requested from the user during classification of test examples.

The advantages of learning a hierarchy of classes led to the proposal of a learning system called KAHLE [Yeg 90] (Knowledge Acquisition using Hierarchical Learning from Examples), which first learns a reference class description. Each class description is learnt using the reference class description and its own learning examples. Subsequently, relationships among the various classes is learnt using a distance measure. The description of each class is represented as a binary tree called Generation Tree (GT), and distance is evaluated by tree traversal and comparison of corresponding nodes. If the distance between two classes is low, then they are combined to form an intermediate class which is more general than the constituent classes. The comparison of classes is continued till no more combination is possible. The description of an intermediate class is learnt from the description of its constituent classes. In this paper, the method of learning individual class descriptions and the relationships among these classes based on the "closeness" of the classes is described. Each class description is learnt using a reference description and the "closeness" of classes is measured using a distance metric defined on the class descriptions learnt.

Section 2 of the paper describes learning of the reference class description from examples of all the classes. In section 3, learning of each class description from the reference class description and its own learning examples is described. Section 4 explains the evaluation of

the distance between class descriptions. Section 5 describes learning of a hierarchy of classes using the class descriptions and the distance measure. In section 6, the results of applying this learning methodology to some problems is described. Section 7 concludes the work with some suggestions for future work.

2 Learning Reference Class Description

The description and relative comparison of a given set of classes is facilitated by learning about the class of classes, which is referred to herein as the reference class. Learning of the reference class description is particularly useful to identify attributes which take similar values across a set of classes and attributes which take dissimilar values across this set of classes. The attributes which take similar values characterise this set of classes and the attributes which take dissimilar values discriminate each class from every other class in this set. 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 (GT) in the sequel 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.

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 individual class descriptions relative to this reference GT. The reference GT is constructed for attributes taking binary, real/integer and nominal values.

2.1 Generation Tree for Binary Valued Attributes

The concept of Generation Tree was first reported in [Doct 85]. However, the procedure used to create the GT from examples in this proposal had certain flaws which was rectified in a modified proposal [Arun 90]. This later proposal took care of only binary valued attributes. The procedure used in this proposal was later extended to non-binary valued attributes [Yeg 90].

In this section, this procedure which is used to learn the description of the reference class is briefly described. The process of learning the elementary class description is described in the next section.

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 called expansion threshold, 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.

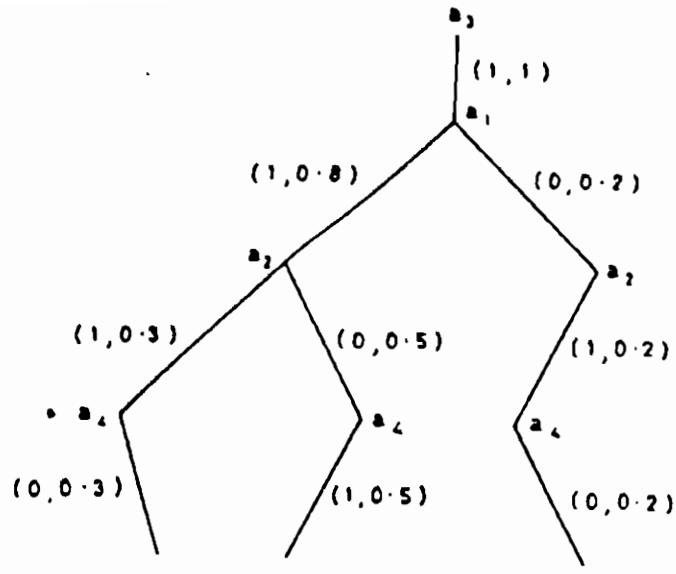


Figure 1: GT for binary valued attributes

$$S = \{(1011),(0110),(1011),(1110),(1011),$$
$$(0110),(1110),(1011),(1110),(1011)\}$$

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

$$\{(|a_1 = 1| = 8),(|a_1 = 0| = 2),(|a_2 = 1| = 5),$$
$$(|a_2 = 0| = 5),(|a_3 = 1| = 10),(|a_3 = 0| = 0),$$
$$(|a_4 = 1| = 5),(|a_4 = 0| = 5)\}$$

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.

2.2 Real/Integer and Nominal Valued Attributes

2.2.1 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 α*least count, where α 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.

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 2 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 2.

2.2.2 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}.

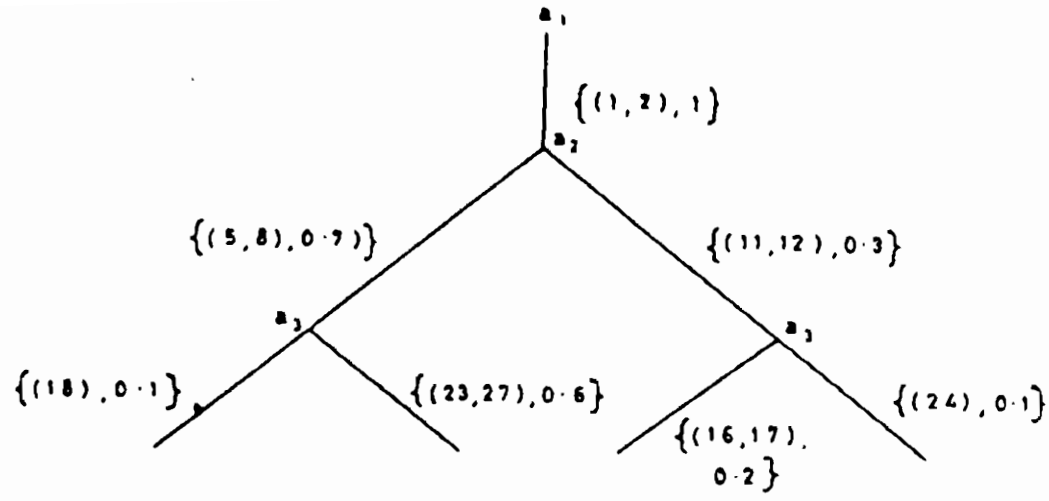


Figure 2: GT for integer valued attributes

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.

Example 3 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:

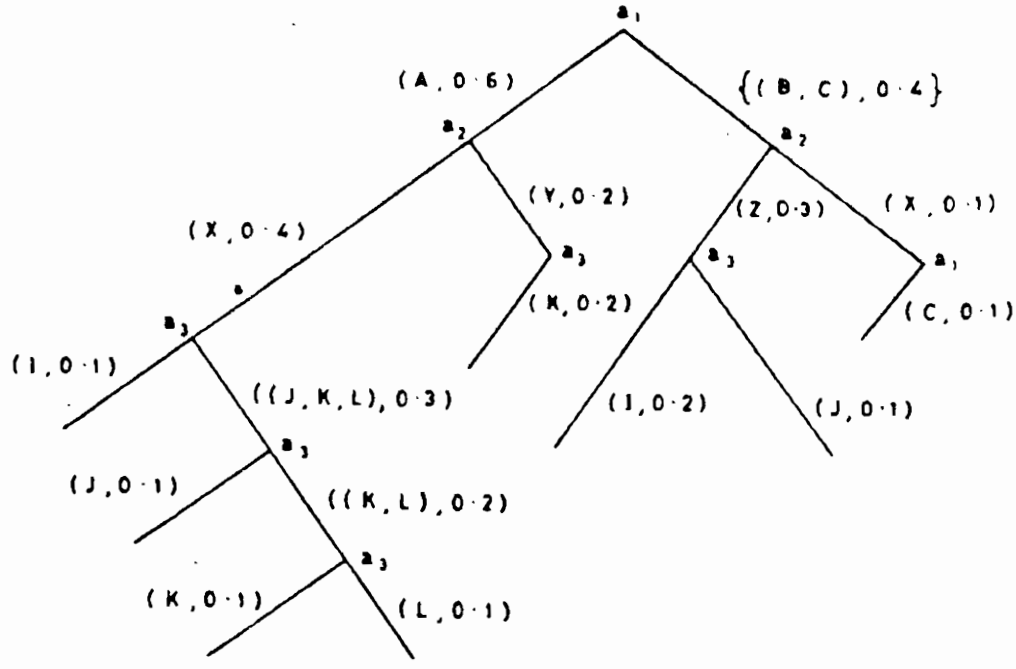


Figure 3: GT for nominal valued attributes

```
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 3.

3 Learning Elementary Class Description

The description of an elementary class (a class for which examples are given) 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.

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 4 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.

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

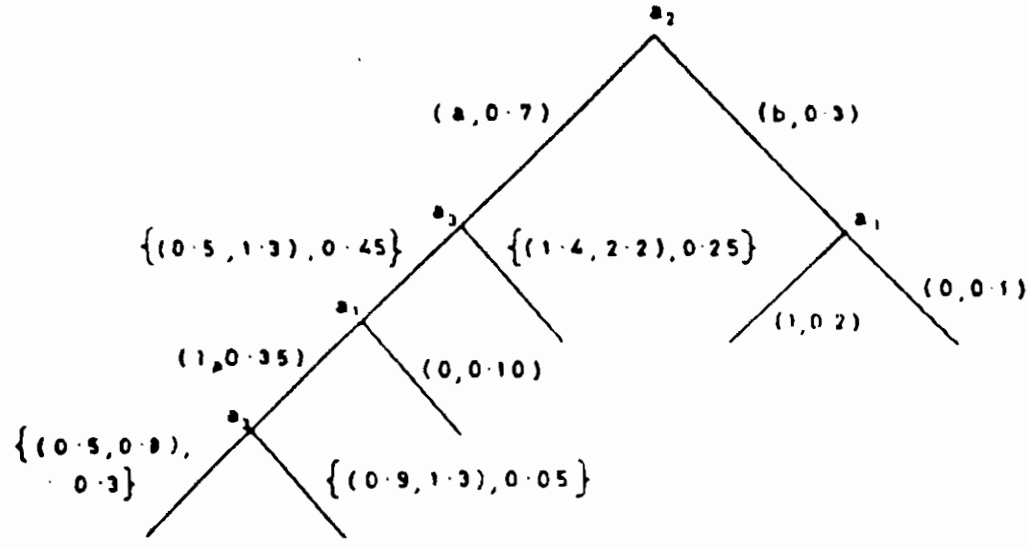


Figure 4: The Reference GT

$(0\ a\ 2.2), (0\ b\ 2.5)\}$

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

The reference tree generated using the maximum representation learning criterion is shown in Figure 4.

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 5 and the GT corresponding to class 2 in Figure 6.

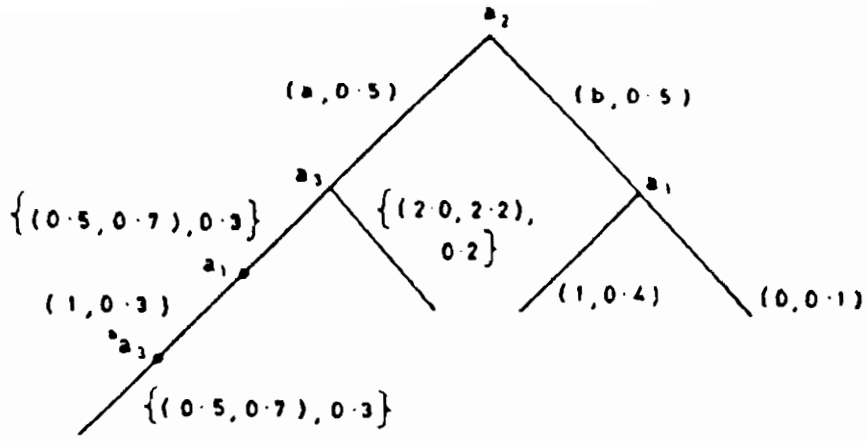


Figure 5: GT for class 1

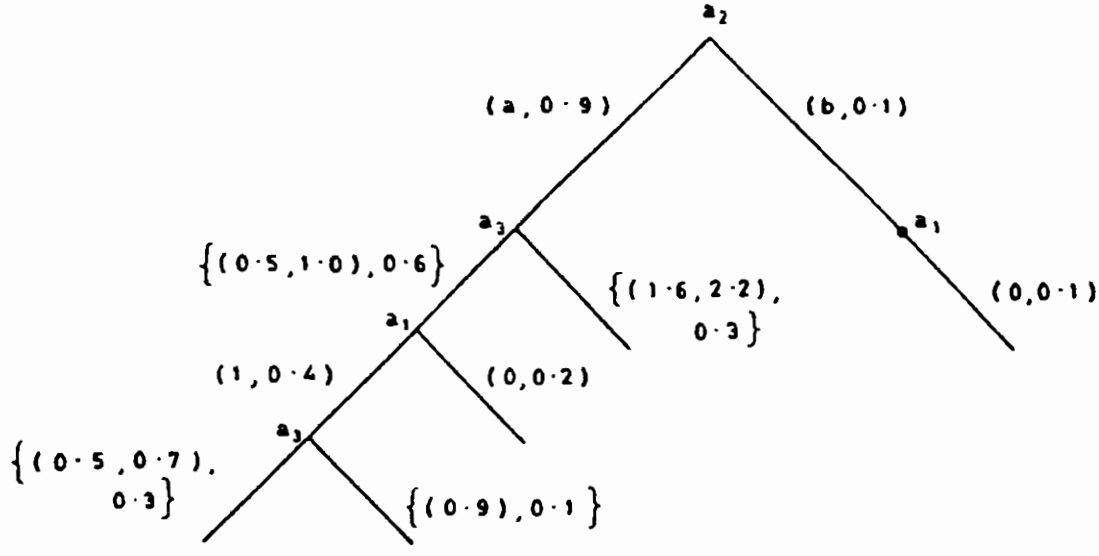


Figure 6: GT for class 2

4 A Distance Measure for Generation Trees

The evaluation of the distance between two class descriptions is required to find the degree of closeness of the classes. The distance for GTs is evaluated based on the values/value-ranges of the corresponding arcs and the frequencies. In this section, a distance measure for GTs is defined and shown to be a metric.

4.1 Preliminaries

Definition 1 The level of a node in a GT is equal to the difference of the length of the path (in terms of the number of hops) of the longest path from root to leaf of the GT (which is the depth of the GT) and the length of the path from the root up to the node under consideration.
Example 5 The level of each node of the GT whose depth is 3 is as indicated in Figure 7.

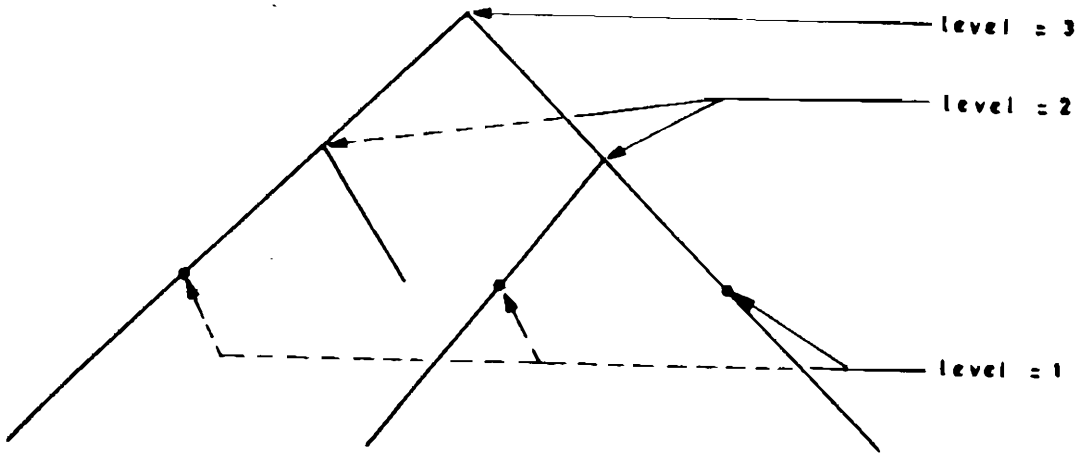


Figure 7: The levels of the nodes of a GT

Definition 2 The root node is numbered 1. The left child of a node numbered p is numbered 2p and the right child is numbered 2p+1.
Example 6 In Figure 8, all the nodes are numbered as per definition 2.
Definition 3 Two nodes are said to be corresponding if their node numbers are equivalent.
Definition 4 Two arcs are said to be corresponding if the nodes they connect are corresponding.

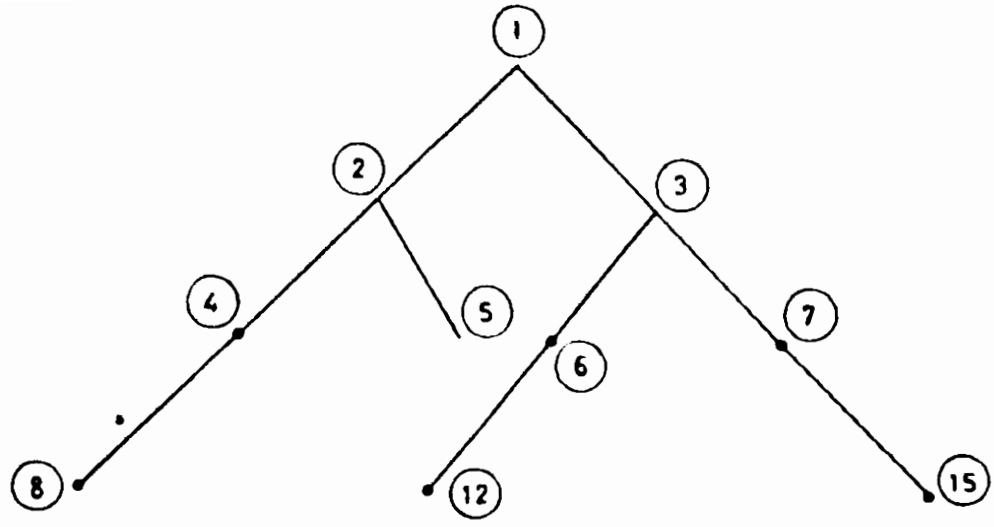


Figure 8: Node numbers of a GT

Definition 5 The source node of an arc is that node (of the two nodes which the arc connects) which has a lower node numbering. The other node is the sink node of the arc.

The process of distance evaluation of any two GTs of two classes involves a depth first tree traversal of the two GTs, comparison of the corresponding arcs and calculation of the contribution due to left and right arcs.

4.2 Distance Measure for Binary Attributes

Distance Contribution

The distance contribution of corresponding arcs q (whose source node numbered p is at level l) of two GTs T_i and T_j denoted by C^{ij}_{lpq} is proportional to the frequency difference at the arcs and the level l. Since only binary trees are involved, q takes a value of either 0 or 1.

The distance between two GTs T_i and T_j is defined as follows:

$$D(T_i, T_j) = \frac{\sum_{l=1}^{n} \sum_{p=2^{n-l}}^{2^{n-l+1}-1} \sum_{q=1}^{2} C^{ij}_{lpq}}{(n * (n+1))} \tag{1}$$

where

D(T_i,T_j) is the distance between GTs T_i and T_j, the outer summation is over all the levels, the second summation is over all source nodes p at level l, the innermost summation is over all the

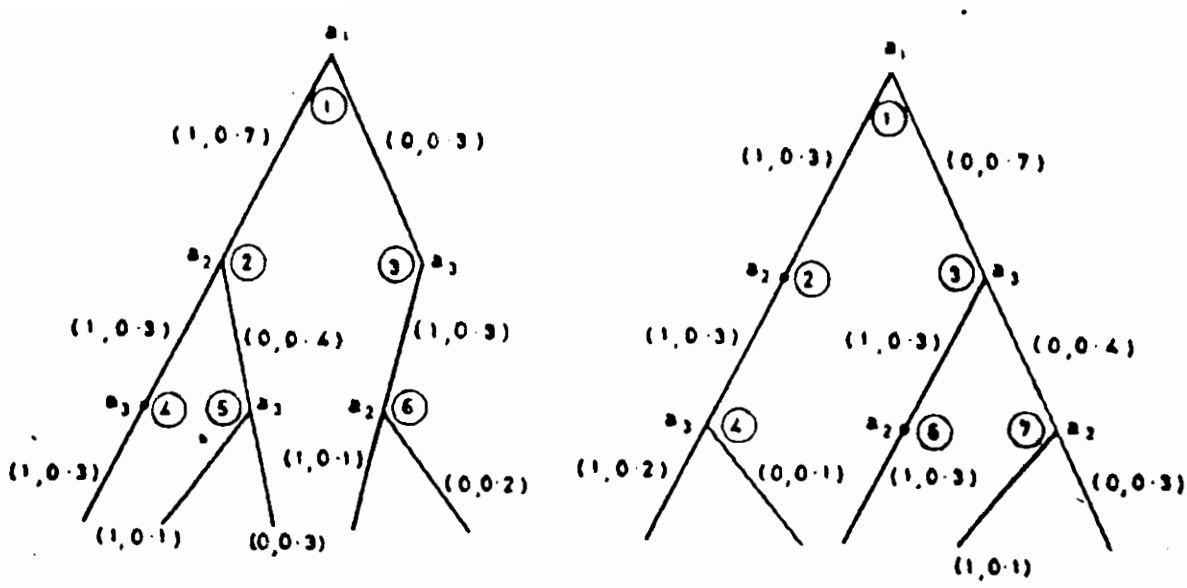


Figure 9: GTs for distance calculation

arcs q at node p and n is equal to max(depth(T_i),depth(T_j)) (which is the number of attributes if there are no irrelevant nodes at least along one path) and the distance contribution C^{ij}_{lpq} for the different cases is as given below in eqns. (2) and (3).

If the corresponding arcs exist in both the GTs T_i and T_j, then eqn. (2) is applicable, else eqn. (3) is applicable.

$$C^{ij}_{lpq} = \mid f^i_{lpq} - f^j_{lpq} \mid * \, l \tag{2}$$

$$C^{ij}_{lpq} = f^i_{lpq} * l \tag{3}$$

where

f^i_{lpq} is the frequency at arc q with source node p at level l for GT T_i, f^j_{lpq} is the frequency at arc q with source node p at level l for GT T_j and l is the level.

Note Since the distance is evaluated between two GTs T_i and T_j built with respect to the same reference tree, the attributes at the corresponding nodes, if present in both GTs T_i and T_j will be the same.

Example 7 The GTs for class T_i and T_j are as shown in Figure 9.

Here n=3.

$D(T_i, T_j) = (C^{ij}_{311} + C^{ij}_{312} + C^{ij}_{221} + C^{ij}_{222} + C^{ij}_{231} + C^{ij}_{232} + C^{ij}_{141} +$

$C^{ij}_{142} + C^{ij}_{151} + C^{ij}_{152} + C^{ij}_{161} + C^{ij}_{162} + C^{ij}_{171} + C^{ij}_{172})/(n * (n+1))$

$C^{ij}_{311} = \mid 0.7 - 0.3 \mid * 3 = 1.2$

$C^{ij}_{312} = \mid 0.3 - 0.7 \mid * 3 = 1.2$

$C^{ij}_{221} = 0; \qquad C^{ij}_{222} = 0.4 * 2 = 0.8; \qquad C^{ij}_{231} = 0$

$C^{ij}_{232} = 0.4 * 2 = 0.8; \qquad C^{ij}_{141} = \mid 0.3 - 0.2 \mid * 1 = 0.1$

$C^{ij}_{142} = 0.1 * 1 = 0.1; \qquad C^{ij}_{151} = 0.1 * 1 = 0.1$

$C^{ij}_{152} = 0.3 * 1 = 0.3; \qquad C^{ij}_{161} = \mid 0.1 - 0.3 \mid * 1 = 0.2$

$C^{ij}_{162} = 0.2 * 1 = 0.2; \qquad C^{ij}_{171} = 0.1 * 1 = 0.1$

$C^{ij}_{172} = 0.3 * 1 = 0.3$

$$D(T_i, T_j) = \frac{5.4}{3 * 4} = 0.45$$

4.3 Metric Properties

$D(T_i,T_j)$ satisfies the following metric properties [Boor 73] :

a) $0 \leq D(T_i,T_j) \leq 1$

b) $D(T_i,T_j) = 0$, iff $T_i = T_j$.

c) $D(T_i,T_j) = D(T_j,T_i)$

d) $D(T_i,T_j) + D(T_j,T_k) \geq D(T_i,T_k)$

Property 1

a) $0 \leq D(T_i,T_j) \leq 1$

