High Computational Complexity of the Decision Tree Induction with many Missing Attribute Values

Rafał Latkowski
Institute of Computer Science, Warsaw University
ul. Banacha 2, 02–097 Warsaw, Poland
rlatkows@mimuw.edu.pl

Abstract

The decision tree induction is a widely applied technique in machine learning. Algorithms based on such technique with careful implementation can reach the computational complexity $\Omega(n \log n)$. However, a special treatment of decision trees is needed in case of missing values. The most popular and widely used approach to this problem is to divide such cases among all subsets (sub-nodes). Such an approach is implemented, e.g., in *C4.5*, *RSES-Lib*, *WEKA* and many other machine learning programs. In this paper we show that this strategy leads to a significant increase of computational complexity of the decision tree induction algorithm for data sets with many missing values.

Keywords: decision trees, missing attribute values, computational complexity

1 Introduction

The decision tree induction is a widely used technique in machine learning. It has a lot of advantages in comparison to other machine learning methods like, e.g., low computational complexity, high accuracy or readability of used data structures. The decision tree induction algorithm was proposed by Hunt in 1966 [5] and as a general scheme of recursive partitioning remains actual up to present-day. Although there exist other decision tree methods (like, e.g., lazy decision trees) in this paper we are focusing on the standard decision trees that are induced by recursive partitioning, i.e., TDIDT — Top Down Induction of Decision Trees.

There exist a plenty of different algorithms for decision tree induction and their implementations. With careful implementation such algorithms can reach the computational complexity $\Omega(n \log n)$ with respect to the number n of cases in training set (see, e.g., [8]). Of course the computational complexity depends not only on the number of cases, but also on the number of attributes (columns in data table), on the number of decision classes and on the number of possible branches at the tree nodes. These factors, however, add a liner component into the complexity estimation and are neglected in the paper to simplify the considerations. We are making further simplifications and only binary decision trees are considered. It is a known fact that trees with more than two branches can be represented by slightly bigger binary trees. The second simplification is that each inner node can test the value of only one attribute. The decision trees that consider one attribute in each test are very standard and commonly used. All other decision trees can be represented by such trees with "unary-tests" after the attribute basis transformation. This transformation is naturally constructed from formulas included in complex tests.

Initially the decision tree induction algorithms were not capable to deal with missing attribute values, like, e.g., Quinlan's ID3 [9]. With continuous advance in data gathering and machine

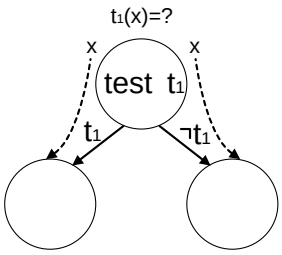

Figure 1: The simple partitioning strategy. If the value of the test t_1 in the decision tree node cannot be determined due to the missing values in object x (i.e., $t_1(x) =?$), then object x is assigned into all subsets (sub-nodes) of this node.

learning it turned out that the problem of missing attribute values is very significant (see, e.g., [2, 4]). Thanks to the simple scheme of decision tree algorithm it was rather easy to adapt this algorithm scheme to the case of missing values. A lot of different ideas on this problem were prosed. Quinlan in [10] summarized and empirically evaluated the approaches how to deal with missing attributes in decision tree induction. All of these approaches have different impact on the computational complexity of the decision tree induction.

The approaches dealing with missing values in decision tree induction can be characterized by three independent parts (c.f. [10]):

- Evaluation of a test (in a tree node). This concern the strategy of how to evaluate different tests when each attribute has a different amount of missing values.
- Partitioning the training set using a test. This concern the strategy where assign a case with missing value on an attribute considered in a test.
- Classifying a new case with unknown value of a tested attribute. As distinct from previous two parts it has nothing to do with the induction itself.

This paper concerns only the second part — partitioning strategy, because it has the most important, but frequently disregarded impact on the computational complexity of the algorithms for decision tree induction. The presented results are valid independently from the employed strategy evaluating tests and classifying new cases with missing attribute values.

