Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 368 – 377

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

Minimization of decision tree average depth for decision tables with many-valued decisions

Mohammad Azad*, Mikhail Moshkov

Computer, Electrical & Mathematical Sciences & Engineering Division
King Abdullah University of Science and Technology
Thuwal 23955-6900, Saudi Arabia

Abstract

The paper is devoted to the analysis of greedy algorithms for the minimization of average depth of decision trees for decision tables such that each row is labeled with a set of decisions. The goal is to find one decision from the set of decisions. When we compare with the optimal result obtained from dynamic programming algorithm, we found some greedy algorithms produces results which are close to the optimal result for the minimization of average depth of decision trees.

Peer-review under responsibility of KES International.
Keywords: Many-valued decisions; Optimization; Decision Tree; Dynamic Programming; Greedy Algorithm

1. Introduction

Many-valued decision table (MVDT) is a decision table where each row is labeled with a set of decisions. It is common to have such tables in our real life because of incomplete information of the domain. Furthermore, it is natural to have such data sets in optimization problems such as finding a Hamiltonian circuit with the minimum length in the traveling salesman problem, finding nearest post office in the post office problem[1]; in this case we give input with more than one optimal solutions. It also arises when we study problem of semantic annotation of images[2], music categorization into emotions[3], functional genomics[4], and text categorization[5].

Many-valued decision tables are often considered for the problems of classification and prediction[6,7,8,9,10], however, in this paper our aim is to study decision trees for MVDT in case of knowledge representation, and as algorithms for problem solving.

We studied a greedy algorithm for construction of decision trees for MVDT using the heuristic based on the number of boundary subtables[11,12]. Besides, we have studied this algorithm in the cases of most common decision,

* Corresponding author.
E-mail address: mohammad.azad@kaust.edu.sa

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.117

and generalized decision approaches in the paper[13] (in that paper, we considered decision tables with one-valued decisions as MVDT where sets of decisions attached to rows have one element).

This paper is a continuation of the conference publication[13]. In this paper, we have introduced some greedy heuristics 'misclassification error', 'absent', 'combined' whose performances are as good as the previous one. We also adapt 'many-valued entropy', 'sorted entropy' heuristics from literature. Our aim is to compare the performance among these algorithms. We have done experiments using modified data sets from UCI Machine Learning Repository[14]. Based on the results of the experiments, we have presented rankings among the algorithms in the form of critical difference diagram[15]. Furthermore, we have shown the average relative difference between greedy and optimal results to describe how close they are. Hence, our approach is efficient enough to choose one greedy algorithm which produce results close to the optimal.

This paper consists of six sections. Section 2 contains the related background study of this problem. In Sect. 3 contains the important definitions related to our study. We present the dynamic and greedy algorithms for construction of decision trees in Sect. 4, and we compare among algorithms using Friedman with the corresponding post-hoc test in Sect. 5. Section 6 contains results of experiments and Sect. 7 concludes the paper.

2. Related work

In literature, often, problems that are connected with multi-label data are considered for classification: multi-label learning[16], multi-instance learning[10] etc. In multi-label learning, the output for each instances can be a set of decisions, whereas in our framework, we choose only one decision as output for each instance. In multi-instance learning, bag of instances are labeled rather than individual example which is far away from our problem. There is also semi-supervised learning[17] where some examples are labeled but some are not, but we deal with examples that are labeled with multiple decisions.

Furthermore, some learning problems deal with many-valued data sets in different ways such as partial learning[18], ambiguous learning[19], and multiple label learning[20]. The difference is that we consider all labels that are attached to an object are correct, whereas, these problems considers only one label is correct and others are incorrect. Additionally, these papers only focus on classification results rather than optimization of data model.

In this paper, we consider the problem from the point of view of knowledge representation and optimization of data model. Therefore, our goal is to choose a data model which will be optimized and will give us one decision from the set of decision attached to each row.

3. Main definitions

3.1. Many-valued decision table (MVDT)

A *many-valued decision table (MVDT)*, T is a rectangular table filled by nonnegative integers. Columns of this table are labeled with conditional attributes $f_1, \ldots, f_n$. If we have strings as values of attributes, we have to encode the values as nonnegative integers. We do not have any duplicate rows, and each row is labeled with a nonempty finite set of natural numbers (set of decisions). We denote by $N(T)$ the number of rows in the table T. We denote a row by r_i where $i = 1, \ldots, N(T)$. For example, r_1 means first row, r_2 means second rows and so on.

