Unified View of Decision Tree Learning Machines for the Purpose of Meta-learning

Krzysztof Grąbczewski

Abstract. The experience gained from thorough analysis of many decision tree (DT) induction algorithms, has resulted in a unified model for DT construction and reliable testing. The model has been designed and implemented within Intemi – a versatile environment for data mining. Its modular architecture facilitates construction of all the most popular algorithms by combining proper building blocks. Alternative components can be reliably compared by tests in the same environment. This is the start point for a manifold research in the area of DTs, which will bring advanced meta-learning algorithms providing new knowledge about DT induction and optimal DT models for many kinds of data.

Keywords: Decision trees, meta-learning, object oriented design.

1 Introduction

Numerous articles on decision tree (DT) induction have been published so far and new ones appear from time to time. Each proposed algorithm includes a number of solutions, that can be applied in different combinations, not only the ones originally presented.

There is still a need for thorough research on advantages and drawbacks of different approaches. The most popular books on the subject [2, 22] are devoted to particular algorithms and have been published a long time ago. A newer book publication [24] has also undertaken the subject, but the area remains not exhaustively examined. Some comparisons of DT algorithms have also been published, but they are restricted to some split criteria [3, 16] or pruning methods [4, 17, 21].

The major reason of the lack of satisfactory comparative research seems to be the lack of a versatile environment that would make the research really simple.

Krzysztof Grąbczewski
Department of Informatics, Nicolaus Copernicus University, Toruń, Poland
e-mail: `mailto:kg@is.umk.pl`
`http://www.is.umk.pl/~kg`

Some approaches to extensive DT methods libraries have been undertaken (like MLC++ [13], TDDT [23]), but they also suffer from significant limitations (they are restricted to DTs only, so comparison with other methods is not straightforward, they are not ready for deep advanced analysis at the meta-level, etc.).

Recently, a system called Intemi [9, 10] has been designed and implemented. It constitutes a perfect framework for such tasks. Therefore, this framework has been used to implement the unified model of DT algorithms, presented in this article.

The general approach to DTs is the first step towards advanced meta-learning in the area of DTs. With such a tool, it is not difficult to examine the influence of particular elements of the algorithms to the whole model accuracy, and to gain meta knowledge on how to construct learning algorithms to succeed in the analysis of a given dataset.

This article presents a result of the effort related to two branches of science: computational intelligence on one side and object oriented design on the other. Deep analysis of many DT algorithms and their applications brought a general object oriented framework for construction and testing of a variety of learners. Like in most DT algorithms, the focus is put on classification tasks, however any application of the general model is possible by providing alternative components.

Below, section 2 reviews the most popular DT algorithms, section 3 presents the unified model of DT learning machines, section 4 shows which kinds of building blocks compose particular well known algorithms and section 5 signals the advantages of the framework in some applications.

2 Decision Tree Algorithms

The literature provides an abundance of DT algorithms of quite different nature and their different applications. One of the first algorithms to propose hierarchical splits of feature spaces resulting in the form of DTs was ID3 [20]. It dealt with categorical data features only and used information gain measure to determine the splits. The most popular DT classification learners are C4.5 [22] (an extension of ID3) and CART [2]. They split DT nodes with the information gain ratio and the Gini index criterion respectively. They offered some solutions in DT pruning, validation, dealing with missing values in the data etc. Another approach has used the SSV criterion [7, 8] for node splitting.

Other interesting algorithms of slightly different nature are Cal5 [19] and the family of "statistical" DTs: FACT [14], QUEST [15] and CRUISE [12].

All of the methods listed above are capable of building so-called univariate trees, i.e. trees which split nodes on the basis of the values of a single data feature. Some of them (e.g. CART or FACT) were also proposed in multivariate (usually linear) versions.

Most common multivariate DTs perform hierarchical data splits by finding linear combinations of data features and adequate threshold values for the splits. The linear discrimination learning is the fundamental part of each of these algorithms.

The most popular and most successful approaches (apart from CART, FACT and CRUISE) include OC1 [18], LMDT [25], LDT [26], LTree, Qtree and LgTree [5, 6], DT-SE, DT-SEP and DT-SEPIR [11] and the algorithm exploiting the dipolar criterion [1].

