International Journal of Hybrid Intelligent Systems 2 (2005) 149–160
IOS Press

Fast split selection method and its application in decision tree construction from large databases

Hung Son Nguyen[a,*] and Sinh Hoa Nguyen[b]
[a]*Institute of Mathematics, Warsaw University, Banacha 2, Warsaw 02-097, Poland*
[b]*Polish-Japanese Institute of Information Technology, Koszykowa 86, 02-008, Warszawa, Poland*

Abstract. We present an efficient method for decision tree construction from large data sets, which are assumed to be stored in database servers, and be accessible by SQL queries. The proposed method minimizes the number of simple queries necessary to search for the best splits (cut points) by employing "divide and conquer" search strategy. To make it possible, we develop some novel evaluation measures which are defined on intervals of attribute domains. Proposed measures are necessary to estimate the quality of the best cut in a given interval. We propose some applications of the presented approach in discretization and construction of "soft decision tree", which is a novel classifier model.

Keywords: Data mining, decision tree, large databases, discernibility measure

1. Introduction

The problem of searching for optimal partitions of real value attributes (defined by cut points) has been studied by many authors (see e.g., [3,9,15]). The main goal is to discover those *splitting points* (cuts) that can be used to construct decision trees or decision rules of high quality with respect to some predefined quality measures (e.g., quality defined by classification accuracy of new unseen objects or quality defined by the decision tree height, its support and confidence). In general, these optimization problems are hard from a computational point of view. For example, it has been shown in [9] that the problem of searching for a minimal and consistent set of cuts is NP-hard. Therefore, many heuristics have been proposed to develop efficient approximate solutions for those problems. Usually, existing heuristics are based on creating some measures to estimate the quality of cut points.

One can mention some problems that occur in existing decision tree induction algorithms. First problem is related to the efficiency of the algorithm that searches for an optimal partition of real value attributes – assuming that the training data set is too large and must be stored in the relational data base. In many practical applications, there is a need of decision tree construction at the terminal of a client-server network system, and access to the database (located on the server) is enabled by SQL queries. The critical factor for time complexity of algorithms solving the discussed problem is the number of simple SQL queries necessary to find out the best partition. One can assume that the complexity can be measured by the number of queries related to the intervals of attribute values, e.g.,

*Corresponding author. E-mail: son@mimuw.edu.pl.

```
SELECT COUNT FROM aTable
  WHERE (anAttribute BETWEEN aValue1 AND aValue2)
  AND (some additional conditions)
```

The sequential approach to searching for optimal partition (with respect to a given quality measure) requires as least $O(N)$ queries, where N is the number of pre-assumed partitions of the searching space. In cases where N is a large number, even the linear complexity of the algorithm is not acceptable.

The second problem is related to the use of crisp cuts as a condition for object discerning. This can lead to misclassification of new objects which are very close to the cut points, and this fact can result in low quality of new object classifications.

The first problem is a big challenge for data mining researchers. The existing methods are based either on sampling technique [1] (i.e., repeating the decision tree induction algorithm for small, randomly chosen subset of data, until meeting the decision tree of sufficient quality [5]) or parallelization technique [2,16] using computer network architecture. In previous papers [10–12] we have presented some approximation measures that make the construction of semi-optimal partition very efficient.

In this paper we propose a novel approach based on *soft cuts* which make possible to overcome the second problem. Decision trees using soft cuts as test functions are called *soft decision trees*. The new approach leads to new efficient strategies in searching for the best cuts (both soft and crisp cuts) using all data. We show some properties of considered optimization measures that allows to reduce the size of searching space. Moreover, we prove that using only $O(\log N)$ simple queries, one can construct soft partitions that are very close to the optimal.

2. Preliminaries

Let us fix some notations related to the data model and decision tree induction methods.

