DOI: 10.15439/2014F256
ACSIS, Vol. 2

Minimizing Size of Decision Trees for Multi-label Decision Tables

Mohammad Azad and Mikhail Moshkov
Computer, Electrical & Mathematical Sciences & Engineering Division
King Abdullah University of Science and Technology
Thuwal 23955-6900, Saudi Arabia
{mohammad.azad, mikhail.moshkov}@kaust.edu.sa

***Abstract*—We used decision tree as a model to discover the knowledge from multi-label decision tables where each row has a set of decisions attached to it and our goal is to find out one arbitrary decision from the set of decisions attached to a row. The size of the decision tree can be small as well as very large. We study here different greedy as well as dynamic programming algorithms to minimize the size of the decision trees. When we compare the optimal result from dynamic programming algorithm, we found some greedy algorithms produce results which are close to the optimal result for the minimization of number of nodes (at most 18.92% difference), number of nonterminal nodes (at most 20.76% difference), and number of terminal nodes (at most 18.71% difference).**

I. Introduction

NOW a days, multi-label decision tables have gained attention in problem of semantic annotation of images [1], music categorization into emotions [2], functional genomics [3], and text categorization [4]. 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 [5], finding nearest post office in the post office problem [5]; in this case we give input with more than one optimal solutions.

In multi-label decision tables, each row is labeled with a set of decisions. It is common to have such tables in our real life because we do not have enough number of attributes of the domain to separate rows. Thus we have objects with equal values of conditional attributes but with different decisions. In literature, often, decision trees and other classifiers for multi-label data are considered for prediction (multi-label classification problem) [6], [7], [8], [9]. However, in this paper our aim is to study decision trees for multi-label decision tables for knowledge representation, and as algorithms for problem solving.

In [10] we studied a greedy algorithm for construction of decision trees for multi-label decision tables using the heuristic based on the number of boundary subtables. Besides, in [11] we have studied this algorithm in the cases of most common decision, and generalized decision approaches (in that paper, we considered decision tables with one-valued decisions as multi-label decision tables where sets of decisions attached to rows have one element).

This paper is a continuation of the conference publication [11]. We have introduced new greedy heuristics 'misclassification error', 'absent', 'combined' whose performances are as good as the previous one. Also we adapt, 'multi-label entropy', 'sorted entropy' heuristics from literature. We compared the performance among themselves for the cost function of number of nodes, number of nonterminal nodes and number of terminal nodes. We have done experiments using modified data sets from UCI Machine Learning Repository [12]. Based on the results of the experiments, we have presented rankings among the algorithms in the form of critical difference diagram [13]. Furthermore, we have shown the average relative difference between greedy and optimal results to describe how close they are. Hence, our goal is to choose some of the greedy heuristics which are close to the optimal results.

This paper consists of six sections. Section II contains the related background study of this problem. Section III contains the important definitions related to our study. After that, in Sect. IV, we presented the dynamic and greedy algorithms for construction of decision trees, then in Sect. V we gave comparison among algorithms using Friedman test. Section VI contains results of experiments and Sect. VII concludes the paper.

II. Related Work

In literature, often, problems that are connected with multi-label data are considered for classification: multi-label learning [14], multi-instance learning [9] etc. In multi-label learning, the output for each instances can be a set of decisions, whereas in our framework, we chose 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 [15] where some examples are labeled but some are not labeled, 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 [16], ambiguous learning [17], and multiple label learning [18]. These problems consider only one label as correct and others as incorrect, but we consider all labels that are attached to an object as correct labels for that object. In [16], [18], the authors showed probabilistic methods to solve the learning problem whereas in [17], the author used standard heuristic approach to exploit inductive bias to disambiguate label information.

Additionally, these papers only focus on classification results rather than optimization of data model.

In this paper, we consider the problem 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 arbitrary decision from the set of decision attached with each row.

III. Main Definitions

A. Multi-label Decision Tables

A *multi-label decision table* T is a rectangular table filled by nonnegative integers. Columns of this table are labeled with conditional attributes $f_1, \dots, f_n$. 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 the number of rows in the table T by $N(T)$. We denote row i by r_i where $i = 1, \dots, N(T)$. For example, r_1 means the first row, r_2 means the second row and so on.