3 Unified View of Learning Decision Trees

A thorough analysis of all the algorithms listed above (and also some other less popular approaches) has brought numerous conclusions on their similarities and differences. The conclusions have resulted in the uniform view described below. From the topmost point of view, the tasks related to building decision tree models can be split into two separate groups:

- algorithms of **tree construction**,
- methods of **tree refinement** that are applied on top of different tree construction algorithms, including various techniques of post-pruning and approaches like iterative refiltering.

They are discussed separately in the following subsections.

3.1 Decision Tree Construction

Following the top-down approach of object-oriented analysis and design, we can determine components of the tree construction algorithms:

- **search strategy** that composes the tree node by node,
- **node splitter** i.e. a procedure responsible for splitting the nodes,
- **stop criterion** i.e. the rule that stops the search process,
- **split perspective estimator** i.e. a procedure that defines the order in which the nodes of the tree are split – when using some stop criteria, the order of splitting nodes may be very important, but in most cases it is irrelevant,
- **decision making module** which provides decisions for data items on the basis of the tree,
- optional **data transformations** that prepare the training data at the start of the process or convert the parts of data at particular nodes.

A sketch of dependencies between the modules is presented in figure 1. It can be viewed in terms of classes and interfaces: each box represents an interface and presents a group of classes implementing the interface. Although one could regard UML diagrams as more appropriate form of presentation for the object oriented design of the system, this form is not used here, because it would take much more space and would be less readable. In the figure, the solid arrows represent submodule relation while the dotted ones show the modules used by the search process.

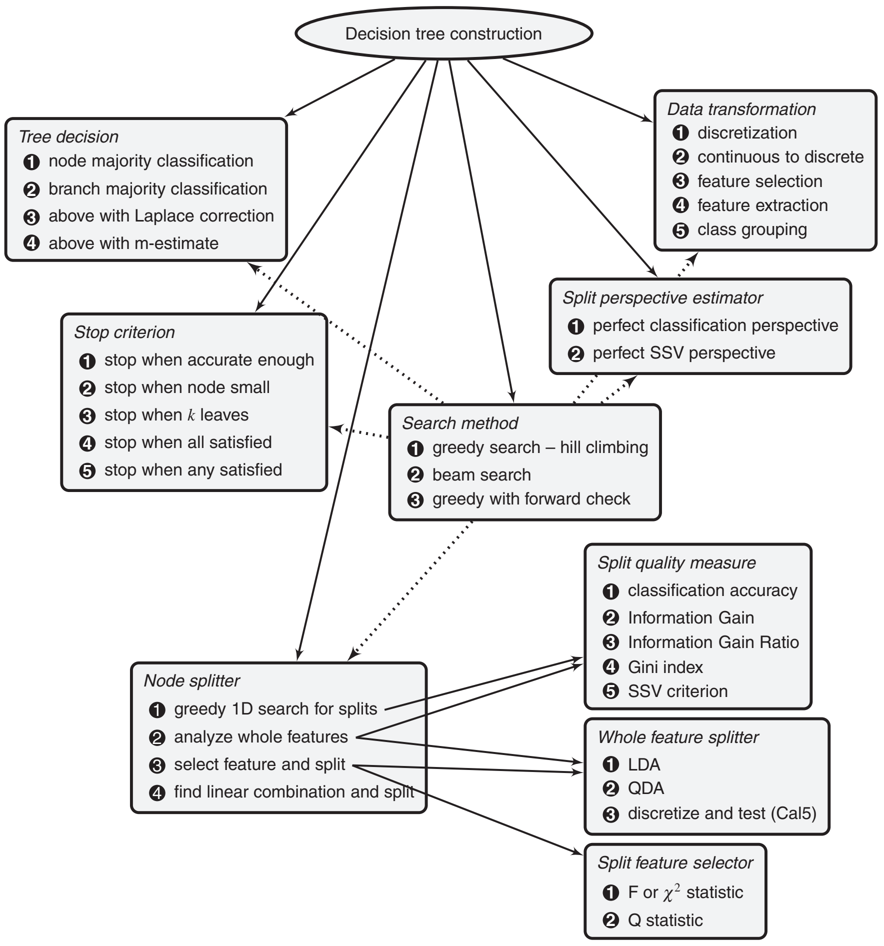
Decision tree construction
Tree decision
❶ node majority classification
❷ branch majority classification
❸ above with Laplace correction
❹ above with m-estimate
Data transformation
❶ discretization
❷ continuous to discrete
❸ feature selection
❹ feature extraction
❺ class grouping
Split perspective estimator
❶ perfect classification perspective
❷ perfect SSV perspective
Stop criterion
❶ stop when accurate enough
❷ stop when node small
❸ stop when k leaves
❹ stop when all satisfied
❺ stop when any satisfied
Search method
❶ greedy search – hill climbing
❷ beam search
❸ greedy with forward check
Split quality measure
❶ classification accuracy
❷ Information Gain
❸ Information Gain Ratio
❹ Gini index
❺ SSV criterion
Node splitter
❶ greedy 1D search for splits
❷ analyze whole features
❸ select feature and split
❹ find linear combination and split
Whole feature splitter
❶ LDA
❷ QDA
❸ discretize and test (Cal5)
Split feature selector
❶ F or χ² statistic
❷ Q statistic