The objective of many learning problems is searching for approximation of a concept (also called target concept, decision function or classes) defined on a given *universe* of objects X. Usually, objects from X are described by a finite set of *attributes* $A = \{a_1, \ldots, a_k\}$, called predictive features or exogenous variables. Every attribute a_i can be treated as function defined for all objects $a_i : X \to V_{a_i}$, where V_{a_i} is called *the domain* of attribute a_i. Hence, every object $x \in X$ is characterized by an information vector

$$inf_A(x) = [a_1(x), \ldots, a_k(x)]$$

There are some types of attributes (depending on their domains). An attribute $a_i \in A$ is called

1. *numerous* or *continuous* if its domain V_{a_i} is a set of real numbers in the sense that they can be arranged in a linear order. Age and salary of bank customers are examples of numerous attributes;
2. *symbolic* if V_{a_i} consists of incomparable (or partially ordered) elements; job and hobbies of bank customers are examples of symbolic attributes;
3. *boolean* or *binary* if $V_{a_i} = \{0, 1\}$;

In this paper, we concentrate on the concept approximation problem which is a common issue in many research areas like data mining, machine learning, statistical discriminant analysis or pattern recognition. The concept approximation problem can be formulated as a problem of searching for the best approximation of the unknown target function $t : X \to V_t$. We also limit our consideration to the case when the target concept t has a finite domain, i.e. $|V_t| < \infty$.

	SepalLength	SepalWidth	PetalLength	PetalWidth	Class
1	5.0	2.0	3.5	1.0	Iris-versicol
2	6.0	2.2	5.0	1.5	Iris-virginica
3	6.0	2.2	4.0	1.0	Iris-versicol
4	6.2	2.2	4.5	1.5	Iris-versicol
5	4.5	2.3	1.3	0.3	Iris-setosa
6	5.0	2.3	3.3	1.0	Iris-versicol
7	5.5	2.3	4.0	1.3	Iris-versicol
8	6.3	2.3	4.4	1.3	Iris-versicol
9	4.9	2.4	3.3	1.0	Iris-versicol
10	5.5	2.4	3.8	1.1	Iris-versicol
11	5.5	2.4	3.7	1.0	Iris-versicol
12	5.7	2.5	5.0	2.0	Iris-virginica
13	5.5	2.5	4.0	1.3	Iris-versicol
14	5.1	2.5	3.0	1.1	Iris-versicol
15	4.9	2.5	4.5	1.7	Iris-virginica
16	5.6	2.5	3.9	1.1	Iris-versicol
17	6.3	2.5	4.9	1.5	Iris-versicol
18	6.3	2.5	5.0	1.9	Iris-virginica
19	6.7	2.5	5.8	1.8	Iris-virginica
20	5.7	2.6	3.5	1.0	Iris-versicol

Fig. 1. An example of decision table: a part of the "Iris" classification problem.

In the *inductive learning approach* to concept approximation problem, we assume that the target function $t : X \to V_t$ is given on a finite sample $U = \{x_1, \ldots, x_n\} \subset X$ of *training objects*. The set:

$$\mathbb{S} = \{\langle inf_A(x_1), t(x_1)\rangle, \langle inf_A(x_2), t(x_2)\rangle, \ldots, \langle inf_A(x_n), t(x_n)\rangle\}$$

is called *the training set*. The problem is to construct a classifying algorithm C, called a *classifier*, that is a good approximation of the target concept t. The algorithm that induces a classifier from training data is called *the learning algorithm* or *the learner*. Having the information vector $inf_A(x)$ of any object $x \in X$, the classifier C must return a value $C(x) \in V_t$ which is called the predicted decision.

Any pair (a, c), where a is a numerous attribute and $c \in V_a$ is a real value, is called *a cut*. We say that *"a pair of objects x, y is discerned by a cut (a, c))"* if either $a(x) < c \leqslant a(y)$ or $a(y) < c \leqslant a(x)$.

2.1. Data model

Formally, the training set can be described by a triple $\mathbb{S} = (U, A \cup \{dec\})$, called *decision table*, where U is a non-empty, finite set of training objects, A is a nonempty, finite set of attributes, and $dec \notin A$ is a special attribute, called *decision*. The decision encodes the target concept in the mentioned above concept approximation problem. Any decision table can be interpreted as a rectangle data matrix, where objects are related to rows, columns are related to attributes and one special column is related to decision.

Without loss of generality one can assume that $V_{dec} = \{1, \ldots, d\}$. For any $1 \leqslant m \leqslant d$, the set $CLASS_m = \{x \in U : dec(x) = m\}$ is called the m^{th} *decision class of* $\mathbb{S}$.

We assign to every set of objects $X \subset U$ a d-tuple of integers $f(X) = \langle x_1, .., x_d\rangle$ where $x_k = card(X \cap DEC_k)$ for $k \in \{1, \ldots, d\}$. The vector $f(X)$ is called *the class distribution* of X. If the set of objects X is defined by $X = \{u \in U : p \leqslant a(u) < q\}$ for some $p, q \in \mathbb{R}$ then the class distribution of X is called *the class distribution of U in $[p; q)$*.