Proof Since the distance measure $D(T_i,T_j)$ is the sum of non-negative terms only, $D(T_i,T_j) \geq 0$. Since the maximum contribution at level l of the GTs T_i and T_j from all the arcs with source node at that level is at most 2l, $D(T_i,T_j)$ is at most equal to $2 * (1 + 2 + 3 + ... + n)$ which is equal to $n * (n + 1)$. Therefore $D(T_i, T_j) \leq 1$. □

Note $D(T_i, T_j) = 1$ iff there are no corresponding arcs in the two GTs T_i and T_j and each path in both the GTs is of length equal to n.

Property 2

b) $D(T_i, T_j) = 0$ iff $T_i = T_j$.

Proof The proof for the if part is straightforward; we prove that,

$$D(T_i, T_j) = 0 \Longrightarrow T_i = T_j.$$

Since each of the terms of $D(T_i, T_j)$ is non-negative, it must be equal to 0. The terms can be that of one of the two cases described in Section 4.2. If it is of the case (i) type, then

$$| f^i_{lpq} - f^j_{lpq} | * l = 0$$

$$\Longrightarrow \quad f^i_{lpq} = f^j_{lpq}$$

If the term is of the case (ii) type, then

$$f^i_{lpq} * l = 0$$

$$\Longrightarrow \quad f^i_{lpq} = 0$$

The first type says that the two frequencies are equal. The second type says that the frequency at the node of GT T_i is 0, which means it does not exist in GT T_i also. □

Property 3 It follows from the structure, and the assignment, of the values of the GTs.

Property 4

d) $D(T_i, T_j) + D(T_j, T_k) \geq D(T_i, T_k)$

Proof Let NC^{ij}_{lpq} be the normalised distance contribution at source node p, level l and arc q between GTs T_i and T_j, i.e.,

$$NC^{ij}_{lpq} = \frac{(C^{ij}_{lpq})}{(n * (n+1))}$$

We demonstrate that at each arc the sum of the corresponding normalised contribution of distance between GTs T_i and T_j, and T_j and T_k is greater than the normalised contribution of the distance between T_i and T_k. We prove for each case.

1. The corresponding arc is present in the three GTs T_i, T_j and T_k. Then,

$$NC^{ij}_{lpq} = \frac{(| f^i_{lpq} - f^j_{lpq} | * l)}{(n * (n+1))} \tag{4}$$

$$NC^{jk}_{lpq} = \frac{(| f^j_{lpq} - f^k_{lpq} | * l)}{(n * (n+1))} \tag{5}$$

$$NC^{ik}_{lpq} = \frac{(| f^i_{lpq} - f^k_{lpq} | * l)}{(n * (n+1))} \tag{6}$$

It follows from eqns. (4), (5) and (6) that

$$NC^{ij}_{lpq} + NC^{jk}_{lpq} \geq NC^{ik}_{lpq}$$

2. A particular arc q exists only in two of the three GTs. The worst case, i.e., when $NC^{ij}_{lpq} + NC^{jk}_{lpq}$ is the least and NC^{ij}_{lpq} is the greatest, is when the arc q exists for T_j and T_k but not for T_i. (or when the arc q exists for T_i and T_j, but not for T_k.)

$$NC^{ij}_{lpq} = \frac{f^j_{lpq}}{(n*(n+1))} \tag{7}$$

$$NC^{jk}_{lpq} = \frac{(\mid f^j_{lpq} - f^k_{lpq} \mid)}{(n*(n+1))} \tag{8}$$

$$NC^{ik}_{lpq} = \frac{f^k_{lpq} * l}{(n*(n+1))} \tag{9}$$

It follows from eqns (7), (8) and (9) that,

$$NC^{ij}_{lpq} + NC^{jk}_{lpq} \geq NC^{ik}_{lpq}$$

Similarly, the above can be proved for the remaining two cases.

3. A particular arc q exists only in one of the three GTs. The worst case is when the arc q exists in T_i, but not in T_j and T_k.

$$NC^{ij}_{lpq} = NC^{ik}_{lpq} \quad and \quad NC^{jk}_{lpq} = 0$$

Therefore,

$$NC^{ij}_{lpq} + NC^{jk}_{lpq} \geq NC^{ik}_{lpq}$$

Similarly, the above can be proved for the remaining two cases also.

Hence,

$$D(T_i, T_j) + D(T_j, T_k) \geq D(T_i, T_k) \quad \square$$

4.4 The Distance Measure for Non-Binary Attributes

The distance $D(T_i, T_j)$ between two GTs T_i and T_j is defined by eqn. (1), where the contribution is the same as the binary valued case if one of the corresponding arcs are missing, and is as defined below if the corresponding arcs are present.

$$C^{ij}_{lpq} = (\mid f^i_{lpq} - f^j_{lpq} \mid + 2 * \left(1 - \frac{m^{ij}_{lpq}}{u^{ij}_{lpq}}\right) * (min(f^i_{lpq}, f^j_{lpq}))) * l \tag{10}$$

where

m^{ij}_{lpq} is the difference of the lower and upper bounds of the matching values of the arc q between GTs T_i and T_j with source node p at level l (for nominal valued attributes, it is equal to the cardinality of the intersection of values) and u^{ij}_{lpq} is the range of the corresponding sets with source node p at level l and arc q (for nominal valued attributes, it is equal to the cardinality of the union of the values).

Remark 1 If the (attribute, value-range) pairs are equal at all arcs in both GTs T_i and T_j, then the second term in eqn. (10) vanishes and the distance evaluation is similar to the binary valued case.

Remark 2 The distance measure defined above for the non-binary valued attributes satisfies all the metric properties (Properties 1,2,3 and 4 listed in Section 4.3).

The proofs for properties 1, 2, and 3 are extensions of the binary valued case, and only the proof of the triangle inequality is given below:

Proof of triangle inequality

$$C^{ij} = (\mid f^i - f^j \mid + 2 * \left(1 - \frac{m^{ij}}{u^{ij}}\right) * (min(f^i, f^j)) * l$$

$$C^{jk} = (\mid f^j - f^k \mid + 2 * \left(1 - \frac{m^{jk}}{u^{jk}}\right) * (min(f^j, f^k)) * l$$

$$C^{ik} = (\mid f^i - f^k \mid + 2 * \left(1 - \frac{m^{ik}}{u^{ik}}\right) * (min(f^i, f^k)) * l$$

When the corresponding arc is absent in any of the GTs, the proof is straightforward. When the corresponding arcs are present in all the GTs, we have,

$$C^{ij} + C^{jk} \geq \left(f^i + f^k - 2 * f^j * \left(\frac{m^{ij}}{u^{ij}} + \frac{m^{jk}}{u^{jk}} - 1\right)\right) \tag{11}$$

and assuming $f^k < f^i$,

$$C^{ik} = f^i + f^k - 2 * \left(\frac{m^{ik}}{u^{ik}}\right) * f^k \tag{12}$$

The proof is required only for positive values of $\frac{m^{ij}}{u^{ij}} + \frac{m^{jk}}{u^{jk}} - 1$. If this is negative, then the RHS of eqn. (11) is greater than that of eqn. (12) for all cases. When $f^k \geq f^j$, we have to

