Level Construction of Decision Trees in a Partition-based Framework for Classification

Y.Y. Yao, Y. Zhao and J.T. Yao
Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
E-mail: {yyao, yanzhao, jtyao}@cs.uregina.ca

Abstract

A partition-based framework is presented for a formal study of consistent classification problems. An information table is used as knowledge representation. Solutions to, and solution space of, classification problems are formulated in terms of partitions. Algorithms for finding solutions are modeled as searching in a space of partitions under a refinement order relation. We focus on a particular type of solutions called conjunctively definable partitions. Two level construction methods for decision trees are investigated. Experimental results are reported to compare the two level construction methods.

1. Introduction

Classification is one of the main tasks in machine learning, data mining and pattern recognition [1, 3, 4]. It deals with classifying labeled objects. Knowledge for classification can be expressed in different forms, such as classification rules, discriminant functions, decision trees and decision graphs.

Classification by decision trees is a popular method. The typical algorithms for decision tree learning are the ID3 algorithm [6] and its descendent, the C4.5 algorithm [7]. Typically, ID3-like algorithms build a decision in a top-down, depth-first mode. Furthermore, the node splitting criteria are based on local optimization. When splitting a node, an attribute is chosen based on only information about this node, but not on any other nodes in the same level. Consequently, different nodes in the same level may use different attributes, and the same attribute may be used at different levels. The use of local optimal criteria makes it difficult to judge the overall quality of the partial decision tree during its construction process.

The main objective of the paper is to study a top-down, breadth-first, level-wise mode for constructing decision trees. Two types of algorithms are proposed and studied. One is based on local optimization node splitting criteria, and the other is based on global optimization criteria. The former is referred to as the level construction version of ID3 and is denoted by LID3. The latter is in fact a level-wise, reduct based methods and is denoted by kLR. The kLR algorithm combines the methods for constructing decision trees and the methods for searching for reducts [4, 5, 9].

The rest of the paper is organized in two layers. A formal framework for classification is presented in Section 2, which sets the stage for the algorithmic studies. Level construction algorithms are discussed in Section 3 and their experimental evaluations are reported in Section 4. The proposed methods offer a complementary approach to depth-first ID3. The decision trees obtained from the algorithms enables us to see the different aspects of knowledge embedded in data.

2. Consistent Classification Problems

2.1. Information tables

An information table provides a convenient way to describe a finite set of objects by a finite set of attributes. It deals with the issues of knowledge representation for classification problems [5, 11]. An information table S is the tuple:

$$S = (U, At, \mathcal{L}, \{V_a \mid a \in At\}, \{I_a \mid a \in At\}),$$

where U is a finite nonempty set of objects, At is a finite nonempty set of attributes, $\mathcal{L}$ is a language defined by using attributes in At, V_a is a nonempty set of values for a where $a \in At$, and $I_a : U \rightarrow V_a$ is an information function.

Formulas of $\mathcal{L}$ are defined by the following two rules: (i) An atomic formula ϕ of $\mathcal{L}$ is a descriptor $a = v$, where $a \in At$ and $v \in V_a$; (ii) The well-formed formulas (wff) of $\mathcal{L}$ is the smallest set containing the atomic formulas and closed under $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\equiv$.

If ϕ is a formula, the set $m_S(\phi)$ defined by

$$m_S(\phi) = \{x \in U \mid x \models \phi\},$$

is called the meaning of the formula ϕ in S. If S is understood, we simply write $m(\phi)$. The meaning of a formula ϕ is the set of all objects having the properties expressed by the formula ϕ. A connection between formulas of $\mathcal{L}$ and subsets of U is thus established.

The notion of definability of subsets in an information table is essential to data analysis. In fact, definable subsets are the basic units that can be described and discussed, upon which other notions can be developed. A subset $X \subseteq U$ is called a definable granule in an information table S if there exists at least one formula ϕ such that $m(\phi) = X$. For a subset of attributes $A \subseteq At$, X is an A-definable granule if there exists at least one formula ϕ_A using only attributes from A such that $m(\phi_A) = X$.

