

Decision Making in Information Systems Using Decision Diagrams

Abstract: In this paper we study the problem of data representation using decision diagrams approaches in information systems. Firstly we represent data using binary decision diagrams. Also, we used ID3-Decision tree learning algorithm to reduce information systems. Many algorithms are built and developed to refine the process of decision making in information systems. Many illustrative figures and examples are presented and many application examples are given.

Keywords: ID3 Learning Algorithm; Data Representations ; Decision making; Decision Diagram; information System.

1. Introduction

We often meet decision making problems in our daily life or working environment. Sometimes it is very difficult for us to make good decisions. In practice, we usually use our past experience as a form of performing experiments costs time and money. Fortunately, the developments of computer technologies and automatic learning techniques can make this easier and more efficient. There exist many approaches of decision making techniques, such as decision trees or decision diagrams, artificial neural networks and Bayesian learning. In this chapter, we focused on the decision diagrams especially on Binary Decision Diagram (BDD), Binary Decision Diagram Information System (BDDIS) and Decision Tree Learning Algorithm (DTLA) [40,42,43].

There exist many methods to do decision analysis. Each method has its own advantages and disadvantages [8,10,12,13,41].

The approach of BDD is used for binary data representation and it is very efficient for Boolean function representation such that relations, relational algebra, midterms, cubes, sets, sets of sets, state machines, partitions, set systems, graphs, covering, tables and matrices, but it is not understood for non-technical people.

The approach of BDDIS is more generalized than the BDD, such that it can represent any type of data but it has the same disadvantage of BDD.

The approach of DTLA is used in many areas because it has many advantages, it is the quickest, especially under the condition that the concept space is large. Furthermore, it is easy to do the data representation and to understand for non-technical people. Another advantage is that it can classify both categorical and numeric data.

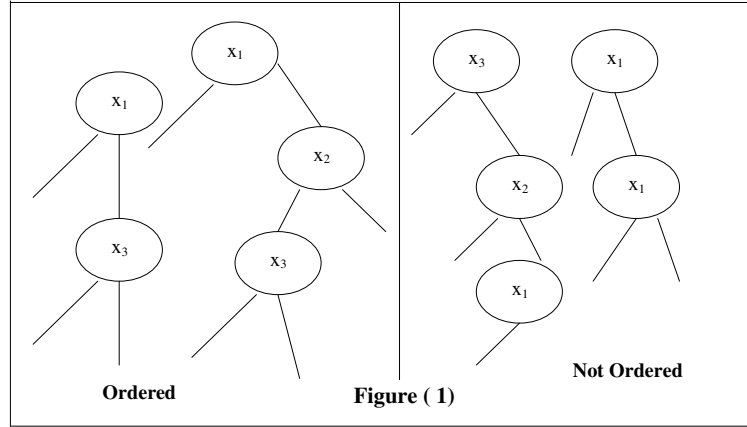
For data in an information system, the acquisition of knowledge and reasoning may involve vagueness, incompleteness, and granularity. In order to deal with the incomplete and vague information in classification, concept formulation, and data analysis, researchers have proposed many methods other than classical logic, for example, rough fuzzy sets [17,18,19], rough set theory and its generalizations [2,7,8,9-11,14,15,20], computing with words, granular computing, formal concept analysis, quotient space theory, and computational theory for linguistic dynamic systems. The advantage of the rough set method is that it does not need any additional information about the data, like probability in statistics or membership in fuzzy set theory. The main idea of applications of the rough theory in knowledge discovery comes from W. Ziarko's work [40-43]. Many researchers made contributions to this theory. Applications of the rough set and fuzzy set theories can be found in [11,14-17,40-43].

2. Decision making using Binary Decision Diagrams (BDD)

The main aim of this section is to describe how data is represented by BDD and how to obtain reducts using it. A BDD is a rooted, directed acyclic graph, with two sink nodes labeled by constants 0 and 1, and whose internal nodes have two outgoing edges. Each internal node v is labeled by a binary variable $\text{var}(v) = x_i$, $i = 1, \dots, n$. It has two outgoing edges

which are labeled by 0 and 1 , and are usually depicted as dotted and solid lines, respectively. Let the two corresponding successor nodes of v be denoted by $low(v)$ and $high(v)$, respectively.

A BDD is ordered (OBDD) if , on all paths through the graph , the variables respect a given liner order $x_1 \prec x_2 \prec \dots \prec x_n$ i.e., for each edge leading from a node labeled by x_i to a node labeled by x_j , it holds that $x_i \prec x_j$. We can say that variable must appear in ascending order along all paths (see Figure 1).



Note that, when using BDDs to represent a set of functions, some total ordering of the input variable must be selected. Since the size of a BDD is very sensitive to this choice, finding a suitable ordering is critical inability to do so will in general lead to unacceptable running times or even memory overflow. Several exact minimization algorithms have proposed, but, considering that finding an optimal order is an Np-hard problem , they are often too costly for larger problem instances. Hence heuristic approaches(selecting some ordering based on available data) or local search techniques (which aim at improving a given variable order, e.g., by moving variables up or down the graph) have been widely investigated.

A Boolean function is a function from any set A to the set $B = \{true, false\}$ [11]. An OBDD is meant to represent an n – variable Boolean function , $\{0,1\}^n \rightarrow \{0,1\}$. For a given assignment to the variables, the function value is determined by tracing a path from the root to the sink, there by following the edges indicated by the values assigned to the variables. The label of the sink node specifies the function value assigned for that input case.

Figure (2) displays an example of an OBDD for a three variables function given by the Boolean formula:

$f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge x_3$, with respect to the variable order $x_1 \prec x_2 \prec x_3$. Table 1 shows the values of this function.

x_1	x_2	x_3	$(x_1 \vee x_2) \wedge x_3$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Table 1

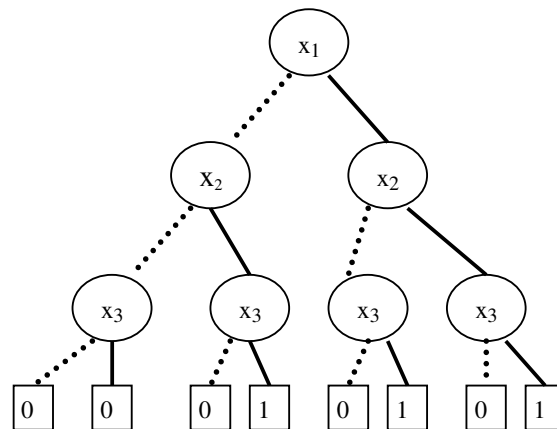
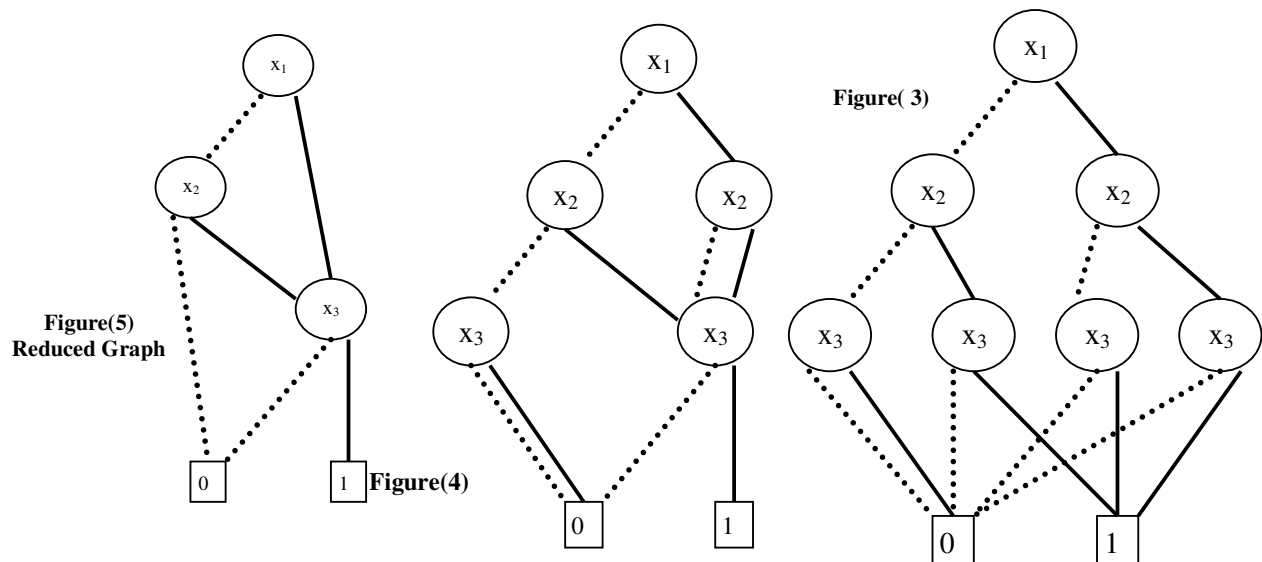


Figure (2)

Up to here, OBDD are not yet uniquely determined for each function. However, by further restricting the representation, a canonical form of BDDs. We observed that reduced OBDD (ROBDD) are a canonical representation of Boolean functions.