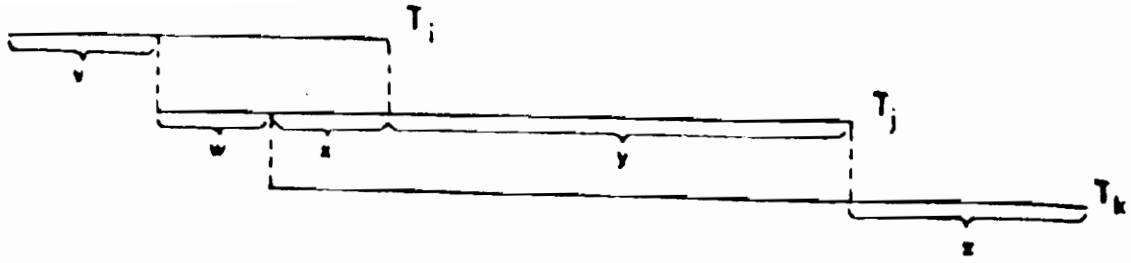


Figure 10: Attribute value overlap for GTs T_i, T_j and T_k

prove that :

$$\left(\frac{m^{ij}}{u^{ij}}\right) + \left(\frac{m^{jk}}{u^{jk}}\right) - 1 \leq \left(\frac{m^{ik}}{u^{ik}}\right)$$

or

$$\left(\frac{m^{ij}}{u^{ij}}\right) + \left(\frac{m^{jk}}{u^{jk}}\right) - \left(\frac{m^{ik}}{u^{ik}}\right) \leq 1 \tag{13}$$

Suppose the matches are as described in the Figure 10, where v, w, x, y, z are all greater than or equal to 0. Then,

$$\frac{m^{ij}}{u^{ij}} = \frac{(w+x)}{(v+w+x+y)}$$

$$\frac{m^{jk}}{u^{jk}} = \frac{(x+y)}{(w+x+y+z)}$$

$$\frac{m^{ij}}{u^{ij}} = \frac{x}{(v+w+x+y+z)}$$

Substituting in LHS of eqn. (13), we have,

$$\frac{(w+x)}{(v+w+x+y)} + \frac{(x+y)}{(w+x+y+z)} - \frac{x}{(v+w+x+y+z)}$$

This expression reduces to

$$1 - \frac{((v+w+x+y+z)(vw+vz+yz)+vxz)}{((v+w+x+y+z)(v+w+x+y)(w+x+y+z))}$$

Since v, w, x, y and z are all non-negative and none of the terms in the denominator is equal to zero (because the case being considered is when the attribute is present in all the trees), this expression is ≤ 1. Similarly, for any other combination of match for the three trees, the triangle inequality can be proved.

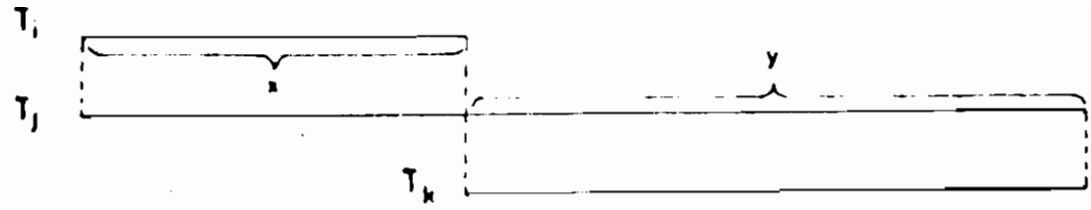


Figure 11: Attribute value overlap for GTs T_i, T_j and T_k

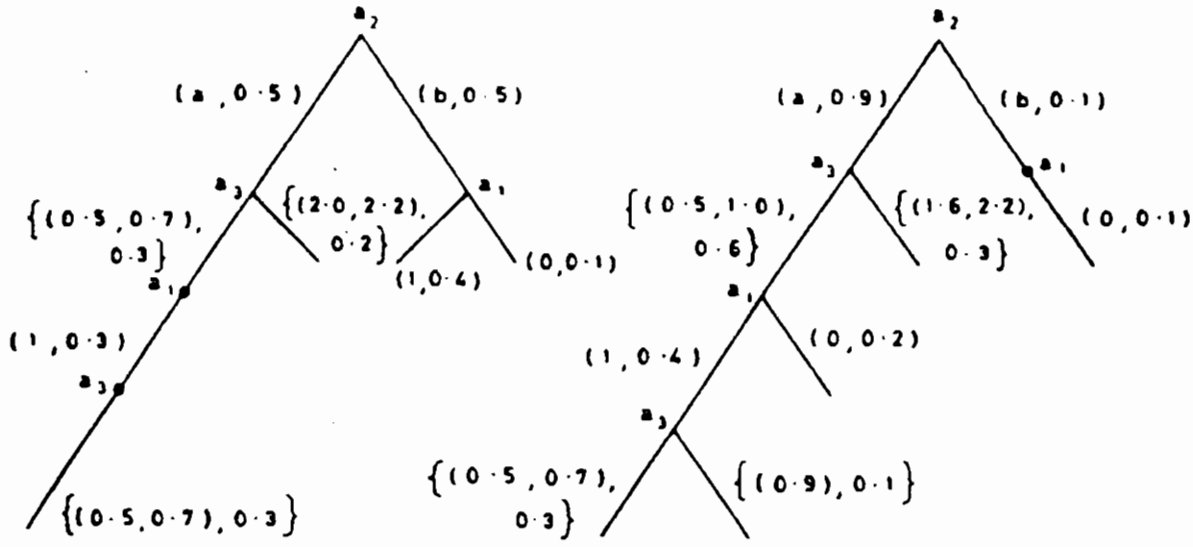


Figure 12: GTs T_i and T_j involving non-binary attributes

When $f^k < f^j$, we have to check if $\left(\frac{m^{ij}}{u^{ij}} * f^j + \frac{m^{jk}}{w^{jk}} * f^k - f^j\right)$ is less than $\left(\frac{m^{ik}}{u^{ik}} * f^k\right)$. The maximum value of the former is $\frac{y}{(x+y)} * (f^k - f^j)$ which is less than or equal to 0 and the minimum value of the latter is 0. Here x and y are as described in Figure 11. □

Example 8 Consider the two GTs T_i and T_j in Figure 12 corresponding to classes C_i and C_j, respectively, where attribute a_1 is of binary type, a_2 of nominal type and a_3 of integer type. The distance between GTs T_i and T_j is evaluated as follows:

$$\begin{aligned} D(T_i, T_j) \;=\; & ((\mid 0.5 - 0.9 \mid + \mid 0.5 - 0.1 \mid) * 4 + ((\mid 0.3 - 0.6 \mid + 2 * (1 - 0.3/0.6) * 0.3) + \\ & (\mid 0.2 - 0.3 \mid + 2 * (1 - 0.3/0.7) * 0.2) + 0.4) * 3 + (\mid 0.4 - 0.3 \mid + 0.2) * 2 + \\ & (0.1))/(4 * (4 + 1)) = 0.396 \end{aligned}$$

Note The evaluation of the distance between GTs T_i and T_j is bounded by $(N_i + N_j)$, where N_i and N_j are the number of examples in classes T_i and T_j respectively.

In the following section, this distance metric is used to find a relationship among the elementary classes.

5 Learning Relationships Among Classes

Description of individual classes in terms of GT representation was dealt in Section 3. In many applications we are interested in learning about higher level concepts that link specific classes. When such a concept is represented as a tree, we refer to it as a Concept Tree (CT). In this tree, a class is related to a set of classes. This set of related classes are combined to form a general class. All relationships in a CT are belongs-to relations.

5.1 Concept Tree

A node in a CT corresponds to a class and a node higher up corresponds to a class which is more general than all the classes corresponding to nodes below it. A class is more general than another class if after some generalisation (i.e., dropping of conditions from some sub-descriptions), the two classes are identical (or nearly so - i.e., based on the distance between the corresponding GTs) with respect to the sub-descriptions and their frequencies. The first class is referred to as general and the second as specific in this division into two classes. Any arc in the CT therefore signifies a belongs-to relation. The leaf nodes in this tree stands for the most specific classes. The CT is built in a bottom-up manner starting from the most specific classes and proceeding up to the most general class.

Each node of the CT has a corresponding class description representable as a GT. The CT is learnt from the GT of the specific classes. The training examples are given only for the most specific classes which correspond to the leaves of the CT. The inherent hierarchy is learnt progressively. This is particularly suited for diagnostic systems where the expert diagnoses from general to specific. Also, in such a setup, examples for learning are given only for the specific classes. The intermediate nodes in the CT are learnt using the GTs of two or more classes and the distance metric. The hierarchy not only aids in sequential diagnosis, but also in improving explanations.

5.2 Explanation of the Method to Build CTs

The CT is built as a gradual hierarchy. Two or more classes similar in many respects and dissimilar in a few respects, are combined to form a general class, so that the hierarchy is gradual. This implies that classes which are close are combined first and then the resultant general class is combined with other general classes. If the distance threshold is high, then it is likely that we may be combining two dissimilar or more distant classes right at the start. The aim is to combine similar or closer classes first and then relate these to dissimilar classes at higher levels of the CT. These concepts are operationalised by using two thresholds which are explained below.