In many classification algorithms, one is only interested in formulas of a certain form. Suppose we restrict the connectives of language $\mathcal{L}$ to only the conjunction connective $\wedge$. A subset $X \subseteq U$ is a conjunctively definable granule in an information table S if there exists a conjunctor ϕ such that $m(\phi) = X$.

2.2. Partitions in an information table

Classification involves the division of the set U of objects into many classes. The notion of partitions provides a formal means to describe classification.

Definition 1 *A partition π of a set U is a collection of nonempty and pair-wise disjoint subsets of U whose union is U. The subsets in a partition are called blocks.*

Different partitions may be related to each other. A partition π_1 is a refinement of another partition π_2, or equivalently, π_2 is a coarsening of π_1, denoted by $\pi_1 \preceq \pi_2$, if every block of π_1 is contained in some block of π_2. The refinement relation is a partial order, namely, it is reflexive, antisymmetric, and transitive. It defines a partition lattice $\Pi(U)$.

A partition π is called a definable partition in an information table S if every block of π is a definable granule. A partition π is called a conjunctively definable partition if every equivalence class of π is a conjunctively definable granule. Consider two special families of conjunctively definable partitions below.

The first family is the uniformly conjunctively definable partitions, which has been studied extensively in databases [2]. A partition π is called a uniformly conjunctively definable partition in an information table S if, given a subset A of attributes in a certain order, the blocks are further refined in a process that the attributes are one-by-one added to the conjunctor. $\pi_\emptyset = U$ is the coarsest partition, and π_{At} is the finest partition, for any $A \subseteq At$, we have $\pi_{At} \preceq \pi_A \preceq \pi_\emptyset$.

The second family is the non-uniformly conjunctively definable partitions, which has been used extensively in machine learning [6]. A partition π is called a non-uniformly conjunctively definable partition in an information table S if the partition blocks select their own optimal attributes to further division, according to a consistent selection criteria. This strategy results in that the partition blocks at the same level may use different attributes for refinement partition. Let Π_{tree} be the set of non-uniformly conjunctively definable partitions. The partial order $\preceq$ can be carried over to Π_{tree}. Suppose $\pi_t \in \Pi_{tree}$. It can be easily verified that $\pi_{At} \preceq \pi_t \preceq \pi_\emptyset$.

2.3 Solutions to consistent classification problems

In an information table for classification problems, we have a set of attribute $At = F \cup \{\mathbf{class}\}$. The problem can be formally stated in terms of partitions.

Definition 2 *An information table is said to define a consistent classification if objects with the same description have the same class value, namely, for any two objects $x, y \in U$, $I_F(x) = I_F(y)$ implies $I_{\mathbf{class}}(x) = I_{\mathbf{class}}(y)$.*

Suppose π is an A-definable partition, that is, each block of π is an A-definable granule. We say that π is a solution to the consistent classification problem, if $\pi \preceq \pi_{\mathbf{class}}$. A solution π is called a most general solution if there does not exist another solution π' such that $\pi \prec \pi' \preceq \pi_{\mathbf{class}}$, where $\pi \prec \pi'$ stands for $\pi \neq \pi'$ and $\pi \preceq \pi'$.

Suppose π is a solution to the classification problem, namely, $\pi \preceq \pi_{\mathbf{class}}$. For a pair of equivalence classes $X \in \pi$ and $C \in \pi_{\mathbf{class}}$ with $X \subseteq C$, we can derive a classification rule $Des(X) \Longrightarrow Des(C)$, where $Des(X)$ and $Des(C)$ are the formulas that describe sets X and C, respectively.

Let Π_{sol} be the set of all solutions called solution space. The partition π_F is the minimum element of Π_{sol}. For two partitions with $\pi_1 \preceq \pi_2$, if π_2 is a solution, then π_1 is also a solution. For two solutions π_1 and π_2, $\pi_1 \wedge \pi_2$ is also a solution. The solution space Π_{sol} contains the trivial solution π_F, and is closed under meet $\wedge$. The solution space is a meet sub-lattice.