TABLE I
A multi-label decision table 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 does not have rows or it has a common decision. For example, T' is a degenerate table as shown in Table II, where the common decision is 1.

TABLE II
A degenerate multi-label decision table, 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 each of its proper subtable is degenerate. We denote the number of boundary subtables of the table T by $nBS(T)$. Below are examples 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	$\{2\}$

The subtable of T which consists of rows that have values $a_1, \dots, a_m$ at the intersection with columns $f_{i_1}, \dots, f_{i_m}$ is denoted by $T(f_{i_1}, a_1), \dots, (f_{i_m}, a_m)$. 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^0 (see Table I), it will consist of rows $1, 2$, and 5. Similarly, $T^0(f_1, 0)(f_2, 0)$ subtable will consist of rows 1, and 5 (see Table III).

TABLE III
Example of subtables of multi-label decision table 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}

The set of attributes (columns of table T), such that each of them has different values is denoted by $E(T)$. 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 the set of values from the column f_i by $E(T, f_i)$. 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 the number of rows for which the set of decisions contains the most common decision for T by $N_{mcd}(T)$. For the table T^0, $N_{mcd}(T^0) = 3$.

B. 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 $\{f_1, \dots, f_n\}$. A number of edges start from each nonterminal node which are labeled with different non-negative integers (e.g. two edges labeled with 0 and 1 if the nonterminal node is labeled with 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 define a subtable $T(v)$ of T corresponding to the node v. 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 node 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 $f_i \in E(T(v))$, and if $E(T(v), f_i) = \{a_1, \dots, a_k\}$, then k outgoing edges from node v are labeled with $a_1, \dots, a_k$.

An example of a decision tree for the table T can be found in Fig. 1. If v is the node labeled with the 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 $T(f_1, 0)(f_3, 0)$.

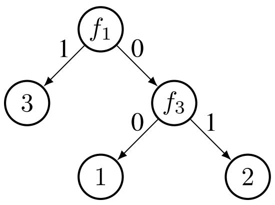

Fig. 1. A decision tree for the multi-label decision table, T^0

The number of nodes in the decision tree Γ is denoted by $N(\Gamma)$. The number of terminal and nonterminal nodes in the decision tree Γ are denoted by $N^t(\Gamma)$, and $N^n(\Gamma)$ respectively.

C. *Impurity Functions and Uncertainty Measures*

In greedy algorithm, we need to choose attributes to divide 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 of the considered subtable. If we have a common decision, then there is no uncertainty in the data, hence we get uncertainty measure as zero, otherwise we will get positive values for it. We have used six different uncertainty measures, and four different impurity function types for our experiments.

1) Uncertainty Measures: Uncertainty measure U is a function from the set of nonempty multi-label decision tables to the set of real numbers such that $U(T) \geq 0$ for any decision table T, and $U(T) = 0$ if and only if T is degenerate.

Let T be a multi-label decision table having n conditional attributes, $N = N(T)$ rows and its rows be 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 $p_i = \frac{N_i}{N}$. 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 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 $d_1, \dots, d_{i-1}$ by $N_i^{'}$, and $p_i^{'} = \frac{N_i^{'}}{N}$. 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 the difference between total number of rows and number of rows with the most common decision.
- Sorted entropy: $entSort(T) = -\sum_{i=1}^{m} p_i^{'} \log_2 p_i^{'}$ (see [17]). First we sort the probability for each decision. Then, for each row, we keep the decision having maximum probability and discard others. After that, we calculate entropy for this modified decision table.
- Multi-label 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, $q_i = 1 - p_i$. (see [19]). It measures entropy for multi-label decision table.
- $nBS(T)$: number of boundary subtables in T. We calculate number of boundary subtables using brute force approach by checking all possible subtables of T (see [10]).
- Absent: $abs(T) = q_1 \cdots q_m$, where $q_i = 1 - p_i$. It measures the absent probability q_i, and multiplies all of 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 rows and three rows, respectively. It is the combination of uncertainty measures.