Table 1. A decision table with many-valued decisions T^0

$T^0 =$

	f_1	f_2	f_3	
r_1	0	0	0	{1}
r_2	0	1	1	{1,2}
r_3	1	0	1	{1,3}
r_4	1	1	0	{2,3}
r_5	0	0	1	{2}

If there is a decision which belongs to the set of decisions attached to each row of T, then we call it a *common decision* for T. We will say that T is a *degenerate* table if T is empty or it has a common decision. For example, T' is degenerate table as shown in Table 2, where the common decision is 1.

Table 2. A degenerate decision table with many-valued decisions T'

$T' =$

	f_1	f_2	f_3	
r_1	0	0	0	{1}
r_2	0	1	1	{1,2}
r_3	1	0	1	{1,3}

A table obtained from T by removing some rows is called a subtable of T. There is a special type of subtable called *boundary subtable*. The subtable T' of T is a *boundary subtable* of T if and only if T' is not degenerate but its proper subtable is degenerate. We denote by $nBS(T)$ the number of boundary subtables of the table T. Below is an example of all boundary subtables of T_0:

$T_1 =$

	f_1	f_2	f_3	d
r_2	0	1	1	$\{1,2\}$
r_3	1	0	1	$\{1,3\}$
r_4	1	1	0	$\{2,3\}$

$T_2 =$

	f_1	f_2	f_3	d
r_1	0	0	0	$\{1\}$
r_4	1	1	0	$\{2,3\}$

$T_3 =$

	f_1	f_2	f_3	d
r_3	1	0	1	$\{1,3\}$
r_5	0	0	1	$\{2\}$

$T_4 =$

	f_1	f_2	f_3	d
r_1	0	0	0	$\{1\}$
r_5	0	0	1	$\{2\}$

We denote by $T(f_{i_1}, a_1), \ldots, (f_{i_m}, a_m)$ a *subtable* of T which consists of rows that at the intersection with columns $f_{i_1}, \ldots, f_{i_m}$ have values $a_1, \ldots, a_m$. Such nonempty subtables (including the table T) are called separable subtables of T. For example, if we consider subtable $T^0(f_1, 0)$ for table T (see Table 1), it consists of rows $1, 2$, and 5. Similarly, $T^0(f_1, 0)(f_2, 0)$ subtable consists of rows 1, and 5.

Table 3. Example of subtables of decision table with many-valued decisions T^0

$T^0(f_1, 0) =$

	f_1	f_2	f_3	
r_1	0	0	0	{1}
r_2	0	1	1	{1,2}
r_5	0	0	1	{2}

,

$T^0(f_1, 0)(f_2, 0) =$

	f_1	f_2	f_3	
r_1	0	0	0	{1}
r_5	0	0	1	{2}

We denote by $E(T)$, the set of attributes (columns of table T), such that each of them has different values. For example, if we consider table T^0, $E(T^0) = \{f_1, f_2, f_3\}$. Similarly, $E(T^0(f_1, 0)) = \{f_2, f_3\}$ for the subtable $T^0(f_1, 0)$, because the value for the attribute f_1 is constant (=0) in subtable $T^0(f_1, 0)$. For $f_i \in E(T)$, we denote by $E(T, f_i)$ the set of values from the column f_i. As an example, if we consider table T^0 and attribute f_1, then $E(T^0, f_1) = \{0, 1\}$.

The minimum decision which belongs to the maximum number of sets of decisions attached to rows of the table T is called the *most common decision* for T. For example, the most common decision for table T^0 is 1. Even though both 1 and 2 appears 3 times in the sets of decisions, but 1 is the minimum decision, so we choose 1 as the most common decision. We denote by $N_{mcd}(T)$ the number of rows for which the set of decisions contains the most common decision for T. For the table T^0, $N_{mcd}(T^0) = 3$.

3.2. Decision tree

A *decision tree* over T is a finite tree with root in which each terminal node is labeled with a decision (a natural number), and each nonterminal node is labeled with an attribute from the set $\{f_1, \ldots, f_n\}$. A number of edges start from each nonterminal node which are labeled with the values of that attribute (e.g. two edges labeled with 0 and 1 for the binary attribute) .