This paper is organized as follows. In the next section the description of the most commonly used method for partitioning cases is provided. The third section describes the algorithm and its computational complexity. The fourth section provides an estimation of the presented computational complexity. Conclusion and remarks are presented in the final, fifth section.

2 Partitioning cases

There exist a plenty of different algorithms for decision tree induction and their implementations. But almost all of them are based on the same idea of recursive data partitioning described in early works of Hunt and Friedman [4, 5]. The same goes for the strategies of dealing with missing attribute values, especially in partitioning the training set. Although there exist a lot of different strategies of partitioning the training set, only one of them turned out to be helpful in all applications. A good summary of ideas how to partition the training set using a test is provided by Quinlan in [10]. His work influenced a lot the machine learning community in all three aspects of handling missing values presented in previous section. As a result of his experimental evaluation came out that the only superior method for partitioning cases with missing values is dividing such

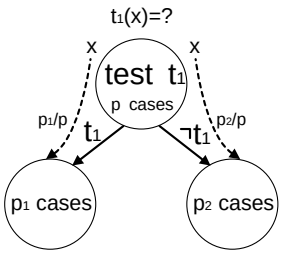

Figure 2: The more advanced partitioning strategy. If the value of the test t_1 in the decision tree node cannot be determined due to the missing values in object x (i.e. $t_1(x) =?$), then object x is partitioned into all subsets (sub-nodes) of this node proportional to the number of cases with known value in each subset (i.e. $\frac{p_1}{p}$ into the sub-node for cases that satisfy test t_1 and $\frac{p_2}{p}$ into the sub-node for cases that do not satisfy test t_1).

cases among all subsets. This scheme of proceeding indeed improve quality and is commonly used not only in Quinlan's *C4.5* algorithms, but also in many other algorithms and software systems like, e.g., Rough Set Exploration System [1] or Waikato Environment for Knowledge Analysis.

The considered approach how to deal with missing attribute values has two different variants that differ a lot in achieved quality but are almost identical in implementation, thus in complexity.

The first method comes from Friedman [4] and proceed as follows (see fig. 2). When partitioning the training set using a test on an attribute a and a training case has unknown value of attribute a, assign this case to each subset. In a usual situation, when a case has a proper value of attribute a, then this case is assigned to only one subset.

The second method comes from Kononenko et al. [6] and using it we proceed as follows (see fig. 2). When partitioning the training set using a test on attribute a and a training case has unknown value of attribute a, assign a fraction of this case to each subset with probability proportional to the number of cases with the known value in each subset.

In real data we are not able to directly represent "a fraction of a case." The only way to represent a fraction of a case is to assign a numerical weight to each case. Initially each case has its weight equals to one. When the algorithm performs partition into all subsets, the weight of a case is reduced proportional to the empirical frequency of a such partition estimated over the cases without missing values on considered attribute (see fig. 2). Such "a fraction of a case" can be partitioned many times, so in some cases "a fraction of a fraction of a case" will be evaluated, which lead to standard computations on rational numbers. Apart from weights, the cases (i.e., their descriptions) must be physically copied as entities that does not support any segmentation. This leads to a conclusion that the both methods (i.e., partitioning "whole cases" and partitioning "fraction of cases") physically distribute all cases with missing description to all subsets (i.e., sub-nodes of a tree). Therefore we can consider only a simplified, first version of this method as the difference in the computational complexity is negligible (i.e., $\mathcal{O}(1)$).