2) Impurity Functions: Let $f_i \in E(T)$, and $E(T, f_i) = \{a_1, \dots, a_t\}$. The attribute f_i divides the table T into t subtables: $T_1 = T(f_i, a_1), \dots, T_t = T(f_i, a_t)$. 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: $I(T, f_i) = \max_{1 \leq j \leq t} U(T_j) N(T_j)$. For this type, we take the maximum among all the uncertainties of tables $T_1, \dots, T_t$ multiplied by the weights of its number of rows.
- ws: $I(T, f_i) = \sum_{j=1}^{t} U(T_j) N(T_j)$. For this type, we take the sum over all the uncertainties of tables $T_1, \dots, T_t$ multiplied by the weights of its number of rows.
- Div_ws: $I(T, f_i) = (\sum_{j=1}^{t} U(T_j) N(T_j)) / \log_2 t$. For this type, we divide the weighted sum impurity type (wt_sum) by the logarithmic function of number of branches ($\log_2 t$).

- *Mult_ws*: $I(T, f_i) = (\sum_{j=1}^{t} U(T_j)N(T_j)) \cdot \log_2 t$. For this type, we multiply the weighted sum impurity type (wt_sum) by the logarithmic function of number of branches ($\log_2 t$)

As a result, we have 24 (4 types multiplied by 6 uncertainty measures) impurity functions.

IV. Algorithms for Decision Tree Construction

In this section, we consider dynamic programming algorithm and greedy algorithms. Dynamic programming algorithm gives us optimal solution whereas greedy algorithms give us suboptimal solutions. As dynamic programming is highly time consuming, we need to choose some greedy algorithms which will be fast enough as well as their performances will be comparable to the optimal one.

A. Dynamic Programming Algorithm

We now describe an algorithm A_d which, for a given multi-label decision table constructs a decision tree with minimum size (number of nodes, or number of nonterminal nodes, or number of terminal nodes). This algorithm is based on dynamic programming approach [20], [5], and the complexity of this algorithm in the worst case is exponential.

Let T contains n conditional attribute $f_1, \dots, f_n$. The set of all separable subtables of the table T including the table T is denoted by $S(T)$. The first part of the algorithm A_d constructs the set $S(T)$ (see Algorithm 1). For each subtable from $S(T)$, the second part of the algorithm A_d constructs a decision tree with minimum size (see Algorithm 2). Note that here size refers to either the number of nodes in the tree, or the number of nonterminal nodes in the tree, or the number of terminal nodes in the tree.

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

```
Require: A multi-label decision table $T$ with conditional
    attributes $f_1, \dots, f_n$.
Ensure: 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 $E(T_s) = \phi$ then
        Mark the table $T_s$ as treated;
      else
        Add to the set $S(T)$ all subtables of the form
        $T_s(f_i, \delta)$, where $f_i \in E(T_s)$, and $\delta \in E(T_s, f_i)$
        which were not in $S(T)$, mark the table $T_s$ as
        treated, and new subtables $T_s(f_i, \delta)$ as untreated.
      end if
    end if
  end while
```

After that, A_d returns the minimum size of the optimal tree which corresponds to the table T.

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

```
Require: A multi-label decision table $T$, with conditional
    attributes $f_1, \dots, f_n$, and the set $S(T)$.
Ensure: Decision tree $A_d(T)$ for $T$.
  while (true) do
    if $T$ has been assigned a decision tree then
      Return this tree as $A_d(T)$;
    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 $f_i \in E(T_s)$ and each $\delta \in E(T_s, f_i)$,
        we denote the decision tree assigned to the table
        $T_s(f_i, \delta)$ by $\Gamma(f_i, \delta)$. We now define a decision tree
        $\Gamma_{f_i}$ with a root labeled by the attribute $f_i$ where
        $f_i \in E(T_s)$, and $E(T_s, f_i) = \{\delta_1, \dots, \delta_r\}$. The root
        has exactly $r$ edges $d_1, \dots, d_r$ which are labeled by
        the numbers $\delta_1, \dots, \delta_r$, respectively. The roots of
        the decision trees $\Gamma(f_i, \delta_1), \dots, \Gamma(f_i, \delta_r)$ are ending
        points of the edges $d_1, \dots, d_r$, respectively. Assign
        to the table $T_s$ one of the trees $\Gamma_{f_i}, f_i \in E(T_s)$,
        having minimum size.
      end if
    end if
  end while
```

B. Greedy Algorithms

Let I be an impurity function. For a given multi-label decision table T, the greedy algorithm A_I 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. In case of 'number of boundary subtables' uncertainty measure, we will only consider those tables where the maximum number of decisions are bounded.