An OBDD is said to be reduced, if it does not contain a node v whose successor nodes are identical (i.e., $low(v) = high(v)$) and no two distinct nodes u and w exist such that the sub graphs rooted in u and w are isomorphic, i.e., for which $var(u) = var(w)$, $low(u) = low(w)$ and $high(u) = high(w)$.

The ROBDD representation of any function is uniquely determined (up to isomorphism), so that several properties (e.g. functional equivalence, constant functions, etc.) become easily testable. Conceptually, a reduced diagram can be interpreted as the result of the repeated application of three types of transformations. One reduction rule is merge equivalent leaves. This rule means use only one terminal label for 0 and one terminal label for 1 and redirect all lines from n to the respective node. Figure (3) represents this reduction rule for the same function given in Figure (2).



Second reduction rule is to merge isomorphic nodes. The second rule means if two terminal nodes x, y are on the same level and $low(x) = low(y)$ and $high(x) = high(y)$, then eliminate one of them, say x and redirect all lines into x to y . Figure(4) represents the second rule.

The last reduction rule is eliminate redundant tests. This means if $low(x) = high(x)$ for some one terminal x , then remove x and redirect all lines into x to $low(x)$. Figure(5) shows the last reduced graph.

The result of applying these rules to an OBDD until none of them can be applied any more is called a reduced OBDD.

Reduced Algorithm

- Visit OBDD bottom up and label each vertex with an identifier.
- If $id(low(v)) = id(high(v))$, then vertex v is redundant and set $id(v) = id(low(v))$.
- If $id(low(v)) = id(high(w))$, and $id(high(v)) = id(high(w))$ then set $id(v) = id(u)$.
- A different identifier is given to each vertex at level i .
- Terminated when root is reached.
- An ROBDD is identified by a subset of vertices with different identifiers.

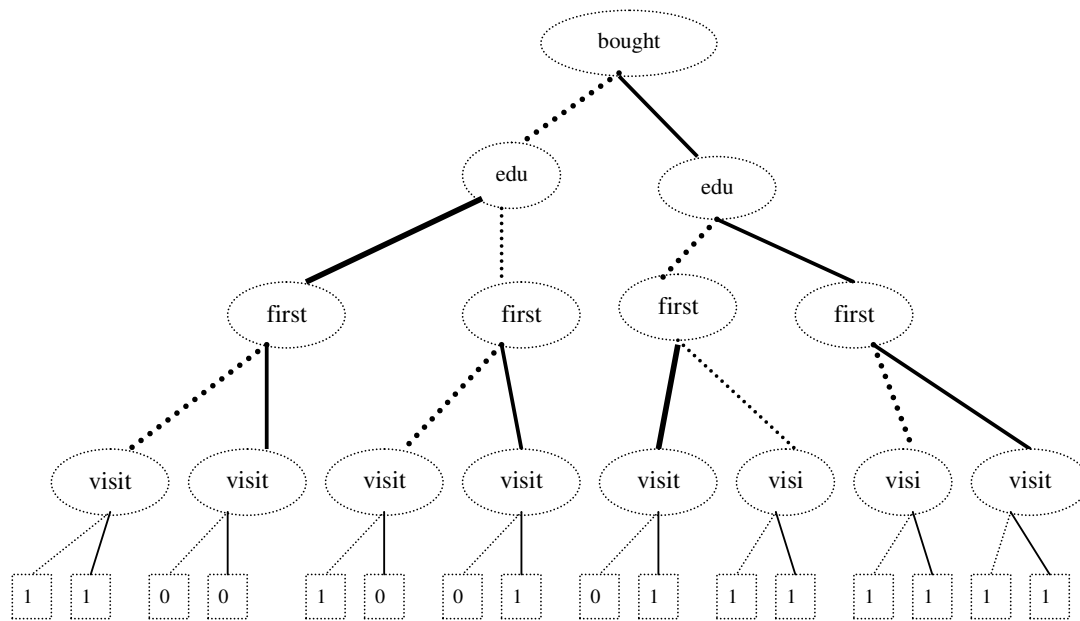
As we mentioned before the BDD is very efficient method for represents binary data ,we will introduce an example from our daily life and we will try to make a reduction for container data using ROBDD.

Example 2.1 According to Table 2 in electric commerce applications we want to make predictions about what a user will do. Consider the following made-up data used to predict whether someone will ask for more information (More-info) based on whether they accessed from an educational domain (Edu.), whether this is a first visit (First), whether they have bought goods from an affiliated company (Bought), and whether they have visited a famous online information store (Visited).

Examples	Bought	Edu.	First	Visited	More-info
e ₁	false	true	false	false	true
e ₂	true	false	true	false	false
e ₃	false	false	true	true	true
e ₄	false	false	true	false	false
e ₅	false	false	false	true	false
e ₆	true	false	false	true	true
e ₇	true	false	false	false	true
e ₈	false	true	true	true	false
e ₉	false	true	true	false	false
e ₁₀	true	true	true	false	true
e ₁₁	true	true	false	true	true
e ₁₂	false	false	false	false	true
e ₁₃	true	true	false	false	true
e ₁₄	false	true	false	true	true
e ₁₅	true	false	true	true	true
e ₁₆	true	true	true	true	true

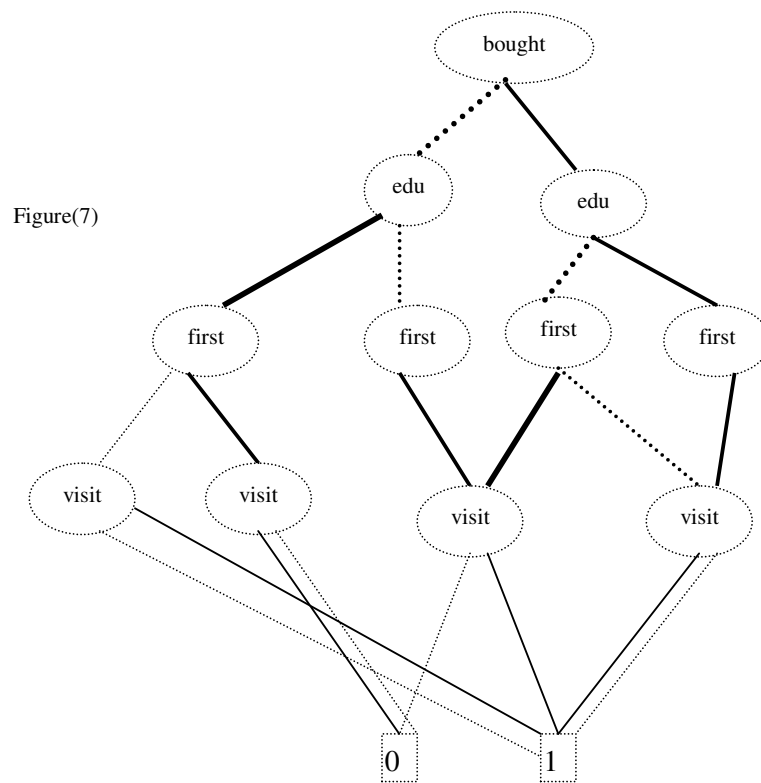
Table 2

By using BDD we consider the attribute value true replaced by 1 and false by 0, Figure(6) is the BDD for the data given in Table(2) using the order Bought $\prec$ Edu $\prec$ First $\prec$ Visited.