In most cases, we are interested in the most general solutions, instead of the trivial solution π_F. In many practical situations, one is satisfied with an approximate solution of the classification problem, instead of an exact solution.

Definition 3 *Let $\rho : \pi \times \pi \longrightarrow \Re^+$, where $\Re^+$ stands for non-negative reals, be a function such that $\rho(\pi_1, \pi_2)$ measures the degree to which $\pi_1 \preceq \pi_2$ is true. For a threshold*

α, *a partition* π *is said to be an approximate solution if* $\rho(\pi, \pi_{\mathbf{class}}) \geq \alpha$.

The measure ρ can be defined to capture various aspects of classification. Two such measures are discussed below, they are the *ratio of sure classification* (RSC) , and the *accuracy* of classification.

Definition 4 *For the partition* $\pi = \{X_1, X_2, \ldots, X_m\}$*, the ratio of sure classification (RSC) by* π *is given by:*

$$\rho_1(\pi, \pi_{\mathbf{class}}) = \frac{\sum_{i=1}^{m} |\{X_i \in \pi \mid \exists C_j \in \pi_{\mathbf{class}}, X_i \subseteq C_j\}|}{|U|}, \quad (1)$$

where $|\cdot|$ denotes that cardinality of a set. The ratio of sure classification represents the percentage of objects that can be classified by π without any uncertainty. The measure $\rho_1(\pi, \pi_{\mathbf{class}})$ reaches the maximum value 1 if $\pi \preceq \pi_{\mathbf{class}}$, and reaches the minimum value 0 if for all blocks $X_i \in \pi$ and $C_j \in \pi_{\mathbf{class}}$, $X_i \subseteq C_j$ does not hold. For two partitions with $\pi_1 \preceq \pi_2$, we have $\rho_1(\pi_1, \pi_{\mathbf{class}}) \geq \rho_1(\pi_2, \pi_{\mathbf{class}})$.

Definition 5 *For the partition* $\pi = \{X_1, X_2, \ldots, X_m\}$*, the accuracy of classification by a partition is defined by:*

$$\rho_2(\pi, \pi_{\mathbf{class}}) = \frac{\sum_{i=1}^{m} |X_i \cap C_{j(X_i)}|}{|U|}, \quad (2)$$

where $C_{j(X_i)} = arg\max\{|C_j \cap X_i| \mid C_j \in \pi_{\mathbf{class}}\}$. The accuracy of π is in fact the weighted average accuracies of individual rules. The measure $\rho_2(\pi, \pi_{\mathbf{class}})$ reaches the maximum value 1 if $\pi \preceq \pi_{\mathbf{class}}$, and reaches the minimum value $|C_{k_0}|/|U|$, where C_{k_0} is the class with the maximum number of objects. For two partitions with $\pi_1 \preceq \pi_2$, we have $\rho_2(\pi_1, \pi_{\mathbf{class}}) \geq \rho_2(\pi_2, \pi_{\mathbf{class}})$.

Additional measures can also be defined based on the properties of partitions. For example, one may use information-theoretic measures [12].

2.4. Classification as search

Conceptually, finding a solution to a classification problem can be modeled as a search in the space of A-definable partitions under the order relation $\preceq$. A difficulty with this straightforward search is that the space is too large to be practically applicable. One may avoid such a difficulty in several ways, for instance, only searching the space of conjunctively definable partitions. In particular, in searching the conjunctively definable partitions, two special cases deserve consideration, namely, the space of uniformly conjunctively definable partitions, and the space of non-uniformly conjunctively definable partitions.

Searching solutions in the space of uniformly conjunctively definable partitions can be achieved by rough set based classification methods [5, 9]. Suppose all the attributes of F have same priorities. The goal of solution searching is to find a subset of attributes A so that π_A is a most general solution to the classification problem. The important notions of rough set based approaches are summarized below.

Definition 6 *An attribute* $a \in A$ *is called a core attribute, if* $\pi_{F-\{a\}}$ *is not a solution, i.e.,* $\neg(\pi_{F-\{a\}} \preceq \pi_{\mathbf{class}})$.