Let Γ be a decision tree over T and v be a node of Γ. There is one to one mapping between node v and subtable of T i.e. for each node v, we have a unique subtable of T. We denote $T(v)$ as a subtable of T that is mapped for a node v of decision tree Γ. If node v is the root of Γ then $T(v) = T$ i.e. the subtable $T(v)$ is the same as T. Otherwise, $T(v)$ is the subtable $T(f_{i_1}, \delta_1) \dots (f_{i_m}, \delta_m)$ of the table T where attributes $f_{i_1}, \dots, f_{i_m}$ and numbers $\delta_1, \dots, \delta_m$ are respectively nodes and edge labels in the path from the root to node v.

We will say that Γ is a decision tree for T, if for any node v of Γ

- If $T(v)$ is degenerate then v is labeled with the common decision for $T(v)$,
- If $T(v)$ is not degenerate then v is labeled with an attribute $f_i \in E(T(v))$. In this case, if $E(T(v), f_i) = \{a_1, \dots, a_k\}$, then k outgoing edges from node v are labeled with $a_1, \dots, a_k$.

It is clear that for any row r of T there exists exactly one terminal node v in Γ such that r belongs to $T(v)$. The decision attached to v will be considered as the result of the work of Γ on the row r. We denote by $l_\Gamma(r)$ the length of the path from the root of Γ to the node v. We denote by $\Delta(T)$, the set of rows of T. We denote by $h_{avg}(\Gamma)$ the average depth of Γ which is equal to $\sum_{r \in \Delta(T)} \frac{l_\Gamma(r)}{N(T)}$.

An example of a decision tree for the table T can be found in Fig. 1. If v is the node labeled with the nonterminal attribute f_3, then subtable $T(v)$ corresponding to the node v will be the subtable $T(f_1, 0)$ of table T. Similarly, the subtable corresponding to the node labeled with 2 will be $T(f_1, 0)(f_3, 0)$.

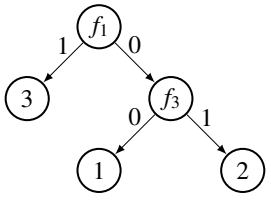

Fig. 1. A decision tree for the decision table with MVDT, T^0

3.3. Impurity functions and uncertainty measures

In greedy algorithm, we need to choose attributes to partition the decision table into smaller subtables until we get degenerate table which then be used to label the terminal node. To choose which partition to consider for tree construction, we need to evaluate the quality of partition. Impurity function is the criterion for the evaluation of quality of partition. We assume that, the smaller the impurity function value, the better is the quality of partition. We can calculate impurity function based on uncertainty measure value for the considered subtables corresponding to the partitioning. The uncertainty measure evaluates the uncertainty or randomness of the considered subtable. If we have a common decision, then there is no uncertainty in the data, hence we get its value as zero, otherwise we will get nonnegative values for it. We have used six different uncertainty measures for our experiment.

3.3.1. Uncertainty measures

Uncertainty measure U is a function from the set of nonempty MVDT to the set of real numbers such that $U(T) \geq 0$, and $U(T) = 0$ if and only if T is degenerate. Let T be a decision table with many-valued decisions having n conditional attributes, and $N = N(T)$ rows. Its rows are labeled with sets containing m different decisions $d_1, \dots, d_m$. For $i = 1, \dots, m$, let N_i be the number of rows in T that has been attached with sets of decisions containing the decision d_i, and $p_i = N_i/N$. Let $d_1, \dots, d_m$ be ordered such that $p_1 \geq \dots \geq p_m$, then for $i = 1, \dots, m$, we denote by $N_i^{'}$ the number of rows in T such that the set of decisions attached to row contains d_i, and if $i > 1$ then this set does not contain $d_1, \dots, d_{i-1}$, and $p_i^{'} = N_i^{'}/N$. We have the following six uncertainty measures (we assume $0 \log_2 0 = 0$):

- Misclassification error: $ME(T) = N(T) - N_{mcd}(T)$; It measures difference between total number of rows and number of rows with most common decision.

- Sorted entropy: $entSort(T) = -\sum_{i=1}^{m} p_i' \log_2 p_i'$; First, we sort[19] the probability for each decision. After that, for each row, keep the decision having maximum probability and discard others. Then we calculate entropy for this modified decision table.
- Many-valued entropy: $entML(T) = 0$, if and only if T is degenerate, otherwise, it is equal to $-\sum_{i=1}^{m}(p_i \log_2 p_i + q_i \log_2 q_i)$, where, $q_i = 1 - p_i$; It measures entropy[21] for decision table with many-valued decisions.
- $nBS(T)$: number of boundary subtables in T; We calculate number of boundary subtables as brute force approach by checking all subtables of T.
- Absent: $abs(T) = q_1 \cdots q_m$, where $q_i = 1 - p_i$; It measures the absent probability (q_i), and multiply all q_i's.
- Combined: $comb(T) = ME(T) + B_2 + B_3$, where B_2 and B_3 are the number of boundary subtables with two and three rows, respectively. It is the combination of 'misclassification error', and 'boundary subtables' uncertainty measures.

