Sorting by Decision Trees with Hypotheses

Mohammad Azad [1], Igor Chikalov [2], Shahid Hussain[3] and Mikhail Moshkov [4]

[1]Jouf University, Sakaka 72441, Saudi Arabia

[2]Intel Corporation, Arizona 85226, USA

[3]Institute of Business Administration, University Road, Karachi 75270, Pakistan

[4]King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia

28 September 2021

Agenda

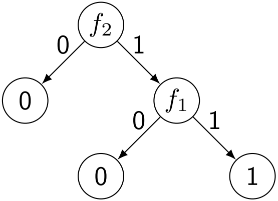
Three domains

- Decision trees are widely used in many areas of computer science as classifiers, as a means for knowledge representation, and as algorithms to solve various problems of computational geometry, combinatorial optimization, etc.
- They are studied in
  - Test theory (initiated by **Chegis** and **Yablonskii**),
  - Rough set theory (initiated by **Pawlak**),
  - Exact learning (initiated by **Angluin**).
- These theories are closely related.

- Our aim (mainly theoretical) is to understand whether it is possible to decrease the number of nodes as well as the depth in decision trees if we use (additionally) hypotheses.

Attribute vs. Hypothesis

$$T = \begin{array}{|cc|c|} \hline f_1 & f_2 & \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ \hline \end{array}$$

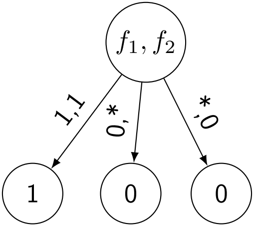


- Let T be a decision table with n conditional attributes $f_1, \ldots, f_n$ having values from the set $\omega = \{0, 1, 2, ...\}$.
- The rows of this table T are pairwise different and each row is labeled with a decision from ω.
- For a given row of T, we should recognize the decision attached to this row.
- To this end, we can use decision trees based on two types of queries.
- We can ask about the value of an attribute $f_i \in \{f_1, \ldots, f_n\}$ on the given row.
- We will obtain an answer of the kind $f_i = \delta$, where δ is the number at the intersection of the given row and the column f_i.

Attribute vs. Hypothesis...

$T =$

f_1	f_2	
0	0	0
0	1	0
1	0	0
1	1	1



- We can also ask if a hypothesis $f_1 = \delta_1, \ldots, f_n = \delta_n$ is true, where $\delta_1, \ldots, \delta_n$ are numbers from the columns $f_1, \ldots, f_n$ respectively.
- Either this hypothesis will be confirmed or we obtain a counterexample in the form $f_i = \sigma$, where $f_i \in \{f_1, \ldots, f_n\}$ and σ is a number from the column f_i different from δ_i.
- The considered hypothesis is called proper if $(\delta_1, \ldots, \delta_n)$ is a row of the table T.

Attribute vs. Hypothesis...

- Decision trees using hypotheses can be more efficient than the decision trees using only attributes.
- As an example, let us consider the problem of computation of the conjunction $x_1 \wedge \cdots \wedge x_n$. The minimum number of realizable nodes in a decision tree solving this problem using the attributes $x_1, \ldots, x_n$ is equal to $2n + 1$.
- However, the minimum number of realizable nodes in a decision tree solving this problem using proper hypotheses is equal to $n + 2$: it is enough to ask only about the hypothesis $x_1 = 1, \ldots, x_n = 1$. If it is true, then the considered conjunction is equal to 1. Otherwise, it is equal to 0. The obtained decision tree contains one non terminal node and $n + 1$ terminal nodes.

Types of Decision Trees

We consider the following **five types** of decision trees. Decision trees that use:

- **Type 1:** *only attributes*.
- **Type 2:** *only hypotheses*.
- **Type 3:** *both attributes and hypotheses*.
- **Type 4:** *only proper hypotheses*.
- **Type 5:** *both attributes and proper hypotheses*.

