Classification and Optimization of Decision Trees for Inconsistent Decision Tables Represented as MVD Tables

Classification and Optimization of Decision Trees for Inconsistent Decision Tables Represented as MVD 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*—Decision tree is a widely used technique to discover patterns from consistent data set. But if the data set is inconsistent, where there are groups of examples (objects) with equal values of conditional attributes but different decisions (values of the decision attribute), then to discover the essential patterns or knowledge from the data set is challenging. We consider three approaches (generalized, most common and many-valued decision) to handle such inconsistency. We created different greedy algorithms using various types of impurity and uncertainty measures to construct decision trees. We compared the three approaches based on the decision tree properties of the depth, average depth and number of nodes. Based on the result of the comparison, we choose to work with the many-valued decision approach. Now to determine which greedy algorithms are efficient, we compared them based on the optimization and classification results. It was found that some greedy algorithms (*$Mult\_ws\_entSort$*, and *$Mult\_ws\_entML$*) are good for both optimization and classification.**

I. Introduction

OFTEN in a decision table, we have different examples with the different values of decision and we call such table as a consistent decision table or single-valued decision table. But it is pretty common in real life problems to have inconsistent decision tables where there are groups of examples (objects) with equal values of conditional attributes and different decisions (values of the decision attribute).

In this paper, instead of the group of examples with equal values of conditional attribute, we consider only one example for this group and attach the set of decisions to it. We will call such tables as many-valued decision tables.

In the rough set theory [1], generalized decision (GD) has been used to handle inconsistency. In this case, an inconsistent decision table is transformed into the many-valued decision table and after that, each set of decisions has been encoded by a number (decision) such that equal sets are encoded by equal numbers and different sets by different numbers (see Figure 1). We have also used another approach named the most common decision (MCD) which is derived from the concept of using most common value in case of missing value [2]. Instead of a group of equal examples with (probably) different decisions, we consider one example given by values of conditional attributes and we attach to this example the most common decision for examples from the group (see Figure 1).

T^0 =

f_1	f_2	f_3	
1	1	1	1
0	1	0	1
0	1	0	3
1	1	0	2
0	0	1	2
0	0	1	3
1	0	0	1
1	0	0	2

T^0_{MVD} =

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

T^0_{GD} =

f_1	f_2	f_3	
1	1	1	1
0	1	0	2
1	1	0	3
0	0	1	4
1	0	0	5

T^0_{MCD} =

f_1	f_2	f_3	
1	1	1	1
0	1	0	1
1	1	0	2
0	0	1	2
1	0	0	1

Fig. 1: Transformation of inconsistent decision table T^0 into decision tables T^0_{MVD}, T^0_{GD} and T^0_{MCD}

In our approach, we can say that for a given example, we have multiple decisions that can be attached to the example but the goal is to find a single decision for each example. We refer this approach as many-valued decision (MVD) approach (see Figure 1). This approach is used for classical optimization problems (finding a Hamiltonian circuit with the minimum length or finding nearest post office [3]) where we have multiple optimal solutions but we have to give only one optimal output.

We studied a greedy algorithm for construction of decision trees for many-valued decision tables using the heuristic based on the number of boundary subtables in [4]. Besides, we have studied this algorithm in the cases of most common decision, and generalized decision approaches in [5]. In addition to this, we studied various greedy algorithms as well as dynamic programming algorithm to minimize the average depth in [6], minimize depth in [7], and as well as minimize size of the constructed decision tree in [8].

This paper is a continuation of the current research. We have compared three approaches MVD, MCD, and GD by comparing the complexity of constructed decision trees. We choose MVD approach based on the result of the comparison. After that, we have shown the average relative difference between greedy algorithm results and optimal results obtained by dynamic programming algorithms for the depth, average depth, and number of nodes of the constructed decision trees. Subsequently, we compare the performance of the classification error rates among classifiers constructed by the various greedy algorithms. We have presented results in the form of critical difference diagram [9] as well as average error rates using data sets from UCI ML Repository [10] and KEEL [11] repository. Finally, we found some of the greedy algorithms are in the top list for both optimization and classification tasks.

