A soft decision tree

Hung Son Nguyen

Institute of Mathematics,
Warsaw University,
Banacha 2, Warsaw 02095, Poland

Abstract. Searching for binary partition of attribute domains is an important task in Data Mining, particularly in decision tree methods. The most important advantage of decision tree methods are based on compactness and clearness of presented knowledge and high accuracy of classification. In case of large data tables, the existing decision tree induction methods often show to be inefficient in both computation and description aspects. The disadvantage of standard decision tree methods is also their instability, i.e., small deviation of data perhaps cause a total change of decision tree. We present the novel "soft discretization" methods using "soft cuts" instead of traditional "crisp" (or sharp) cuts. This new concept allows to generate more compact and stable decision trees with high classification accuracy. We also present an efficient method for soft cut generation from large data bases.

Key words: Data Mining, Rough set, Decision tree, Soft Discretization.

1 Introduction

The main step in methods of decision tree construction is to fine optimal partitions of the set of objects. The problem of searching for optimal partitions of real value attributes, defined by so called cuts, has been studied by many authors (see e.g. [1,2,10],[4]), where optimization criteria are defined by e.g. height of the obtained decision tree, the number of cuts or the classification accuracy of decision tree on new unseen objects. In general, all those problems are hard from 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 approximate measures estimating the quality of extracted cuts. In rough set and Boolean reasoning based methods, the quality is defined by the number of pairs of objects discerned by the partition.

One can mention some problems occurring in existing decision tree induction algorithms. The first is related to efficiency of searching for optimal partition of real value attributes assuming that the large data table is represented in relational data base. Using straightforward approach to the 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 pre-assumed partitions of the searching space. In such case, even the linear complexity is not acceptable because of the time needed for one step. The critical factor for time complexity of algorithms solving the discussed problem is the number of simple SQL queries like *SELECT COUNT FROM ... WHERE attribute BETWEEN ...* (related to some interval of attribute values) necessary to construct such partitions. Moreover, the existing (traditional) methods are using crisp conditions for object discerning. This can lead to misclassification of new objects which are, e.g., close to the separating boundary, and this fact can provide to low quality of new object classification.

The first problem is a big challenge for data mining researchers. Almost all existing methods are based on sampling technique, i.e., building a decision tree for small, randomly chosen subset of data, and then evaluate the quality of decision tree for whole data [3]. If the quality of generated decision tree is not sufficient enough, we have to repeat this step for new sample.

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 tree.* The new approach leads to new efficient strategies in searching for the best cuts (both soft and crisp cuts) using the whole data. We show some properties of considered optimization measures allowing to reduce the size of searching space. Moreover, we prove that using only $O(\log N)$ simple queries, one can construct the partition very close to optimal.

2 Basic notions

An *information system* [8] is a pair $\mathbb{A} = (U, A)$, where U is a non-empty, finite set called the *universe* and A is a non-empty finite set of *attributes*, 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*. 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 *decision table* where $dec \notin A$ is called *decision.* Without loss of generality we assume that $V_{dec} = \{1, \ldots, d\}$. Then the set $DEC_k = \{x \in U : dec(x) = k\}$ will be called the k^{th} *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 *"the 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 Decision tree construction from Decision tables

The decision tree for a given decision table is (in simplest case) a binary directed tree with *test functions* (i.e. Boolean functions defined on the information vectors of objects) labeled in internal nodes and decision values labeled in leaves. In this paper, we consider decision trees using cuts as test functions. Every cut (a, c) is associated with 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 $a(u) > c$. Every decision tree can be treated as a decision algorithm. The new object $u \in U$ can be classified by a given decision tree as follows: "*We start from root of 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 will be continued for any node until we reach any external node." The decision tree is called *consistent* with the decision table

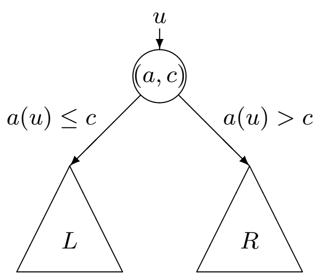

Fig. 1. Decision tree approach.