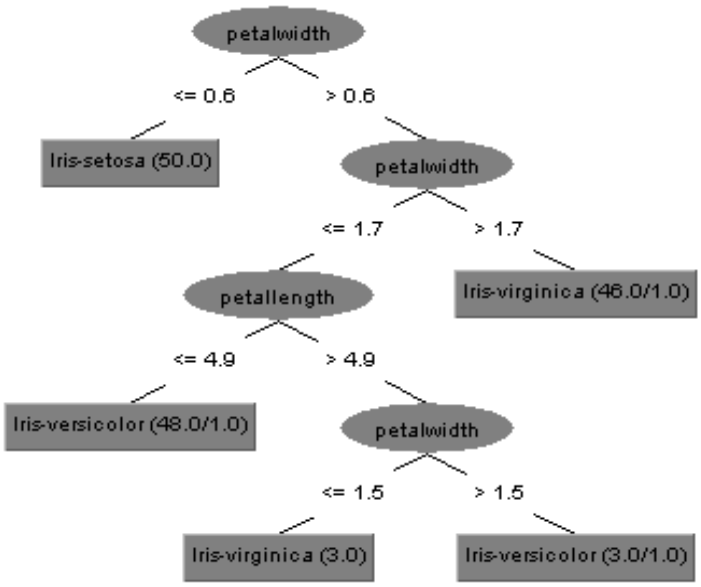

Fig. 2. An illustration of decision tree for Iris data. The decision tree was created and illustrated by Weka system (http://www.cs.waikato.ac.nz/ml/weka).

2.2. Construction of decision tree from decision table

Decision tree is one of the most popular and most accurate classification algorithms. This model is a tree like structure, where each *internal node* is labeled by a *test function* which should be determined on the set of information vectors, and every *terminal none*, also called leave, is labeled either by a decision class or by a class distribution. Every test function (or briefly *test*) defines a ramification of a given node, and each *branch* represents an outcome of the test. Figure 1 represent an example of decision table for famous "Iris" classification problem [6]. The corresponding decision tree for this problem is presented in Fig. 2.

In order to classify an unknown example, the information vector of this example is tested against the decision tree. The path is traced from the root to a leaf node that holds the class prediction for this example.

In this paper, we consider decision trees using cuts as test functions. Every cut (a, c) is associated with a test function $f_{(a,c)}$ such that for any object $u \in U$ the value of $f_{(a,c)}(u)$ is equal to 1 (true) if and only if $a(u) > c$.

The decision tree **T** is called *consistent* with a given decision table $\mathbb{S}$ if it properly classifies all objects from $\mathbb{S}$. The decision tree **T** is called *optimal* for a given decision table $\mathbb{S}$ if it has smallest height among decision trees that are consistent with $\mathbb{S}$. The cut c on attribute a is called *optimal cut* if (a, c) labels one of internal nodes of optimal decision trees.

Unfortunately, the problem of searching for optimal decision tree for a given decision table is NP-hard. Hence, developing any decision tree induction method we should solve the following problem: "*For a given set of candidate cut points $C_a = \{c_1, \ldots, c_N\}$ on an attribute a, find the cut (a, c_i) that most probably belongs to the set of optimal cuts*", e.g., see [3,15].

Usually, to solve this problem, we should use a *measures* (*or quality functions*)

$$\mathcal{F} : \{c_1, \ldots, c_N\} \to \mathbb{R}$$

to estimate the quality of partitions determined by those cuts. For a given measure $\mathcal{F}$, the *straightforward algorithm* of searching for best cut should compute values of $\mathcal{F}$ for all cuts: $\mathcal{F}(c_1), .., \mathcal{F}(c_N)$. The cut c_{Best} which optimizes the value of function $\mathcal{F}$ is selected as the result of searching process.

The basic heuristic for the construction of the decision tree (for example ID3 or later C4.5 – see [15]) is based on the top-down recursive strategy described as follows:

1. It starts with a tree with one node representing the whole training set of objects.
2. If all objects have the same decision class, the node becomes leaf and is labeled with this class.
3. Otherwise, the algorithm selects from the set of all possible tests $F = \{f_1, f_2, \ldots, f_m\}$ the best one.
4. The current node is labeled by the selected test and it is branched accordingly to values of the selected test function. Also, the set of objects is partitioned and assigned to new created nodes.
5. The algorithm uses the same processes (Steps 2, 3, 4) recursively for each new nodes to form the whole decision tree.
6. The partitioning process stops when either all examples in a current node belong to the same class, or no test function has been selected in Step 3.