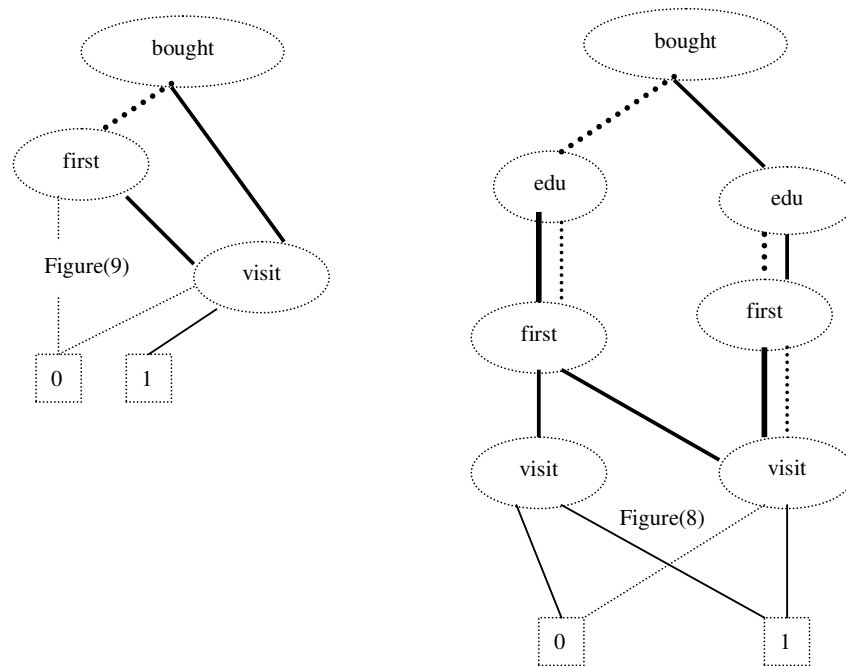


Figure(6)

According to reduction algorithm we have the following reduct figures.



Figure(7)



A BDD is an efficient data structure for represents binary data (Boolean function). We use it to obtain reducts of our daily life binary data. But for the normal user, he cannot understand what is the meant by the final graph. So we can delete the redundant attributes from the table and rewrite it. By looking at the final graph the attribute Edu. can eliminated.

3. Decision making using BDDIS

A BDDIS is a new information system representation. The basic structure of this representation is a binary decision diagram. A BDDIS is a quasi-reduced shared BDD with represents both data and indiscernibility within an information system.

A BDDIS has one or more root nodes each unique root node has a sub-tree that may share nodes with other objects and represents one or more indiscernible objects. The level root containing nodes which is defined as the object-top level ; a top level element is referenced by $T(x)$, $x \in U$.

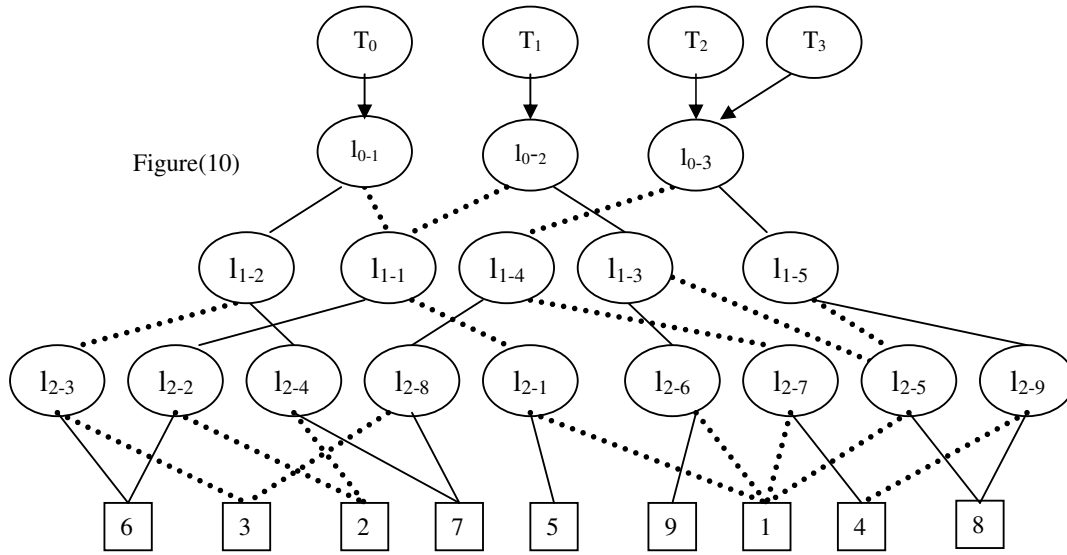
The shared nature of the BDDIS structure provides the mechanism for which indiscernibility inherent. Also every unique attribute value can be represented by a unique node on one level of the roots sub-trees defined as the attribute-top level.

The following information system can be represented by BDDIS using object-top level.

U	a	b	c	d
x ₁	15	26	36	27
x ₂	15	26	18	19
x ₃	14	37	18	48
x ₄	14	37	18	48

Table(3)

The following diagram represents the data in Table 3 using BDDIS.



Each column within the information system is represented by a path composed of a unique series of l_o and/or h_i branches arriving at attribute top level node.

In Figure(9) the first column of Table 3 is represented by the path $l_o - l_o$, the second column is represented by the path $l_o - h_i$ and the third is represented by the path $h_i - l_o$. Each node is labeled with $l_n - x$ for level number n and node number x .

For a given data set, the size of the BDDIS decreases as the level of indiscernibility increases. The worst case data that causes the BDDIS to grow as large as possible relative to the size of the information system is data containing all unique attribute values. The quasi-reduced structure is chosen to reduce the complexity associated with determining indiscernibility for the initial implementation.

Two methods can be taken to reduce the size of the BDDIS. One can reduce the BDDIS by applying the ROBDD rules described before. Alternatively, one can create the BDDIS using a coded information system in which each attribute value V_a of the original information system is coded to value $0, 1, 2, \dots, k-1$, $k = |V_a|$; here each unique coded value represents a unique attribute values. Using the same coding for different entries will reduce the size of the BDDIS, where the cost of the coded table will negligible.

A coded decision system can be efficiently derived by creating a temporary BDDIS representing only one attribute. The top nodes will represent the object indiscernibility with respect to one attribute and derive a coding for the column.

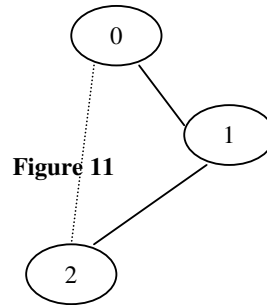
We observe that very large data with random structure can be treated as a basis for rough set data analysis with the chosen BDDIS representation.

We can reduce the data of Table 3 by using the coded method. The corresponding coded data is given in Table 4.

U	a	b	c	d
x_1	0	0	0	0
x_2	0	0	1	1
x_3	2	2	1	2
x_4	2	2	1	2

Table 4

We can represent the coded table by attribute top level using BDDIS. By make a coded table we observe that the attribute values of attribute a is the same for the attribute value for attribute b, then we can represent BDDIS without attribute b. the reduced graph of the coded data is shown in Figure 10.



4. Decision making using decision trees

A decision tree (DT) is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a decision [12,13]. Decision tree is commonly used for gaining information for the purpose of decision making. DT

starts with a root node on which it is for users to take actions. From this node , users split each node recursively according to DT in which each branch represents a possible scenario of decision and its outcome.

DT learning algorithm is a method for approximating discrete values target function, in which the learned function is presented by decision tree. DT learning algorithm is one of the most widely used and practical methods for inductive inference.

Decision Tree Learning Algorithm:

```

function DECISION –TREE-LEARNING (examples, attributes, default)
    returns a decision tree
inputs: examples, set of examples, attributes, set of attributes
    Default, default value for the goal predicate
if (examples is empty) then return default
else if (all examples have the same classification) then return the classification.
else if (attributes is empty) then return MAJORITY-VALUE(examples)
else
    best ← CHOOSE-ATTRIBUTE( attributes, examples)
    tree ← a new decision tree with root test best
    for each value  $v_i$  of best do
        example  $s_i$  ← {elements of examples with best =  $v_i$  }
        subtree ← DECISION-TREE-LEARNING(example  $s_i$ , attributes- best,
        MAJORITY-VALUE(examples) add a branch to tree with label  $v_i$  and subtree sub-tree.
    end

```

An example is described by the values of the attributes and the value of the goal predicate. We call the value of the goal predicate the classification of the example. If the goal predicate is yes for some examples, we call these examples is a positive examples. Otherwise they called a negative examples.