Definition 7 *A subset* $A \subseteq F$ *is called a reduct, if* π_A *is a solution and for any subset* $B \subseteq A$, π_B *is not a solution. That is,*

(i) $\pi_A \preceq \pi_{\mathbf{class}}$*;*
(ii) *for any proper subset* $B \subset A$, $\neg(\pi_B \preceq \pi_{\mathbf{class}})$.

Each reduct provides one solution to the classification problems. There may exist more than one reduct. A core attribute must be presented at each reduct, namely, a core attribute is in every solution to the classification problem. The set of core attributes is the intersection of all reducts. The bias of searching solutions in the space of uniformly conjunctively definable partitions is to find the reduct, a set of individually necessary and jointly sufficient attributes.

The ID3-like algorithms search the space of non-uniformly conjunctively definable partitions. Typically, a classification is constructed in a depth-first manner until the leaf nodes are subsets that consist of elements of the same class with respect to **class**. By labeling the leaves by the class symbol of **class**, we obtain a decision tree for classification. The bias of searching solutions in the space of non-uniformly conjunctively definable partitions is to find the shortest tree construction.

One can combine these two searches together, i.e., construct a classification tree by using a reduct set of attributes derived from a reduct-based algorithm.

3. Level Construction of Decision Trees

Two level construction methods are discussed in this section. One is the ID3-like approach, and the other is the reduct-based approach.

3.1. The LID3 algorithm

The ID3 algorithm [6] is perhaps one of the most studied depth-first method for constructing decision trees. It starts with the entire set of objects and recursively divides the set by selecting one attribute at a time, until each node is a subset of objects belong to one class.

A level construction method based ID3 is given below.

LID3: A level construction version of ID3

1. Let $k = 0$.

2. The k-level, $k > 0$, of the classification tree is built based on the $(k-1)^{th}$ level described as follows:
if a node in $(k-1)^{th}$ level does not consist of only elements of the same class, **then**

2.1 Choose an attribute based on a certain criterion $\beta : At \longrightarrow \Re$;

2.2 Divide the node based on the selected attribute and produce the k^{th} level nodes, which are the subsets of that node;

2.3 Label the node by the attribute name, and label the branches coming out from the node by values of the attribute.

The selection criterion used by ID3 is an information-theoretic measures called conditional entropy. Let S denote the set of objects in a particular node at level $(k-1)$. The conditional entropy of **class** given an attribute a is denoted by:

$$\begin{aligned} H_S(\mathbf{class}|a) &= \sum_{v \in V_a} P_S(v) H(\mathbf{class}|v) \\ &= -\sum_{v \in V_a} P_S(v) \sum_{d \in V_{\mathbf{class}}} P_S(d|v) \log P_S(d|v) \\ &= -\sum_{d \in V_{\mathbf{class}}} \sum_{v \in V_a} P_S(d,v) \log P_S(d|v), \quad (3) \end{aligned}$$

where the subscript S indicates that all quantities are defined with respect to the set S. An attribute with the minimum entropy value is chosen to split a node.

For the information Table 1, we obtain a decision tree shown in Figure 1, and the analysis of RSC and accuracy is summarized in Table 2.

	A	B	C	D	**class**
1	a_1	b_1	c_1	d_2	-
2	a_1	b_1	c_2	d_2	-
3	a_1	b_2	c_1	d_1	+
4	a_1	b_2	c_2	d_1	+
5	a_2	b_1	c_1	d_2	-
6	a_2	b_1	c_2	d_1	-
7	a_2	b_2	c_1	d_2	-
8	a_2	b_2	c_2	d_1	+
9	a_3	b_1	c_1	d_2	+
10	a_3	b_1	c_2	d_1	-
11	a_3	b_2	c_1	d_1	+
12	a_3	b_2	c_2	d_1	+