V. Comparison of Algorithms

To compare the algorithms statistically, we use Friedman test with the corresponding Nemenyi post-hoc test as suggested in [13]. Let we have k greedy algorithms $A_1, \dots, A_k$ for constructing trees and M decision tables $T_1, \dots, T_M$. For each decision table T_i, $i = 1, \dots, M$, we rank the algorithms $A_1, \dots, A_k$ on T_i based on their performance scores (from the point of view of cost functions: number of nodes, or number of nonterminal nodes, or number of terminal nodes of constructed

Algorithm 3 Greedy algorithm A_I

```
Require: A multi-label decision table $T$ with conditional attributes $f_1, \dots, f_n$.
Ensure: 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 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 $f_i \in E(T')$, we compute the value of the impurity function $I(T', f_i)$;
        Choose the attribute $f_{i_0} \in E(T')$, 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 $\delta \in E(T', f_i)$, add to the tree $G$ the node $v_\delta$ and mark this node with the subtable $T'(f_{i_0}, \delta)$;
        Draw an edge from $v$ to $v_\delta$ and mark this edge with $\delta$.
      end if
    end if
  end while
```

trees), where we assign the best performing algorithm as the rank 1, the second best as the 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 $j = 1, \dots, k$, 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 significant 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 (see [13]).

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

$$ARD_j^N = \frac{1}{M} \sum_{i=1}^{M} \frac{N_{ij} - N_i^{opt}}{N_i^{opt}} \times 100\%.$$

Similarly, for number of nonterminal nodes ($N_n(\Gamma)$), we have

$$ARD_j^{N^n} = \frac{1}{M} \sum_{i=1}^{M} \frac{N_{ij}^n - {N^n}_i^{opt}}{{N^n}_i^{opt}} \times 100\%.$$

Similarly, for number of terminal nodes ($N_t(\Gamma)$), we have

$$ARD_j^{N^t} = \frac{1}{M} \sum_{i=1}^{M} \frac{N_{ij}^t - {N^t}_i^{opt}}{{N^t}_i^{opt}} \times 100\%.$$

VI. Experimental Results

We consider 16 decision tables from UCI Machine Learning Repository [12]. 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 multi-label decision table format, we removed the more conditional attributes from these tables. 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 multi-label decision table can be found in Table IV. Modified decision table has been renamed in Table IV by the name of initial table plus an index equal to the number of removed conditional attributes. Table IV also contains the number of rows (column "Rows"), the number of attributes (column "Attr"), and the spectrum of the corresponding decision table (column "Spectrum"). Spectrum of a multi-label decision table is a sequence #1, #2,..., where #i, $i = 1, 2, \dots$, is the number of rows labeled with sets of decisions with the cardinality equal to i.

TABLE IV
Characteristics of multi-label decision tables

Decision	Rows	Attr	Spectrum					
table T			#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				