A set of examples $x_1, x_2, \dots, x_{12}$ are given in Table (5). The positive examples are ones where the goal will wait is Yes, namely the examples $x_1, x_3, x_4, x_6, x_8, x_{12}$ and the negative examples are $x_2, x_5, x_7, x_{10}, x_{11}$. The complete set of examples is called the training set.

Examples	Attributes										Goal
	a ₁	a ₂	a ₃	a ₄	a ₅	a ₆	a ₇	a ₈	a ₉	a ₁₀	Will wait
x ₁	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	Yes
x ₂	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	No
x ₃	No	Yes	No	No	Some	\$	No	No	Burger	0-10	Yes
x ₄	Yes	No	Yes	Yes	Full	\$	No	No	Thai	10-30	Yes
x ₅	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	No
x ₆	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	Yes
x ₇	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	No
x ₈	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	Yes
x ₉	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	No
x ₁₀	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	No
x ₁₁	No	No	No	No	None	\$	No	No	Thai	0-10	No
x ₁₂	Yes	Yes	Yes	Yes	Full	\$	No	Yes	Burger	30-60	Yes

Table 5

The examples given in the above table (Table 5) are 12 customers of a given restaurant according to the following instructions:

- a_1 is an alternate attribute and asked about if there is a suitable alternative restaurant nearby.
- a_2 asked about if the restaurant has a comfortable bar area to wait in.
- a_3 asked if the restaurant opens on Fridays and Saturdays.
- a_4 asked if the customer is hungry.
- a_5 asked about the number of customers (none, some or full).
- a_6 asked about the money which one customer will pay (\$,\$\$or \$\$\$).
- a_7 asked if it raining outside or not.
- a_8 asked if you have reservation before or not.
- a_9 asked about the type of mail of this restaurant (French, Italian, Thai or Burger).
- a_{10} time wait estimated by the host (0-10, 10-30),30-60 or >60 minutes).

DTs are fully expressive within the class of propositional languages, that is , any Boolean function can be written as a decision tree. This can be done trivially by having each row in the truth table for the function corresponds to a path in the tree. This would not necessarily be a good way to represent the function, because the truth table is exponentially large in the number of attributes. Clearly, decision trees can represent many functions with much smaller trees.

Decision trees are good for some kinds of functions , and bad for others. Consider the set of all Boolean functions on n attributes. How many different functions are in this set? This is just the number of different truth tables that we can write down, because the function is defined by its truth table. The truth table has 2^n rows, because each input case is described by n attributes.

We can consider the answer column of the table as a 2^n bit number that defines the function. No matter, what representation we use for functions, some of the functions (almost all of them in fact) are going to require at least this many bits to represent.

If it takes 2^n bits to define the function, this means that there are 2^{2^n} different functions on n attributes. This is a scary number. For example, with just six attributes, there are about 2×10^{19} different functions to choose from. We will need some ingenious algorithms to find consistent hypotheses in such a large space.

The following decision tree (Figure 12) represents one of multiple trees which represents the restaurant problem given in Table 5.

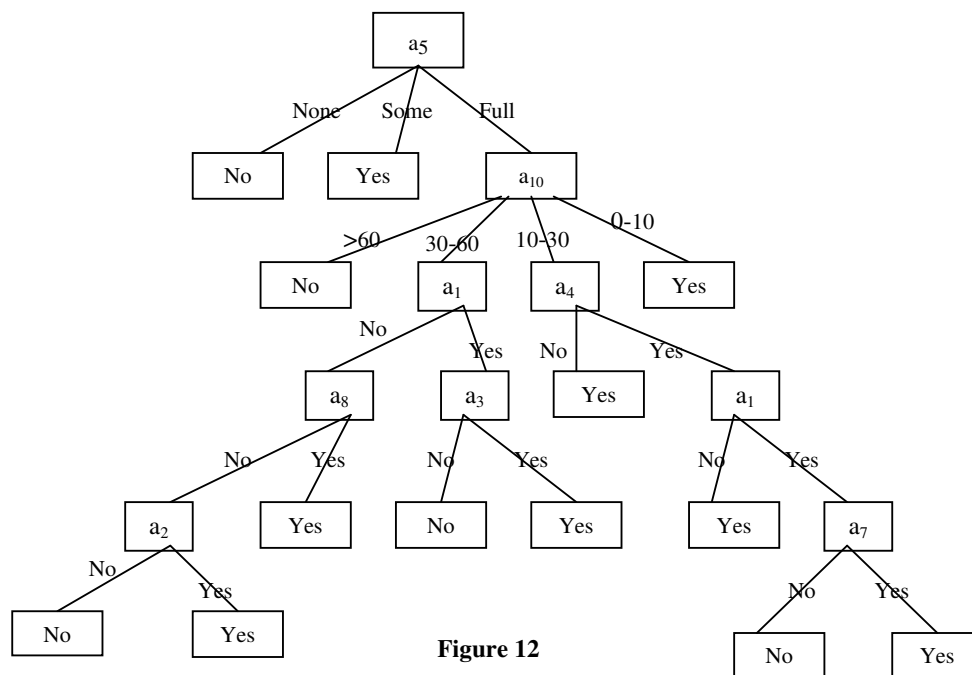


Figure 12

By using decision tree method, we can find many numbers of decision trees that represent the same problem. Our aim is to find the smallest decision tree and it is an intractable problem. But with some simple heuristic, we can do a good job of finding a smallish one. The basic idea behind the decision tree learning algorithm is to test the most important attribute first. By the "most attribute", we mean the one that makes the most difference to the classification of an example.

This way, we hope to get to the correct classification with a small number of tests, meaning that all paths in the tree will be short and the tree as whole will be small.

Figure 12 shows the algorithms gets started. We are given 12 training examples, which we classify into positive and negative sets. We then decide which attribute to use as the first test in the tree.

Figure (13-a) shows that patrons is a fairly important attribute, because if the value is none or some, then we are left with example sets for which we can answer definitively (no and yes, respectively). If the value is full, we will need additional tests. In Figure (13-b) we see that type is poor attribute, because it leaves us with four possible outcomes, each of them has the same number of positive and negative answers. We consider all possible attributes in this way, and choose the most important one as the root test. The importance of DT is measured in the next sections.

After the first attribute test splits up the examples, each outcome is a new decision tree learning problem in itself, with fewer examples and one fewer attribute. There are four cases to consider for these recursive problems are:

1. if there are some positive and negative examples, then choose the best attribute to spilt them. Figure (13-c) shows that attribute Hungry being used to spilt the remaining examples.
2. if all the remaining examples are positive(or all are negative), then we can answer by yes or no. Figure (13-c) shows examples of this in the none and some cases.
3. if there are no examples left, it means that no such example has been observed, and we return a default value calculated from the majority classification at the node's parent.
4. if there are no attributes left, but both positive and negative examples, we have a problem. It means that these examples have exactly the same description but different classification. This happens when some of data are incorrect; we say there is noise in the data. It also happens when the attributes do not give enough information to fully describe the situation, or when the domain is truly nondeterministic. One simple way out of the problem is to use a majority vote.

We continue to apply the DECISION-LEARNING algorithm until we get the tree shown in Figure (14). The tree is distinctly different from the original tree shown in Figure (12). Despite the fact that the data were actually generated from an agent using the original tree.

One might conclude that the learning algorithm looks at the examples, not at the correct function, and in fact, its hypothesis (See Figure (14)) not only agrees with all the examples, but is considerably simpler than the original tree. The learning algorithm has no reason to include tests for raining and reservation. It has also detected an interesting regularity in data (that was not even suspected).

If we were to gather more examples, we might induce a tree more similar to the original. The tree in Figure (14) is bound to make a mistake ; for instance, it has never seen a case where the wait is 0-10 minutes but the restaurant is full. For a case where hungry is false, the tree says not to wait , but the author(customer) would certainly wait. This raises an obvious question : if the algorithm induces a consistent but incorrect tree from the examples, how incorrect will the tree be? The next section answer this question and discuss how to analyze this experimentally.