Table 1. An information table

	Rules	RSC	Accuracy
k = 1 B	$b_1 \Rightarrow 5/6 -$ $b_2 \Rightarrow 5/6 +$	0.00	0.83
k = 2 ABD	$b_1 \wedge a_1 \Rightarrow 2/2 -$ $b_1 \wedge a_2 \Rightarrow 2/2 +$ $b_1 \wedge a_3 \Rightarrow 1/2 +$ $b_2 \wedge d_1 \Rightarrow 5/5 +$ $b_2 \wedge d_2 \Rightarrow 1/1 -$	0.83	0.92
k = 3 $ABCD$	$b_1 \wedge a_3 \wedge c_1 \Rightarrow 1/1 +$ $b_1 \wedge a_3 \wedge c_2 \Rightarrow 1/1 -$	1.00	1.00

Table 2. Rules generated by ID3

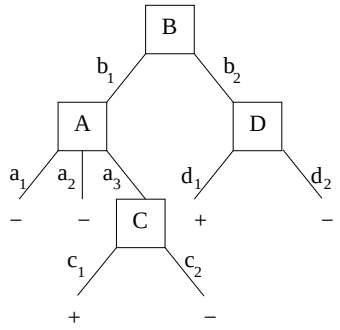

Figure 1. An ID3 decision tree

3.2. The kLR algorithm

Recall that a reduct is a set of individually necessary and jointly sufficient attributes that correctly classify the objects. An algorithm for finding a reduct can be easily extended into a level construction method for decision trees called kLR.

kLR: A reduct-based level construction method

1. Let $k = 0$.

2. The k-level, $k > 0$, of the classification tree is built based on the $(k-1)^{th}$ level described as follows:
if there is a node in $(k-1)^{th}$ level that does not consist of only elements of the same class **then**

2.1 Choose an attribute based on a certain criterion $\gamma : At \longrightarrow \Re$;

2.2 Divide all the inconsistent nodes based on the selected attribute and produce the k^{th} level nodes, which are subsets of the inconsistent nodes;

2.3 Label the inconsistent nodes by the attribute name, and label the branches coming out from the inconsistent nodes by the values of the attribute.

Note that when choosing an attribute, one needs to consider all the inconsistent nodes. In contrast, LID3 only con-

	Rules	RSC	Accuracy
$k=1$ B	$b_1 \Rightarrow 5/6-$ $b_2 \Rightarrow 5/6+$	0.00	0.83
$k=2$ BD	$b_1 \wedge d_1 \Rightarrow 2/2-$ $b_1 \wedge d_1 \Rightarrow 3/4-$ $b_2 \wedge d_1 \Rightarrow 5/5+$ $b_2 \wedge d_2 \Rightarrow 1/1-$	0.67	0.92
$k=3$ ABD	$b_1 \wedge d_2 \wedge a_1 \Rightarrow 2/2-$ $b_1 \wedge d_2 \wedge a_2 \Rightarrow 1/1-$ $b_1 \wedge d_2 \wedge a_3 \Rightarrow 1/1+$	1.00	1.00

Table 3. Rules generated by kLR

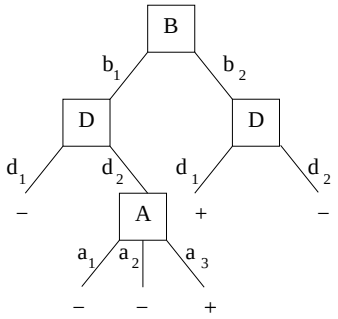

Figure 2. A kLR decision tree

siders one inconsistent node at a time.

Conditional entropy can also be used as the selection criterion γ. In this case, the subset of examples considered at each level is the union of all inconsistent nodes. Let $A_{(k-1)}$ be the set of attributes used from level 0 to level $(k-1)$. The next attribute a for level k can be selected based on the following conditional entropy:

$$H(\mathbf{class}|A_{(k-1)} \cup \{a\}). \quad (4)$$

The use of $A_{(k-1)}$ ensures that all inconsistent nodes at level $k-1$ are considered in the selection of level k attribute [9].