3.3.2. Impurity functions

Let $f_i \in E(T)$, and $E(T, f_i) = \{a_1, \ldots, a_t\}$. The attribute f_i divides the table T into t subtables: $T_1 = T(f_i, a_1), \ldots, T_t = T(f_i, a_t)$. We now define an impurity function I which gives us the impurity $I(T, f_i)$ of this partition. Let us fix an uncertainty measure U from the set {*ME*, *entSort*, *entML*, *nBS*, *abs*, *comb*}, and type of impurity function from {weighted sum (*ws*), weighted max (*wm*), divided weighted sum (*Div_ws*), multiplied weighted sum (*Mult_ws*)}. Then:

- wm, $I(T, f_i) = \max_{1 \le j \le t} U(T_j) N(T_j)$
- ws, $I(T, f_i) = \sum_{j=1}^{t} U(T_j) N(T_j)$
- Div_ws, $I(T, f_i) = (\sum_{j=1}^{t} U(T_j) N(T_j)) / \log_2 t$
- Mult_ws, $I(T, f_i) = (\sum_{j=1}^{t} U(T_j) N(T_j)) \cdot \log_2 t$

Impurity function can be a weighted sum (*ws*) of values of uncertainty measure for the subtables of partitioning. It can also be weighted max (*wm*) values of uncertainty measures of the subtables. Also, we can have new types of impurity functions by dividing (*Div_ws*) or multiplying (*Mult_ws*) the weighted sum impurity types by the number of branches for a partition.

As a result, we have 24 ($= 4 \times 6$) impurity functions.

4. Algorithms for decision tree construction

In this paper, we consider dynamic programming and greedy algorithms. Dynamic programming algorithm give us optimal solution whereas greedy algorithms give us suboptimal solutions. As dynamic programming is highly time consuming, we need to choose some greedy algorithms whose performances are comparable to the optimal result.

4.1. Dynamic programming algorithm

We now describe an algorithm A_d which, for a given decision table with many-valued decisions constructs a decision tree with minimum average depth. This algorithm is based on dynamic programming approach[22,1], and the complexity of this algorithm in the worst case is exponential.

Let T contains n conditional attribute $f_1, \ldots, f_n$. We denote by $S(T)$ the set of all separable subtables of the table T including the table T. The first part of the algorithm A_d constructs the set $S(T)$ (see Algorithm 1). The second part of the algorithm A_d constructs, for each subtable from $S(T)$, a decision tree with minimum average depth (see Algorithm 2).

After that A_d returns the minimum average depth of the optimal tree which corresponds to the table T.

4.2. Greedy algorithms

Let I be an impurity function. The greedy algorithm A_I, for a given decision table with many-valued decisions T, constructs a decision tree $A_I(T)$ for T (see Algorithm 3).

It constructs decision tree sequentially in a top-down fashion. It greedily chooses one attribute at each step based on uncertainty measure and type of the impurity function. We have total 24 algorithms. The complexities of these algorithms are polynomially bounded above by the size of the table.

Algorithm 1 Construction of the set of separable subtable $S(T)$

Input: A decision table with many-valued decisions T, and conditional attributes $f_1, \ldots, f_n$.
Output: The set $S(T)$

```
Assign S(T) = {T}, and mark T as not treated;
while (true) do
  if No untreated tables in S(T) then
    Return S(T);
  else
    Choose a table T_s in S(T) which is not treated;
    if E(T_s) = ∅ then
      Mark the table T_s as treated;
    else
      Add to the set S(T) all subtables of the form T_s(f_i, δ), where f_i ∈ E(T_s), and δ ∈ E(T_s, f_i) which were not
      in S(T), mark the table T_s as treated, and new subtables T_s(f_i, δ) as untreated.
    end if
  end if
end while
```

Algorithm 2 Construction of a decision tree with minimum average depth for each table from $S(T)$