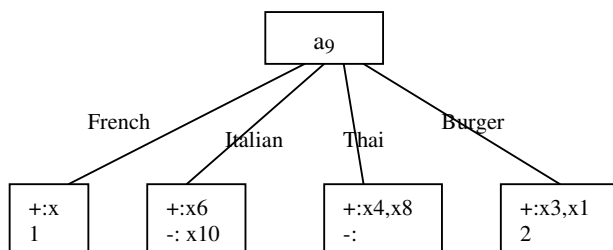


Figure (13-b)

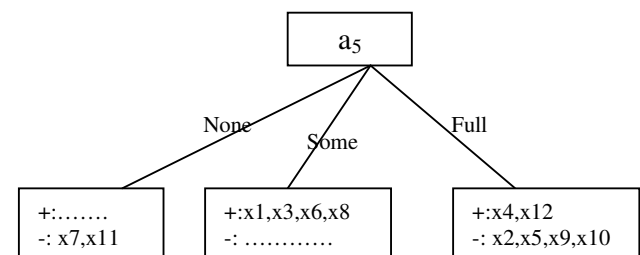


Figure (13-a)

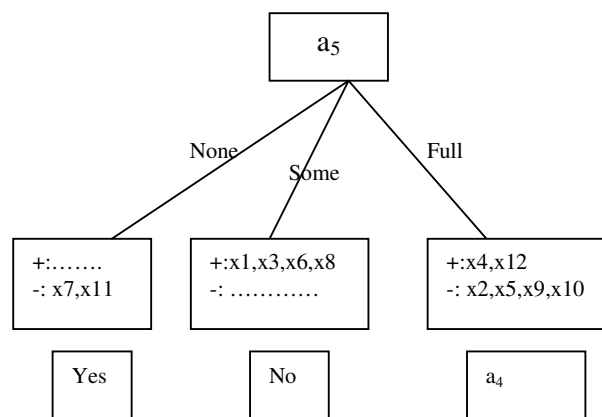


Figure (13-c)

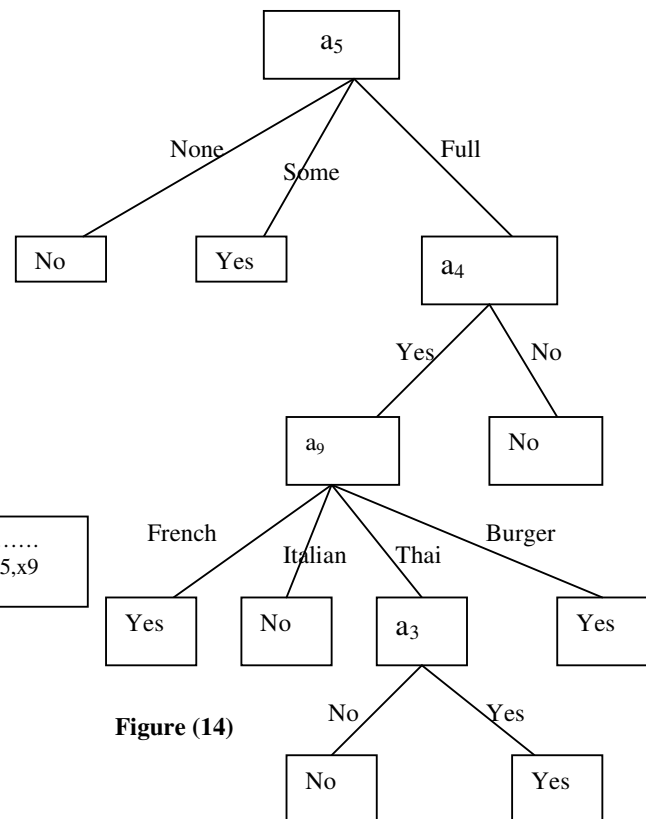


Figure (14)

5. ID3-decision tree learning algorithm

Interactive Dichotomize 3 (ID3 for short) algorithm is a simple decision tree learning developed by Ross Quinlan based on the concept learning system (CLS) algorithm. It builds the decision tree from some fixed or historic symbolic data in order to learn to classify them and predict the classification of new data. The data must have several attributes with different values, this data also has to belong to diverse predefined discrete classes (i.e., yes/no).

The basic idea of ID3 algorithm is to construct the decision tree by employing a top- down greedy search [23]. Through the given sets to test each attribute at every tree node. In order to select the attribute that is most useful for classifying a given sets. Decision tree chooses the attributes for decision making by using information gain (IG).

To find an optimal way to classify a learning set, what we need to do is to minimize the questions asked (i.e. minimizing the depth of the tree). Thus , we need some function which can measure which questions provide the most balanced splitting the IG is such a function.

The basic ID3 method selects each instance attribute classification by using statically method beginning in the top of the tree. The core question of the method is how to select the attribute of pitch point of the tree. A statistical property called information gain is defined to measure the worth of the attribute. The statistical quantity entropy is applied to define the information gain , to choose the best attribute from the candidate attributes.

The entropy measure is defined as follows:

$$H(S) = \sum_{i=1}^N -p_i^+ \log_2(p_i^+) - p_i^- \log_2(p_i^-) \quad \text{where } p_i^+ \text{ is the ratio of the positive examples } S^+ \text{ with}$$

respect to the set of all examples S i.e., $p_i^+ = \frac{|S^+|}{|S|}$. And p_i^- is the ratio of the negative examples S^- with

respect to the set of all examples S i.e., $p_i^- = \frac{|S^-|}{|S|}$, where $|S|$ is the cardinality of S .

For example, we assume that a sample set S has 14 members altogether ($|S| = 14$), including 9 positive ($|S^+| = 9$) and 5 negative examples. Then the entropy of S is :

$$H(S) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.94.$$

Below we discuss the entropy in the special case of the Boolean classification.

- If all the members of a set S belong to the identical kind (have the same decision value), then the entropy is null. That means that there is no classification uncertainty.
- If the number of the positive examples equals to the number of the negative examples, then entropy takes its maximum value. In this case, the classification uncertainty is maximum.

These results express separately that the same set has no uncertainty (the decision is clear); or it is 100% uncertain for decision making. If the number of positive examples is not equal to the number of the negative examples, then the value of entropy is measured between 0 and 1.

To carry on the attribute expansion, which is based on the data of this sample set, we must define a standard measure called information gain. An information gain of an attribute is the final information content, which is a result of the reduction of the sample set entropy after using this attribute to divide the sample set. The definition of the information gain of an attribute a relates to the sample set S is :

$$G(S, a) = H(S) - \sum_{v \in V_a} \frac{|S_v|}{|S|} \bullet H(v), \quad \text{where } S_v = \{x \in S : V_a(x) = v, v \in V_a\} \text{ and } H(v) \text{ is the}$$

entropy of the a -attribute value v .

In the following example, people decide to go to zoo or stay at home according to the weather, rain and traffic situation as shown in Table 6. Just like Table 6, the set $S = \{D_1, \dots, D_{14}\}$ contains in total 14 examples (9 positive and 5 negative examples). We can calculate the information gain of the attribute rain as follows:

$$G(S, \text{rain}) = H(S) - \frac{8}{14} H(\text{weak}) - \frac{6}{14} H(\text{strong}), \text{ where}$$

$$H(\text{weak}) = -\frac{6}{8} \log_2\left(\frac{6}{8}\right) - \frac{2}{8} \log_2\left(\frac{2}{8}\right) \text{ and } H(\text{strong}) = -\frac{3}{6} \log_2\left(\frac{3}{6}\right) - \frac{3}{6} \log_2\left(\frac{3}{6}\right).$$

Then $G(S, \text{rain}) = 0.048$.

Day	Weather	Rain	Traffic-jam	Go to zoo
D ₁	Hot	Weak	Long	No
D ₂	Hot	Strong	Long	No
D ₃	Hot	Weak	Long	Yes
D ₄	Mild	Weak	Long	Yes
D ₅	Cool	Weak	Short	Yes
D ₆	Cool	Strong	Short	No
D ₇	Cool	Strong	Short	Yes
D ₈	Mild	Weak	Long	No
D ₉	Cool	Weak	Short	Yes
D ₁₀	Mild	Weak	Short	Yes
D ₁₁	Mild	Strong	Short	Yes
D ₁₂	Mild	Strong	Long	Yes
D ₁₃	Hot	Weak	Short	Yes
D ₁₄	Mild	Strong	Long	No

Table 6

Using the same principle, we can calculate the information gain to the remaining attributes and we have: $G(S, weather) = 0.029$ and $G(S, traffic) = 0.94$.

