User-Driven Data Preprocessing for Decision Support

Olivier Parisot, Pierrick Bruneau, Yoanne Didry, and Thomas Tamisier

Département Informatique, Systèmes et Collaboration (ISC)
Centre de Recherche Public – Gabriel Lippmann
Belvaux, Luxembourg
parisot@lippmann.lu

Abstract. Decision trees are helpful decision support tools, due to their graphical nature and the easiness to obtain them from data. Unfortunately, decision tree size tends to grow according to the complexity of the learning data, which may be problematic in real world settings. This paper proposes an original solution to reduce the size of decision trees by taking user preferences into account. More specifically, we present a user-driven algorithm that automatically transforms data in order to construct simpler decision tree. A prototype has been implemented, and the benefits are shown on several UCI datasets.

Keywords: decision tree, data preprocessing, user-driven.

1 Introduction

Originally used as an analytical decision support tool, decision trees are a convenient way to model the logic behind the data. They provide an overview of the rules underlying the data by using a form which is intuitive and easy to understand for domain experts [1]. In order to support decision making, trees can be built from data by using well-known induction techniques [1, 4] or by using recent algorithms [8].

From a pragmatic point of view, decision trees are known for their tendency to grow excessively large [6], due to the complexity of real world data (size, outliers, missing values, etc.). On the one hand, a fit solution to this problem is decision tree pruning; even if it is an old domain in the classification area [5, 6], recent works have tried to use pruning to build simpler models according to user preferences [7]. On the other hand, reducing decision trees size can be done by using data preprocessing before decision tree induction [6]. Commonly used as a preliminary for machine learning [2], data preprocessing transforms a dataset so that its use in a given task (visualisation, knowledge extraction, support for modelling etc.) is facilitated. More precisely, data preprocessing is a generic term which regroups several kinds of data manipulation tasks [3]: '*cleaning'* (treatment of noise/extreme/redundancy/unknown value, etc.), '*dimensionality altering*' (construction/transformation/filtering of features, etc.) and '*quantity altering*' (selection/sampling of the data records). As data preprocessing generally causes information loss (example: feature or records deletion, etc.), it has to be carefully used, and a user has to be able to control information loss.

2 Contribution

This paper presents a method to find an ordered combination of data preprocessing operations which transforms a given dataset so that the modified dataset leads to a decision tree with a lower complexity [6].

The method is driven by the user's preferences, materialized by the *acceptable data completeness factor*. This indicator reflects the acceptable loss of information for the user, which can occur when using data preprocessing methods (records removal, loss of precision after discretization, etc.). As an example, if the user accepts to lose up to 10% of information, then he should specify 90% as the acceptable data completeness factor.

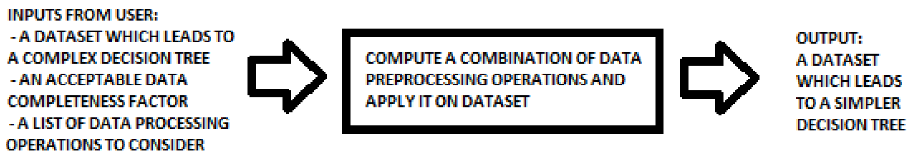
INPUTS FROM USER:
- A DATASET WHICH LEADS TO A COMPLEX DECISION TREE
- AN ACCEPTABLE DATA COMPLETENESS FACTOR
- A LIST OF DATA PROCESSING OPERATIONS TO CONSIDER
COMPUTE A COMBINATION OF DATA PREPROCESSING OPERATIONS AND APPLY IT ON DATASET
OUTPUT:
A DATASET WHICH LEADS TO A SIMPLER DECISION TREE

Fig. 1. Method overview

In practice, the current data completeness of a modified dataset is obtained by computing the count of values (i.e. the count of cells) which are not modified during data preprocessing.

Given the user's inputs (Fig 1), the method consists of trying to apply on the dataset different combinations of data processing operations (the cardinal of possible operations is small, and data-independent, so a backtracking approach is applied). After each operation application, the current data completeness is checked; if it is still lower than the acceptable data completeness factor, then the method tries to apply another preprocessing operation. If not, then a decision tree is built from the modified dataset set, by using a decision tree induction algorithm (like C4.5 [1, 4]).

Table 1. Pseudo code to find data preprocessing operations according to the user preferences

```
foreach operationsList in possibleOperationsSet
 modifiedDataset = copy of dataset
 currentCompleteness = 100%
 appliedOperations = empty list of data preprocessing operations
 foreach operation in operationsList
   modifiedDataset = apply operation on modifiedDataset
   currentCompleteness = computeCompleteness(modifiedDataset,dataset)
   if (completenessFactor < ACCEPTABLE_DATA_COMPLETENESS_FACTOR) break
   else appliedOperations = add operation into appliedOperations
 foreach
 decision_tree = compute decision tree from newdataset
endforeach
select appliedOperations for which decision_tree.size is minimum
```

With this algorithm, the combination of operations that leads to the simplest decision tree within acceptability bounds is finally obtained. If several combinations lead to decision trees with the same size, then the error-rate of the decision tree is used to make a choice: it is a measure which reflects the accuracy of the decision tree [7]. In this case, the error-rate is related to the whole transformed dataset.