The distance between two GTs being considered for combination should be less than a specified threshold 't'. This threshold (whose value lies between 0 and 1), to be specified by the user, is based on the number of attributes used to discriminate a set of related classes. If 'm' attributes are used for discrimination of two specific classes whose GTs are of depth 'n', then the threshold 't' could be fixed equal to or marginally greater than $(m*(m+1))/(n*(n+1))$. The number of attributes used for discrimination is higher for more specific classes than for less specific classes, since the depth is higher for more specific classes and 't' is a constant. This enables more classes to be combined into one at lower levels of the CT than at higher levels and therefore, the number of levels in the CT does not become unduly large.

The distance contribution for the top (n-m) levels should be less than a threshold 't1' which is approximately equal to zero. This is because we want the two classes to be ideally identical at the top (n-m) levels. This threshold 't1' is a fraction of the earlier threshold depending on what distance can be tolerated at the top levels. So, 't1' is equal to $\alpha * t$ (where $0 \leq \alpha < 1$).

The first threshold called discriminating threshold ensures that the classes are not farther than the case where the lowest 'm' levels of the GTs are completely discriminating. The second threshold called identical threshold ensures that the two classes are almost identical at the top (n-m) levels. The two thresholds together make sure that the lowest 'm' levels of the GTs are almost completely discriminating and the top (n-m) levels of the GTs are almost identical. If these two constraints are satisfied, the two classes can be combined. The resultant GT for the general class is a weighted average of the top (n-m) levels of the GTs of classes C_i and C_j. Ideally, the top (n-m) levels of the GTs of the general class and classes C_i and C_j should be identical. The lower 'm' levels are dropped using the principle of redundancy of an attribute

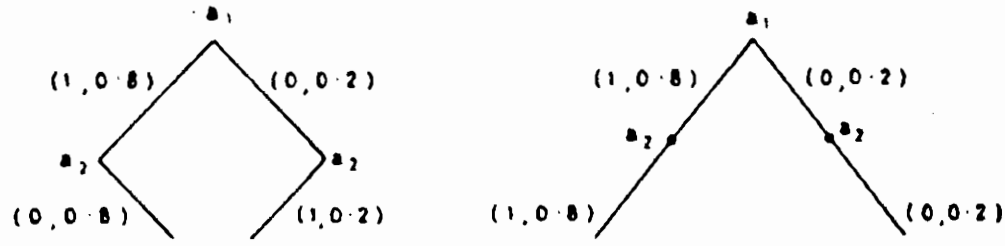


Figure 13: GTs corresponding to example sets S_i and S_j

at a node, which states that if an attribute at the lowest node of the path assumes both the values (since the GT is a binary tree) with equal frequency at a node, then it is redundant at that node. Since these thresholds can only be used as guidelines, the user should test the system for various values of these thresholds.

In the following example, we illustrate that combining two GTs after dropping of a level is equivalent to constructing the GT of the general class from union of examples of specific classes and dropping redundant nodes.

Example 9 Consider the two classes C_i and C_j for which the following two sets of examples are given.

$$S_i = \{(10),(10),(10),(10),(10),(10),(10),(10),(01),(01)\}$$

$$S_j = \{(11),(11),(11),(11),(11),(11),(11),(11),(00),(00)\}$$

The GTs of C_i and C_j are as shown in Figure 13.

If t=0.35 and t1=0.00, then T_i and T_j can be combined and the GT of the resultant class C_{ij} is as described in Figure 14.

If we build the GT of class C_{ij} using the union of example sets S_i and S_j, we get the GT in Figure 15. This reduces to the GT in Figure 14 due to the redundancy definition which states that an attribute is redundant at a node if both its descendants are leaf nodes and if the frequency of its left and right arcs are equal. In Figure 15, the attribute a_2 is redundant at both the nodes at level 1 since at both these nodes the frequencies of the left and right arcs are equal. Hence, the GT in Figure 15 reduces to that shown in Figure 14. The two procedures are hence equivalent.

Figure 14: GT corresponding to resultant class C_{ij}

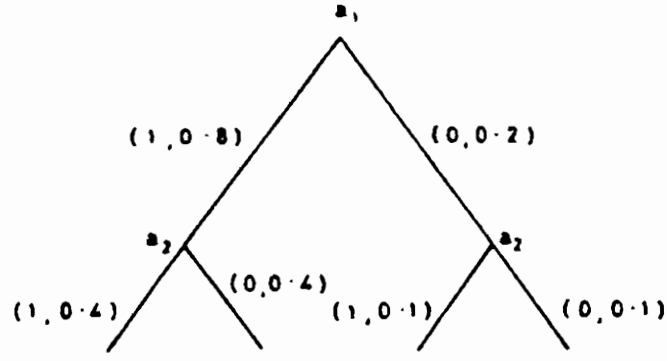


Figure 15: GT corresponding to union of example sets S_i and S_j

5.3 Combination of Classes

Preliminary Definitions

Definition 6 A class C_i is said to be more general than a class C_j, if after GT corresponding to C_j is reduced in depth to that of C_i, the two classes are identical or the distance between the GTs is within the identical threshold. Here, C_j is said to be the specific class and C_i the general class.

Definition 7 A class C_i is called maximally general class, if there is no other class C_j which is more general than C_i.

Definition 8 A class C_i is called maximally specific class, if there is no other class C_j which is more specific than C_i.

Definition 9 Two classes C_i and C_j are said to be equally specific, if after the discriminating levels are dropped from the two GTs, the two classes are identical or the distance between the GTs is within the identical threshold.

Two classes are combined at a time. The constraints for combination are checked only for

two classes at a time and classes are combined two at a time. So, if k classes are to be combined at one level, then they are combined in at most k-1 steps. The logic is that if these k classes can be combined, then at each step a specific class should be able to combine with a general class generated in the previous step. Hence, in this case we could either be combining two equally specific classes, or a specific class and a general class. So, the nodes of the given set of specific classes will progressively get connected to the node of the general class in the CT. We show in Corollary 1 that if a set of classes satisfy the constraints for combination for all sequences, then the GT of the resultant general class is invariant of the order of combination.

Algorithm for Combination of Classes