Input: A decision table with many-valued decisions T, and conditional attributes $f_1, \ldots, f_n$, and the set $S(T)$.
Output: Decision tree $A_d(T)$ for T.

```
while (true) do
  if T has been assigned a decision tree then
    Return tree A_d(T);
  else
    Choose a table T_s in the set S(T) which has not been assigned a tree yet and which is either degenerate or all
    separable subtables of the table T_s already have been assigned decision trees.
    if T_s is degenerate then
      Assign to the table T_s the decision tree consisting of one node. Mark this node with the common decision
      for T_s;
    else
      For each f_i ∈ E(T_s) and each δ ∈ E(T_s, f_i), we denote by Γ(f_i, δ), the decision tree assigned to the table
      T_s(f_i, δ). We now define a decision tree Γ_{f_i} with a root labeled by the attribute f_i where f_i ∈ E(T_s), and
      E(T_s, f_i) = {δ_1, ..., δ_r}. The root has exactly r edges d_1, ..., d_r which are labeled by the numbers δ_1, ..., δ_r,
      respectively. The roots of the decision trees Γ(f_i, δ_1), ..., Γ(f_i, δ_r) are ending points of the edges d_1, ..., d_r,
      respectively. Assign to the table T_s one of the trees Γ_{f_i}, f_i ∈ E(T_s), having minimum average depth for T_s.
    end if
  end if
end while
```

5. Comparison of algorithms

To compare the algorithms statistically, we use Friedman test with the corresponding Nemenyi post-hoc test as suggested in [15]. Let we have k greedy algorithms $A_1, \ldots, A_k$ for constructing trees and M decision tables $T_1, \ldots, T_M$. For each decision table T_i, $i = 1, \ldots, M$, we rank the algorithms $A_1, \ldots, A_k$ on T_i based on their performance scores

Algorithm 3 Greedy algorithm A_I

Input: A decision table with many-valued decisions T, and conditional attributes $f_1, \ldots, f_n$.
Output: Decision tree $A_I(T)$ for T.

```
Construct the tree G consisting of a single node labeled with the table T;
while (true) do
  if No any node of the tree G is labeled with a table then
    Denote the tree G by A_I(T);
  else
    Choose a node v in G which is labeled with a subtable T' of the table T;
    if U(T') = 0 then
      Instead of T' mark the node v with the common decision for T';
    else
      For each f_i ∈ E(T'), we compute the value of the impurity function I(T', f_i);
      Choose the attribute f_{i_0} ∈ E(T'), where i_0 is the minimum i for which I(T', f_i) has the minimum value;
      Instead of T' mark the node v with the attribute f_{i_0};
      For each δ ∈ E(T', f_i), add to the tree G the node v_δ and mark this node with the subtable T'(f_{i_0}, δ);
      Draw an edge from v to v_δ and mark this edge with δ.
    end if
  end if
end while
```

(from the point of view average depth of constructed trees), where we assign the best performing algorithm the rank of 1, the second best rank 2, and so on. We break ties by computing the average of ranks. Let r_i^j be the rank of the j-th of k algorithms on the decision table T_i. For $j = 1, \ldots, k$, we correspond to the algorithm A_j the average rank

$$R_j = \frac{1}{M} \cdot \sum_{i=1}^{M} r_i^j.$$

For a fixed significance level α (in our work $\alpha = 0.05$), the performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k\,(k+1)}{6M}}$$

where q_α is a critical value for the two-tailed Nemenyi test depending on α and k [15].

We can also compare performance scores of algorithms $A_1, \ldots, A_k$ with optimal results obtained by dynamic programming algorithm. For $j = 1, \ldots, k$ and $i = 1, \ldots, M$, we denote, by h_{ij}^{avg} the average depth of the decision tree constructed by the algorithm A_j on the decision table T_i. For $i = 1, \ldots, M$, we denote by $h_i^{avg^{opt}}$ the value of the minimum possible average depth of a decision tree for T_i. Thus, we can compute the average relative difference in percentage for average depth as

$$ARD_j^{h^{avg}} = \frac{1}{M} \sum_{i=1}^{M} \frac{h_{ij}^{avg} - h_i^{avg^{opt}}}{h_i^{avg^{opt}}} \times 100.$$