II. Related Works

In literature, these types of tables are often referred as multi-label decision tables [12]. These tables are found in the problem of semantic annotation of images and videos, music categorization into emotions, functional genomics (gene and protein functions), and text classification (news article, email, bookmarks). There are two ways to solve the classification problem from these data sets: first one is algorithm adaptation method where usual classification methods are adapted or modified to handle multi-label data, and the second one is the problem transformation method where the multi-label data set is transformed into single label data set to work with usual classification methods without any modification of the algorithm. These papers solve the inconsistency of the decision table by dividing full set of decisions into relevant and irrelevant decision set for each example. The goal is to find the relevant set of decisions for unknown object.

There is another way to handle inconsistency which is mentioned in different names in literature: partial learning [13], ambiguous learning [14], and multiple label learning [15]. In this learning problem, each example is associated with multiple labels but only one label is correct, and all others are incorrect. The goal is to find out which label is correct. In [13], [15], the authors shows probabilistic methods to solve the learning problem whereas in [14], the author used standard heuristic approach to exploit inductive bias to disambiguate label information.

Our approach of MVD is different from the above mentioned approaches in two ways:
- we assume that all our decisions are correct and there is no incorrect decisions attached with any of the examples,
- we assume that it is enough to find out one decision from the set of decisions rather than the relevant set of decisions.

Therefore, one can use our approach when it is enough to find one decision from the set of decisions.

III. Preliminaries

A. Many-valued Decision Table

A *many-valued decision table*, T is a rectangular table whose rows are filled by nonnegative integers and columns 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. There are no duplicate rows, and each row is labeled with a nonempty finite set of natural numbers (set of decisions). We denote the number of examples (rows) in the table T by $N(T)$.

TABLE I: A many-valued decision table T'

$T' =$

f_1	f_2	f_3	
0	0	0	{1}
0	1	1	{1,2}
1	0	1	{1,3}
1	1	0	{2,3}
0	0	1	{2}

If there is a decision which belongs to all of the set of decisions attached to examples 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 examples or it has a common decision. We give an example of degenerate table in the Table II where 1 is the common decision.

TABLE II: A degenerate many-valued decision table

$T'' =$

f_1	f_2	f_3	
0	0	0	{1}
0	1	1	{1,2}
1	0	1	{1,3}

A table obtained from T by removing some examples is called a subtable of T. We denote a *subtable* of T which consists of examples that at the intersection with columns $f_{i_1}, \ldots, f_{i_m}$ have values $a_1, \ldots, a_m$ by $T(f_{i_1}, a_1), \ldots, (f_{i_m}, a_m)$. Such nonempty tables (including the table T) are called separable subtables of T. For example, if we consider subtable $T'(f_1, 0)$ for table T', it will consist of examples $1, 2$, and 5. Similarly, $T'(f_1, 0)(f_2, 0)$ subtable will consist of examples 1, and 5.

TABLE III: Example of subtables of many-valued decision table T'

$T'(f_1, 0) =$

f_1	f_2	f_3	
0	0	0	{1}
0	1	1	{1,2}
0	0	1	{2}

$T'(f_1, 0)(f_2, 0) =$

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

We denote the set of attributes (columns of table T), such that each of them has different values by $E(T)$. For example, if we consider table T', $E(T') = \{f_1, f_2, f_3\}$. Similarly, $E(T'(f_1, 0)) = \{f_2, f_3\}$ for the subtable $T'(f_1, 0)$, because the value for the attribute f_1 is constant in subtable $T'(f_1, 0)$. For $f_i \in E(T)$, we denote the set of values from the attribute f_i by $E(T, f_i)$. As an example, if we consider table T' and attribute f_1, then $E(T', f_1) = \{0, 1\}$.