As a result, a sequence of data preprocessing operations is obtained: for instance, the result can be 'discretize numeric features + remove duplicates'. The dataset is then transformed according to the sequence of operations into a modified dataset with acceptable information loss, and leading to a simple decision tree.

3 Experiments and Discussion

In order to validate the approach described in this paper, a standalone tool has been developed in Java. It takes advantage of the Weka data mining library [10], especially its implementation of the C4.5 decision tree induction algorithm.

This prototype was used according to the following procedure: using a selection of ten datasets from UCI [9], tests have been performed for several data completeness factors in order to check the impact on decision tree size (Table 2).

Table 2. Impact on decision trees in each case

Dataset (size)	Data completeness factor										Min/ max error-rate (%)
	100% (orig. data)	90%	80%	70%	60%	50%	40%	30%	20%	10%	
anneal	40	40	40	40	24	22	22	22	22	3	0/0.9
vehicle	151	137	129	129	107	87	83	83	83	83	6.5/20.1
soybean	90	90	90	90	81	75	73	73	72	72	4.9/11.3
autos	64	64	64	64	45	39	23	17	9	9	4/10.7
colic	23	8	8	6	6	6	6	6	6	3	10.9/18.5
dermatology	37	37	37	37	29	29	29	29	25	25	2.7/3.3
diabetes	39	35	35	35	17	17	11	1	1	1	15.9/33.7
glass	43	41	41	35	35	21	19	19	15	5	7.7/56
segment	81	71	71	71	69	29	29	29	13	13	1.1/17.2
sick	51	43	43	24	24	24	24	24	5	5	0.4/2.1

For this evaluation, the following preprocessing operations were used: deletion of meaningless features, numerization of nominal features, supervised and unsupervised discretization of numeric features, and deletion of outliers, extreme values, or duplicates. This list covers different kinds of preprocessing operations (cleaning, dimensions altering, quantity altering), with variable impact on information loss.

According to the results, the method provides a way to progressively reduce the size of the decision tree by transforming the dataset. For the majority of the checked datasets, the size reduction appears with 90% of data completeness, and for all the checked datasets, the size reduction is really effective from 60% of data completeness.

As expected, decreasing data completeness allows at simplify decision trees. We may interpret this parameter as the user preference regarding the compromise between model complexity and information loss.

In addition, the error-rate has been computed in each case from the respective transformed dataset. Thus, it has been observed that the error-rate tends to grow during simplification: it is normal and the tradeoff between decision tree's accuracy and simplicity has been deeply studied in the past [5, 6, 7].

4 Conclusion

In this paper, a method has been proposed in order to reduce the size of decision trees constructed from data. Given a factor which determines the acceptable loss of information for the user, the method aims at finding an ordered list of transformations: by applying these transformations, the user obtains a dataset which leads to a simpler decision tree, independently of the used decision tree algorithm.

An implementation of the method has been developed, and its effectiveness was demonstrated on well-known UCI datasets. The data completeness factor is shown to command a compromise between complexity of decision tree and information loss, allowing a user to set his preference to this respect.

As a real world use-case, the method could be used to discover knowledge from raw business data: the method would enable to interactively obtain variable granularity levels of a given decisional business model.

In future works, we will try to optimize the method by using other techniques like genetic algorithms.

References

1. Murthy, S.K.: Automatic Construction of Decision Trees from Data: A MultiDiciplinary Survey. Data Mining and Knowledge Discovery 2(4), 345–389 (1998)
2. Famili, A., Shen, W.-M., Weber, R., Simoudis, E.: Data preprocessing and intelligent data analysi. Intelligent Data Analysis 1(1-4) (1997)
3. Engels, R., Theusinger, C.: Using a Data Metric for Preprocessing Advice for Data Mining Applications. In: ECAI, pp. 430–434 (1998)
4. Ross Quinlan, J.: Induction of Decision Trees. Machine Learning 1(1), 81–106
5. Ross Quinlan, J.: Simplifying decision trees. International Journal of Man-Machine Studies 27(3), 221–234 (1987)
6. Breslow, L.A., Aha, D.W.: Simplifying decision trees: A survey. The Knowledge Engineering Review 12(1), 1–40 (1997)
7. Stiglic, G., Kocbek, S., Pernek, I., Kokol, P.: Comprehensive Decision Tree Models in Bioinformatics., PLoS ONE 7(3), e33812 (2012)
8. Lomax, S., Vadera, S.: A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys (CSUR) 45(2), 16 (2013)
9. `http://archive.ics.uci.edu/ml/`
10. `http://www.cs.waikato.ac.nz/ml/weka/`