The other well known approach, but not investigated in [10] is to create an additional test for so-called surrogate split that is implemented in *CART* classifier (Breiman et al. [2]). Unlike for the other methods, there are no clear evidence that partitioning to all subsets (i.e., C4.5 classifier) is superior over the method used in CART classifier. There are, however, two important reasons why the surrogate splits are not so spread in other algorithms and software packages. Firstly, the surrogate splits require the reasonable amount of non-missing values in highly correlated attributes. It is not possible to relay on surrogate splits for data with many missing values and such a case is in scope of interest of this paper. Secondly, the surrogate splits are not so easy to implement and provide an additional increase of the computational complexity. For simplicity, many researchers

Figure 3: The partitioning with different amount of missing attribute values. At the figure (a) there are no missing values and the resulting partition is 50%–50%. At the figure (b) there are 50% missing values and the resulting partiton is 75%–75%.

and software developers follow the general scheme of the C4.5 algorithm that is simpler. Presented here results does not apply to the surrogate splits as they do not use object division.

3 Algorithm

The described above method of partitioning cases with missing attribute values influences the algorithm of decision tree induction. The resulting modified algorithm can by summarized in following scheme:

1. Create the decision tree root (starting node) n_0 and assign to it whole training set.
2. Create initially empty queue $\mathcal{Q}$ of nodes and put there the root node n_0.
3. While queue of nodes $\mathcal{Q}$ is not empty, repeat following steps:
   (a) Remove a node n from the queue $\mathcal{Q}$ ($\Theta(1)$).
   (b) Check the stop criterion concerning a training subset s related with node n. If the stop criterion is satisfied then skip further steps (from $\Theta(1)$ to $\Theta(n)$).
   (c) Select the optimal test t_{best} for a training subset s related to the node n ($\Theta(n)$ in a cute implementation).
   (d) Partition objects into two subsets s_l and s_r ($\Theta(n)$).
   (e) Create new nodes n_l and n_r for subsets s_l and s_r and put them into queue $\mathcal{Q}$ The nodes n_l and n_r become children of the node n in the decision tree. ($\Theta(1)$).

For each step of this algorithm its computational cost is provided. As described in introduction, to simplify considerations we concern only a number of objects as variable that influences computational complexity. These computational costs are easy to determine and were determined many times in the past.

The whole algorithm follows the divide&conquer scheme, similarly to, e.g., the Quick Sort algorithm. Here, however, instead of a partitioning element we have a test that assigns objects into two sets and instead of partition procedure we have searching for the optimal test and partitioning elements with selected test. In steps 3.3–3.5 of presented algorithm objects are partitioned into two subsets. In an optimistic case, when all partitions are balanced these subsets are of equal size, and if there are no missing values than they contain 50% of initial set of objects. It is illustrated at the figure 3(a). For such an optimistic case we can easily write a recursive function of computational complexity:

$$\mathrm{T}(n) = n + 2 \cdot \mathrm{T}\left(\left\lceil \frac{n}{2} \right\rceil\right) = n + 2 \cdot \mathrm{T}\left(\left\lceil \frac{1}{2} \cdot n \right\rceil\right). \tag{1}$$

It is well known that $\mathrm{T}(n) = \Theta(n \log n)$. Due to the notation "$\Theta$" it is not important which base of a logarithm we choose, but the original estimation assumes $\Theta(n \log_2 n)$, because each time a set of

objects is partitioned into two equal subsets. Even when the partition is not in the proportion 50%–50%, but for example 90%–10% the computational cost is still $\Theta(n \log n)$, only the hidden constant is slightly higher (see, e.g., [3]). In general we can write that in such a case the computational complexity is $\Omega(n \log n)$, because it is easy to determine the constant for the optimistic case, but in pessimistic case the time raises up to n^2.

Lets consider now the data with the missing attribute values and the algorithm with partitioning of cases with missing values as described in previous section. We will make additional assumptions to concentrate only on brake-down of the computational complexity. Let us assume that missing values are uniformly distributed in the data and the presence of the missing values in different attributes is pairwise independent. Usually this assumption does not reflect the reality, but it is very helpful and there exist some data sets with missing values of natural origin that satisfy such an assumption (see, e.g., [7]). For simplicity, we consider only the optimistical case of a partition, when as many objects as possible are equally partitioned. It is not explicitly written, but it is also useful to assume that the data consists of many attributes (columns), i.e., $k \gtrsim \log n$ (more precisely $k \gtrsim \log_{1/\alpha} n$ for some α defined further). These assumptions are not critical, but they simplify a lot the considerations.