$\mathbb{A}$ if it classifies properly all objects from $\mathbb{A}$. The decision tree is called *optimal* with $\mathbb{A}$ if it has a smallest height among decision tree consistent with $\mathbb{A}$. The cut c on attribute a is called *optimal cut* if (a, c) labels one of internal nodes of optimal decision trees. Developing some decision tree induction methods [2,10] we should often solve the following problem: "*For a given set of candidate cuts $\{c_1, ..., c_N\}$ on an attribute a, find a cut c_i belonging to the set of optimal cuts with highest probability*". Usually, we use some *measure (or quality functions)* $F : \{c_1, ..., c_N\} \to \mathbb{R}$ to estimate the quality of cuts. For a given measure F, the *straightforward algorithm* should compute the values of F for all cuts: $F(c_1), .., F(c_N)$. The cut c_{Best} which optimizes the value of function F is selected as the result of searching process. 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.

We consider the set of all relevant cuts $\mathbf{C}_a = \{c_1, ..., c_N\}$ on an attribute a.

Definition 1. The d-tuple of integers $\langle x_1, .., x_d \rangle$ is called class distribution of the set of objects $X \subset U$ iff $x_k = card(X \cap DEC_k)$ for $k \in \{1, ..., d\}$. 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 is called **the class distribution in** $[p; q)$.

In next sections we recall the most frequently used measures for decision tree induction like *"Entropy Measure" and "Discernibitity Measure"*, respectively.

Entropy measure 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 $\langle N_1, ..., N_d \rangle$, where $N_1 + ... + N_d = N$, 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 Figure 2. The methods based on information entropy are reported in [2,10].

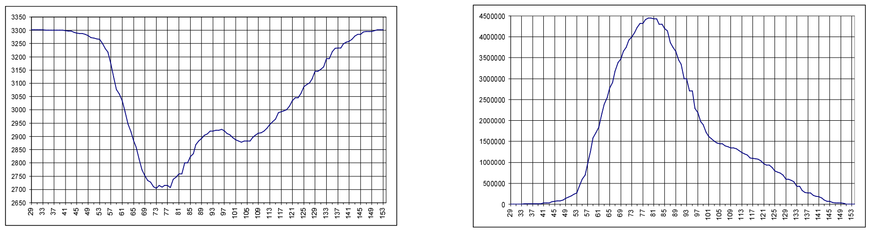

Fig. 2. The illustration of Entropy measure (left) discernibility measure(right) for the same data set. On horizontal axes: ID of consequent cuts; on vertical axes: values of corresponding measures

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 $\langle N_1, ..., N_d \rangle$ be a class distribution of X, then $conflict(X)$ can be computed by $conflict(X) = \sum_{i<j} N_i N_j$. 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 more is number of pairs of objects discerned by the cut (a, c), the larger is chance that c can be chosen to the optimal set of cut. This algorithm is called Maximal-Discernibility heuristics or *the MD-heuristics* for decision tree construction. Figures 2 illustrate the values of Entropy and Discernibility functions over set of possible cuts on one of 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 measure and their comparison with Entropy-based methods has been reported in [4].

2.2 Soft cuts

We have presented so far decision tree methods working with cuts treated as sharp classifiers such, that real values are partitioned by them into disjoint intervals. One can observe that in some situations objects which are close one to other, can be treated as very different. In this section we introduce some notions of *soft cuts* which discern two given values if those values are far enough from the cut. The formal definition of soft cuts is following:

Definition 2. 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 $(l \leq r)$; 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.

2.3 Soft Decision Tree

The test functions defined by cuts can be Here we propose two strategies being modifications of that method by using described above soft cuts (fuzzy separated cuts). They are called *fuzzy decision tree* and *rough decision tree.*

In fuzzy decision tree method instead of checking the condition $a(u) > c$ we have to check how strong is 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.

In the case of rough decision tree, 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 children of considered node. We have to compare their answers taking into account the numbers of supported by them objects. The answer with most number of supported objects is a decision of given object.

3 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. There are modifications of our MD-heuristic described in previous sections. We describe some techniques which have been presented in previous papers (see [5]).