1. Form a list L of the maximally general classes, which at the start is the set of all the specific classes for which examples are given. Let the last element of this list be LAST.

2. We consider the combination of equally specific class first. Combine two GTs T_i and T_j if they are of equal depth and the following two conditions are satisfied.

   (a) D(T_i,T_j) $\leq t$, where D(T_i,T_j) is the distance between GTs T_i and T_j, and 't' is the specified discriminating threshold.

   (b) $D_{n\text{-}m}(T_i, T_j) \leq t1$, where $D_{n\text{-}m}(T_i, T_j)$ is the distance between GTs T_i and T_j after deleting 'm' bottom levels along all the paths. The integer 'm' is the largest positive satisfying $(m * (m+1))/(n * (n+1)) \leq t$. 't1' is the identical threshold.

   We next consider equally specific combination for GTs of unequal depths.

   (c) Two classes (whose GTs are of unequal depths) are equally specific and qualify for combination, if after the GT of larger depth is truncated to that of the other GT, the distance between the two GTs is greater than the identical threshold, but lower than the discriminating threshold. Notationally, $t1 \leq D(T_{i'}, T_j) \leq t$, where $T_{i'}$ is the GT built from GT T_i by truncating each path at the depth of the shorter GT T_j. Repeat step (b) with $T_i = T_{i'}$.

   Note Two GTs T_i and T_j can be of unequal depths only if they have at least one non-binary attribute, or if all the attributes are binary, then there should be at least one redundant node along all the paths for one of the GT.

   (d) Delete T_i and T_j from L and add T_{ij} (GT corresponding to combined class C_{ij}) as (LAST+1)th element. Change LAST to LAST+1.

3. Next, we consider combination of a general and a specific class.

   (a) A class the depth of whose GT is lower than that of another is a general class of the other, if after the lower discriminatory levels are dropped in the GT of the second class, the distance is within the identical threshold. In notational form, we have $D(T_{i'}, T_j) < t1$, where $T_{i'}$ is the GT built from GT T_i by truncating each path beyond the depth of the shorter GT T_j.

   Note The logic used here is that the lowest levels would have been used to discriminate the specific classes, and since after deletion of the lower levels the distance is less than 't1', it is a general-specific combination. If the distance is greater than 't1', then either it is a equally specific combination or unrelated at the current generalisation level.

   (b) Delete T_i from list L (by convention). Note that the weights in GT T_j are updated as explained in Section 5.4.

   (c) Repeat steps 2 - 3 using the new list L (which is the list of the current maximally general classes), till no more combination is possible.

Note The maximum number of distance evaluation in this algorithm is equal to (n-1) + (n-2) + ... + 1 which is of order $O(n^2)$.

Example 10 This example illustrates equally specific combination when the GTs are of equal depth. The GTs are given in Figure 16. Assume t=0.35 and t1=0.05.

$$D(T_i, T_j) = \frac{1.6}{(2 * 3)} = 0.27$$

Here m is equal to 1, since the largest value of m satisfying $(m * (m + 1))/(n * (n + 1)) < t$ is 1.

$$D_{(2\text{-}1)}(T_i, T_j) = 0 < t1$$

Therefore these two GTs T_i and T_j can be combined. The GT of the resultant general class is given in Figure 17.

Example 11 Combination of equally specific classes when the depths are unequal. Suppose t=0.35 and t1=0.05. Suppose the GTs corresponding to the two classes are as in Figure 18.

Set the height of the two GTs equal. So, we build a new GT $T_{i'}$ of height 2 as explained in the above algorithm. Then we evaluate $D(T_{i'}, T_j)$. This is equal to 0.27, which is less than

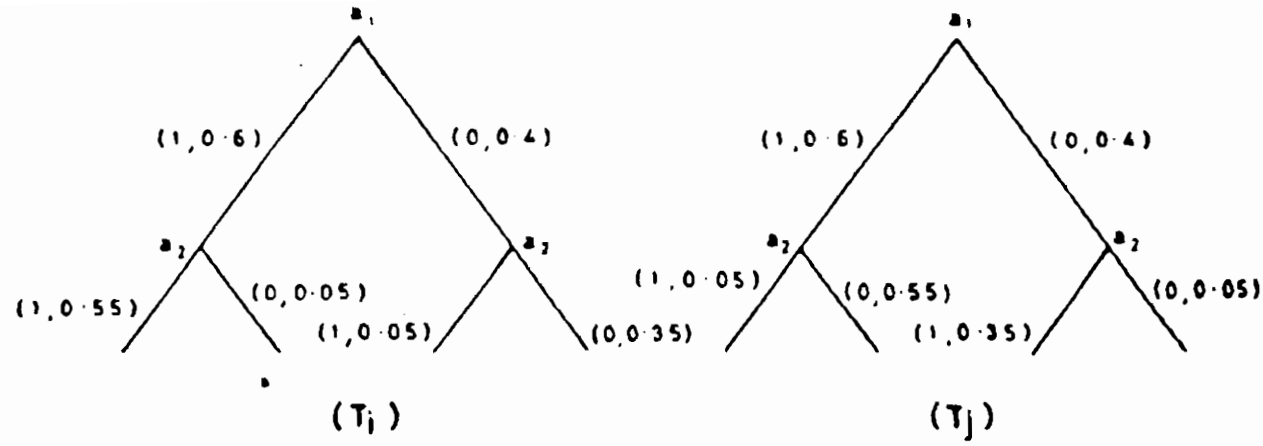


Figure 16: GTs of equal depth

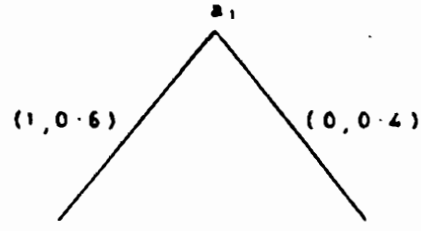


Figure 17: GT of resultant general class C_{ij}

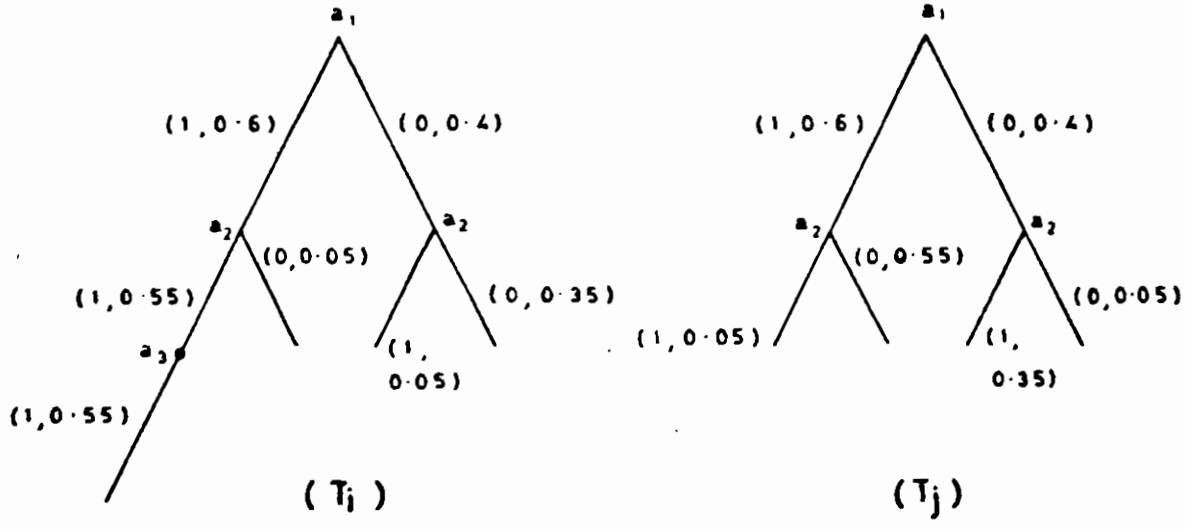


Figure 18: GTs of unequal depth

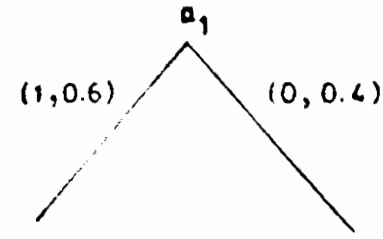


Figure 19: GT of resultant general class C_{ij}

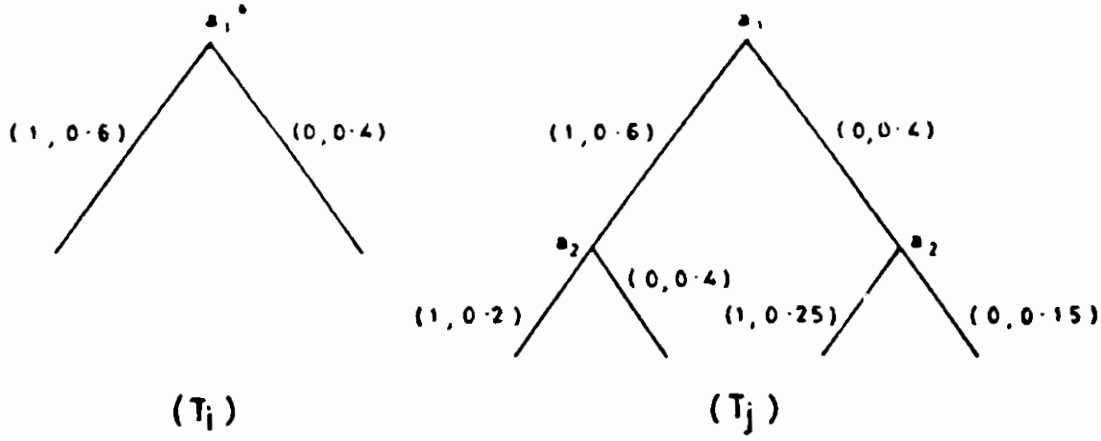


Figure 20: GTs of unequal depth

the threshold 't', but greater than 't1'. The GT of the resultant class is as given in Figure 19.

Example 12 This example highlights combination of a specific and a general class. Suppose t=0.35 and t1=0.05. Suppose the GTs corresponding to the two classes are as shown in Figure 20.

So, we construct a GT of depth 1 corresponding to T_j called $T_{j'}$ as explained in the above algorithm. The distance $D(T_i, T_{j'})$ is equal to zero. Since this is less than 't1', the two classes can be combined. The GT of the resultant class is as shown in Figure 21.

5.4 Learning of Resultant Class Description

The resultant class description is learnt from the class descriptions of the constituent classes (represented as GTs).

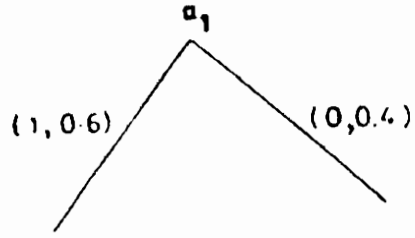


Figure 21: GT of resultant general class C_{ij}

1. Start with the root nodes of GTs $T_{i'}$ and $T_{j'}$.

2. Check if the corresponding arcs exist in both the GTs $T_{i'}$ and $T_{j'}$. If it does, then generate an arc in the new GT with the same attribute name as that of $T_{i'}$ and $T_{j'}$ at its source node and weighted frequency of the corresponding arc of $T_{i'}$ and $T_{j'}$. For example, if the frequency of the arc in GT $T_{i'}$ is f_i and in GT $T_{j'}$ it is f_j, then the weighted frequency is $(f_i * s_i + f_j * s_j)/(s_i + s_j)$ where s_i and s_j are the sample size of classes C_i and C_j respectively.