The minimum decision which belongs to the maximum number of sets of decisions attached to examples of the table

T is called the *most common decision* for T. For example, the most common decision for table T' is 1. Both 1 and 2 appears 3 times in the sets of decisions, but 1 is the most common decision as it is the minimum. We denote the number of examples for which the set of decisions contains the most common decision for T by $N_{mcd}(T)$.

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 $\{f_1, \ldots, f_n\}$. 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. We denote $T(v)$ as a subtable of T that is mapped for a node v of decision tree Γ. If the 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 $T(f_{i_1}, \delta_1) \ldots (f_{i_m}, \delta_m)$ of the table T where attributes $f_{i_1}, \ldots, f_{i_m}$ and numbers $\delta_1, \ldots, \delta_m$ 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 Γ satisfies the following conditions:
- if $T(v)$ is degenerate then v is labeled with the common decision for $T(v)$,
- otherwise v is labeled with an attribute $f_i \in E(T(v))$. In this case, k outgoing edges from node v are labeled with $a_1, \ldots, a_k$ where $E(T(v), f_i) = \{a_1, \ldots, a_k\}$.

An example of a decision tree for the table T can be found in Fig. 2. If the node v is 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 $T(f_1, 0)(f_3, 1)$ and here 2 is the common decision.

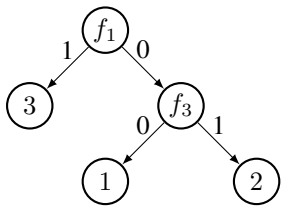

Fig. 2: Decision tree for the many-valued decision table T'

C. 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 by impurity function. We assume that, the smaller the impurity function value, the better is the quality of partition. Impurity function can be calculated based on uncertainty measures for the considered subtables corresponding to the partitions. If we have a common decision, then there is no uncertainty in the data, and uncertainty measure is zero, otherwise uncertainty measure is positive.

