Chapter 13
On Exploring Soft Discretization of Continuous Attributes

Hung Son Nguyen

Institute of Mathematics, Warsaw University, Banacha 2, 02-097 Warsaw , Poland
son@mimuw.edu.pl

Summary. Searching for a binary partition of attribute domains is an important task in data mining. It is present in both decision tree construction and discretization. The most important advantages of decision tree methods are compactness and clearness of knowledge representation as well as high accuracy of classification. Decision tree algorithms also have some drawbacks. In cases of large data tables, existing decision tree induction methods are often inefficient in both computation and description aspects. Another disadvantage of standard decision tree methods is their instability, i.e., small data deviations may require a significant reconstruction of the decision tree. We present novel *soft discretization* methods using *soft cuts* instead of traditional *crisp* (or sharp) cuts. This new concept makes it possible to generate more compact and stable decision trees with high accuracy of classification. We also present an efficient method for soft cut generation from large databases.

1 Introduction

Classification and description of target concepts are among the most important tasks in knowledge discovery in database (KDD) processes. There are many efficient classification techniques, but only approaches based on a decision rule set and a decision tree can be applied to perform both classification and description tasks. The other approaches such as case base reasoning (e.g., the nearest neighbor method) or artificial neural networks are not suitable for description tasks. In practice, one can notice the unpleasant fact that although rule based methods and decision tree methods are more complex than, e.g., the nearest neighbor method, their classification accuracy is not better for some data sets. Let us explain the reason for this phenomenon.

Usually, the existing (traditional) classification methods based on rule set generation require a data preprocessing step called discretization of continuous attributes, which divides the attribute domain into intervals. Decision tree methods can be used for the best partition of attribute domain extraction. The problem of searching for optimal partitions of real value attributes, defined by so-called cuts, has been studied by many authors ([1–3, 5, 12, 22]), where optimization criteria are defined by, e.g., the height of the decision tree obtained, the number of cuts, or the classification accuracy of the decision tree on new unseen objects. In general, all of those problems are hard from the computational point of view. Hence, numerous heuristics have been investigated to develop approximate solutions of these problems. One of the major tasks of these heuristics is to define some approximate measures estimating the quality of extracted cuts. Hence, in both discretization and decision tree construction methods, it is necessary to use crisp conditions expressed by cuts for object discerning. In our opinion, this approach can lead to misclassification of new objects which are, e.g., close to the separating boundary between decision classes and provide low quality of new object classification. Furthermore, in some data the values of attributes are measured by sensors, and we know that these values are not perfect with regard to device errors. In such cases, low classification accuracy can be caused by the impossibility of analyzing noisy data by using crisp cuts.

We propose a novel approach based on *soft cuts*, which makes it possible to overcome this difficulty. Our methods are based on the main approach of *rough set data analysis* methods, i.e., based on *handling the discernibility between objects* [20, 24]. Using rough set and Boolean reasoning-based methods, one can define the quality of cuts by the number of pairs of objects discerned by the partition (called *the discernibility measure*).

In this chapter, we consider a discretization problem defined by *soft cuts* and the rough-fuzzy reasoning scheme. We also propose some modifications of existing rule induction methods by soft cuts.

We also present efficient strategies of searching for the best cuts (both soft and crisp cuts). In this chapter, we discuss two strategies called *local* and *global searching strategies*. These strategies allow us to implement the proposed method for large data tables stored in databases.

There are two strategies for solving the problem of searching for an optimal decision tree or discretization for real value data, assuming that a large data table is represented in the relational database. The most popular strategy is based on the sampling technique, i.e., on building a decision tree for a small, randomly chosen subset of data and then evaluating the quality of the decision tree for all of the data. If the quality of a generated decision tree is not sufficient, we have to repeat this step for a new sample. In this chapter, we propose new methods using all of the data. Using a straightforward approach to optimal partition selection (with respect to a given measure), the number of necessary queries is of order $O(N)$, where N is the number of preassumed partitions of the searching space. For large databases, even linear complexity is not acceptable because of the time necessary for one step. The critical factor for time complexity of algorithms solving the problem discussed is the number of simple structured query language (SQL) queries such as, *select count from ... where attribute between ...*, (related to some interval of attribute values) necessary to construct such partitions. We assume that the answering time for such queries does not depend on the interval length. We show some properties of considered optimization measures allowing us to reduce the searching space size. Moreover, we prove that using only $O(\log N)$ simple queries, one can construct a partition very close to optimal. We have shown that the main part of the formula estimating the quality of the best cut for independent variables from [18] is the same for fully dependent variables.

This chapter is organized as follows: in Sect. 2 we present the main notations related to rough set theory, discretization, and the decision tree problem. The definition and application of soft cuts in classification problems are presented in Sect. 3. In Sect. 4 we present efficient searching methods for the best soft and crisp cuts. Conclusions and remarks are presented in Sect. 5.

2 Basic Notions

An *information system* [20] is a pair $\mathbb{A} = (U,A)$, where U is a nonempty, finite set called the *universe* and A is a nonempty finite set of *attributes* (or *features*), i.e., $a : U \to V_a$ for $a \in A$, where V_a is called *the value set of* a. Elements of U are called *objects* or *records*. Two objects $x,y \in U$ are said to be *discernible* by attributes from A if there exists an attribute $a \in A$ such that $a(x) \neq a(y)$.