Decision Tables

- Subtable TS of T is obtained by removing one or more rows from it;
- $T(f_{i_1} = a_1) \dots (f_{i_m} = a_m)$ is a separable subtable of T;

$T =$

	f_1	f_2	
r_1	1	0	1
r_2	0	0	2
r_3	0	1	3

$; T(f_1 = 0) =$

	f_1	f_2	
r_2	0	0	2
r_3	0	1	3

- $|SEP(T)|$ is the number of different separable subtables of T.

Parameters of decision trees

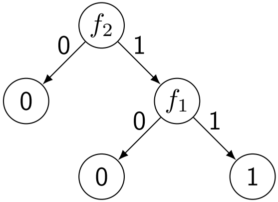
Let Γ be a decision tree for T. As the space complexity of the decision tree Γ, we consider the number of its realizable relative to T nodes. A node v of Γ is called realizable relative to T if and only if the subtable $TS(\Gamma, v)$ is nonempty. We denote by $L(T, \Gamma)$ the number of nodes in Γ that are realizable relative to T.

As the time complexity of a decision tree, we consider its depth that is the maximum number of working nodes in a complete path in the tree. We denote by $h(\Gamma)$ the depth of a decision tree Γ.

- $h^{(k)}(T)$ denotes the minimum depth of a decision tree of the type k for T, $k = 1, \ldots, 5$.
- $L^{(k)}(T)$ denotes the minimum number of nodes realizable relative to T in a decision tree of the type k for T, $k = 1, \ldots, 5$.

Decision Trees

$$T = \begin{array}{|cc|c|} \hline f_1 & f_2 & \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ \hline \end{array}$$



- Greedy heuristics;
- Ant colony algorithms;
- Genetic algorithms;
- Branch and bound techniques;
- **Directed acyclic graph (DAG) construction using dynamic programming (DP)**.

Agenda

1. Preliminaries
2. Design of decision trees
3. Results of Experiments
4. Discussion

DAG $\Delta(T)$

Algorithm $\mathcal{B}_0$

1. Construct the graph that consists of one node T which is not marked as processed.
2. If all nodes of the graph are processed, then the work of algorithm is finished. Return the resulting graph as $\Delta(T)$. Otherwise, choose a node (table) Θ that has not been processed yet.
3. a. If Θ is degenerate, mark the node Θ as processed and proceed to step 2.

   b. If Θ is not degenerate, then for each $f_i \in E(\Theta)$, draw a bundle of edges from the node Θ (this bundle of edges will be called the f_i-*bundle*). Let $E(\Theta, f_i) = \{a_1, \ldots, a_k\}$. Then, draw k edges from Θ and label these edges with the pairs $(f_i, a_1), \ldots, (f_i, a_k)$. These edges enter nodes $\Theta(f_i, a_1), \ldots,$ $\Theta(f_i, a_k)$, respectively. If some of the nodes $\Theta(f_i, a_1), \ldots, \Theta(f_i, a_k)$ are not present in the graph, then add these nodes to the graph. Mark the node Θ as processed and return to step 2.