Algorithm 1 Greedy algorithm A_I

Input: A many-valued decision table T with 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 one 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'), 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
```

1) Uncertainty Measures: Uncertainty measure U is a function from the set of nonempty many-valued decision tables 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 many-valued decision table having n conditional attributes, $N = N(T)$ examples and its examples be labeled with sets containing m different decisions $d_1, \ldots, d_m$. For $i = 1, \ldots, m$, let N_i be the number of examples in T that has been attached with sets of decisions containing the decision d_i, and $p_i = N_i/N$. Let $d_1, \ldots, d_m$ be ordered such that $p_1 \geq \cdots \geq p_m$, then for $i = 1, \ldots, m$, we denote by $N_i^{'}$ the number of examples in T such that the set of decisions attached to example contains d_i, and if $i > 1$ then this set does not contain $d_1, \ldots, d_{i-1}$, and $p_i^{'} = N_i^{'}/N$. We have the following four 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 examples and number of examples with most common decision.
- Sorted entropy: $entSort(T) = -\sum_{i=1}^{m} p_i^{'} \log_2 p_i^{'}$ ([14]). First we sort the probabilities for all decisions. After that, for each example, keep the decision having maximum probability and discard others. Then 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, $q_i = 1 - p_i$. ([16]).

- Absent: $abs(T) = \prod_{i=1}^{m} q_i$, where $q_i = 1 - p_i$. It measures the multiplication of all absent probability q_i's.

2) Impurity Functions: Let U be an uncertainty measure, $f_i \in E(T)$, 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 three types of impurity function I which gives us the impurity $I(T, f_i)$ of this partition.

- Weighted max (wm): $I(T, f_i) = \max_{1 \le j \le t} U(T_j)N(T_j)$.
- Weighted sum (ws): $I(T, f_i) = \sum_{j=1}^{t} U(T_j)N(T_j)$.
- Multiplied weighted sum ($Mult\_ws$): $I(T, f_i) = (\sum_{j=1}^{t} U(T_j)N(T_j)) \times \log_2 t$.

IV. Greedy Algorithms for Decision Tree Construction

Let I be an impurity function based on the uncertainty measure U. The greedy algorithm A_I, for a given many-valued decision table T, constructs a decision tree $A_I(T)$ for T (see Algorithm 1).

It constructs decision tree sequentially in a top-down fashion. It greedily chooses one attribute at each step based on the considered impurity function. We have total 12 (= 4×3) algorithms. The complexities of these algorithms are polynomially bounded above by the size of the tables.

V. Data Sets

We consider five decision tables from UCI Machine Learning Repository [10]. 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 example. For the sake of experiments, we removed from these tables more conditional attributes. As a result, we obtained inconsistent decision tables which contain equal examples with equal or different decisions. The information about obtained inconsistent (represented as many-valued decision) tables can be found in Table IV. These modified tables have been renamed as the name of initial table with an index equal to the number of removed conditional attributes.

We also consider five decision tables from KEEL [11] multi-label data set repository. Note that, these tables are already in many-valued decision format. The information about these table can be found in Table V. The decision table 'genbase' has one attribute with unique value for each example and therefore, it was removed, and renamed as 'genbase-1'.