Any information system of the form $\mathbb{A} = (U,A \cup \{dec\})$ is called a *decision table* where $dec \notin A$ is called a *decision attribute*. Without loss of generality, we assume that $V_{\text{dec}} = \{1,\ldots,d\}$. Then the set $DEC_k = \{x \in U : dec(x) = k\}$ will be called the kth *decision class of* $\mathbb{A}$ for $1 \leq k \leq d$. Any pair (a,c), where a is an attribute and c is a real value, is called *a cut*. We say that *cut* (a,c) *discerns a pair of objects* x, y if either $a(x) < c \leq a(y)$ or $a(y) < c \leq a(x)$.

2.1 Rough Set Based Discretization Method

Discretization is a process of determining the partition of attribute domains into intervals. Such a partition can be defined uniquely by some set of cuts.

Definition 1. For a given decision table $\mathbb{A} = (U,A \cup \{dec\})$, the set of cuts $\mathbf{C}$ is called $\mathbb{A}$-*consistent*, if for any pair of objects (x,y) such that $dec(x) \neq dec(y)$ and x,y are *discernible* by A, there exists a cut $(a,c) \in \mathbf{C}$ discerning x from y. The set of cuts $\mathbf{P}^{irr}$ is $\mathbb{A}$-*irreducible* if $\mathbf{P}$ is not $\mathbb{A}$-consistent for any $\mathbf{P} \subset \mathbf{P}^{irr}$. The set of cuts $\mathbf{P}^{opt}$ is $\mathbb{A}$-*optimal* if $card\,(\mathbf{P}^{opt}) \leq card\,(\mathbf{P})$ for any $\mathbb{A}$-consistent set of cuts $\mathbf{P}$.

Rough set based discretization methods are oriented to searching for an optimal set of cuts. In previous papers, we have shown the following theorems:

Theorem 1. *[12] For a given decision table $\mathbb{A}$ and an integer k: The decision problem of checking if there exists an $\mathbb{A}$-irreducible set of cuts* **P** *such that* $card(\mathbf{P}) < k$ *is NP-complete. The problem of searching for an $\mathbb{A}$-optimal set of cuts is NP-hard.*

To prove this theorem, one can construct efficient heuristics using a so-called Boolean reasoning approach.

For a given decision table $\mathbb{A} = (U, A \cup \{d\})$, a new decision table,

$$\mathbb{A}^* = (U^*, A^* \cup \{d^*\}),$$

is constructed as follows:

- $U^* = \left\{(u,v) \in U^2 : d(u) \neq d(v)\right\} \cup \{\perp\}$.
- $A^* = \{c : c \text{ is a cut on } \mathbb{A}\}$; $c(\perp) = 0$.
  $c[(u_i,u_j)] = \begin{cases} 1 & \text{if cut } c \text{ discerns } u_i, u_j \\ 0 & \text{otherwise.} \end{cases}$
- $d(\perp) = 0; d^*(u_i,u_j) = 1$.

$\mathbb{A}^*$	c_1	c_2	$\cdots$	c	$\cdots$	d^*
(u_1,u_2)	1	0	$\cdots$	$\cdots$	$\cdots$	1
$\vdots$						
(u_i,u_j)	0			1		1
$\vdots$			$\cdots$			
$\perp$	0	0	$\cdots$	0	$\cdots$	0

It has been shown that any set of cuts is $\mathbb{A}$-irreducible if and only if it is a reduct of A^*.

Our maximal discernibility heuristic (MD) based on searching for cuts with a maximal number of object pairs discerned by this cut [12] is described as follows:

> *From the set of all possible cuts A^*, the cut discerning the maximal number of pairs of objects from different decision classes is chosen, and this step is repeated until no two objects from different decision classes discerned by some cut can be found.*

This heuristic can be realized quite efficiently. In [15], we have shown that the total time of discretization is $O(nk \cdot |\mathbf{C}|)$, where **C** is a final set of cuts for discretization and n and k are numbers of objects and attributes in the decision table.

2.2 Decision Tree Construction from Decision Tables

The decision tree for a given decision table is (in the simplest case) a binary directed tree with *test functions* (i.e., Boolean functions defined on the information vectors of objects) labeling internal nodes and decision values labeling leaves. In this chapter, we consider decision trees using cuts to represent test functions. Any 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 typical algorithm for decision tree induction can be described as follows:

1. For a given set of objects U, select a cut (a, c_{Best}) of high quality among all possible cuts and all attributes.
2. Induce a partition U_1, U_2 of U by (a, c_{Best}).
3. Recursively apply Step 1 to both sets U_1, U_2 of objects until some stopping condition is satisfied.

In developing some decision tree induction methods [5,22] and some supervised discretization methods [1,3,12,15], it is often necessary to solve the following problem:

> *For a given real value attribute a and set of candidate cuts $\{c_1, \ldots, c_N\}$, find a cut (a, c_i) belonging to the set of optimal cuts with the highest probability.*

Usually, we use some *measure* (or *quality function*) $F : \{c_1, \ldots, c_N\} \to \mathbb{R}$ to estimate the quality of cuts. For a given measure F, any *straightforward algorithm* should compute the values of F for all cuts: $F(c_1), \ldots, F(c_N)$. The cut c_{Best} that maximizes or minimizes the value of function F is selected as the result of the searching process. Let us consider the attribute a and the set of all relevant cuts $\mathbf{C}_a = \{c_1, \ldots, c_N\}$ on a.