The decision trees generated by the kLR algorithm are level-constructed trees. The idea is similar to oblivious decision trees, in which all nodes at the same level test the same attributes according to a given order. While the order is not given, we can use the function $\gamma : At \longrightarrow \Re$ to decide an order. The criterion γ can be one of the information measures, for example, conditional entropy (as shown above) or mutual information, which indicate how much information the attributes contribute to the decision attribute **class**; or the statistical measures, for example, the χ^2 test or binomial distribution, which indicates the dependency level between the test attribute and the decision attribute **class**. We can get such an order by testing all the attributes. However, we need to update the order level by level. There are two reasons for level-wise updating. First, in respect that some nodes of a classification tree are intended to halt the partition when the solutions or the approximate solutions are found. The search space is possibly changed for different levels. Second, for each test that partitions the search space into uneven-sized blocks, the value of function γ is the sum of the function value of γ for each block multiplies the probability distribution of the block.

Consider the earlier example, based on the conditional entropy, the decision tree built by kLR algorithm is shown in Figure 2. The analysis of RSC and accuracy is given by Table 3.

Comparing these two decision trees in Figures 1 and 2, we notice that kLR decision tree can construct a tree possessing the same RSC and accuracy as the LID3 decision tree with fewer attributes involved. In this example, the set of attributes $\{A, B, D\}$ is a reduct, since $\pi_{\{A,B,D\}} \preceq \pi_{\mathbf{class}}$, and for any proper subset $X \subset \{A, B, D\}$, $\neg(\pi_X \preceq \pi_{\mathbf{class}})$.

4. Experimental evaluations

In order to evaluate level construction methods, we choose some well-known datasets from UCI machine learning repository [10]. SGI's MLC++ utilities 2.0 [8] is used to discretize the datasets into categorical attribute sets. Set the accuracy threshold $\alpha = 100\%$.

Dataset 1: Credit - 690 training objects, 14 attributes and 2 classes. Using all the attributes, neither LID3 nor kLR can 100% consistently classify all the training instances. kLR discovers that one attribute contributes little to the classification, namely, we can have the same RSC and accuracy values with only 13 attributes. kLR generates more rules than its counterpart. However comparing the trees where the same number of attributes are used, kLR obtains higher RSC and accuracy.

Dataset 2: Vote - 435 training objects, 16 attributes and 2 classes. It achieves 100% of RSC and accuracy for classification. In the case of LID3, the total tree length is 8 levels, but 15 attributes were required for 100% of RSC and accuracy. In the case of kLR, we can reach the same level of RSC and accuracy by a 9-level-tree with only 9 related attributes.

Dataset 3: Cleve - 303 training objects, 13 attributes and 2 classes. It cannot be 100% consistently classified either by all its 13 attributes. In this dataset, the kLR tree is short than the LID3 tree. The kLR tree discovered consists of 11 attributes. This number is less than that is used for constructing a full LID3 tree. This observation is shown in all the other experiments.