Table IV and V also contain the number of examples (column "Row"), the number of attributes (column "Attr"), the total number of decisions (column "Label"), the cardinality of decision (column "lc"), the density of decision (column "ld"), and the spectrum of this table (column "Spectrum"). The decision cardinality, lc, is the average number of decisions for each example in the table. The decision density, ld, is the average number of decisions for each example divided by the total number of decisions. If T is a many-valued decision table with N examples (x_i, D_i) where $i = 1, \ldots, N$, then $lc(T) = \frac{1}{N}\sum_{i=1}^{N} |D_i|$, where $|D_i|$ is the cardinality of decision set in i-th example, and $ld(T) = \frac{1}{|L|} lc(T)$, where L is the total number of decisions in T. Spectrum of a many-valued decision table is a sequence $\#1$, $\#2$,..., where $\#i$, $i = 1, 2, \ldots$, is the number of examples labeled with sets of decisions with the cardinality equal to i. For some tables (marked with * in Table V), the spectrum is too long to fit in the page width. Hence, we show what allows in the page width limit.

VI. Comparison of Three Approaches

We compared the three approaches MVD, MCD, and GD to handle inconsistency. We have published the results of the comparison using the decision tree complexity (depth, average depth and number of nodes) in [17]. For the sake of the discussion, we reproduced the result in Table VI for the above 10 decision tables using the algorithm A_I (see Algorithm 1) which uses misclassification error uncertainty measure and weighted sum impurity type.

Data sets from KEEL are already in MVD format. These tables are converted into formats MCD (in this case, the first decision is selected from the set of decisions attached to a row) and GD by the procedure described in Section I. Conversely, inconsistent tables from UCI ML Repository were converted into MVD, MCD and GD. We then used such data sets to construct decision trees by the algorithm A_I, and further we listed the depth, average depth and number of nodes in the constructed decision trees. Note that, we interpreted single valued decision tables, i.e. T_{GD}, T_{MCD}, as many-valued decision tables where each row is labeled with a set of decisions that has one element. Hence, we can apply the same algorithm for all three cases.

Table VI shows the result of depth, average depth and number of nodes for decision trees $A_I(T_{MVD})$, $A_I(T_{GD})$ and $A_I(T_{MCD})$. Moreover, we took average among the 10 data sets. Since, the result varies in the range of the parameter, we took the normalized average. The normalization has been done by taking the value and dividing by the maximum of three approaches. For example, the maximum depth of the three approaches for the table 'bibtex' is 43. Then the normalized depth of MVD approach will be 39/43 = 0.91. Similarly, the normalized depth of MCD approach will be 42/43 = 0.98, and for GD will be 1.

If we look at the result, the MVD approach in many cases gives minimum depth, average depth and number of nodes. When we took the normalized average, this claim is pretty clear. If our goal is to represent knowledge from the given data using the decision tree, we should use MVD approach as it produces simpler trees compared to other approaches. Therefore, we have used MVD approach for the rest of the paper.

TABLE IV: Characteristics of modified UCI inconsistent data represented in MVD format

Decision table T	Row	Attr	Label	lc	ld	Spectrum		
						#1	#2	#3
CARS-1	432	5	4	1.43	0.36	258	161	13
FLAGS-5	171	21	6	1.07	0.18	159	12	
LYMPHOGRAPHY-5	122	13	4	1.07	0.27	113	9	
NURSERY-1	4320	7	5	1.34	0.27	2858	1460	2
ZOO-DATA-5	42	11	7	1.14	0.16	36	6	

TABLE V: Characteristics of KEEL multi-label data

Decision table T	Row	Attr	Label	lc	ld	Spectrum								
						#1	#2	#3	#4	#5	#6	#7	#8	#9
$bibtex^*$	7355	1836	159	2.41	0.015	2791	1825	1302	669	399	179	87	46	18
COREL5K	4998	499	374	3.52	0.009	3	376	1559	3013	17	0	1	0	0
$enron^*$	1561	1001	53	3.49	0.066	179	238	441	337	200	91	51	15	3
GENBASE-1	662	1186	27	1.47	0.054	560	58	31	8	2	3	0	0	0
MEDICAL	967	1449	45	1	0.027	741	212	14	0	0	0	0	0	0

TABLE VI: Depth, average depth, and number of nodes for decision trees $A_I(T_{MVD})$, $A_I(T_{GD})$ and $A_I(T_{MCD})$ for UCI and KEEL data sets using misclassification error uncertainty measure and weighted sum impurity type

Decision table T	Depth			Average Depth			Number of Nodes		