3. If an arc with source node labeled a_r exists in one GT (say T_i) and does not exist in the other, then generate a corresponding arc with source node labeled a_r in the new GT with the weighted frequency of the arc i.e. $\frac{s_i}{s_i+s_j} * f_i$.

4. Proceed in a depth first manner till all the arcs of GTs $T_{i'}$ and $T_{j'}$ are traversed.

Note For non-binary valued attributes, the same algorithm can be used for construction of GT of the general class, except that in step 2 the value range of the arcs in the new GT is the union of the range of the corresponding arcs of T_i and T_j.

Example 13 Consider the two GTs in Figure 22 for combination. Assuming the sample sizes of both the classes C_i and C_j are equal, the GT corresponding to class C_{ij} constructed using the above algorithm is as depicted in Figure 23.

Theorem 1 The GT corresponding to the general class built using the algorithm above converges in the limit to the true description, for each 't' and 't1'.

Proof Since the attribute at corresponding nodes of all the constituent classes are the same, it remains the same for the general class. The arc frequency is the weighted frequency of the corresponding arc of the constituent classes. The arc range in case of non-binary attributes is the union of the ranges of the constituent classes.

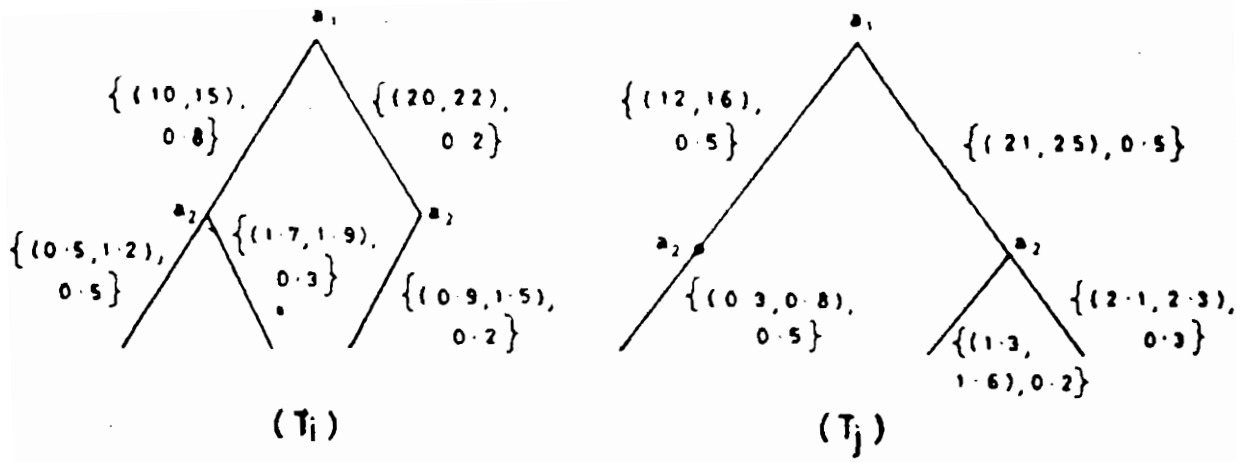


Figure 22: GTs involving integer/real valued attributes

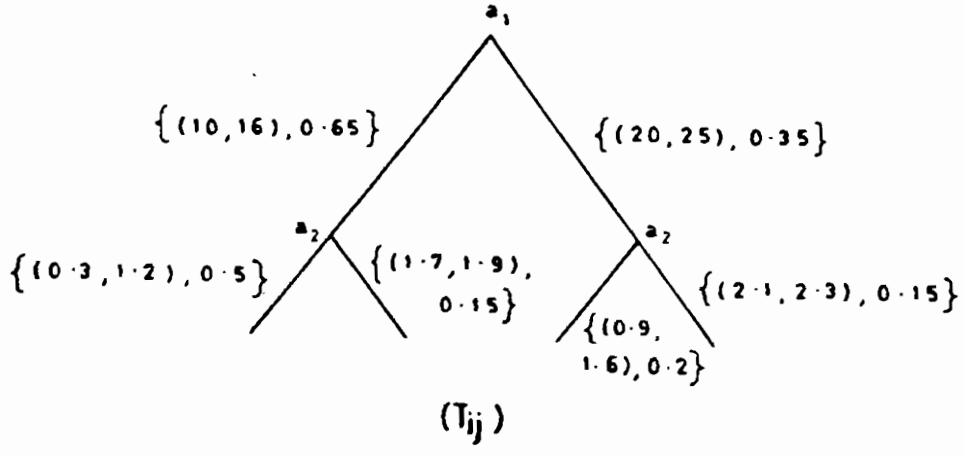


Figure 23: The resultant GT

Using Glivenko-Cantelli theorem [DeGr 87] it can be proved that the reference tree and the GTs of the constituent classes converge. This theorem states that: $| F_n(x) - F(x) | \longrightarrow 0$, 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, $\forall n > n(\epsilon)$. 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 ϵ 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 $\forall n > n(\epsilon)$. Since the attribute, frequency and ranges of all the constituent classes has converged, this will be true for the general class as well. □

Lemma 1 If the corresponding arcs with source node labeled a_r of two GTs T_i and T_j exist, then taking the union of the range of corresponding arcs and evaluating the weighted frequency, is equivalent to finding the range and evaluating the frequency for the corresponding arc with source node labeled a_r, for the GT with the examples of the union of the two classes at that node.

Proof The attribute selected at the source node in both the cases is the same, since both the algorithms use the reference tree to select the attribute at a node. The frequency of the examples at this node that falls within the range of the arc under consideration of the reference tree is equal to $(s_i * f_i + s_j * f_j)/(s_i + s_j)$. The set of examples at this node is the same as the union of examples at the corresponding node of GTs T_i and T_j. Hence, the range and frequency of the corresponding arc in the new GT is the same in both the algorithms. Thus, the arc generated in the new GT by both the methods are equivalent. □

Theorem 2 Building the GT of a general class using algorithm in Section 5.4 is equivalent to building the GT of this general class with examples of the constituent classes.

Proof There are two different cases to be considered for each node. It is to be noted that for all the cases the attributes at the corresponding source nodes are the same, since both the GTs T_i and T_j are constructed from the same reference tree.

1. The first case, is when the corresponding arcs exist in both the GTs. In this case, by Lemma 1 finding the range and evaluating frequency from the arcs of the two GTs and generating the arc from examples at that node for the new GT are equivalent and, hence the two methods are equivalent.

2. If the node exists in one of the GTs (say T_i) but not in the other, then the weighted average of the frequency of all the arcs below it is taken. This is equivalent to subtree building at that node with examples of that class only. Hence, the two algorithms are equivalent. □

This completes the proof for two classes. Extending the arguments, this can be proved for n classes.

From Theorem 2 we get,

Corollary 1 Given that a set of classes satisfy the constraints for combination for all sequences, the GT of the resultant general class would be the same irrespective of the order of combination.

Theorem 3 The hierarchy learnt converges in the limit.

Proof The proof follows from theorem 2 and the algorithm to build the hierarchy. □

In this section, learning of single and multiple relationships among elementary and general classes leading to tree structures, using the proposed distance metric was explained. The hierarchy learnt was proved to converge in the limit.

6 Practical Applications of KAHLE

Two applications were taken to demonstrate learning of a hierarchical relationship among the specified classes. In both these applications KAHLE was able to learn meaningful hierarchies. The results of these experiments are described below.

The inputs to KAHLE are the learning examples, expansion threshold, sample size of each class, attribute names, distance threshold (corresponding to the discriminating threshold and the identical threshold explained in section 5) and the test examples.

6.1 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 first run

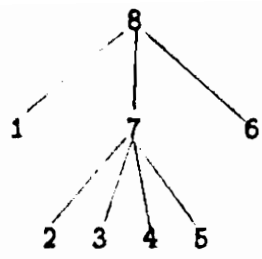


Figure 24: Hierarchy when distance threshold = 0.45

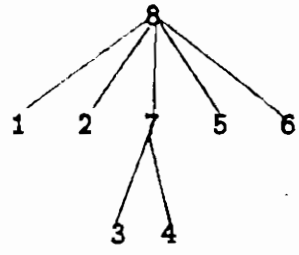


Figure 25: Hierarchy when distance threshold = 0.30

with distance threshold = 0.45 and expansion threshold = 0.10. The hierarchy learnt is as shown in Figure 24.