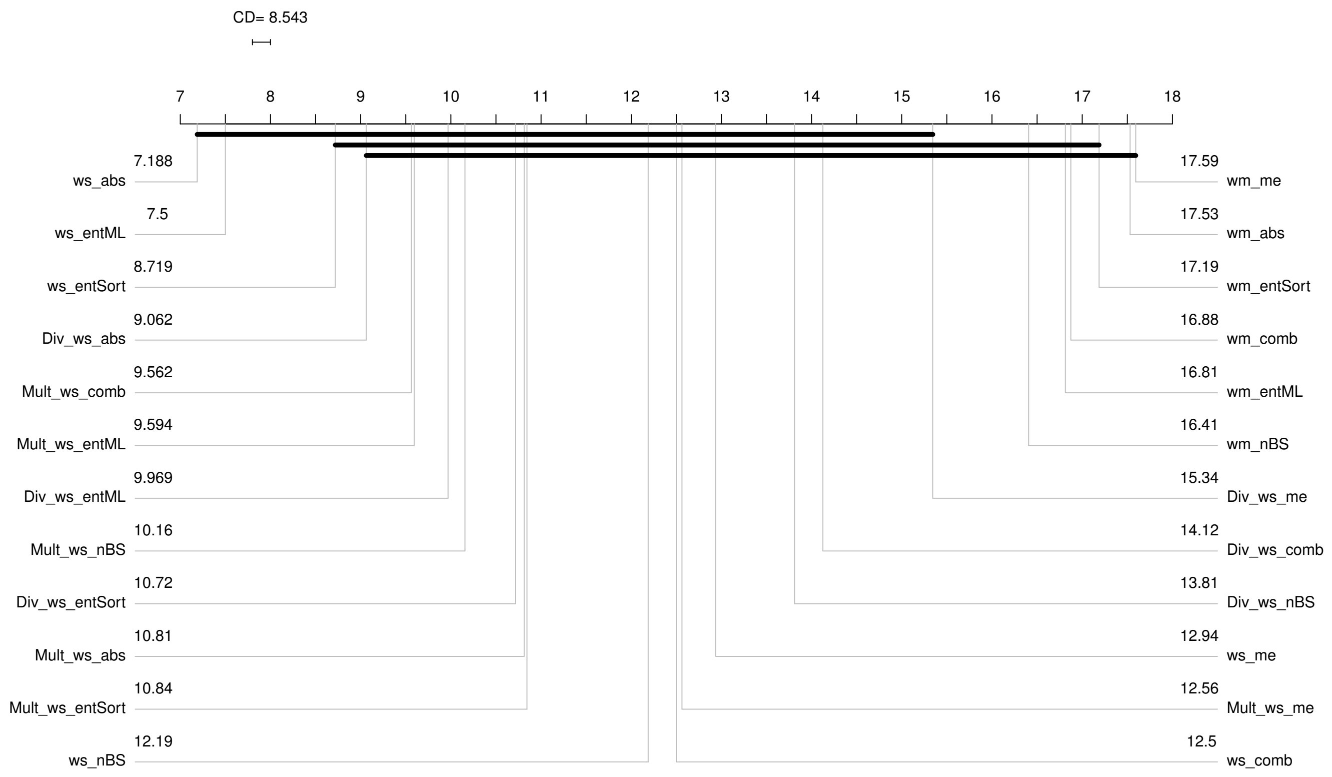

Fig. 2. CDD for number of nodes of decision trees constructed by greedy algorithms

We have six uncertainty measures (me, nBS, abs, $comb$, $entSort$, $entML$) and four types of impurity functions (ws, wm, Div_ws, $Mult\_ws$), so total 24 greedy algorithms have been compared. In the critical difference diagram (CDD), we showed 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 algorithms on the x-axis for significant level of $\alpha = 0.1$. When Nemenyi test cannot identify significant difference between some algorithms, the algorithms are clustered (connected). It is clear from Figure 2 that, 18 algorithms from the 24 algorithms are clustered in the first group (left most algorithms is the best ranked algorithm) which are the leaders among all greedy algorithms for minimization of number of nodes in decision trees, and the best ranked algorithm is ws_abs. We have shown the best three algorithms having minimum ARD in Table V. If we look at the ARD table, we can see the best algorithm that is closer to the optimal results is ws_abs, and the average relative difference is only 18.92%.

Also, we can see from Figure 3 that, 17 algorithms among 24 algorithms are leaders for minimization of number of nonterminal nodes in decision trees, and the best ranked algorithm is ws_abs. We have shown the best three algorithms having minimum ARD for minimization of number of nonterminal nodes in Table VI, and the ARD for ws_abs is only 20.76% relative to the optimal results.