6. Experimental results

We consider 16 decision tables from UCI Machine Learning Repository [14]. There were missing values for some attributes which were replaced with the most common values of the corresponding attributes. Some conditional attributes have been removed that take unique value for each row. To convert such tables into many-valued decision table format, we removed from these tables more conditional attributes. As a result we obtained inconsistent decision

tables which contained equal rows with different decisions. Each group of identical rows was replaced with a single row from the group which is labeled with the set of decisions attached to rows from the group. The information about obtained MVDT can be found in Table 4. This table contains the name of initial table from[14] with an index equal to the number of removed conditional attributes, number of rows (column "Rows"), number of attributes (column "Attr"), and spectrum of this table (column "Spectrum"). Spectrum of a decision table with MVD is a sequence #1, #2,..., where #i, $i = 1, 2, \ldots$, is the number of rows labeled with sets of decisions with the cardinality equal to i. We randomly select 70% of rows 500 times from each table. As a result, we obtain 8000(= $16 \cdot 500$) decision tables.

Table 4. Characteristics of MVDT

Decision table T	Rows	Attr	Spectrum					
			#1	#2	#3	#4	#5	#6
balance-scale-1	125	3	45	50	30			
breast-cancer-1	193	8	169	24				
breast-cancer-5	98	4	58	40				
cars-1	432	5	258	161	13			
flags-5	171	21	159	12				
hayes-roth-data-1	39	3	22	13	4			
lymphography-5	122	13	113	9				
mushroom-5	4078	17	4048	30				
nursery-1	4320	7	2858	1460	2			
nursery-4	240	4	97	96	47			
spect-test-1	164	21	161	3				
teeth-1	22	7	12	10				
teeth-5	14	3	6	3	0	5	0	2
tic-tac-toe-4	231	5	102	129				
tic-tac-toe-3	449	6	300	149				
zoo-data-5	42	11	36	6				

We have six uncertainty measures and four types of impurity functions, so total 24 greedy algorithms have been compared. In the critical difference diagram (CDD), we show the names of the algorithms as combined name of heuristic and impurity function types separated by '_'. For example, if the algorithm name is *wm_nBS*, this means it uses *wm* as a type of impurity function and *nBS* as uncertainty measure.

Figure 2 shows the CDD containing average (mean) rank for each algorithm on the x-axis for significance level of $\alpha = 0.05$. The best ranked algorithm are shown in the leftmost side of the figure. When Nemenyi test cannot identify significant difference between some algorithms, then those are clustered (connected). It is clear that, *ws_entML*, *ws_entSort*, and *Div_ws_entSort* are the leaders among all greedy algorithm for minimization of decision tree average depth. If we look at the ARD Table 5, we can see the best algorithm that is closer to the optimal result is the *ws_entML*.

Table 5. Average relative difference between results of greedy and dynamic algorithms (with sampling)

Algorithm	*ARD*
ws_entML	3.26%
ws_entSort	3.49%
Div_ws_entSort	4.53%

Now, we cannot guarantee that our data sets are independent after sampling, therefore, we have done the experiments without sampling. Hence, in Figure 3 we have shown the experimental results for 12 UCI data sets (we removed 'breast-cancer-5', 'nursery-5', 'teeth-5', and 'tic-tac-toe-4' data sets from Table 4) without sampling. These are obviously independent to each other, and the average relative difference are shown in Table 6. We can see that now lowest average relative difference is 4.58% which can be obtained when we use the algorithm *ws_entSort*.