In next sections, we recall some of most frequently used measures for construction of decision trees like *"Entropy Measure" and "Discernibitity Measure"*, respectively.

2.2.1. Entropy measure

A number of methods based on entropy measure formed a strong group of works in the domain of decision tree induction and discretization [3,15]. This concept uses class-entropy as a criterion to evaluate the list of best cuts which together with the attribute domain induce the desired intervals. The class information entropy of the set of N objects X with class distribution $\langle N_1, \ldots, N_d \rangle$, where $N_1 + \ldots + N_d = N$, is defined by

$$Ent(X) = -\sum_{j=1}^{d} \frac{N_j}{N} \log \frac{N_j}{N}$$

Hence, the entropy of the partition induced by a cut point c on attribute a is defined by

$$E(a, c; U) = \frac{|U_L|}{n} Ent(U_L) + \frac{|U_R|}{n} Ent(U_R) \tag{1}$$

where $\{U_L, U_R\}$ is a partition of U defined by c. For a given feature a, the cut $c_{\min}$ which minimizes the entropy function over all possible cuts is selected see Fig. 3. The methods based on information entropy are reported in [3,15].

2.2.2. Discernibility measure

Discernibility measure is based on Rough Set and Boolean reasoning approach.

Intuitively, energy of the set of objects $X \subset U$ can be defined by the number of pairs of objects from X to be discerned called $conflict(X)$. Let $\langle N_1, \ldots, N_d \rangle$ be the class distribution of X, then $conflict(X)$ can be computed by