	MVD	MCD	GD	MVD	MCD	GD	MVD	MCD	GD
BIBTEX	39	42	43	11.52	12.24	12.97	9357	10583	13521
CARS-1	5	5	5	1.958	2.583	3.813	43	101	280
COREL5K	156	156	157	36.1	36.41	36.29	6899	8235	9823
ENRON	28	26	41	9.18	9.62	11.18	743	1071	2667
FLAGS-5	6	6	6	3.754	3.801	3.836	210	216	223
GENBASE-1	12	12	11	4.718	4.937	5.762	43	49	81
LYMPHOGRAPHY-5	7	7	7	3.787	4.115	4.311	77	94	112
MEDICAL	16	16	16	8.424	8.424	8.424	747	747	747
NURSERY-1	7	7	7	2.169	3.469	4.127	198	832	1433
ZOO-DATA-5	4	7	7	3.214	3.714	4.119	19	25	41
AVERAGE	28	28.4	30	8.48	8.93	9.48	1833.6	2195.3	2892.8
NORMALIZED AVERAGE	0.92	0.96	0.99	0.82	0.9	0.99	0.56	0.7	1

VII. Decision Tree Optimization

We can optimize the depth, average depth and number of nodes of the constructed decision tree based on dynamic programming algorithms as shown in [6], [8], [7]. It builds all possible separable subtables from the root to the leaf. After that, it considers all possible decision trees by the bottom up approach based on the given criteria of minimizing depth or average depth, or number of nodes. We have compared the average relative difference (in %) $ARD = \frac{greedy-optimal}{optimal} \times 100$ between the sub-optimal results from the greedy algorithms and optimal result from the dynamic programming algorithm. ARD shows how close the greedy result compared to the optimal solution. We have produced the ARD results in Table VIIa, VIIb and VIIc for 3 top algorithms from 12 algorithms (see Section IV).

TABLE VII: ARD (in %) between results of greedy and dynamic algorithms

Algorithm	ARD
$ws\_entSort$	4.58
ws_entML	5.47
ws_me	6.03

(a) Average depth

Algorithm	ARD
wm_me	12.08
$wm\_entSort$	12.08
ws_me	12.08

(b) Depth

Algorithm	ARD
$Mult\_ws\_entML$	21.22
$Mult\_ws\_entSort$	24.58
ws_abs	25.03

(c) Number of nodes

VIII. Decision Tree Classifier

The examples in the many-valued decision table T have been attached with sets of decisions $D \subset L$ where L is the set of all possible decisions in the table T. We denote $D(x)$ as the set of decisions attached to the example x. If X is the domain of the examples to be classified, the goal is to find a classifier $h : X \to L$ such that $h(x) = d$, where $d \in D(x)$, that means to find a decision from the ground truth set of decisions attached to the example. To solve the problem, we use decision tree as our model. We construct different kinds of decision trees using various impurity functions.

A. Evaluation Measure

Here we use the common evaluation measure of classification error percentage. Let us assume, we have unknown instance x' and corresponding decision set is $D(x')$. The classifier h is applied on the new instance x' and it gives the decision $d = h(x')$. If $d \in D(x)$ then $error(x') = 0$, otherwise $error(x') = 1$. Let us assume, we have total M unknown instances to classify, then the error rate will be $\frac{1}{M}\sum_{i=1}^{M} error(x_i)$.

B. Methodology

Let T be a many-valued decision table with conditional attributes $f_1, \ldots, f_n$, and the decision attribute D. We have to divide the initial subtable into three subtables: training subtable T_1, validation subtable T_2, and test subtable T_3. The subtable T_1 is used for construction of initial classifier. The subtable T_2 is used for pruning of the initial tree. Let Γ is a decision tree for T_1. For each node v of Γ, we construct a subtable $T_1(v)$ of the table T_1. If v is the root, then $T_1(v) = T_1$, otherwise $T_1(v) = T_1(f_{i_1}, a_1) \ldots (f_{i_m}, a_m)$ where $f_{i_1}, \ldots, f_{i_m}$ are the conditional attributes attached to nodes of the path from the root of Γ to v, and $a_1, \ldots, a_m$ are numbers attached to edges of this path.

We denote $\alpha(v) = U(T_1(v))/U(T_1)$, where $U(T_1)$ is the misclassification error uncertainty of table T_1. Let Γ contain t nonterminal nodes, and $v_1, \ldots, v_t$ be all nonterminal nodes of Γ in an order such that $\alpha(v_1) \leq \alpha(v_2) \ldots \leq \alpha(v_t)$. For any $i \in \{1, \ldots, t-1\}$, if $\alpha(v_i) = \alpha(v_{i+1})$ then the distance from the root of Γ to v_i is at least the distance from the root to v_{i+1}. We now construct a sequence of decision trees $\Gamma_0, \Gamma_1, \ldots, \Gamma_t$ where $\Gamma_0 = \Gamma$ (initial tree). The procedure of such decision tree construction is described below in an inductive way:

Let assume that, for some $i \in 0, \ldots, t-1$, the decision tree Γ_i is already constructed. We now construct the decision tree Γ_{i+1} from the decision tree Γ_i. Let D be a subtree of Γ_i with the root v_{i+1}. We remove all nodes and edges of D from Γ_i with the exception of v_{i+1}. After that, we transform the node v_{i+1} into a terminal node which is labeled with the most common decision for $T_1(v_{i+1})$. As a result, we obtain the decision tree Γ_{i+1}.

For $i = 0, \ldots, t$, we used the decision tree Γ_i to calculate the classification error rate for the table T_2. We choose the tree Γ_i which has the minimum classification error rate (in case of tie, we choose the tree with smaller index). Now this tree can be used as the final classifier and we can evaluate the test error rate by using this tree to classify the examples in table T_3.

IX. Statistical Comparison of Greedy Algorithms

To compare the algorithms statistically, we used Friedman test with the corresponding Nemenyi post-hoc test as suggested in [9]. 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 of classification error rates, 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, 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.

X. Classification Results

We used 3-fold cross validation to separate test and training data set for each decision table. The data set is divided into 3 folds, we run the experiment for 3 times. At i-th ($i = 1, 2, 3$) iteration, i-th fold is used as the test subset, and the rest of data is partitioned randomly into train (70%) and validation subset (30%). The validation subset is used to prune the trained tree. We successively prune the nodes of the trained decision tree model based on the accuracy of the classifier from validation data set unless its accuracy is maximum. After pruning, we used trained decision tree model to predict the decisions for test data sets. For each fold, we repeat the experiment 5 times and take the average of 5 error rates.

We have four uncertainty measures (me, abs, $entSort$, $entML$) and three types of impurity functions (ws, wm, $Mult\_ws$). So, 12 greedy algorithms have been compared. We show the names of the algorithms as combined name of heuristic and impurity function types separated by '_' in CDD. For example, if the algorithm name is wm_me, this means it uses wm as a type of impurity function and me as uncertainty measure. Figure 3 shows the CDD containing average 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, $Mult\_ws\_abs$ is the best ranked algorithm to minimize the test error. We have shown classification error rate for each data sets for the three best ranked algorithm in Table VIII as well as the average error rate (AER) among all the data sets. We can see that for most of the data sets $Mult\_ws\_abs$ gives minimum error rate than others. On average it gives the best result. We have also shown the overall execution time for the three best ranked algorithm in the Table IX and found that the $Mult\_ws\_entML$ algorithm is faster than other algorithms.

Also note that, the $Mult\_ws\_entML$ algorithm is the best for minimizing the number of nodes in the tree (see Section VII), and it is also one of the top algorithms for minimizing the classification error rates. Also there are two algorithms ($Mult\_ws\_entML$, and $Mult\_ws\_entSort$) for minimizing number of nodes in the tree intersects with the same two

algorithms for minimizing the classification error rates. This result is interesting as we can relate the classification and optimization problem. It looks like the quality of classification is connected with the quality of minimizing the number of nodes.