As a result, we may choose the attribute of highest information gain to used to expand the tree. We consider the attribute traffic-jam as the root for the decision tree.

Two sub-nodes are generated. The left and the right sub-node of the root separately contain the samples with the attribute value long and short. We then delete the attribute traffic-jam of the samples in these sub-nodes and compute the entropy and the information gain to expand the tree using the attribute with highest gain value. We repeat this process until the entropy of the node equals zero. At that moment, the samples in this node belong to the same class. In the end we get the following decision tree (Figure 15).

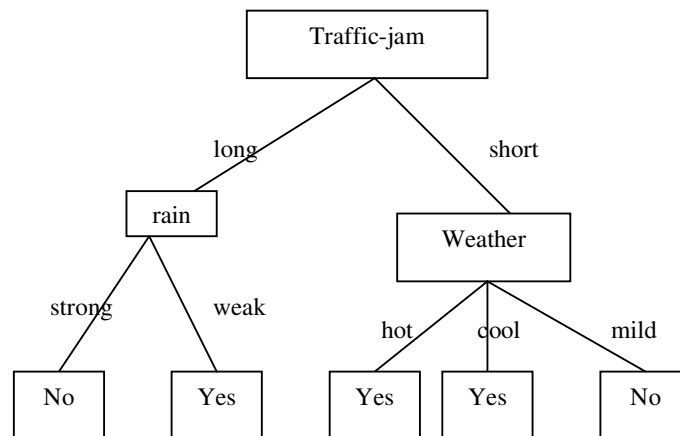


Figure 15

From the above figure as a result of using ID3 algorithm the following rules will be generated :

- If traffic-jam is long and rain is strong then people will choose to stay at home.
- If traffic-jam is long and rain is weak then people will choose to go to zoo. In logical form $(traffic, long) \wedge (rain, weak) \Rightarrow (go\ to\ zoo, No)$.
- If traffic-jam is short and weather is hot then people will choose to go to zoo. In logical form $(traffic, short) \wedge (weather, hot) \Rightarrow (go\ to\ zoo, Yes)$.
- If traffic-jam is short and weather is cool then people will choose to go to zoo. In logical form $(traffic, short) \wedge (weather, cool) \Rightarrow (go\ to\ zoo, Yes)$.
- If traffic-jam is short and weather is mild then people will choose to stay at home. In logical form $(traffic, short) \wedge (weather, mild) \Rightarrow (go\ to\ zoo, No)$.

In order to extend decision tree induction to a wider variety of problems, a number of issues must be addressed such as:

- Missing data: In many domains not all the attribute values will be known for every example. The values may not have been recorded, or they may be too expensive to obtain. First, given a complete decision tree, how should one classify an object that is missing one of the test attributes?
- Multi-valued attributes: When an attribute has a large number of possible values, the information gain measure gives an inappropriate indication of the attribute's usefulness. Consider the extreme case where every example has a different value for the attribute. For instance, if we were to use an attribute restaurant name in Example (3.1). In such a case, each subset of examples is a singleton and therefore has a unique classification, so the information gain measure would have its highest value for this attribute. However, the attribute may be irrelevant or useless.
- Multi-valued data: The multi-valued data is a generalization for a single valued one, when an attribute value has a number of values. How we can represent this problem by the decision tree and how the information gain measure this case is an open problem.
- Continuous valued attributes: Attributes such as height and weight have a large or infinite set of possible values. They are therefore not well-suited for decision tree learning algorithms.

A decision tree learning algorithm must be able to handle all of these problems. We will attempt to handle the multi-valued data problem by describing a new method to classify the attribute value which has multi-valued data into sub-classes. Then we will calculate the information gain for the new set which contains all the sub-classes, and we will consider that all sub-classes is the new training set.

We introduce the following example which studies the above cases.

Example 5.1 In this example, a company needs some employees. It decided to accept the person if it has some languages, some scientific degrees, experience and background knowledge according to Table 7.

Person	Language	Degree	Experience	Background	Decision
P ₁	{F,D}	{Bs,Ms,PhD}	Medium	Excellent	Accept
P ₂	{H,R}	{Bs}	Low	Neutral	Reject
P ₃	{F,D,S}	{Bs, Ms}	Low	Good	Reject
P ₄	{F}	{Bs, Ms}	High	Neutral	Accept
P ₅	{D}	{Bs}	Medium	Neutral	Reject
P ₆	{F,D}	{Bs, Ms}	High	Excellent	Accept
P ₇	{H,R}	{Bs}	High	Good	Accept

Table 7

From the above table , $S = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$ is the set of examples. The set { Language, Degree , Experience , Background } is the condition attributes of the company and Decision is the decision attribute. The attribute Language has the value set {F,D, H, R, S} which is some languages in symbolic $V_{language} = \{F, D, H, R, S\}$. Also $V_{degree} = \{Bs, Ms, Ph.D\}$, $V_{experience} = \{low, medium, high\}$ and $V_{background} = \{neutral, good, excellent\}$.

Now, we need to represent this problem by using DTLA. If we consider the attribute values {F,D} of the attribute Language as a separately attribute value we can apply DTLA easily but the result is not good where we take more than one attribute value in the same class.

The entropy for the set S of all persons (S contains 4 positive examples and 3 negative examples) is :

$$H(S) = -\frac{4}{7} \log_2\left(\frac{4}{7}\right) - \frac{3}{7} \log_2\left(\frac{3}{7}\right) = 0.985$$

Then the information gain of each attribute is:

$$G(S, language) = H(S) - \frac{2}{7} H(\{F, D\}) - \frac{2}{7} H(\{H, R\}) - \frac{1}{7} H(\{F, D, S\}) - \frac{1}{7} H(\{F\}) - \frac{1}{7} H(\{D\}) = 0.985 - 0.2857 = 0.69928$$

$$G(S, degree) = H(S) - \frac{1}{7} H(\{Bs, Ms, Ph.D\}) - \frac{3}{7} H(\{Bs\}) - \frac{3}{7} H(\{Bs, Ms\}) = 0.985 - \frac{3}{7} \left[-\frac{2}{3} \log_2\left(\frac{2}{3}\right) - \frac{1}{3} \log_2\left(\frac{1}{3}\right) \right] = 0.19789$$

$$G(S, experience) = H(S) - \frac{2}{7} H(medium) - \frac{2}{7} H(low) - \frac{3}{7} H(high) = 0.69928$$

$$G(S, background) = H(S) - \frac{2}{7} H(excellent) - \frac{3}{7} H(neutral) - \frac{2}{7} H(good) = 0.3.$$

According to the result , the selected attribute is language or experience because it have the same high value. If we take language as the root, we have the first classification as Figure 16 shows:

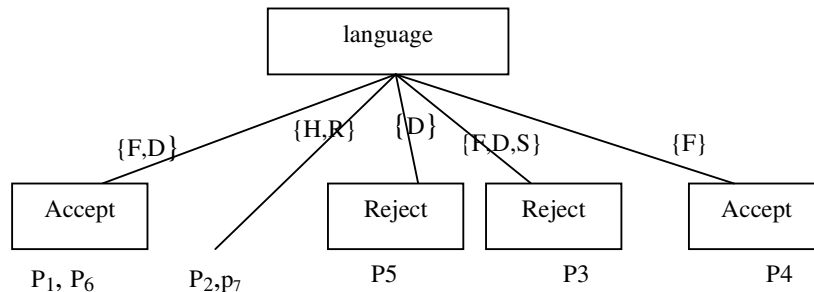


Figure 16

We choose for the second classification the attribute value set {H,R} and neglect the attribute language and take the attribute experience as a root as shown in Figure 17 below.