Definition 2. The d-tuple of integers $\langle x_1, \ldots, x_d \rangle$ is called the class distribution of the set of objects $X \subset U$ iff $x_k = card(X \cap DEC_k)$ for $k \in \{1, \ldots, d\}$. If the set of objects X is defined by $X = \{u \in U : p \leq a(u) < q\}$ for some $p, q \in \mathbb{R}$, then the class distribution of X can be called the class distribution in $[p;q)$.

Any cut $c \in \mathbf{C}_a$ splits the domain $V_a = (l_a, r_a)$ of the attribute a into two intervals: $I_L = (l_a, c); I_R = (c, r_a)$. We will use the following notation:

- U_{L_j}, U_{R_j} — the sets of objects from the j^{th} class in I_L and I_R where $j \in \{1, \ldots, d\}$.
- $U_L = \bigcup_j U_{L_j}$, and $U_R = \bigcup_j U_{R_j}$.
- $\langle L_1, \ldots, L_d \rangle$ and $\langle R_1, \ldots, R_d \rangle$ — class distributions in U_L and U_R.
- $L = \sum_{j=1}^{d} L_j$, and $R = \sum_{j=1}^{d} R_j$.
- $C_j = L_j + R_j$ — number of objects in the j^{th} class.
- $n = \sum_{i=1}^{d} C_j = L + R$ — the total number of objects.

In the following sections, we recall the most frequently used measures for decision tree induction such as *"entropy function" and "discernibility measure."*

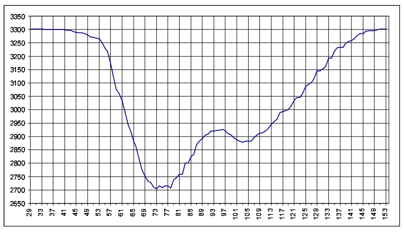
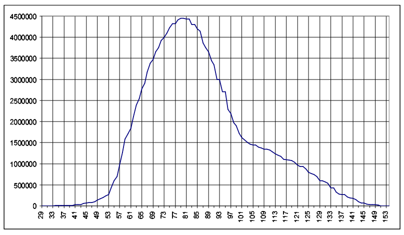

Fig. 1. Illustration of entropy measure (left) and discernibility measure (right)

Entropy methods A number of methods based on entropy measure have been developed in the domain of decision tree induction and discretization. These methods use class entropy as a criterion to evaluate the list of best cuts, which together with the attribute domain induce the relevant 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

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

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. 1. The methods based on information entropy are reported in [1, 2, 6, 22].

Maximal discernibility principle Cuts in Boolean reasoning methods are treated as Boolean variables, and the searching problem for an optimal set of cuts can be characterized by a Boolean function $f_{\mathbb{A}}$ (where $\mathbb{A}$ is a given decision table). Any set of cuts is $\mathbb{A}$-consistent if and only if the corresponding evaluation of variables in $f_{\mathbb{A}}$ returns the value $True$ [12]. We have shown that the quality of cuts can be measured by their *discernibility properties*. Intuitively, the energy of the set of objects $X\subset U$ can be defined by the number, called $conflict(X)$, of pairs of objects from X to be discerned. Let $\langle N_1,\ldots,N_d\rangle$ be a class distribution of X, then $conflict(X)$ can be computed by

$$conflict(X)=\sum_{i<j}N_iN_j.$$

The cut c, which divides the set of objects U into U_1 and U_2 is evaluated by

$$W(c)=conflict(U)-conflict(U_1)-conflict(U_2),$$

i.e., the higher the number of pairs of objects discerned by the cut (a,c), the larger is the chance that c can be added to the optimal set of cuts. Hence, the decision tree induction algorithms based on the rough set and the Boolean reasoning approach use the quality of a given cut c defined by

$$W(c) = \sum_{i \neq j}^{d} L_i R_j = \sum_{i=1}^{d} L_i \sum_{i=1}^{d} R_i - \sum_{i=1}^{d} L_i R_i. \tag{1}$$

The algorithm based on such a measure is called maximal-discernibility heuristics (MD-heuristics) for decision tree construction. Figure 1 illustrates changes of values of entropy and discernibility functions over the set of possible cuts on one of the attributes of SatImage data. One can see that the cuts preferred by both measures are quite similar. The high accuracy of decision trees constructed by using discernibility measures and their comparison with entropy-based decision methods are reported in [16, 17].

3 Discretization by Soft Cuts

So far, we have presented discretization methods working with sharp partitions defined by cuts, i.e., domains of real values are partitioned by them into disjoint intervals. One can observe that in some situations, similar (class) objects can be treated by cuts as very different. In this section, we introduce *soft cuts* discerning two given values if those values are far enough from the cut. The formal definition of soft cuts follows:

> *A soft cut is any triple $p = \langle a, l, r \rangle$, where $a \in A$ is an attribute, $l, r \in \mathbb{R}$ are called the left and right bounds of p ($l \leq r$); and the value $\varepsilon = \frac{r-l}{2}$ is called the uncertainty radius of p. We say that a soft cut p discerns the 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 whether v is 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. One can see that any normal cut is a soft cut of radius equal to zero.