$$conflict(X) = \sum_{i<j} N_i N_j$$

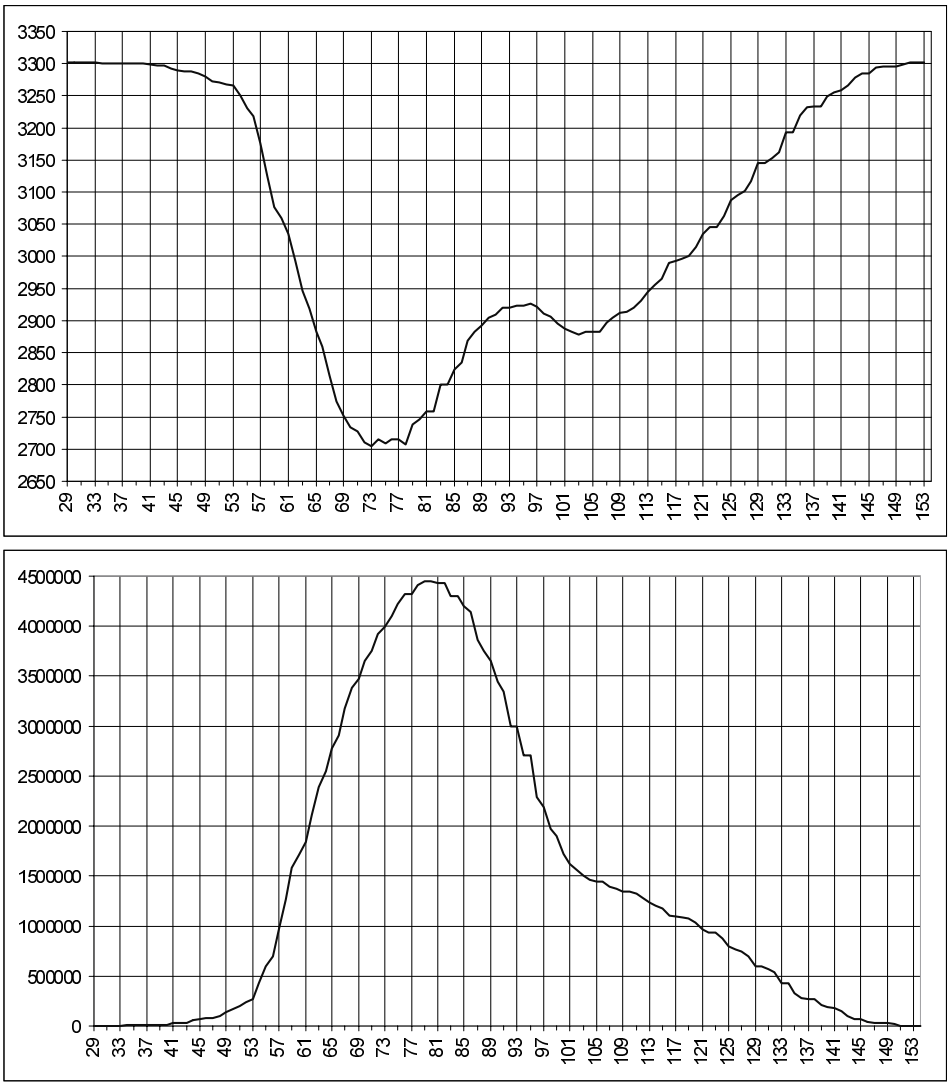

Fig. 3. The illustration of Entropy measure (up) and discernibility measure (low) for the same data set. On horizontal axes: ID of consequent cuts; on vertical axes: values of corresponding measures.

Any cut (a, c) that divides the set of objects U into U_1, and U_2 is evaluated by

$$W(c) = conflict(U) - conflict(U_1) - conflict(U_2) \tag{2}$$

i.e. the larger number of pairs of objects is discerned by the cut (a, c), the bigger is the chance that c can be chosen as one of optimal cuts. This algorithm is called Maximal-Discernibility heuristics or *MD-heuristics* for decision tree construction. Figure 3 illustrates the graph of Entropy and Discernibility functions over the set of possible cuts on one of the attributes of SatImage data. One can see that both measures preferred quite similar cuts as optimal. The high accuracy of decision trees constructed by using discernibility measure and their comparison with Entropy-based methods has been reported in [9].

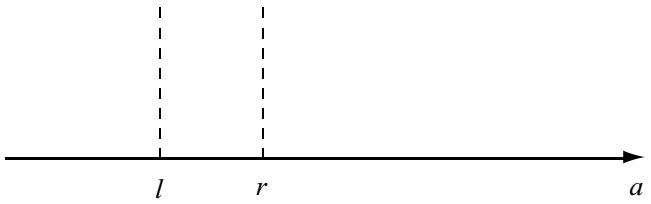

Fig. 4. The soft cut.

3. Soft cuts and soft decision trees

So far, we have presented decision tree methods working with cuts that are treated as sharp partition, since real values are partitioned into disjoint intervals. One can observe that some objects which are very similar can be treated as very different because they are in different sides of a cut. In this section we introduce some notions of *soft cuts* that discern two objects if they are located far enough from cuts.

3.1. Soft cuts

The formal definition of soft cuts is following:

Definition 3.1 A soft cut is any triple $p = \langle a, l, r\rangle$, where $a \in A$ is an attribute, $l, r \in \Re$ are called the left and right bounds of p ($l \leqslant r$); the value $\varepsilon = \frac{r-l}{2}$ is called the uncertain radius of p. We say that a soft cut p discerns pair of objects x_1, x_2 if $a(x_1) < l$ and $a(x_2) > r$.

The intuitive meaning of $p = \langle a, l, r\rangle$ is such that there is a real cut somewhere between l and r. So, we are not sure where one can place the real cut in the interval $[l, r]$. Hence for any value $v \in [l, r]$ we are not able to check if v is either on the left side or on the right side of the real cut. Then we say that the interval $[l, r]$ is an uncertain interval of the soft cut p. Any normal cut can be treated as soft cut of radius equal to 0.

Any set of soft cuts splits the real axis into intervals of two categories: the intervals corresponding to new nominal values and the intervals of uncertain values called boundary regions. The problem of searching for minimal set of soft cuts with a given uncertain radius can be solved in a similar way to the case of sharp cuts. We propose some heuristic for this problem in the last section of the paper. The problem becomes more complicated if we want to obtain as small as possible set of soft cuts with the radius as large as possible. We will discuss this problem in the next paper.

3.2. Soft decision tree

The test functions defined by traditional cuts can be replaced by soft cuts. We have proposed two strategies as modifications of standard classification method for decision tree with soft cuts (fuzzy separated cuts). They are called *fuzzy decision trees* and *rough decision trees*.