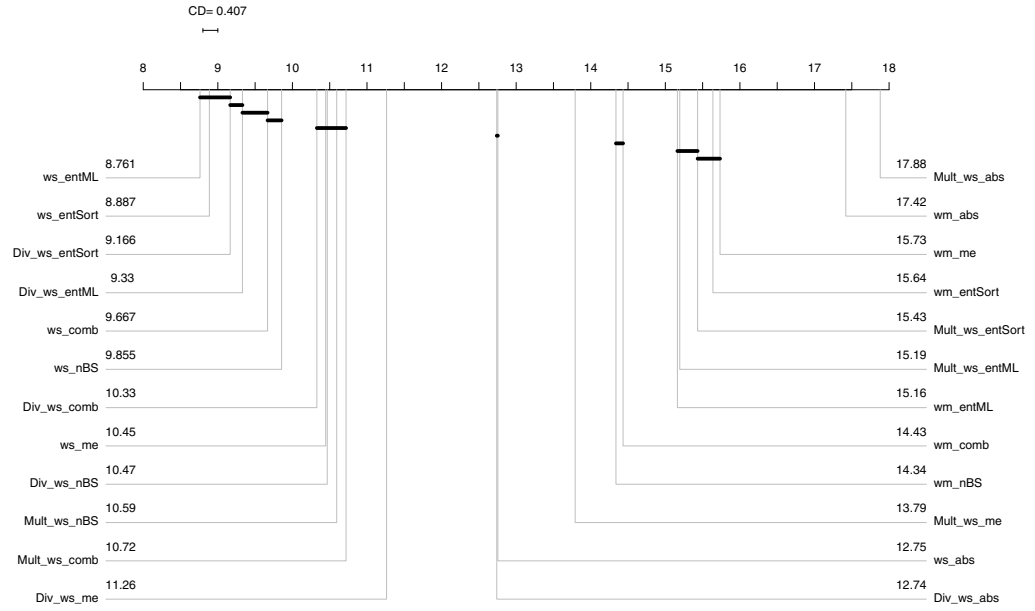

Fig. 2. CDD for average depth of decision trees constructed by greedy algorithms (with sampling)

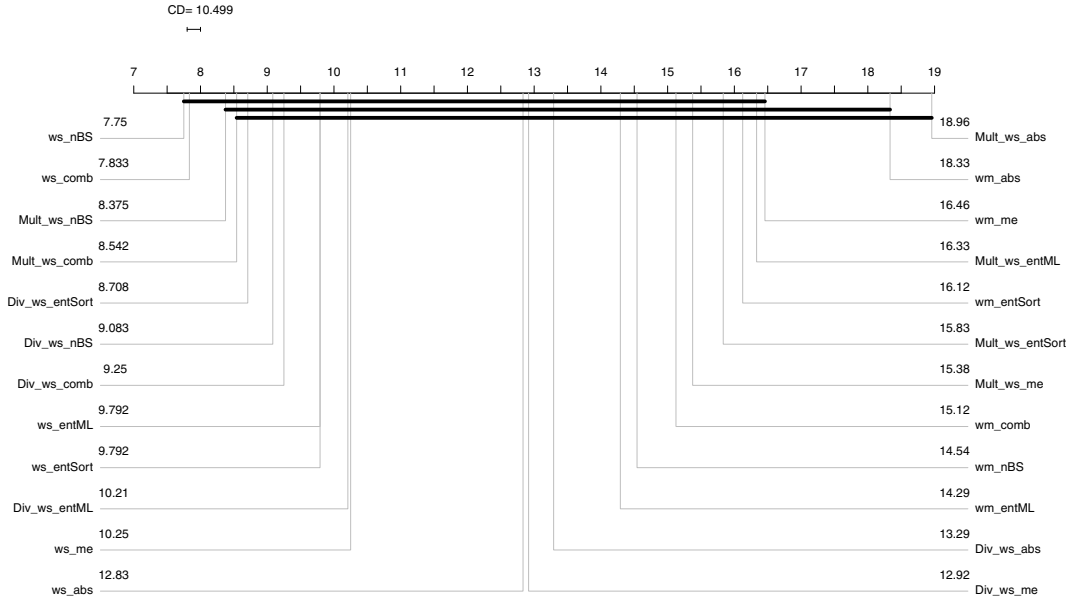

Fig. 3. CDD for average depth of decision trees constructed by greedy algorithms (without sampling)

7. Conclusion

In this paper, we studied greedy algorithms for decision tree construction which are based on various impurity functions. We compared these algorithms from the point of view of minimization of average depth of decision trees

Table 6. Average relative difference between results of greedy and dynamic algorithms (without sampling)

Algorithm	*ARD*
ws_entSort	4.58%
ws_entML	5.47%
Div_ws_entSort	5.65%

for MVDT, and considered also the average relative difference between optimal and greedy results. We found that, for the best greedy algorithm, ARD is at most 3.2% in case of with sampling and 4.58% in case of without sampling.

Acknowledgement

Research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST).