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 a minimal set of soft cuts with a given uncertainty radius can be solved in a way similar to the case of sharp cuts. We propose some heuristic for this problem in the last section of the chapter. The problem becomes more complicated if we want to obtain the smallest set of soft cuts with a radius as large as possible. We will discuss this problem later. Now, we recall some existing rule induction methods for real value attribute data and their modifications using soft cuts.

Instead of sharp cuts (see previous sections), soft cuts determine additionally some uncertainty regions. Assume that $\mathbf{P} = \{p_1, p_2, \ldots, p_k\}$ is a set of soft cuts on attribute $a \in A$, where $p_i = (a, l_i, r_i); l_i \leq r_i$ and $r_i < l_{i+1}$for $i = 1, \ldots, k-1$. The set of soft cuts $\mathbf{P}$ defines on $\mathbb{R}$ a partition,

$$\mathbb{R} = (-\infty, l_1) \cup [l_1, r_1] \cup (r_1, l_2) \cup \ldots \cup [l_k, r_k] \cup (r_k, +\infty),$$

and at the same time defines a new nominal attribute $a^{\mathbf{P}} : U \to \{0, 1, \ldots, k\}$, such that $a^{\mathbf{P}}(x) = i$ if and only if $a(x) \in (r_i, l_{i+1})$; $i = 1, \ldots, k$. We are proposing some possible classification methods using soft discretization. These methods are based on the fuzzy set approach, rough set approach, clustering approach, and decision tree approach.

3.1 Fuzzy Set Approach

In the fuzzy set approach, one can treat the interval $[l_i, r_i]$ for any $i \in \{1, \ldots, k\}$ as a kernel of some fuzzy set Δ_i. The membership function $f_{\Delta_i} : \mathbb{R} \to [0, 1]$ is defined as follows:

1. $f_{\Delta_i}(x) = 0$ for $x < l_i$ or $x > r_{i+1}$.
2. $f_{\Delta_i}(x)$ increases from 0 to 1 for $x \in [l_i, r_i]$.
3. $f_{\Delta_i}(x)$ decreases from 1 to 0 for $x \in [l_{i+1}, r_{i+1}]$.
4. $f_{\Delta_i}(x) = 1$ for $x \in (r_i, l_{i+1})$.

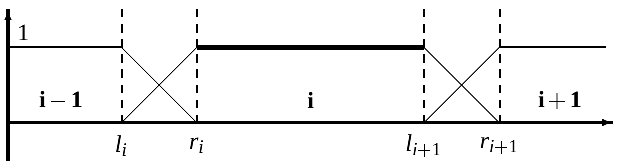

Fig. 2. Membership functions of intervals

Having defined membership function, one can use the idea of a *fuzzy graph* [4] to represent the knowledge discovered.

3.2 Rough Set Approach

The boundary interval $[l_i, r_i]$ can be treated as an uncertainty region for a real sharp cut. Hence, using the rough set approach, the intervals (r_i, l_{i+1}) and $[l_i, r_{i+1}]$ are treated as the lower and the upper approximations of any set X. Hence, we use the following notation: $\mathbf{L}_a(X_i) = (r_i, l_{i+1})$ and $\mathbf{U}_a(X_i) = [l_i, r_{i+1}]$, such that $(r_i, l_{i+1}) \subseteq X \subseteq [l_i, r_{i+1}]$.

Having approximations of nominal values of all attributes, we can generate an upper and a lower approximation of decision classes by taking the Cartesian product of rough sets. For instance, let the set X be given by its rough representation $[\mathbf{L}_B(X), \mathbf{U}_B(X)]$ and the set Y by $[\mathbf{L}_C(Y), \mathbf{U}_C(Y)]$, and let $B \cap C = \emptyset$. One can define a rough representation of $X \times Y$ by $[\mathbf{L}_{B\cup C}(X \times Y), \mathbf{U}_{B\cup C}(X \times Y)]$, where

$$\mathbf{L}_{B\cup C}(X \times Y) = \mathbf{L}_B(X) \times \mathbf{L}_C(Y)$$

and

$$\mathbf{U}_{B\cup C}(X \times Y) = \mathbf{U}_B(X) \times \mathbf{U}_C(Y)\,.$$

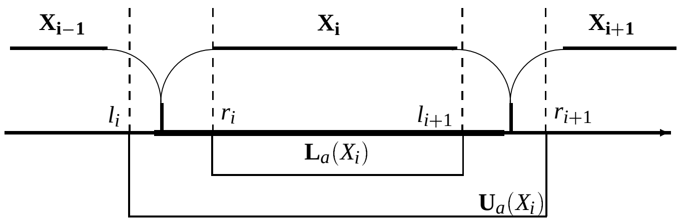

Fig. 3. Illustration of soft cuts

3.3 Clustering Approach

Any set of soft cuts $\mathbf{P}$ defines a partition of real values of attributes into disjoint intervals, which determine a natural equivalence relation $IND(\mathbf{P})$ over the set of objects. New objects belonging to the boundary regions can be classified by applying the rough set membership function to test the hypothesis that the new object belongs to a certain decision class.