In fuzzy decision tree method, instead of checking the condition $a(u) > c$ we have to check how strong is the hypothesis that u is on the left or right side of the cut (a, c). This condition can be expressed by $\mu_L(u)$ and $\mu_R(u)$, where μ_L and μ_R are membership function of left and right intervals (respectively). The values of those membership functions can be treated as a probability distribution of u in the node labeled by soft cut $(a, c-\varepsilon, c+\varepsilon)$. Then one can compute the probability of the event that object u is reaching a leaf. The decision for u is equal to decision labeling the leaf with largest probability.

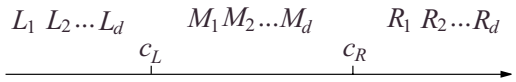

Fig. 5. The evaluation measure should be determined from class distributions of three intervals: $(-\infty; c_L)$, $[c_L; c_R]$ and $[c_R; \infty)$.

In the case of rough decision trees, when we are not able to decide to turn left or right (the value $a(u)$ is too close to c), we do not distribute the probability to the children of considered node. We have to compare their answers taking into account the numbers of objects supported by them. The answer with the highest number of supported objects is a decision of the given object.

4. Searching for best cuts in large data tables

In this section we present the efficient approach to searching for optimal cuts in large data tables. The main idea is to apply the *"divide and conquer"* strategy to determine the best cut $c_{Best} \in \{c_1, \ldots, c_n\}$ with respect to a given quality function. First, we divide the set of possible cuts into k intervals (e.g. $k = 2, 3, ..$). Then, we choose the interval to which the best cut may most probably belong (using some evaluation measures). This process is repeated until the considered interval consists of one cut. Then the best cut can be chosen between all visited cuts.

The problem arises on how to define the measure evaluating the quality of the interval $[c_L; c_R]$ having class distributions: $\langle L_1, \ldots, L_d \rangle$ in $(-\infty; c_L)$; $\langle M_1, \ldots, M_d \rangle$ in $[c_L; c_R)$; and $\langle R_1, \ldots, R_d \rangle$ in $[c_R; \infty)$. The approximation measure $Eval([c_L; c_R])$ should estimate the quality of the best cut among those belonging to the interval $[c_L; c_R]$ with respect to discernibility or entropy measures. In Section 4.1 we will present different approximation measures. The general algorithm (with respect to the given approximate measure $Eval(.)$) can be described as follows:

Hence the number of queries necessary for running our algorithm is of order $O(dk \log_k N)$. If we want to minimize the number of queries, we should set $k = 3$, because the function $f(k) = dk \log_k N$ has minimum over positive integers for $k = 3$. For $k > 2$, instead of choosing the best interval $[p_{i-1}; p_i]$, one can select the best union $[p_{i-m}; p_i]$ of m consecutive intervals in every step for a predefined parameter $m < k$. The modified algorithm needs more – but still of order $O(\log N)$ – simple questions only.

4.1. Approximation measures

Approximation measure for *Gini's index* has been introduced and applied in CLOUDS algorithm (see [1]), but it has been determined by sampling technique.

In previous papers [10–12], we considered two assumptions about distributions of objects in $[c_L, c_R]$. The "*full independency assumption* supposes that distributions of objects from each decision class in $[c_L, c_R]$ are pairwise independent. The "*full dependency assumption*" supposes that the values $x_1, \ldots, x_d$ are proportional to $M_1, \ldots, M_d$, i.e., $\frac{x_1}{M_1} \simeq \frac{x_2}{M_2} \simeq \ldots \simeq \frac{x_d}{M_d}$.

We are going to present the approximations of discernibility and entropy measures under full independent and full dependent assumptions. Detail justifications of those results can be found in our previous papers (see [11,12]).

ALGORITHM 1: Searching for semi-optimal cut

INPUT: Attribute a; the set of candidate cuts $\mathbf{C}_a = \{c_1, .., c_N\}$ on a;

OUTPUT: The optimal cut $c \in \mathbf{C}_a$

begin

$Left \leftarrow 1$; $Right \leftarrow N$;

while ($Left < Right$)

1.Divide $[Left; Right]$ into k intervals with equal length by $(k+1)$ boundary points i.e.

$$p_i = Left + i * \frac{Right - Left}{k};$$

for $i = 0, .., k$

2.**for** $i = 1, .., k$

Compute $Eval([c_{p_{i-1}}; c_{p_i}])$;
Let $[p_{j-1}; p_j]$ be the interval with maximal value of $Eval(.)$;