The experimental results of the above three datasets are

Dataset 1		
Goal	LID3	kLR
$Accu. \geq 85.00\%$	$k = 1$ (1 attr)	$k = 1$ (1 attr)
$Accu. \geq 90.00\%$	$k = 4$ (14 attrs)	$k = 5$ (5 attrs)
$Accu. \geq 95.00\%$	$k = 6$ (14 attrs)	$k = 8$ (8 attrs)
$Accu. = 98.55\%$	$k = 11$ (14 attrs)	$k = 13$ (13 attrs)
$RSC \geq 85.00\%$	$k = 6$ (14 attrs)	$k = 8$ (8 attrs)
$RSC \geq 90.00\%$	$k = 7$ (14 attrs)	$k = 9$ (9 attrs)
$RSC \geq 95.00\%$	$k = 10$ (14 attrs)	$k = 12$ (12 attrs)
$RSC = 95.80\%$	$k = 11$ (14 attrs)	$k = 13$ (13 attrs)
Dataset 2		
Goal	LID3	kLR
$Accu. \geq 95.00\%$	$k = 1$ (1 attr)	$k = 1$ (1 attr)
$Accu. = 100.00\%$	$k = 8$ (15 attrs)	$k = 9$ (9 attrs)
$RSC \geq 95.00\%$	$k = 5$ (11 attrs)	$k = 6$ (6 attrs)
$RSC = 100.00\%$	$k = 8$ (15 attrs)	$k = 9$ (9 attrs)
Dataset 3		
Goal	LID3	kLR
$Accu. \geq 85.00\%$	$k = 3$ (6 attrs)	$k = 3$ (3 attrs)
$Accu. \geq 90.00\%$	$k = 5$ (11 attrs)	$k = 6$ (6 attrs)
$Accu. \geq 95.00\%$	$k = 6$ (12 attrs)	$k = 8$ (8 attrs)
$Accu. = 98.35\%$	$k = 13$ (12 attrs)	$k = 11$ (11 attrs)
$RSC \geq 80.00\%$	$k = 6$ (12 attrs)	$k = 7$ (7 attrs)
$RSC \geq 85.00\%$	$k = 7$ (12 attrs)	$k = 8$ (8 attrs)
$RSC \geq 90.00\%$	$k = 8$ (12 attrs)	$k = 9$ (9 attrs)
$RSC = 94.72\%$	$k = 13$ (12 attrs)	$k = 11$ (11 attrs)

Table 4. Experimental results of the datasets used in this paper

reported in Table 4.

¿From the results of experiments, we can have the following observations. The difference of local and global selection causes different tree structures. Normally, LID3 may obtain a shorter tree. On the other hand, if we restrict the height of decision trees, LID3 may use more attributes than kLR. With respect to the RSC measure, LID3 tree is normally better than kLR tree at the same level. With respect to the accuracy measure, LID3 tree is not substantially better than kLR tree at the same level. With respect to different levels of two trees with the same number of attributes, kLR obtains much better accuracy and RSC than LID3. The main advantage of kLR method is that it uses fewer number of attributes to achieve the same level of accuracy.

5. Conclusion

The contribution of this paper is twofold, the development of a formal model and algorithms for level construction methods for building decision trees. The formal framework is based on partitions in an information table. Within the framework, we are able to define precisely and concisely many fundamental notions. The concepts of solution and solution space are discussed. The structures of several search spaces are studied. Two level construction methods are suggested: a breath-first version of ID3 called LID3 , which searches the space of non-uniformly conjunctively definable partitions; and a reduct-based method called kLR, which searches solutions in the space of uniformly conjunctively definable partitions. Experimental results are reported to compare these two methods. They show that one needs to pay more attention to the less studied level construction methods.

References

[1] Duda, R.O. and Hart, P.E. *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[2] Lee, T.T. An information-theoretic analysis of relational databases - part I: data dependencies and information metric, *IEEE Transactions on Software Engineering*, **SE-13**, 1049-1061, 1987.

[3] Michalski, J.S., Carbonell, J.G., and Mirchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann, Palo Alto, CA, 463-482, 1983.

[4] Mitchell, T.M., *Machine Learning*, McGraw-Hill, 1997.

[5] Pawlak, Z., *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht, 1991.

[6] Quinlan, J.R., Induction of decision trees, *Machine Learning*, **1**, 81-106, 1986.

[7] Quinlan, J.R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc., 1993.

[8] SGI's MLC++ utilities 2.0: the discretize utility. http://www.sgi.com/tech/mlc

[9] Wang, G.Y., Yu, H. and Yang, D.C., Decision table reduction based on conditional information entropy, *Chinese Journal of Computers*, **25(7)**, 2002. 3.

[10] UCI Machine Learning Repository. http://www1.ics.uci.edu/∼mlearn/MLRepository.html

[11] Yao, J.T. and Yao, Y.Y., Induction of classification rules by granular computing, *Proceedings of International Conference on Rough Sets and Current Trends in Computing*, 331-338, 2002.

[12] Yao, Y.Y., Information-theoretic measures for knowledge discovery and data mining, *Entropy Measures, Maximum Entropy and Emerging Applications*, Karmeshu (Ed.), Springer, Berlin, 115-136, 2003.