Now, for the minimization of number of terminal nodes, we can see from Figure 4 that 19 algorithms from 24 algorithms are in the best group, and the best ranked algorithm is $Mult\_ws\_entML$. Also from ARD Table VII, we can see that $Mult\_ws\_entML$ is closer to the optimal results by only 18.71%.

TABLE V
ARD IN PERCENTAGE BETWEEN RESULTS OF GREEDY AND DYNAMIC ALGORITHMS FOR TOTAL NUMBER OF NODES

Algorithm	ARD
ws_abs	18.92%
$Mult\_ws\_entML$	19.73%
ws_entML	20.58%

TABLE VI
ARD IN PERCENTAGE BETWEEN RESULTS OF GREEDY AND DYNAMIC ALGORITHMS FOR NUMBER OF NONTERMINAL NODES

Algorithm	ARD
ws_abs	20.76%
ws_entML	22.6%
$ws\_entSort$	23.7%

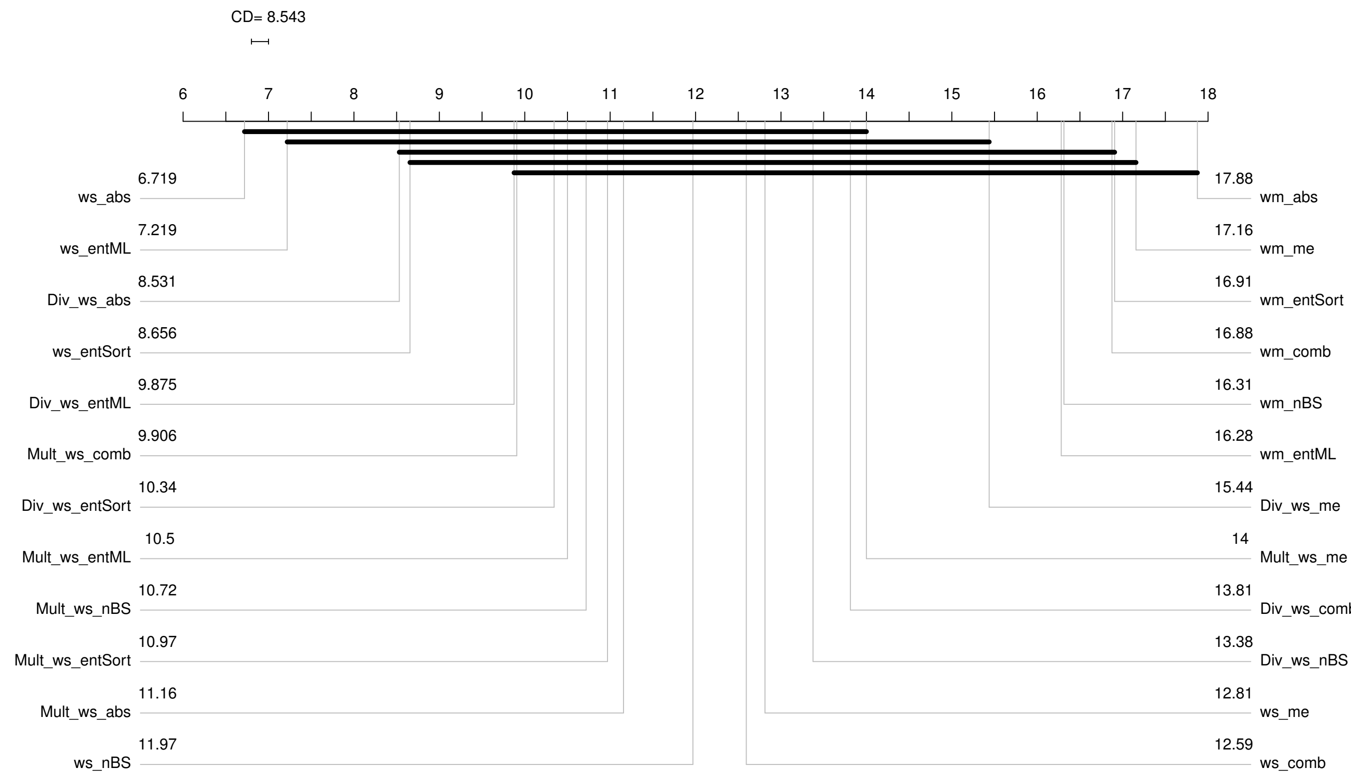

Fig. 3. CDD for nonterminal nodes of decision trees constructed by greedy algorithms