3.$Left \leftarrow p_{j-1}$; $Right \leftarrow p_j$;

endwhile;

return the cut c_{Left};

end

4.1.1. Approximation of discernibility measures

In our previous papers (see [11,12]) we have proposed the following measures to estimate the quality of the best cut in $[c_L; c_R]$:

$$Eval\left([c_L; c_R]\right) = \frac{W(c_L) + W(c_R) + conflict([c_L; c_R])}{2} + \Delta \tag{3}$$

where the value of Δ is defined by:

$$\Delta = \frac{[W(c_R) - W(c_L)]^2}{8 \cdot conflict([c_L; c_R])}$$

in the dependent model, and

$$\Delta = \alpha \cdot \sqrt{\sum_{i=1}^{n} \left[\frac{M_i(M_i+2)}{12} \left(\sum_{j \neq i} (R_j - L_j) \right)^2 \right]}$$

in the independent model.

The choice of Δ and the value of parameter α from $[0; 1]$ can be tuned in learning process or are given by expert. One can see that, in order to determine the value $Eval\left([c_L; c_R]\right)$, we need only $O(d)$ simple SQL queries of the form:

```
SELECT COUNT
FROM data_table
WHERE (a BETWEEN c_L AND c_R) GROUPED BY d.
```

4.1.2. Approximation of entropy measures

In general, it is harder to approximate entropy measure than discernibility measure. In previous paper (see [12]) we have shown that it is still possible, but it requires more complicated computation efforts.

In the independent model, quality of the best cut in $[c_L; c_R]$ with respect to entropy measure can be evaluated by:

$$Eval\left([c_L; c_R]\right) = H(L, M) + H(R, M) - \sum_{j=1}^{d}[H(L_j, M_j) + H(R_j, M_j)]$$

where

$$H(a, b)) \simeq \frac{1}{b}\int_0^b (a + x)\log(a + x)dx$$

$$= \frac{(a + b)^2\log(a + b) - a^2\log(a)}{2b} - \frac{2a + b}{4\ln 2}$$

In the independent model, we did not find any "nice formula" estimating the quality of best cut in the given interval. But, in [12], we have proposed the numerical method to compute this estimation under independent assumptions.

4.2. Searching for soft cuts

One can modify the algorithm presented in the previous section to determine "soft cuts" in large databases. The modification is based on changing the stop condition. In every iteration of the Algorithm 1, the current interval $[Left; Right]$ is divided equally into k smaller intervals, and the best smaller interval will be chosen as the current interval. In the modified algorithm one can either select one of smaller intervals as the current interval or stop the algorithm and return the current interval as a result.

Intuitively, the Divide and Conquer Algorithm is stopped and it results the interval $[c_L; c_R]$ if the following conditions hold:

- The class distribution in $[c_L; c_R]$ is too stable, i.e. there is no sub-interval of $[c_L; c_R]$ which is considerably better than $[c_L; c_R]$ itself;
- The interval $[c_L; c_R]$ is sufficiently small, i.e. it contains a small number of cuts;
- The interval $[c_L; c_R]$ does not contain too much objects; (because the large number of uncertain objects cans result in larger decision tree and then it prolongs the time of decision tree construction)

5. Experiments

In previous papers, we have reported the experimental results showing, that for many benchmark datasets, the best cut found by using approximate discernibility measure is very close to the best cut found by using exact measure.

Table 1
The experimental results of different decision tree algorithms on benchmark data

Data sets	#objects×#attr.	SLIQ	ENT	MD	MD*
Australian	690 × 14	84.9	85.2	86.2	86.2
German	1000 × 24	–	70	69.5	70.5
Heart	270 × 13	–	77.8	79.6	79.6
Letter	20000 × 16	84.6	86.1	85.4	83.4
SatImage	6435 × 36	86.3	84.6	82.6	83.9
Shuttle	57000 × 9	99.9	99.9	99.9	98.7

In this section, we present the accuracy evaluation of our approach. The main goal is to compare the accuracy of soft decision tree built from semi-optimal cuts with other decision tree techniques.

We have implemented three decision tree algorithms called "ENT" (based on entropy measure, similar to C4.5 [15]), "MD" (based on discernibility measure [9]) and "MD*"(soft tree based on approximate discernibility measure). The experiments are done on "small" data set (from STATLOG project [8]) only, since the first two algorithms handle the data sets, that fit into the memory. We also recall the experiment results achieved by SLIQ algorithm (see [7]).