XI. Conclusion

We studied three different approaches to handle inconsistent decision tables and found MVD approach performs better. We also have created different greedy algorithms based on various uncertainty measures and impurity types to construct decision trees, and compared the results with the optimal results. Finally, we compared these greedy algorithms statistically for classification task to get best ranked classifier and considered also the average error rate across all data sets. We found that $Mult\_ws\_abs$ gives lowest classification error rate than others for most of the data sets. We also found $Mult\_ws\_entML$ algorithm is faster than other top algorithms and also good for both classification and optimization of number of nodes.

In the future, our goal is to construct ensemble of decision trees to work with larger data sets efficiently. Also we are planning to consider more sophisticated pruning methods based on Pareto-optimal points using dynamic programming algorithms.

Acknowledgement

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

References

[1] K. DembczyÅĎski, S. Greco, W. KotÅĆowski, and R. SÅĆowiÅĎski, "Optimized generalized decision in dominance-based rough set approach," in *Rough Sets and Knowledge Technology*, ser. Lecture Notes in Computer Science, 2007, vol. 4481, pp. 118–125.

[2] J. Mingers, "An empirical comparison of selection measures for decision-tree induction," *Machine Learning*, vol. 3, no. 4, pp. 319–342, 1989. doi: 10.1007/BF00116837. [Online]. Available: http://dx.doi.org/10.1007/BF00116837

[3] 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

[4] 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*. CEUR-WS.org, 2012, vol. 928.

[5] 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.

[6] M. Azad and M. Moshkov, "Minimization of decision tree average depth for decision tables with many-valued decisions," *Procedia Computer Science*, vol. 35, no. 0, pp. 368 – 377, 2014. doi: http://dx.doi.org/10.1016/j.procs.2014.08.117

[7] ——, "Minimization of decision tree depth for multi-label decision tables," in *Granular Computing (GrC), 2014 IEEE International Conference on*, vol. 0. IEEE, 2014. doi: 10.1109/GRC.2014.6982798

[8] ——, "Minimizing size of decision trees for multi-label decision tables," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, vol. 0. IEEE, 2014. doi: 10.15439/2014F256

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

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

[11] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

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

[13] 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.

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

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

[16] 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.

[17] M. Azad and M. Moshkov, "'misclassification error' greedy heuristic to construct decision trees for inconsistent decision tables," in *International Conference on Knowledge Discovery and Information Retrieval*. SCITEPRESS, 2014, pp. 184–191.

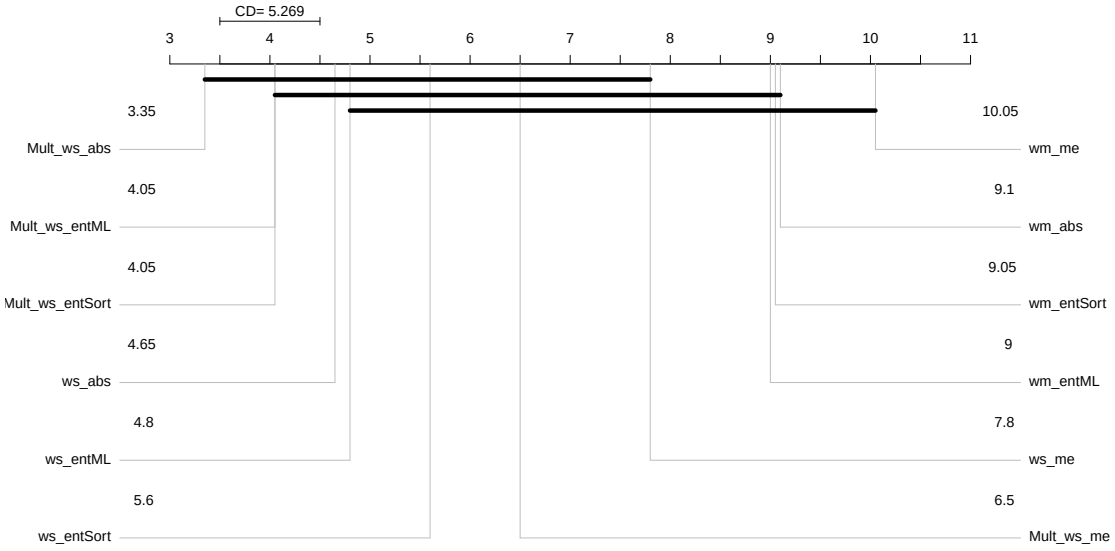

Fig. 3: Critical difference diagram for classification error rates

TABLE VIII: Classification error rate (in %)

Filename	*Mult_ws_abs*	*Mult_ws_entSort*	*Mult_ws_entML*
BIBTEX	56.87	60.09	57.09
CARS-1	3.33	4.49	5.56
COREL5K	74.3	76.57	77.72
ENRON	36.9	26.96	29.69
FLAGS-5	58.83	62.11	63.27
GENBASE-1	5.73	3.79	3.69
LYMPHOGRAPHY-5	27.18	25.4	25.87
MEDICAL	24.05	26.66	26.6
NURSERY-1	2.06	2.69	2.62
ZOO-DATA-5	22.86	27.62	25.24
AER (AVERAGE ERROR RATE)	31.21	31.64	31.73

TABLE IX: Overall execution time (in sec)

Filename	*Mult_ws_abs*	*Mult_ws_entSort*	*Mult_ws_entML*
BIBTEX	285.42	2012.34	117.55
CARS-1	0.0046	0.006	0.0042
COREL5K	127.95	853.3	82.19
ENRON	6.82	14.35	5.45
FLAGS-5	0.0098	0.0176	0.0102
GENBASE-1	0.1174	0.17	0.111
LYMPHOGRAPHY-5	0.0046	0.0056	0.0042
MEDICAL	1.2352	6.9238	1.1488
NURSERY-1	0.0408	0.0622	0.0416
ZOO-DATA-5	0.0024	0.0028	0.0028
AVERAGE	42.16	288.72	20.65