TABLE VII
ARD IN PERCENTAGE BETWEEN RESULTS OF GREEDY AND DYNAMIC ALGORITHMS FOR NUMBER OF TERMINAL NODES

Algorithm	ARD
$Mult\_ws\_entML$	18.71%
ws_abs	21%
$ws\_entSort$	22.49%

VII. Conclusion

In this paper, we studied greedy algorithms for decision tree construction which are based on various impurity functions. We compared these algorithms to find out which algorithm gives us minimum number of nodes, minimum number of nonterminal nodes and minimum number of terminal nodes. We also considered the average relative difference between optimal results and results obtained by the greedy algorithms. We found that, for the best greedy algorithm, it is at most 18.92% ARD for the minimization of number of nodes, at most 20.76% ARD for the minimization of number of nonterminal nodes, and 18.71% ARD for the minimization of terminal nodes, which are promising results. In future, our goal is to compare the above best greedy algorithm for the problem of prediction i.e. to minimize the prediction error.

Acknowledgement

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

References

[1] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004. doi: 10.1016/j.patcog.2004.03.009

[2] A. Wieczorkowska, P. Synak, R. A. Lewis, and Z. W. Ras, "Extracting emotions from music data," in *ISMIS*, 2005. doi: 10.1007/11425274 pp. 456–465.

[3] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, and A. Clare, "Decision trees for hierarchical multilabel classification: A case study in functional genomics," in *PKDD 2006, Berlin, Germany, Proceedings*, ser. LNCS, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Springer, 2006, vol. 4213, pp. 18–29.

[4] Z.-H. Zhou, K. Jiang, and M. Li, "Multi-instance learning based web mining," *Appl. Intell.*, vol. 22, no. 2, pp. 135–147, 2005. doi: 10.1007/s10489-005-5602-z

[5] M. Moshkov and B. Zielosko, *Combinatorial Machine Learning - A Rough Set Approach*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 360. ISBN 978-3-642-20994-9

[6] F. D. Comité, R. Gilleron, and M. Tommasi, "Learning multi-label alternating decision trees from texts and data," in *MLDM*, 2003. doi: 10.1007/3-540-45065-3 pp. 35–49.

[7] E. Loza Mencía and J. Fürnkranz, "Pairwise learning of multilabel classifications with perceptrons," in *IJCNN*, 2008. doi: 10.1109/IJCNN.2008.4634206 pp. 2899–2906.

[8] G. Tsoumakas, I. Katakis, and I. P. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*, 2010, pp. 667–685.