3.1 Tail cuts can be eliminated

In [5] we have shown the following techniques for irrelevant cut eliminating. For given set of candidate cuts $\mathbf{C}_a = \{c_1, .., c_N\}$ on a such that $c_1 < c_2... < c_N$, by median of the k^{th} decision class we mean the cut $c \in \mathbf{C}_a$ which minimizes the value $|L_k - R_k|$. The median of the k^{th} decision class will be denoted by $Median(k)$. Let $c_{min} = \min_i\{Median(i)\}$ and $c_{max} = \max_i\{Median(i)\}$. We have shown in [5] the following theorem.

Theorem 1 *The quality function* $W : \{c_1,..,c_N\} \rightarrow \mathbb{N}$ *defined over the set of cuts is increasing in* $\{c_1, ..., c_{min}\}$ *and decreasing in* $\{c_{max}, ..., c_N\}$. *Hence* $c_{Best} \in \{c_{min}, ..., c_{max}\}$

This property is interesting because one can use only $O(d \log N)$ queries to determine the medians of decision classes by using 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.

3.2 Divide and Conquer Strategy

The main idea is to apply the *"divide and conquer"* strategy to determine the best cut $c_{Best} \in \{c_1, ..., c_n\}$ 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 belong with the highest probability. We will use some approximating measures to predict the interval which probably contains the best cut with respect to discernibility measure. 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 how to define the measure evaluating the quality of the interval $[c_L; c_R]$ having class distributions: $\langle L_1, ..., L_d\rangle$ in $(-\infty; c_L)$; $\langle M_1, ..., M_d\rangle$ in $[c_L; c_R)$; and $\langle R_1, ..., R_d\rangle$ 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 previous papers (see [5,7]) we proposed the following measures to estimate the quality of the best cut in $[c_L; c_R]$

$$Eval\left([c_L; c_R]\right) = \frac{W(c_L) + W(c_R) + conflict([c_L; c_R])}{2} + \Delta \tag{1}$$

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 $[0; 1]$ can be tuned in learning process or are given by expert. One can see that 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.
```

3.3 Local and Global Search

We present two strategies of searching for the best cut using formula 1 called *local* and *global search*. In local search algorithm, first we discover the best cuts on every attribute separately. Then we compare them to find out the global best one. The details of local algorithm can be described as follows:

```
ALGORITHM 1: Searching for semi-optimal cut
PARAMETERS: $k \in \mathbb{N}$ and $\alpha \in [0; 1]$.
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
  $Left \leftarrow$ min; $Right \leftarrow$ max;    {see Theorem 1}
  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 $Eval([c_{p_{i-1}}; c_{p_i}], \alpha)$ using Formula (1). 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
```

Hence the number of queries necessary for running our algorithm is of order $O(dk \log_k N)$. In practice we set $k = 3$ because the function $f(k) = dk \log_k N$ has minimum over positive integers for $k = 3$. For $k > 2$, instead 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.

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 1), i.e. we have a collection of all potential intervals

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

Next we iteratively run the following procedure

- remove the interval $I = (a, c_L, c_R)$ having highest probability of containing the best cut (using Formula 1);
- divide interval I into smaller ones $I = I_1 \cup I_2 ... \cup I_k$;
- insert $I_1, I_2, ..., I_k$ to **Interval_Lists**.

This iterative step can be continued until we have one–element interval or the time limit of searching algorithm is exhausted. This strategy can be simply implemented using priority queue to store the set of all intervals, where priority of intervals is defined by Formula 1.

3.4 Example

We consider a randomly generated data table consisting of 12000 records. Objects are classified into 3 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 has generated class distributions as shown in Figure 3. The medians of three decision classes are c_{166}, c_{414} and c_{189}, respectively. The median of every decision class has been determined by *binary search algorithm* using $\log N = 9$ simple queries. Applying Theorem 1 we conclude that it is enough to consider only cuts from $\{c_{166}, ..., c_{414}\}$. Thus 251 cuts have been eliminated by using 27 simple queries only.

In Figure 3 we show the graph of $W(c_i)$ for $i \in \{166, ..., 414\}$ and we illustrated the outcome of application of our algorithm to the reduce 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: $Eval([c_{166}, c_{290}]) = 23927102$, $Eval([c_{290}, c_{414}]) = 24374685$. Hence, the best cut is predicted to belong 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 of single cut only. 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 shown that the cut c^* returned by the algorithm is close to the best cut c_{Best} (i.e. $\frac{W(c^*)}{W(c_{Best})} \cdot 100\%$ is about 99.9%).

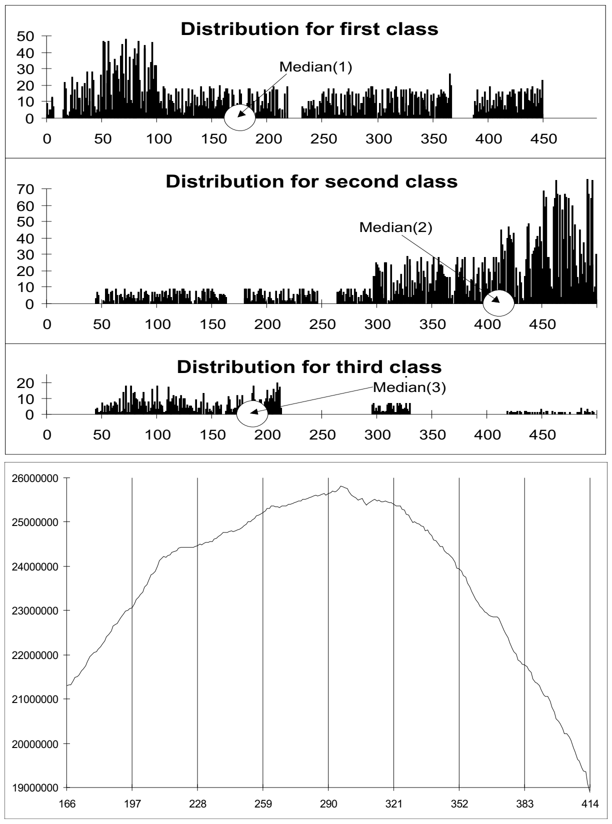

Fig. 3. Distributions for decision classes 1, 2, 3 and graph of $W(c_i)$ for $i \in \{166, .., 414\}$

3.5 Searching for soft cuts

One can modify the Algorithm 1 presented in the previous section to determine "soft cuts" in large data bases. 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 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 $[c_L; c_R]$ him self;
- 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 prolongs the time of decision tree construction)

4 Conclusions

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

Acknowledgement: This paper has been partially supported by Polish State Committee of Research (KBN) grant No 8T11C02519 and grant of the Wallenberg Foundation.

References

1. Dougherty J., Kohavi R., Sahami M.: Supervised and unsupervised discretization of continuous features. In. Proc. of the 12th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1995, pp. 194–202.
2. Fayyad, U. M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8**, 1992, pp. 87–102.
3. John, G. H., Langley, P.: Static vs. dynamic sampling for data mining. Proceedings of the Second International Conference of Knowledge Discovery and Data Mining, Portland, AAAI Press 1996, pp. 367–370.
4. Nguyen, H. S.: Discretization Methods in Data Mining. In L. Polkowski, A. Skowron (Eds.): *Rough Sets in Knowledge Discovery* **1**, Springer Physica-Verlag, Heidelberg, 1998, pp. 451–482.
5. Nguyen, H. S.: Efficient SQL-Querying Method for Data Mining in Large Data Bases. Proc. of 16th International Joint Conference on Artificial Intelligence, IJCAI-99, Morgan Kaufmann Publishers, Stockholm, Sweden, 1999, pp. 806-811.
6. Nguyen, H. S.: 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, pp. 316-323.
7. Nguyen, H. S.: On Efficient Handling of Continuous Attributes in Large Data Bases, Fundamenta Informatica **48(1)**, pp. 61-81
8. Pawlak Z.: *Rough sets: Theoretical aspects of reasoning about data*, Kluwer Dordrecht, 1991.
9. Polkowski, L., Skowron, A. (Eds.): *Rough Sets in Knowledge Discovery* **Vol. 1,2**, Springer Physica-Verlag, Heidelberg, 1998.
10. Quinlan, J. R. *C4.5. Programs for machine learning.* Morgan Kaufmann, San Mateo CA, 1993.
11. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In. R. Słowiński (ed.). Intelligent Decision Support – Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic Publishers, Dordrecht, 1992, pp. 311–362