In our experiments, the standard algorithms based on entropy and discernibility measures are implemented without pruning step. The MD* algorithm is based on approximate discernibility measure (see Formula (3), where Δ was set by 0, i.e.

$$Eval\left([c_L; c_R]\right) = \frac{W(c_L) + W(c_R) + conflict([c_L; c_R])}{2}$$

We used the fuzzy decision tree classification method (described in Section 3.2) in order to classify new objects. From experimental results, we can see that: "Even MD* algorithm constructs decision trees from approximation measure, its accuracy is still comparable with other exact measures".

6. Conclusions

The problem of optimal binary partition of continuous attribute domain for large data sets stored in *relational data bases* has been investigated. We showed that one can reduce the number of simple queries from $O(N)$ to $O(\log N)$ in order to construct the partition very close to the optimal one.

The experimental results are very promising. These results can be improved by setting the parameter Δ in Formula (3) and by applying pruning techniques. The proposed method can be easily implemented in parallel.

Acknowledgement

The research has been partially supported by the grant 3T11C00226 from the Ministry of Scientific Research and Information Technology of the Republic of Poland and by the research grant of Polish-Japanese Institute of Information Technology.

References

[1] A. Khaled, R. Sanjay and S. Vineet, *CLOUDS: A Decision Tree Classifier for Large Datasets*, KDD 1998, 2–8.

[2] M. Beynon, C. Chang, U. Catalyurek, T. Kurc, A. Sussman, H. Andrade, R. Ferreira and J. Saltz, Processing large-scale multidimensional data in parallel and distributed environments, *Parallel Computing* **28**(5) (May 2002), 827–859.
[3] U.M. Fayyad and K.B. Irani, On the handling of continuous-valued attributes in decision tree generation, *Machine Learning* **8** (1992), 87–102.
[4] R. Jin, G. Yang and G. Agrawal, Shared Memory Parallelization of Data Mining Algorithms: Techniques, Programming Interface, and Performance, *IEEE Transactions on Knowledge and Data Engineering* **17**(1) (January 2005), 71–89.
[5] G.H. John and P. Langley, *Static vs. dynamic sampling for data mining*, Proc. of the Second International Conference of Knowledge Discovery and Data Mining, Portland, AAAI Press 1996, 367–370.
[6] M.G. Kendall, A. Stuart and J.K. Ord, *The advanced Theory of Statistics*, (Vol. 3), Design and Analysis and Time Series, Chapter 44, Griffin, London, 1983.
[7] M. Metha, R. Agrawal and J. Rissanen, *SLIQ: A Fast Scalable Classifier for Data Mining*, Proc. of the 5th Int'l Conference on Extending Database Technology (EDBT), Avignon, France, 1996, 18–32.
[8] D. Michie, D.J. Spiegelhalter and C.C. Taylor, *Machine Learning, Neural and Statistical Classifi- cation*, Ellis Horwood, 1994
[9] H.S. Nguyen, Discretization Methods in Data Mining, in: *Rough Sets in Knowledge Discovery*, (Vol. 1), L. Polkowski and A. Skowron, eds, Springer Physica-Verlag, Heidelberg, 1998, pp. 451–482.
[10] H.S. Nguyen, *Efficient SQL-Querying Method for Data Mining in Large Data Bases*, Proc. of IJCAI-99, Morgan Kaufmann Publishers, Stockholm, Sweden, 1999, 806–811.
[11] H.S. Nguyen, *On Efficient Construction of Decision tree from Large Databases*, Proc. of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC'2000). Springer-Verlag, 316–323.
[12] H.S. Nguyen, On Efficient Handling of Continuous Attributes in Large Data Bases, *Fundamenta Informatica* **48**(1), 61–81.
[13] Z. Pawlak, *Rough sets: Theoretical aspects of reasoning about data*, Kluwer Dordrecht, 1991.
[14] L. Polkowski and A. Skowron, eds, *Rough Sets in Knowledge Discovery*, (Vols 1, 2), Springer Physica-Verlag, Heidelberg, 1998.
[15] J.R. Quinlan, *C4.5. Programs for machine learning*, Morgan Kaufmann, San Mateo CA, 1993.
[16] J.C. Shafer, R. Agrawal and M. Mehta, *SPRINT: A Scalable Parallel Classifier for Data Mining*, Proc. of the 22nd VLDB Conference, September, 1996, 544–555.