Example

f_1	f_2	
0	0	0
0	1	0
1	0	0
1	1	1

Example

f_1	f_2	
0	0	0
0	1	0
1	0	0
1	1	1

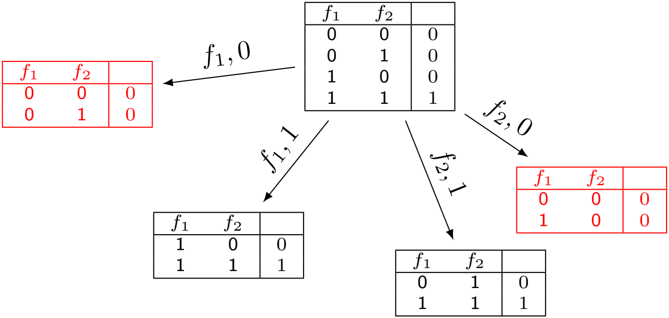
$f_1, 0$

f_1	f_2	
0	0	0
0	1	0

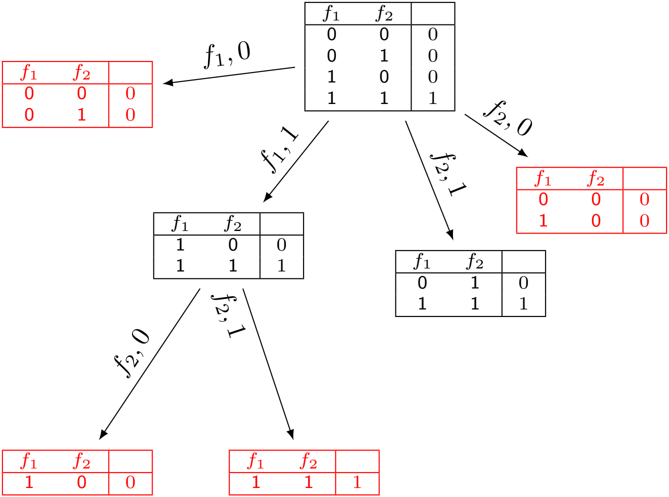
$f_1, 1$

f_1	f_2	
1	0	0
1	1	1

Example

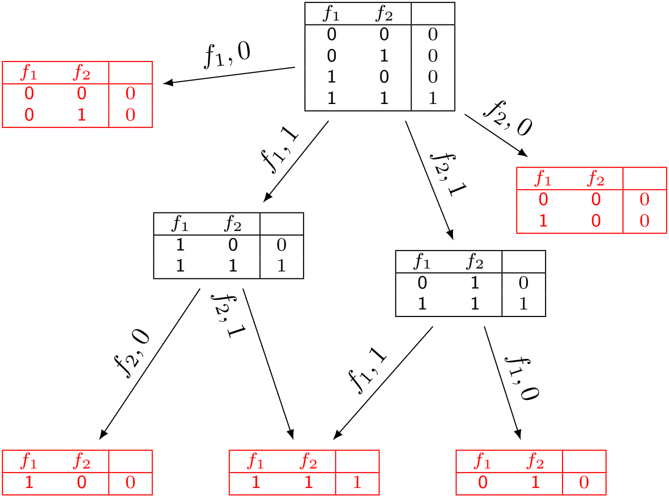

f_1 f_2
0 0 0
0 1 0
1 0 0
1 1 1
$f_1, 0$
f_1 f_2
0 0 0
0 1 0
$f_1, 1$
f_1 f_2
1 0 0
1 1 1
$f_2, 1$
f_1 f_2
0 1 0
1 1 1
$f_2, 0$
f_1 f_2
0 0 0
1 0 0

Example



f1 f2
0 0 0
0 1 0
1 0 0
1 1 1
f1, 0
f1 f2
0 0 0
0 1 0
f1, 1
f1 f2
1 0 0
1 1 1
f2, 1
f1 f2
0 1 0
1 1 1
f2, 0
f1 f2
0 0 0
1 0 0
f2, 0
f1 f2
1 0 0
f2, 1
f1 f2
1 1 1

Example



f1 f2
0 0 0
0 1 0
1 0 0
1 1 1
f1, 0
f1 f2
0 0 0
0 1 0
f2, 0
f1 f2
0 0 0
1 0 0
f1, 1
f1 f2
1 0 0
1 1 1
f2, 1
f1 f2
0 1 0
1 1 1
f2, 0
f1 f2
1 0 0
f2, 1
f1, 1
f1 f2
1 1 1
f1, 0
f1 f2
0 1 0

Time Complexity of Algorithm $\mathcal{B}_0$

We now analyze time complexity of the algorithm $\mathcal{B}_0$.

Proposition

The time complexity of the algorithm $\mathcal{B}_0$ is bounded from above by a polynomial on the size of the input table T and the number $|SEP(T)|$ of different separable subtables of T.

Algorithm $\mathcal{B}_t$ (computation of $L^{(t)}(T)$).

Input: A nonempty decision table T and the directed acyclic graph $\Delta(T)$.

Output: The value $L^{(t)}(T)$.

- If a number is attached to each node of the DAG, then return the number attached to the node T as $L^{(t)}(T)$ and halt the algorithm. Otherwise, choose a node Θ of the graph $\Delta(T)$ without attached number, which is either a terminal node of $\Delta(T)$ or a nonterminal node of $\Delta(T)$ for which all children have attached numbers.
- If Θ is a terminal node, then attach to it the number $L^{(t)}(\Theta) = 1$ and proceed to step 1.