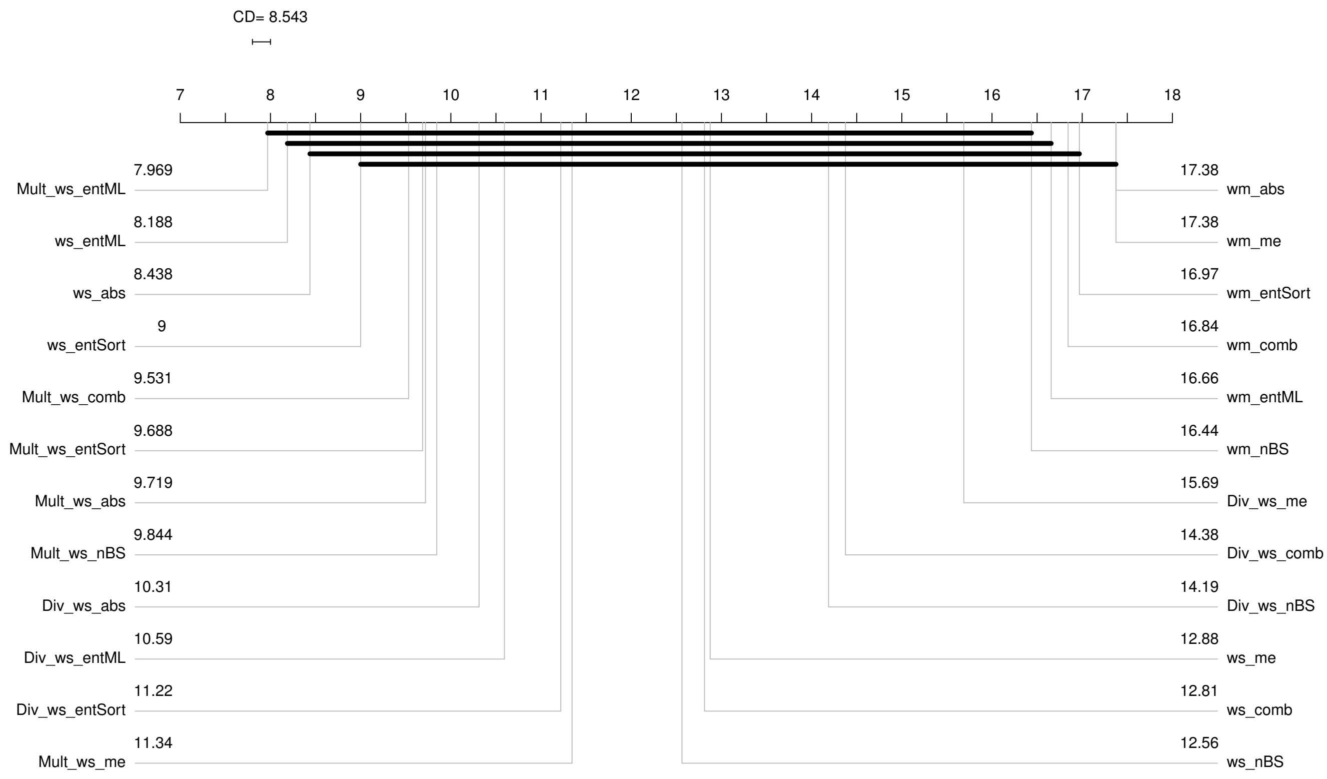

Fig. 4. CDD for terminal nodes of decision trees constructed by greedy algorithms

[9] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li, "Multi-instance multi-label learning," *Artif. Intell.*, vol. 176, no. 1, pp. 2291–2320, 2012. doi: 10.1016/j.artint.2011.10.002

[10] M. Azad, I. Chikalov, M. Moshkov, and B. Zielosko, "Greedy algorithm for construction of decision trees for tables with many-valued decisions," in *Proceedings of the 21th International Workshop on Concurrency, Specification and Programming, Berlin, Germany, September 26-28, 2012*, ser. CEUR Workshop Proceedings, L. Popova-Zeugmann, Ed. CEUR-WS.org, 2012, vol. 928.

[11] M. Azad, I. Chikalov, and M. Moshkov, "Three approaches to deal with inconsistent decision tables - comparison of decision tree complexity," in *RSFDGrC*, 2013. doi: 10.1007/978-3-642-41218-9 pp. 46–54.

[12] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository," http://www.ics.uci.edu/ mlearn/, 2007.

[13] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[14] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *IJDWM*, vol. 3, no. 3, pp. 1–13, 2007.

[15] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[16] T. Cour, B. Sapp, C. Jordan, and B. Taskar, "Learning from ambiguously labeled images," in *CVPR*, 2009. doi: 10.1109/CVPRW.2009.5206667 pp. 919–926.

[17] E. Hüllermeier and J. Beringer, "Learning from ambiguously labeled examples," *Intell. Data Anal.*, vol. 10, no. 5, pp. 419–439, 2006.

[18] R. Jin and Z. Ghahramani, "Learning with multiple labels," in *NIPS*, 2002, pp. 897–904.

[19] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *PKDD*, 2001. doi: 10.1007/3-540-44794-6 pp. 42–53.

[20] M. Moshkov and I. Chikalov, "On algorithm for constructing of decision trees with minimal depth," *Fundam. Inform.*, vol. 41, no. 3, pp. 295–299, 2000. doi: 10.3233/FI-2000-41302