One can also apply the idea of clustering. Any set of soft cuts defines a partition of $\mathbb{R}^k$ into k-dimensional cubes. Using the rough set approach one can classify some of those cubes in the lower approximation of a certain set, and they can be treated as clusters. To classify a new object belonging to any boundary cube one can compare distances from this object to the centers of adjacent clusters (see Fig. 4).

3.4 Decision Tree with Soft Cuts

In [13], we have presented some methods for decision tree construction from cuts (or oblique hyperplanes). Here, we propose two strategies that are modifications of that method using soft cuts (fuzzy separated cuts) described above. They are called *fuzzy decision tree* and *rough decision tree*.

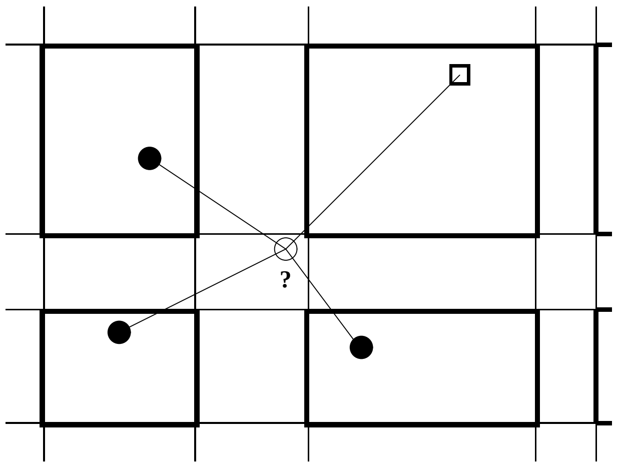

Fig. 4. Clustering approach

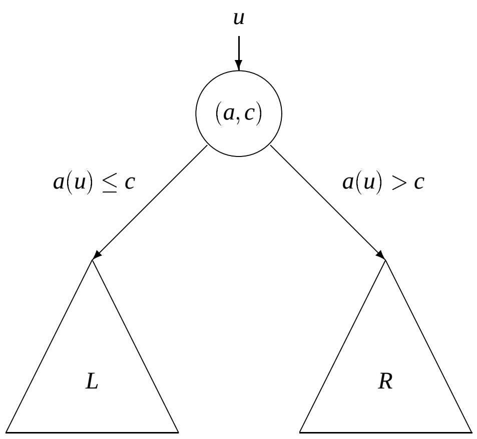

Fig. 5. Standard decision tree approach

The new object $u \in U$ can be classified by a given (traditional) decision tree as follows:

> *We start from the root of the decision tree. Let (a,c) be a cut labeling the root. If $a(u) > c$, we go to the right subtree, and if $a(u) \leq c$, we go to the left subtree of the decision tree. The process is continued for any node until we reach any external node.*

If the fuzzy decision tree method is used, then instead of checking the condition $a(u) > 0$, we have to check the strength of 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 functions 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 the soft cut $(a, c-\varepsilon, c+\varepsilon)$. Then, one can compute the probability of the event that object u reaches a leaf. The decision for u is equal to decision labeling the leaf with the largest probability.

If the rough decision tree is used and we are not able to decide whether to turn left or right, we do not distribute the probabilities to children of the node considered. We have to compare the children's answers, taking into account the numbers of objects supported by them. The answer with the largest number of supported objects determines the decision for a given object.

4 Searching for Best Cuts in Large Data Tables

In this section, we present an efficient approach to searching for optimal cuts in large data tables. We consider some modifications of our MD-heuristic described in previous sections. The next sections describe some techniques that have been presented in previous papers (see [18]).

4.1 Tail Cuts Can Be Eliminated

First, let us consider two cuts $c_L < c_R$. Let $\langle L_1, \ldots, L_d \rangle$ be the class distribution in $(-\infty; c_L)$, $\langle M_1, \ldots, M_d \rangle$ the class distribution in $[c_L; c_R)$, and $\langle R_1, \ldots, R_d \rangle$ the class distribution in $[c_R; \infty)$.

Lemma 1. *The following equation holds:*

$$W(c_R) - W(c_L) = \sum_{i=1}^{d} \left[(R_i - L_i) \sum_{j \neq i} M_j \right]. \qquad (2)$$

This lemma implies the following techniques for eliminating irrelevant cuts:

For a given set of cuts $\mathbf{C}_a = \{c_1, \ldots, c_N\}$ on a, by the median of the kth decision class we mean the cut $c \in \mathbf{C}_a$, which minimizes the value $|L_k - R_k|$. The median of the kth decision class will be denoted by $Median(k)$. Let $c_1 < c_2 \ldots < c_N$ be the set of candidate cuts, and let

$$c_{\min} = \min_i \{Median(i)\} \text{ and } c_{\max} = \max_i \{Median(i)\}.$$

This property is formulated in the following theorem:

Theorem 2. *The quality function $W : \{c_1, \ldots, c_N\} \to \mathbb{N}$ defined over the set of cuts increases in $\{c_1, \ldots, c_{\min}\}$ and decreases in $\{c_{\max}, \ldots, c_N\}$. Hence,*

$$c_{\text{Best}} \in \{c_{\min}, \ldots, c_{\max}\}.$$

This property is interesting because one can use only $O(d \log N)$ queries to determine the medians of decision classes by using the binary search algorithm. Hence, the tail cuts can be eliminated by using using $O(d \log N)$ SQL queries. Let us also observe that if all decision classes have similar medians, then almost all cuts can be eliminated.