Algorithm $\mathcal{B}_t$ (computation of $L^{(t)}(T)$).

- If Θ is not a terminal node, then depending on the value t do the following:
  - In the case $t = 1$, compute the value $L_a^{(1)}(\Theta)$ and attach to Θ the value $L^{(1)}(\Theta) = L_a^{(1)}(\Theta)$.
  - In the case $t = 2$, compute the value $L_h^{(2)}(\Theta)$ and attach to Θ the value $L^{(2)}(\Theta) = L_h^{(2)}(\Theta)$.
  - In the case $t = 3$, compute the values $L_a^{(3)}(\Theta)$ and $L_h^{(3)}(\Theta)$, and attach to Θ the value $L^{(3)}(\Theta) = \min\{L_a^{(3)}(\Theta), L_h^{(3)}(\Theta)\}$.
  - In the case $t = 4$, compute the value $L_p^{(4)}(\Theta)$ and attach to Θ the value $L^{(4)}(\Theta) = L_p^{(4)}(\Theta)$.
  - In the case $t = 5$, compute the values $L_a^{(5)}(\Theta)$ and $L_p^{(5)}(\Theta)$, and attach to Θ the value $L^{(5)}(\Theta) = \min\{L_a^{(5)}(\Theta), L_p^{(5)}(\Theta)\}$.

  Proceed to first step.

Sorting problem

Let $x_1, \ldots, x_n$ be pairwise different elements from a linearly ordered set. We should find a permutation $(p_1, \ldots, p_n)$ from the set P_n of all permutations of the set $\{1, \ldots, n\}$ for which $x_{p_1} < \cdots < x_{p_n}$. To this end, we use attributes $x_i : x_j$ such that $i, j \in \{1, \ldots, n\}$, $i < j$, $x_i : x_j = 1$ if $x_i < x_j$, and $x_i : x_j = 0$ if $x_i > x_j$.

The problem of sorting n elements can be represented as a decision table T_n with $n(n-1)/2$ conditional attributes $x_i : x_j$, $i, j \in \{1, \ldots, n\}$, $i < j$, and $n!$ rows corresponding to permutations from P_n.

For each permutation $(p_1, \ldots, p_n)$, the corresponding row of T_n is labeled with this permutation as the decision. This row is filled with values of attributes $x_i : x_j$ such that $x_i : x_j = 1$ if and only if i stays before j in the tuple $(p_1, \ldots, p_n)$.

Agenda

1. Preliminaries
2. Design of decision trees
3. Results of Experiments
4. Discussion

Experimental Results (Depth)

n	$h^{(1)}(T_n)$	$h^{(2)}(T_n)$	$h^{(3)}(T_n)$	$h^{(4)}(T_n)$	$h^{(5)}(T_n)$
3	3	2	2	2	2
4	5	4	4	4	4
5	7	6	6	6	6
6	10	9	9	9	9

Experimental results (Number of nodes)

n	$L^{(1)}(T_n)$	$L^{(2)}(T_n)$	$L^{(3)}(T_n)$	$L^{(4)}(T_n)$	$L^{(5)}(T_n)$
3	11	13	9	14	9
4	47	253	39	254	39
5	239	15,071	199	15,142	199
6	1,439	2,885,086	1,199	2,886,752	1,199

Agenda

1. Preliminaries
2. Design of decision trees
3. Results of Experiments
4. Discussion

Discussion and future direction

- In this paper, we studied modified decision trees that use queries based on one attribute each and queries based on hypotheses about values of all attributes.
- We proposed dynamic programming algorithms for the minimization of depth and the number of realizable nodes in such decision trees for sorting problem.
- From the obtained experimental results it follows that the decision trees of the types 2–5 can have less depth than the decision trees of the type 1. Decision trees of the types 3 and 5 can have less number of realizable nodes than the decision trees of the type 1. Decision trees of the types 2 and 4 have too many nodes.
- We are planning to study bi-criteria optimization for the decision trees with hypothesis.

Thank You