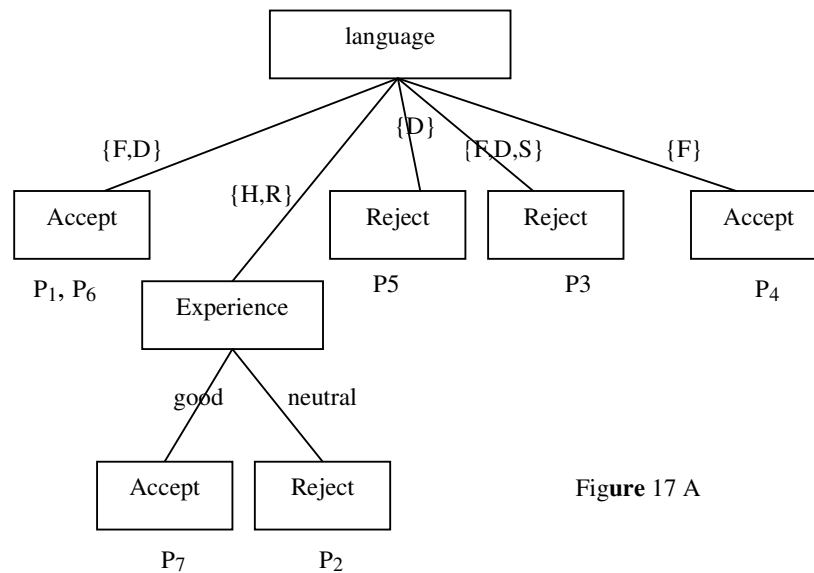


Figure 17 A

This method is not efficient at all because we cannot decide which attribute is important to take the decision.

A new way to dealing with multi-valued data is to classify the attribute value sets of the attributes that has a multi-values sets into sub-classes before constructing the decision tree.

We introduce the following algorithm which called The Multi-valued ID3 algorithm (MID3) to dealing with the above cases.

MID3 Algorithm:

```

function MID3 (examples, attributes, default)
    returns a decision tree
inputs: examples, set of examples, attributes, set of attributes
    Default, default value for the goal predicate
if (examples is empty) then return default
else if (all examples have the same classification ) then return the classification.
else if (attributes is empty) then return MAJORITY-VALUE(examples)
else
do{
if (the attribute values have multi-valued)then
{ classify the attribute values into sub-classes}
best ← CHOOSE-ATTRIBUTE( attributes, examples)
tree ← a new decision tree with root test best
for each value  $v_i$  of best do
example  $s_i$  ← {elements of examples with best =  $v_i$  }
subtree ← MID3(examples  $s_i$ , attributes-best, MAJORITY-VALUE(examples) add a branch to tree with label  $v_i$  and subtree
sub-tree} //end do
end
return tree

```

According to this algorithm Table 7 can then transformed into Table 8.

<i>Person</i>	<i>Classes</i>	<i>Language</i>	<i>Degree</i>	<i>Experience</i>	<i>Background</i>	<i>Decision</i>
P ₁	P ₁₁	F	Bs	Medium	Excellent	Accept
	P ₁₂	F	Ms	Medium	Excellent	Accept
	P ₁₃	F	PhD	Medium	Excellent	Accept
	P ₁₄	D	Bs	Medium	Excellent	Accept
	P ₁₅	D	Ms	Medium	Excellent	Accept
	P ₁₆	D	PhD	Medium	Excellent	Accept
P ₂	P ₂₁	H	Bs	Low	Neutral	Reject
	P ₂₂	R	Bs	Low	Neutral	Reject
P ₃	P ₃₁	F	Bs	Low	Good	Reject
	P ₃₂	F	Ms	Low	Good	Reject
	P ₃₃	D	Bs	Low	Good	Reject
	P ₃₄	D	Ms	Low	Good	Reject
	P ₃₅	S	Bs	Low	Good	Reject
	P ₃₆	S	Ms	Low	Good	Reject
P ₄	P ₄₁	F	Bs	High	Neutral	Accept
	P ₄₂	F	Ms	High	Neutral	Accept
P ₅	P ₅₁	D	Bs	Medium	Neutral	Reject
P ₆	P ₆₁	F	Bs	High	Excellent	Accept
	P ₆₂	F	Ms	High	Excellent	Accept
	P ₆₃	D	Bs	High	Excellent	Accept
	P ₆₄	D	Ms	High	Excellent	Accept
P ₇	P ₇₁	R	Bs	High	Good	Accept
	P ₇₂	H	Bs	High	Good	Accept

Table 8

By applying the Multi-valued ID3 (MID3) algorithm for the above table then the information gain is given as:

$$G(S, language) = H(persons) - \frac{9}{32}H(F) - \frac{8}{23}H(D) - \frac{2}{23}H(H) - \frac{2}{23}H(R) - \frac{2}{23}H(S) = 0.985 - 0.299 - 0.331977 - \frac{4}{23} = 0.1801$$

$$G(S, degree) = H(persons) - \frac{13}{23}H(Bs) - \frac{8}{23}H(Ms) - \frac{2}{23}H(PhD) = 0.985 - \frac{13}{23} \left[-\frac{7}{13} \log_2\left(\frac{7}{13}\right) - \frac{6}{13} \log_2\left(\frac{6}{13}\right) \right] = 0.09$$

$$G(S, experience) = H(persons) - \frac{7}{23}H(medium) - \frac{8}{23}H(low) - \frac{8}{23}H(high) = 0.805$$

$$G(S, background) = H(persons) - \frac{10}{23}H(excellent) - \frac{5}{23}H(neutral) - \frac{8}{23}H(good) = 0.49$$

The attribute experience is of highest value of information gain. Then we take it as a root of our tree as shown in Figure (17).

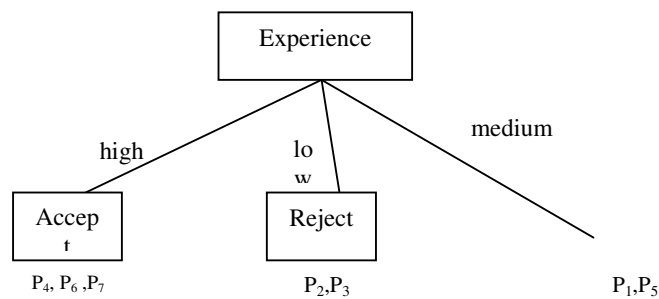


Figure 17 B

Similarly , by calculating the information gain again but with respect to the value medium of experience attribute, the second classification is illustrated in Figure 18.

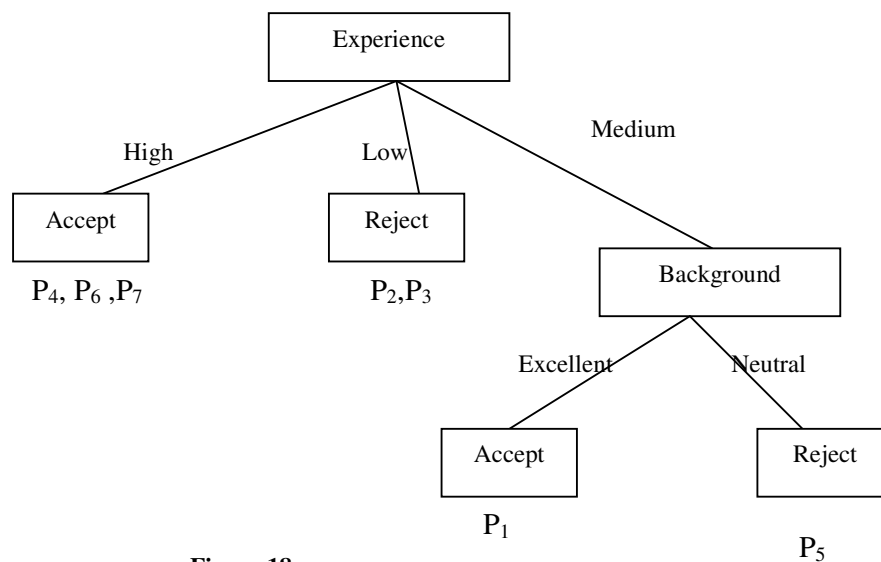


Figure 18

From the last decision tree we can deduce the following decision rules for the company example :

- The persons with high experience are accepted to work in the company.
- The persons with low experience are rejected to work in the company.
- The persons with medium experience and with excellent background are accepted to work in the company.
- The persons with medium experience and with neutral background are rejected to work in the company.