Example. We consider a data table consisting of 12,000 records. Objects are classified into three decision classes with the distribution $\langle 5000, 5600, 1400 \rangle$, respectively. One real value attribute has been selected, and $N = 500$ cuts on its domain have generated class distributions as shown in Fig. 6. The medians of the three decision classes are c_{166}, c_{414} and c_{189}, respectively. The median of every decision class has been determined by a *binary search algorithm* using $\log N = 9$ simple queries. Applying Theorem 2, we conclude that it is enough to consider only cuts from $\{c_{166}, \ldots, c_{414}\}$. In this way, 251 cuts have been eliminated by using only 27 simple queries.

4.2 Divide and Conquer Strategy

The main idea is to apply the *divide and conquer* strategy to determine the best cut $c_{\text{Best}} \in \{c_1, \ldots, c_n\}$ with respect to a given quality function.

First, we divide the set of possible cuts into k intervals, where k is a predefined parameter ($k \geq 2$). Then, we choose the interval to which the best cut may belong with the highest probability. We will use some approximating measures to predict the interval that probably contains the best cut with respect to the discernibility measure. This process is repeated until the interval considered consists of one cut. Then, the best cut can be chosen among all visited cuts.

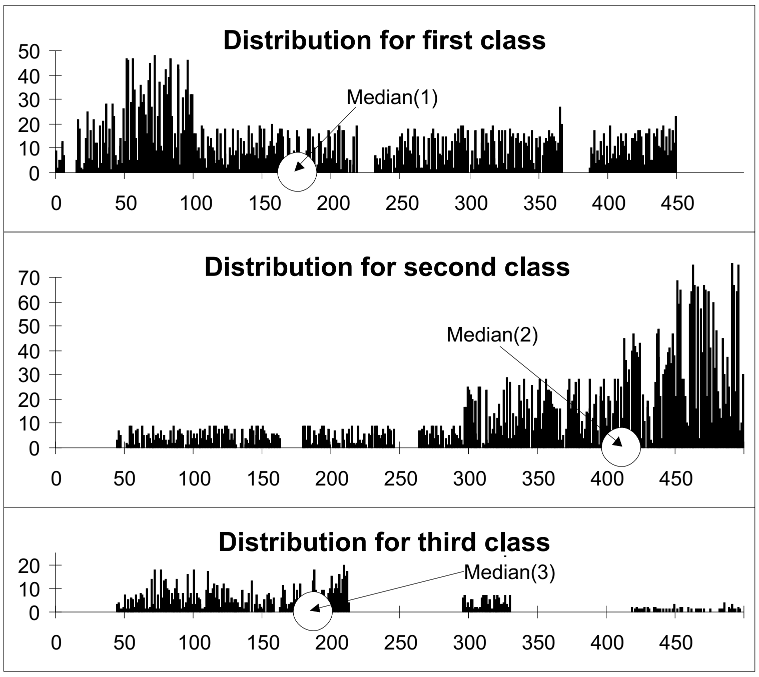

Fig. 6. Distributions for decision classes 1, 2, and 3

The problem arises how to define the measure evaluating the quality of the interval $[c_L; c_R]$ having class distributions: $\langle L_1, \dots, L_d \rangle$ in $(-\infty; c_L)$, $\langle M_1, \dots, M_d \rangle$ in $[c_L; c_R)$, and $\langle R_1, \dots, R_d \rangle$ in $[c_R; \infty)$. This measure should estimate the quality of the best cut among those belonging to the interval $[c_L; c_R]$. In the next section, we present some theoretical considerations about the quality of the best cut in $[c_L; c_R]$. These results will be used to construct the relevant measure to estimate the quality of the whole interval.

Evaluation Measures In our previous papers (see [18]), we have proposed the following measures to estimate the quality of the best cut in $[c_L; c_R]$:

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

where the value of Δ is defined by

$$\Delta = \frac{[W(c_R) - W(c_L)]^2}{8 \cdot conflict([c_L; c_R])} \quad \text{(in the dependent model)},$$

$$\Delta = \alpha \cdot \sqrt{D^2(W(c))} \qquad \text{for some } \alpha \in [0; 1] \quad \text{(in the independent model)}.$$

The choice of Δ and the value of parameter α from the interval $[0;1]$ can be tuned in a learning process or are given by an expert.

4.3 Local and Global Search

We present two searching strategies for the best cut, using formula (3), called the *local* and the *global search* strategies, respectively. Using a local search algorithm, first, we discover the best cuts on every attribute separately. Next, we compare all of the local best cuts to find the global best one. The details of the local algorithm can be described as follows:

```
ALGORITHM 1: Searching for semioptimal cut
PARAMETERS: $k \in \mathbb{N}$ and $\alpha \in [0;1]$.
INPUT: attribute $a$; the set of candidate cuts $\mathbf{C}_a = \{c_1,\dots,c_N\}$ on $a$;
OUTPUT: The optimal cut $c \in \mathbf{C}_a$

begin
 $Left \leftarrow \min$; $Right \leftarrow \max$;      {see Theorem 2}
 while ($Left < Right$)
   1.Divide $[Left;Right]$ into $k$ intervals of equal length defined by
     $(k+1)$ boundary points
                 $p_i = Left + i * \frac{Right - Left}{k}$;
     for $i = 0,\dots,k$.
   2.For $i = 1,\dots,k$ compute $Eval([c_{p_{i-1}};c_{p_i}],\alpha)$ using Formula (3). Let
     $[p_{j-1};p_j]$ be the interval with the maximal value of $Eval(.)$;
   3.$Left \leftarrow p_{j-1}$; $Right \leftarrow p_j$;
 endwhile;
 return the cut $c_{Left}$;
end
```