The program was next run with distance threshold = 0.30 and expansion threshold = 0.10. The hierarchy learnt in this case is as shown in Figure 25. The program when run with distance threshold = 0.35 and the same expansion threshold resulted in the hierarchy described in Figure 26.

It is clear from these experiments that all the hierarchies learnt are meaningful. It is never the case that distant classes (with respect to risk factor) are combined at the lowest levels of the Concept Tree.

6.2 Classification of thyroid diseases

Here the aim is learning of hierarchical disease structure for thyroid diseases. There are six diseases involved - three hypothyroid diseases and three hyperthyroid diseases. The hypothyroid

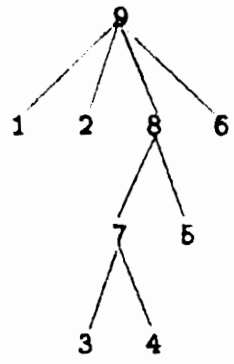


Figure 26: Hierarchy when distance threshold = 0.35

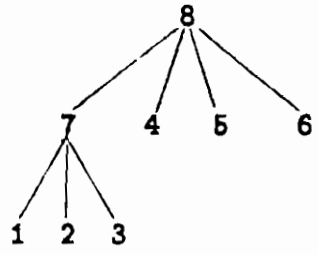


Figure 27: Hierarchy when distance threshold = 0.125

diseases are : primary hypothyroid (class 1), compensated hypothyroid (class 2) and secondary hypothyroid (class 3). The hyperthyroid diseases are : hyperthyroid (class 4), T3toxic (class 5) and goitre (class 6). The first experiment was performed with distance threshold = 0.125 and expansion threshold = 0.05. The hierarchy obtained is as described in Figure 27.

In the next run since secondary hypothyroid had only one example it was dropped. A new class called negative (a class having none of the diseases - class 7) was added to this list. The program was run with distance threshold = 0.10 and expansion threshold = 0.05. The hierarchy obtained for these thresholds is as described in Figure 28.

The two hierarchies are meaningful since intuitively distant classes are not combined at the bottom level of the Concept Tree.

In this section, the results of KAHLE applied to two problems commonly used in machine learning literature was described. The guideline for specifying distance threshold is based on

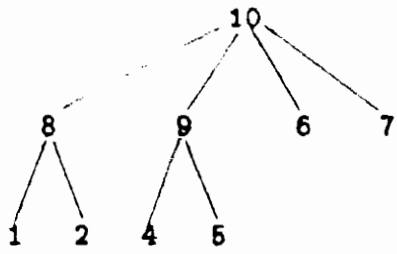


Figure 28: Hierarchy when distance threshold = 0.10

the number of discriminating attributes. The car classification and thyroid diseases experiments were run with different distance thresholds and in all cases the hierarchy learnt was meaningful, though different. These results hence, demonstrate the applicability of KAHLE to practical problems in learning general classes from specific classes.

7 Conclusion

In this paper, learning of a hierarchical relationship among a set of classes specified by examples has been described. The hierarchy learnt is shown to converge in the limit. The description of a general class is learnt from the description of the constituent specific classes. The resultant description has been shown to be equivalent to the description which would result from learning using the examples of all constituent classes. The results of the two applications demonstrate that the distance measure defined on Generation Trees helps learn meaningful hierarchies.

One of the major problems with the proposed learning system is that only one distance measure has been proposed for learning relationships among classes. Alternative distance measures and characterisation of the distance thresholds in learning relationships needs to be explored. The proposed system is non-incremental right now. A procedure to incrementally integrate new examples into the existing description is another interesting research problem.

References

[Arun 90] Arunkumar S., and Yegneshwar S., "Knowledge Acquisition from Examples using

Maximal Representation Learning", in the *7th International Machine Learning Conference*, Texas, USA, 1990, Morgan Kaufmann Publication

[Boor 73] Boorman S.A., and Olivier D.C., "Metrics on Spaces of Finite Trees", *Jnl. of Mathematical Psychology*, vol.10, 1973, pp.26-59.

[Buch 78] Buchnan B.G., and Fiegenbaum E.A., "DENDRAL and Meta-DENDRAL : their application dimension", *Artificial Intelligence*, vol.11, 1978, pp.5-24.

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

[Fu 85] Fu Li-Min and Buchanan B.G., "Inductive Knowledge Acquisition for Rule Based Expert Systems", *Technical Report No. KSL-85-12*, Knowledge Systems Laboratory, Dept. of Computer Science, Stanford University, USA, 1985.

[Lee 86] Lee W.D. and Ray S.R., "Probabilistic Rule Generator:A New Methodology of Variable-Valued Logic Synthesis", *Proc. of I.E.E.E. Conference on Multiple-valued logic*, 1986, pp.164-171.

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

[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.

[Nunz 91] Nunez M., "The Use of Background Knowledge in Decision Tree Induction", *Machine Learning*, vol.6, 1991, pp. 231-250.

[Pao 82] Pao Y.H., and Hu C.H., "A Systematic Procedure for Inductive Inference of Decision rules Applicable to Some Instances of Pattern Recognition", *6th Intl. Conf. on Pattern Recognition*, 1982, pp.1053-1055.

[Pawl 88] Pawlak Z., Wong S.K.M., and Ziarko W., "Rough Sets : Probabilistic versus Deterministic Approach", *Intl. Jnl. of Man-Machine Studies*, vol.29, 1988, pp.81-95.

[Perl 88] Pearl J., "Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference", Morgan Kaufmann Publishers Inc., San Mateo, California, USA, 1988, pp. 381-412.

[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 87b] Quinlan J.R., "Inductive Knowledge Acquisition : A Case Study", in *Applications of Expert System*, ed. J.R.Quinlan, Turing Institute Press and Addison-Wesley, 1987, pp.157-171.

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