Let us consider a data with missing values in proportion to all values ($n \cdot k$) equals to μ. From assumptions about distribution we can say that each attribute has almost the same proportion of missing values. What is happening if we are applying described above partition method on a such data is illustrated in Figure 3(b) for $\mu = 0.5$. The amount of missing values in subsets with respect to the considered attribute is greater or equal[1] μ. The amount of missing values in all other attributes should remain μ, as long as the assumption of pairwise independency holds. Let's summarize and introduce some new notation:

- μ — the frequency of the missing attribute values, i.e., ratio of objects that have to be partitioned in both subsets.
- $\alpha = \alpha(\mu) = \frac{1}{2} + \frac{\mu}{2}$ — the ratio of objects that is assigned to a subset in the best case, i.e., when the partition is balances (see Figure 3(b), it is the same α as in the assumption on the sufficient number of attributes).
- $\gamma = \gamma(\mu) = \log_\alpha \frac{1}{2} = \log_{\frac{1}{\alpha}} 2$ — exponent of the computational complexity increase, as it is described further.

In subsequent partitions, if there are unused attributes the partition can be still made on a subset with μ missing values with respect to a selected attribute. If there are not enough attributes than the proportion of missing values is greater then μ and the described later the best case does not hold, but the computational complexity brake-down is even greater. To focus on the lower bound of the computational complexity the best, optimistic case is chosen, i.e., the all described above assumption hold, all partitions are as much balanced as it is possible, the test is made on an attribute with minimal number of missing values (i.e., with proportion μ as it is explained above). In such a case the recursive function of the computational complexity is:

$$\mathrm{T}(n, \alpha) = n + 2 \cdot \mathrm{T}(\lceil \alpha \cdot n \rceil, \alpha) \tag{2}$$

If the partition is balanced and there are $\mu \cdot n$ objects with missing value on a tested attribute than each subset contains $\lceil n \cdot (\frac{1-\mu}{2} + \mu) \rceil = \lceil \alpha \cdot n \rceil$ objects. This equation reminds the equation 1, but here the subsets are smaller by a factor α instead of a factor $\frac{1}{2}$ (c.f. Figure 3).

[1] It can happened when the partition is not balanced.

4 Estimation

It is hard to imagine the standard form of the equation 2. Probably because of that the impact of this commonly applied partitioning strategy on the computational complexity is frequently disregarded. As we will see this partitioning strategy cause a brake-down of the computational complexity.

To be precise, we should also mention that the $\mathrm{T}(1,\alpha) = 1$ and $\mathrm{T}(0,\alpha) = 0$. We simplify the equation 2 to eliminate the inconvenient rounding:

$$\widehat{\mathrm{T}}(n,\alpha) = n + 2 \cdot \widehat{\mathrm{T}}(\alpha \cdot n, \alpha). \tag{3}$$

It is easy to observe that $\mathrm{T}(n,\alpha) \geq \widehat{\mathrm{T}}(n,\alpha)$, as long as we specify the adequate start conditions, i.e. $\forall x < \frac{1}{2} : \widehat{\mathrm{T}}(x,\alpha) = 0$.