One can see that to determine the value $Eval([c_L;c_R])$, we need only $O(d)$ simple SQL queries of the form:

```
    SELECT COUNT
    FROM data_table
    WHERE attribute BETWEEN c_L AND c_R.
```

Hence the number of SQL queries necessary for running our algorithm is of the order $O(dk\log_k N)$. We can set $k = 3$ to minimize the number of queries, because the function $f(k) = dk\log_k N$ takes the 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 the predefined parameter $m < k$. The modified algorithm needs more — but still of order $O(\log N)$ —

simple questions only.

The global strategy is searching for the best cut over all attributes. At the beginning, the best cut can belong to every attribute; hence, for each attribute, we keep the interval in which the best cut can be found (see Theorem 2), i.e., we have a collection of all potential intervals:

$$\textbf{Interval\_List} = \{(a_1, l_1, r_1), (a_2, l_2, r_2), \ldots, (a_k, l_k, r_k)\}.$$

Next, we iteratively run the following procedure:

- Remove the interval $I = (a, c_L, c_R)$ having the highest probability of containing the best cut [using formula (3)].
- Divide interval I into smaller ones $I = I_1 \cup I_2 \ldots \cup I_k$.
- Insert $I_1, I_2, \ldots, I_k$ into **Interval_List**.

This iterative step can be continued until we have a one-element interval or the time limit of the searching algorithm is exhausted. This strategy can be simply implemented by using a priority queue to store the set of all intervals, where the priority of intervals is defined by formula (3).

4.4 Example

In Fig. 7, we show the graph of $W(c_i)$ for $i \in \{166, \ldots, 414\}$, and we illustrate the outcome of the application of our algorithm to the reduced set of cuts for $k = 2$ and $\Delta = 0$.

First, the cut c_{290} is chosen, and it is necessary to determine to which of the intervals $[c_{166}, c_{290}]$ and $[c_{290}, c_{414}]$ the best cut belongs. The values of function $Eval$ on these intervals is computed:

$$Eval([c_{166}, c_{290}]) = 23927102, \qquad Eval([c_{290}, c_{414}]) = 24374685.$$

Hence, the best cut, it is predicted, belongs to $[c_{290}, c_{414}]$, and the search process is reduced to the interval $[c_{290}, c_{414}]$. The above procedure is repeated recursively until the selected interval consists only of a single cut. For our example, the best cut c_{296} has been successfully selected by our algorithm. In general, the cut selected by the algorithm is not necessarily the best. However, numerous experiments on different large data sets have shown that the cut c^* returned by the algorithm is close to the best cut c_{Best} (i.e., $\frac{W(c^*)}{W(c_{\text{Best}})} \cdot 100\%$ is about 99.9%).

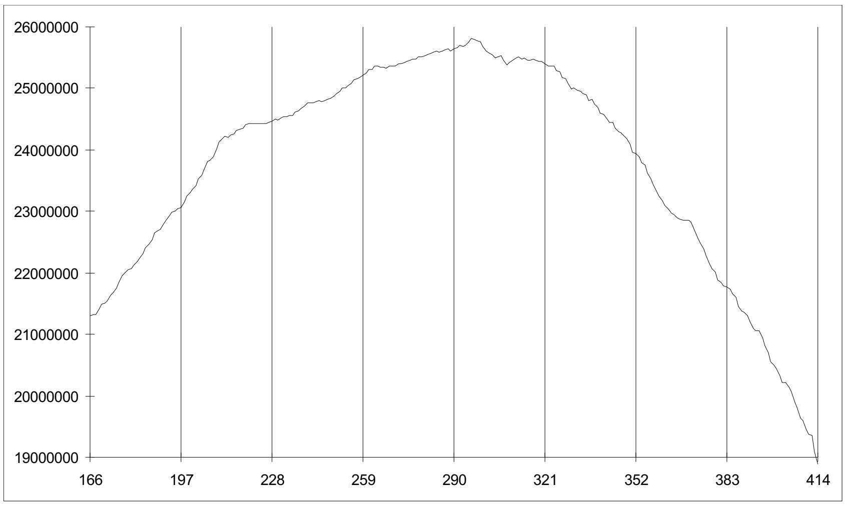

Fig. 7. Graph of $W(c_i)$ for $i \in \{166, \ldots, 414\}$

4.5 Searching for Soft Cuts

One can modify Algorithm 1 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 Algorithm 1, the current interval $[Left; Right]$ is divided equally into k smaller intervals, and the best smaller interval is 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 returns the interval $[c_L; c_R]$ if the following conditions hold:

- The class distribution in $[c_L; c_R]$ is stable, i.e., there is no subinterval of $[c_L; c_R]$ that 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 many objects (because the large number of uncertain objects can result in a larger decision tree that prolongs the time of decision tree construction).

These conditions can be controlled by some parameters. Now we are describing the stability measure for intervals.