References

1. Moshkov, M., Zielosko, B.. *Combinatorial Machine Learning–A Rough Set Approach*; vol. 360 of *Studies in Computational Intelligence*. Heidelberg: Springer; 2011. ISBN 978-3-642-20994-9.
2. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.. Learning multi-label scene classification. *Pattern Recognition* 2004;**37**(9):1757–1771.
3. Wieczorkowska, A., Synak, P., Lewis, R.A., Raś, Z.W.. Extracting emotions from music data. In: Hacid, M.S., Murray, N.V., Raś, Z.W., Tsumoto, S., editors. *Foundations of Intelligent Systems, 15th International Symposium, ISMIS 2005, Saratoga Springs, 2005, Proceedings*; vol. 3488 of *LNCS*. Springer; 2005, p. 456–465.
4. Blockeel, H., Schietgat, L., Struyf, J., Dzeroski, S., Clare, A.. Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M., editors. *PKDD 2006, Berlin, Germany, Proceedings*; vol. 4213 of *LNCS*. Springer; 2006, p. 18–29.
5. Zhou, Z.H., Jiang, K., Li, M.. Multi-instance learning based web mining. *Appl Intell* 2005;**22**(2):135–147.
6. Comité, F.D., Gilleron, R., Tommasi, M.. Learning multi-label alternating decision trees from texts and data. In: Perner, P., Rosenfeld, A., editors. *MLDM 2003, Leipzig, Germany*; vol. 2734 of *LNCS*. Springer; 2003, p. 35–49.
7. Mencía, E.L., Fürnkranz, J.. Pairwise learning of multilabel classifications with perceptrons. In: *Proceedings of the International Joint Conference on Neural Networks, IJCNN, China*. IEEE; 2008, p. 2899–2906.
8. Moshkov, M.J.. Greedy algorithm for decision tree construction in context of knowledge discovery problems. In: Tsumoto, S., Słowinski, R., Komorowski, H.J., Grzymała-Busse, J.W., editors. *Rough Sets and Current Trends in Computing, 4th International Conference, 2004, Proceedings*; vol. 3066 of *LNCS*. Springer; 2004, p. 192–197.
9. Tsoumakas, G., Katakis, I., Vlahavas, I.P.. Mining multi-label data. In: *Data Mining and Knowledge Discovery Handbook, 2nd ed.* Springer; 2010, p. 667–685.
10. Zhou, Z.H., Zhang, M.L., Huang, S.J., Li, Y.F.. Multi-instance multi-label learning. *Artif Intell* 2012;**176**(1):2291–2320.
11. Azad, M., Chikalov, I., Moshkov, M., Zielosko, B.. Greedy algorithm for construction of decision trees for tables with many-valued decisions. In: Popova-Zeugmann, L., editor. *Proceedings of the 21th International Workshop on Concurrency, Specification and Programming, Berlin, Germany, September 26-28, 2012*; vol. 928 of *CEUR Workshop Proceedings*. CEUR-WS.org; 2012, .
12. Moshkov, M., Zielosko, B.. Construction of α-decision trees for tables with many-valued decisions. In: Yao, J., Ramanna, S., Wang, G., Suraj, Z., editors. *RSKT 2011, Canada*; vol. 6954 of *LNCS*. Springer; 2011, p. 486–494.
13. Azad, M., Chikalov, I., Moshkov, M.. Three approaches to deal with inconsistent decision tables - comparison of decision tree complexity. In: Ciucci, D., Inuiguchi, M., Yao, Y., Ślzak, D., Wang, G., editors. *RSFDGrC*; vol. 8170 of *LNCS*. Springer. ISBN 978-3-642-41217-2; 2013, p. 46–54.
14. Asuncion, A., Newman, D.J.. UCI Machine Learning Repository. http://www.ics.uci.edu/ mlearn/, 2007.
15. Demšar, J.. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 2006;**7**:1–30.
16. Tsoumakas, G., Katakis, I.. Multi-label classification: An overview. *IJDWM* 2007;**3**(3):1–13.
17. Zhu, X., Goldberg, A.B.. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers; 2009.
18. Cour, T., Sapp, B., Jordan, C., Taskar, B.. Learning from ambiguously labeled images. In: *CVPR*. 2009, p. 919–926.
19. Hüllermeier, E., Beringer, J.. Learning from ambiguously labeled examples. *Intell Data Anal* 2006;**10**(5):419–439.
20. Jin, R., Ghahramani, Z.. Learning with multiple labels. In: Becker, S., Thrun, S., Obermayer, K., editors. *NIPS*. 2002, p. 897–904.
21. Clare, A., King, R.D.. Knowledge discovery in multi-label phenotype data. In: Raedt, L.D., Siebes, A., editors. *PKDD 2001, Freiburg, Germany, September 3-5, 2001, Proceedings*; vol. 2168 of *LNCS*. Springer; 2001, p. 42–53.
22. Moshkov, M., Chikalov, I.. On algorithm for constructing of decision trees with minimal depth. *Fundam Inform* 2000;**41**(3):295–299.