The function $\widehat{\mathrm{T}}(n,\alpha)$ form a geometric series. We can make a use of this observations if $\alpha \neq \frac{1}{2}$. Moreover from now we will consider $\alpha \in (\frac{1}{2}, 1)$ without boundaries of this interval.

$$\widehat{\mathrm{T}}(n,\alpha) = n + 2 \cdot \widehat{\mathrm{T}}(\alpha \cdot n, \alpha) \tag{4}$$

$$\widehat{\mathrm{T}}(n,\alpha) = n + 2 \cdot ((\alpha \cdot n) + 2 \cdot \widehat{\mathrm{T}}(\alpha \cdot \alpha \cdot n, \alpha)) \tag{5}$$

$$\widehat{\mathrm{T}}(n,\alpha) = n + 2 \cdot \alpha \cdot n + 2^2 \cdot \alpha^2 \cdot n + \ldots + 2^{\lceil \log_{\frac{1}{\alpha}} n \rceil} \cdot \alpha^{\lceil \log_{\frac{1}{\alpha}} n \rceil} \cdot n \tag{6}$$

$$\widehat{\mathrm{T}}(n,\alpha) = n \cdot \sum_{i=0}^{\lceil \log_{\frac{1}{\alpha}} n \rceil} (2\alpha)^i \tag{7}$$

The formula for $w = \lceil \log_{\frac{1}{\alpha}} n \rceil$-th element of a geometric series is:

$$\frac{q^{w+1} - 1}{q - 1} = \frac{q^{\lceil \log_{\frac{1}{\alpha}} n \rceil + 1} - 1}{q - 1} \tag{8}$$

We are making further simplifications to present the compact form of the estimation:

$$\widehat{\mathrm{T}}(n,\alpha) = n \cdot \frac{2\alpha^{\lceil \log_{\frac{1}{\alpha}} n \rceil + 1} - 1}{2\alpha - 1} \geq n \cdot \frac{2\alpha^{(\log_{\frac{1}{\alpha}} n) + 1} - 1}{2\alpha - 1} = \tag{9}$$

$$\ldots = n \cdot \frac{(\frac{n}{\alpha})^{\gamma - 1} - 1}{2\alpha - 1} = \tag{10}$$

$$\ldots = \frac{n^\gamma - (\alpha^{\gamma-1})n}{2\alpha^\gamma - \alpha^{\gamma-1}} \tag{11}$$

In considered domain $\alpha^{\gamma-1} \in (\frac{1}{2}, 1)$. It is easy to observe that formula 11 is $\Theta(n^\gamma)$. Although γ can be small (e.g., in the case of 1% of missing values $\gamma(0.01)$ is about 1.014564), we can easily show that it is $\mathcal{O}(n^\gamma)$ and $\Omega(n^\gamma)$ (continuing example, for $\mu = 0.01$ the inequality $n^\gamma > n^\gamma - n > \frac{1}{10} \cdot n^\gamma$ holds for all $n > 695$). Taking into considerations that $\mathrm{T}(n,\alpha) \geq \widehat{\mathrm{T}}(n,\alpha) \geq \Theta(n^\gamma)$ and that we were considering the optimistic case we can state as follows:

Corollary 1 *The computational complexity of the algorithm for decision tree induction that uses one of the two previously described partitioning strategies is* $\Omega(n^\gamma)$.

This corollary characterizes the computational complexity brake-down when data contains many missing values. The form of the function $\gamma(\mu)$ is in Figure 4. As we can see the $\gamma(\mu)$ is a increasing function. For μ around zero value of γ is around one and as the ratio of missing

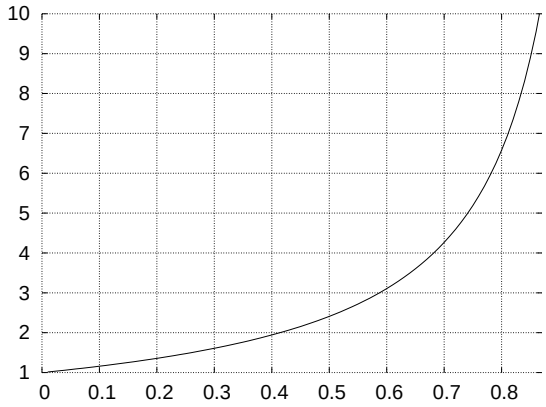

Figure 4: The diagram of the function $\gamma(\mu)$. For $\mu > 0.4$ γ is higher than 2. This provide an explanation why for data with, e.g., 75% missing values the decision trees run enormously long.

values increases up to 1.0 the value of γ increases up to the positive infinity. If we have at least 42% of missing values than the decision tree induction requires more than n^2 time, and similarly for at least 59% of missing values more than n^3. Such a computational cost is unacceptable in many data mining applications, when the number of cases is measured in thousands.