5 Conclusions

The problem of optimal binary partition of a continuous attribute domain for large data sets stored in *relational databases* has been investigated. We show that one can reduce the number of simple queries from $O(N)$ to $O(\log N)$ to construct a partition very close to the optimal one. We plan to extend these results to other measures.

Acknowledgments

This paper has been partially supported by Polish State Committee of Research (KBN) grant No 8T11C02519, a grant of the Wallenberg Foundation, and a British Council/KBN grant "Uncertainty Management in Information Systems: Foundations and Applications of Non-Invasive Methods (1999–2001)."

References

1. J. Catlett. On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff, editor, *Proceedings of the European Working Session on Learning (EWSL'91)*, LNAI 482, 164–178, Springer, Berlin, 1991.
2. M.R. Chmielewski, J.W. Grzymala–Busse. Global discretization of attributes as preprocessing for machine learning. In T. Y. Lin, A. M. Wildberger, editors, *Soft Computing: Rough Sets, Fuzzy Logic Neural Networks, Uncertainty Management, Knowledge Discovery*, 294–297, Simulation Councils, San Diego, CA, 1995.
3. J. Dougherty, R. Kohavi, M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, 194–202, Morgan Kaufmann, San Francisco, 1995.
4. D. Dubois, H. Prade, R.R. Yager, editors. *Readings in Fuzzy Sets for Intelligent Systems*. Morgan Kaufmann, San Francisco, 1993.
5. U.M. Fayyad, K.B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8: 87–102, 1992.
6. U.M. Fayyad, K.B. Irani. The attribute selection problem in decision tree generation. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI 1992)*, 104–110, MIT Press, Cambridge, MA, 1992.
7. S.J. Hong. Use of contextual information for feature ranking and discretization. *IEEE Transactions on Knowledge and Data Engineering*, 9(5): 718–730, 1997.
8. G.H. John, P. Langley. Static vs. dynamic sampling for data mining. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDDM'96)*, 367–370, AAAI Press, Menlo Park, CA, 1996.
9. R. Kerber. Chimerge. Discretization of numeric attributes. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*, 123–128, MIT Press, Cambridge, MA, 1992.
10. H. Liu, R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence (TAI'95)*, 388–391, IEEE Press, Washington, DC, 1995.
11. S. Murthy, S. Kasif, S. Saltzberg, R. Beigel. OC1: Randomized induction of oblique decision trees. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, 322–327, MIT Press, Cambridge, MA, 1993.
12. H.S. Nguyen, A. Skowron. Quantization of real value attributes. In P. P. Wang, editor, *Proceedings of the 2nd Annual Joint Conference on Information Sciences (JCIS'95)*, 34–37, Wrightsville Beach, NC, 1995.

13. S.H. Nguyen, H.S. Nguyen. From optimal hyperplanes to optimal decision tree. In *Proceedings of the 4th International Workshop on Rough Sets, Fuzzy Sets and Machine Discovery (RSFD'96)*, 82–88, Tokyo, 1998.
14. H.S. Nguyen, S.H. Nguyen, A. Skowron. Searching for features defined by hyperplanes. In Z. W. Raś, M. Michalewicz, editors, *Proceedings of the 9th International Symposium on Methodologies for Information Systems (ISMIS'96)*, LNAI 1079, 366–375, Springer, Berlin, 1996.
15. H.S. Nguyen. Discretization methods in data mining. In L. Polkowski, A. Skowron, editors, *Rough Sets in Knowledge Discovery 1*, 451–482, Physica, Heidelberg, 1998.
16. H.S. Nguyen, A. Skowron. Boolean reasoning for feature extraction problems. In Z. W. Raś, A. Skowron, editors, *Proceedings of 10th International Symposium on the Foundations of Intelligent Systems (ISMIS'97)*, LNAI 1325, 117–126, Springer, Heidelberg, 1997.
17. H.S. Nguyen, S.H. Nguyen. From optimal hyperplanes to optimal decision trees. *Fundamenta Informaticae*, 34(1/2): 145–174, 1998.
18. H.S. Nguyen. Efficient SQL-querying method for data mining in large databases. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, 806–811, Morgan Kaufmann, San Francisco, 1999.
19. H.S. Nguyen. On efficient construction of decision tree from large databases. In *Proceedings of the 2nd International Conference on Rough Sets and Current Trends in Computing (RSCTC 2000)*, LNAI 2005, 316–323, Springer, Berlin, 2000.
20. Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer, Dordrecht, 1991.
21. L. Polkowski, A. Skowron, editors. *Rough Sets in Knowledge Discovery*, Vol. 1,2. Physica, Heidelberg, 1998.
22. J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, 1993.
23. A. Skowron, C. Rauszer. The discernibility matrices and functions in information systems. In R. Słowiński, editor, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, 311–362, Kluwer, Dordrecht, 1992.
24. J. Komorowski, Z. Pawlak, L. Polkowski, A. Skowron. Rough sets: A tutorial. In S. K. Pal and A. Skowron, editors, *Rough–Fuzzy Hybridization: A New Trend in Decision Making*, 3–98, Springer, Singapore, 1998.
25. W. Ziarko. Rough sets as a methodology in data mining. In L. Polkowski, A.Skowron, editors, *Rough Sets in Knowledge Discovery 1*, 554–576, Physica, Heidelberg, 1998.