6. Conclusion

Decision trees are also a popular form of classification models. It is well known that classical trees lack the ability of modeling vagueness. By connecting fuzzy systems [38,39] and classical decision trees, we try to achieve classifiers that can model vagueness and are comprehensible. A popular and particularly efficient method for making a decision tree for classification from symbolic data is ID3 algorithm. Revised algorithms for numerical data have been proposed, some of which divide a numerical range into several intervals or fuzzy intervals. Their decision trees, however, are not easy to understand. In this work we proposed a new version of ID3 algorithm to generate a understandable fuzzy decision tree using fuzzy sets is fuzzy ID3.

References

- [1] J. Atkinson-Abutridy, C. Mellish and S. Aitken, Combining information extraction with genetic algorithms for text mining. IEEE Intelligent System, vol. 19, No. 3, pp. 22-30. 2004.
- [2] W. H. Au and K. C. C Chan, An Effective Alogrithm for Discovering Fuzzy Rules in Relation Databases. In proceedings of the 7th IEEE International Conference on fuzzy system, pp. 1314-1319. 1998.
- [3] B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom. Models and Issues Data Stream Systems. In proceedings of the 2002 ACM Symposiumon Principles of Database System (PODS 2002) (Invited Paper). Acn Press, June 2002.
- [4] R. Berghammer, B. Leoniuk and V. Milanese, Implementation of relational algebra using binary decision diagrams, In de swart, H., editor, 6th International Conference Relmics 2001 and 1st workshop of cost Action 274 TARSKI, Oisterwijk, The Netherlands, October 16-21, Volume 2561 of lecture notes in computer science, pages 241-257, Berlin – Heidelberg – New York. Springer ,2002.
- [5] C. Bregler and S.M Omoundro. Non Linear Image Interpolation Using Manifold Learning. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), Advances in Neural Information Processing System 7, pp. 973-980. 1995.
- [6] R. E. Bryant, Graph based algorithms for Boolean function mainpulation, IEEE Trans on Computers, V.C 35, No. 8, pp. 677-691, 1986.
- [7] W. Buntine, Learning Classification Trees, Statistics and Computing, 2:63-73, 1992.
- [8] M. A. Carreira, Continuos latert variable models for dimensionality reduction and sequential data reconstruction. Phd thesis, University of Sheffield, UK. 2001.
- [9] R. L. Chang P. T. Pavlidis, Fuzzy decision tree algorithms. IEEE Transactions on systems, Man, and cybernetics, vol. 7 (1), pp. 28-35, 1997.
- [10] S. Chen, S.L. Lee and C. Lee, A new method for generating fuzzy ruleess from numerical data for handling classification problemes. Applied Arificial Intelligence, vol. 15, No. 7, pp. 645-664. 2001.
- [11] I-Jen Chiang, Jane Yung-jen Hsu. Fuzzy classification trees for data analysis. Fuzzy Sets and Systems , vol.130 ,pp.87 – 99. 2002.
- [12] M. Dash, K. Choi, P. Scheuermann and H. Liu. Feature selection for clustering – A Filter Solution In Proceedings of IEEE International Conference on Data Mining (ICDM), pp. 155-122. 2002.
- [13] M. Dash and H. Liu. Feature Selection for classification. Intelligent Data Analysis, vol. 1, No 3, pp. 131-156. 1997.
- [14] M. David, G. Squire,The ID3 Decision tree Induction Algorithm", MONASH University Faculty of Information Technology. 2004.
- [15] P. Devijver and J. Kittler. Pattern Recognition: A Statistical Approach. Prentice hall. 1982.
- [16] A. Dobra, J. Gehrke, M. Garofalakis, and R. Rastogi. Processing Complex Aggregate Queries Over Data Streams. In proc. of the 2002 ACM SIG MOD Intel. Conf. On Management of Data, June 2002.
- [17] U. M .Fayyed, and B. I. Keki, on the handing of continuous-Valued Attributes in Decision Tree Generation, Machine Learning, 8 87-102. 1992
- [18] U. M. Fayyad, G. Piatetsky-Shapiro and P. Smyth, From data mining to knowledge discovery in databases. Artificial Intelligence Magazine, vol. 17, No. 3, pp. 37-54. 1996.
- [19] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. IEEE Transaction on Computers, vol. C-23, No. 9, pp. 881-890. 1974.
- [20] J. H. Friedman and Wstuetzle. Projection pursuit regression. Journal of the American Statistics Association, vol. 76, pp. 817-823. 1981.
- [21] J. Gehrke, v. Ganti, R. Ramakrishnan, and W. Loh. Boatoptimistic decision tree construction. In Proc. of the ACM conference on knowledge and Data Discovery (SIGKDD), 2000.
- [22] K. T. Harry Wong and John Mylqooulos, Two views of data semantics: Data models in artificial intelligence and database management. INFOR, 15 (3): 344-383, 1977.

- [23] I. Hayashi, An application fuzzy ID3 to wireless lan accesses point optimal location problem. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August ,2005.
- [24] C. Z. Janikow, Fuzzy Decision Trees: Issues and Methods. IEEE Transactions on Systems, Man, and Cybernetics, part B, vol. 28 (1), pp. 1-14. 1998.
- [25] I. Kononenko. Estimating attributes: Analysis and Extensions of RELIEF. Proceedings of The European Conference on Machine Learning, pp. 171-182. 1994.
- [26] B. Kosko. Fuzzy Entropy and Conditioning. Information Science vol. 40 No. 2 pp. 165-174. 1986.
- [27] M. A. Kramer, Non linear principal component analysis using autoassociative Neural networks. AIChE Journal, vol. 37, No. 2, pp. 233-243. 1991.
- [28] J.B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, vol. 29, No. 1, pp. 1-27. 1964.
- [29] A. Muir, Lvo Dünstsch and Günther Gediga. Rough Set Data Representation Using Decision Diagram. RACSAM, Rev. R. Acad. Cien. Serie A. Mat.vol. 98 (1), pp. 197-211. 2004.
- [30] Y. Peng, P.A.Flach, Soft Discretization to enhance the continuous decision tree induction. Integrating Aspects of Data Mining, Decision, Support and Metalearning, pp. 109-118, 2001.
- [31] J. R. Quinlan, Induction of Decision Trees. Macine learning, vol. 1, pp. 81-106, 1986.
- [32] J. R. Quinlan, Decision trees and decision making, IEEE transaction on system, Man and Cybernetic, 20 - 339-346,1990.
- [33] J.R. Quinlan. C4.5: Programs for Machine Learning. The Morgan Kaufmann series in machine learning. Morgan kaufmann publishers, San Mateo, CA. 1993.
- [34] J. R. Quinlan, Improved use of continuous attributes in C4.5, Journal of Artificial Intelligence Research, 4- 77-90,1996.
- [35] B. Raman and T.R. Ioerger. Instance-Based Filter for Feature Selection. Journal of Machine Learning Research, vol. 1, pp. 1-23. 2002.
- [36] M. Sahinoglu and Security Meter. A Practical Decision. Tree Model to Quantify Risk. IEEE Security and privacy, vol. 03 (3), pp. 18-24, May/June, 2005.
- [37] R. Slowinski, Intelligent decision support , handbook of applications and advances of rough set theory, Kluwer Academic Publishers, ISBN, 0-7923-1923-0, 1992.
- [38] L. A. Zadeh, Fuzzy Sets. Information and Control, vol. 8, pp. 338-353. 1965.
- [39] L.A. Zadeh, The concept of a liguistic variable and its application to approximate reasoning. Information Sciences, vol. 8, pp. 199-249, 301-357; vol 9, pp. 43-80. 1975.
- [40] W. Ziarko, The discovery, analysis and representation of data dependencies in databases, In: Piatetsky G. – Shapiro, W, Frawley J. (Eds.), Knowledge Discovery in Databases, AAAI Press / MIT press, pp. 177 – 195. 1991.
- [41] W. Ziarko, Variable Precision Rough Set Model. Journal of Computer and System Sciences, vol. 46, No. 1, pp. 39-59. 1993.
- [42] W. Ziarko, Rough sets and knowledge discovery, An overview. In: Ziarko , pp. 11 – 15. 1994.
- [43] W. Ziarko, Rough Sets, Fuzzy Sets and Knowledge Discovery (RSKD'93) Workshops in Computing, Springer–Verlag & British Computer Society, London, Berlin. (1994).