This estimation is not exact. It is clear if the decision tree induction cannot have the computational complexity below of $n \log n$. The provided estimation works fine for data with many missing values, let say more than 40%. The precise investigation of what is happening for data with a small number of missing values remains open.

5 Conclusions

In general, there has been not to much work done on the computational complexity of the algorithms for the decision tree induction. There are only few publications that consider the the computational cost estimation of decision tree induction, but frequently without any formal analysis. The problem of missing values does not appear at all in those considerations on complexity. This is caused by an intuitive simplicity of such algorithms and easiness of determining the computational complexity for them. Probably because of that nothing has been done at all in the area of the computational complexity of decision tree induction with missing attribute values.

In this paper we discussed briefly one of the problems that is hidden in the decision tree induction over incomplete data. It was observed that for data with many missing values the commonly used algorithms for decision tree induction do not return results in reasonable time [7]. In particular data set "rcd128" that describe the positioning problem in Robotic Soccer satisfy all the made above assumptions and contain 75.1% missing values[2]. For such a data commonly used implementations of the decision tree induction work in time worse than n^5. To illustrate the enormous brake-down of the computational complexity one can compare, that $n \log_2 n$ for one billion is less than n^5 for 125. Moreover, data sets maintained by large companies grow together with they owners are very far from any kind of normalization and contain many missing values. For such a data, however, not all assumptions are satisfied, because the distribution of the missing values reflect the company growth, fusions and business strategy. In such a case the experimenter should discover other methods of classifier induction. For example the decomposition method

[2]This data set is available at `http://www.mimuw.edu.pl/~rlatkows/resources/`.

that is described in [7] makes it possible to use decision trees, but after the decomposition that eliminates the necessity of decision tree induction over data with missing attribute values.

As mentioned, this work is a preliminary step in understanding of decision tree induction with many missing values. The investigation of the algorithm behavior with a small number of missing values and an improvement of provided complexity estimation remains for future work.

Acknowledgments

I would like to thank Nguyen Thi Sinh Hoa and Wit Jakuczun for insightful comments on this work. This work was partially supported by the Katholischer Akademischer Ausländer-Dienst.

References

[1] J. G. Bazan, M. S. Szczuka, and J. Wróblewski. A new version of rough set exploration system. In J. J. Alpigini, J. F. Peters, A. Skowron, and N. Zhong, editors, *Rough Sets and Current Trends in Computing, RSCTC 2002*, *LNAI* 2475, pages 397–404. Springer, 2002.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and P. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[4] J. H. Friedman. A recursive partitioning decision rule for non-parametric classification. *IEEE Trasactions on Computer Science*, 26:404–408, 1977.

[5] E. B. Hunt, J. Marin, and P. T. Stone. *Experiments in Induction*. Academic Press, 1966.

[6] I. Kononenko, I. Bratko, and E. Roškar. Experiments in automatic learning of medical diagnostic rules. Technical report, Jozef Stefan Institute, Ljubljana, 1984.

[7] R. Latkowski. On decomposition for incomplete data. *Fundamenta Informaticae*, 54(1):1–16, 2003.

[8] M. Mehta, R. Agrawal, and J. Risanen. SLIQ: A fast scalable classifier for data mining. In *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, 1996.

[9] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[10] J. R. Quinlan. Unknown attribute values in induction. In A. M. Segre, editor, *Proceedings of the Sixth International Machine Learning Workshop*, pages 31–37. Morgan Kaufmann, 1989.