Fig. 1 Information flow between DT construction algorithm modules.

Search Strategies

The search process is the main part of each tree construction algorithm. It uses the functionalities of other modules to grow the tree. Describing shortly: it uses the node splitter to split subsequent tree nodes. The other modules are used to control some details of the process:

- The stop criterion is used to decide when to stop further growth of the tree. Usually, further splits are rejected when the nodes are pure enough or get too small.
- The split perspective estimator rarely affects the resulting tree. It is used to define the order in which the nodes are split. The order may be important when using a stop criterion that acts globally e.g. sends a stop signal when the tree gets a predefined size.

- The data transformation may be used at each node to somehow prepare the node data before the split. It may be a way to implement the technique of LTree family of algorithms, where new features are generated at each node and then analyzed with a split quality measure like information gain. It can also simplify the data at particular node after the split of its parent—for example a feature can be deleted, when it is useless, e.g. all data vectors share a single value of that feature.
- The decision module of the tree does not drive the search process, but the cooperation with the search module may be very advantageous from technical point of view, because the decision module may prepare some information for further decision making just in time, when the information is available i.e. during the search process. When such information is extracted, some information like the training datasets of particular nodes may be discarded from memory, which improves the efficiency of the process.

Almost all approaches to DT induction use the same search method: they split the nodes recursively from the root node until the leaves are obtained. The search process is called a top-down induction, a recursive splits process, a depth first search, hill climbing etc. All the terms, in this context refer to the same algorithm.

The SSV approach and some others performed some experiments with other search methods like beam search or the hill climbing augmented by some insight into further split possibilities and results, but in fact, a thorough analysis of such approaches with reliable conclusions should still be done.

Node Splitters

In the algorithms listed in section 2, three different approaches to node splitting may be observed. Some methods perform a greedy search through all possible splits (sometimes only binary, sometimes also with more than two subnodes): CART, C4.5, SSV, LMDT, OC1 etc. Some others (FACT, QUEST, CRUISE, Cal5) select a feature to split in a separate procedure, and then split according to the selected feature. A version of Cal5 with entropy based split selection, finds a single split for each feature and then uses a split quality measure to select the best one.

Each of the three techniques requires different set of components to be fully configured—a selection of methods from three groups: split quality measures, whole feature splitters and split feature selectors (see the arrows in the figure 1).

Decision Making Modules

The decision making module is usually a classifier that determines target values according to the class distribution in appropriate tree leaf, however some other approaches can also be found. Some researchers try to introduce simple corrections to this decision function (like Laplace correction or m-estimate), but in fact, they do not change the decision, but just soften it, when probabilities of belonging to different classes are provided. In LTree family of algorithms [5], a decision making strategy is proposed that respects class distributions in the nodes of the whole branch of the tree (responsible for classification of particular data item), so the decision may significantly differ from the one taken on the basis of the leaf only.

3.2 Decision Tree Refinement

Since in noisy domains DT algorithms are prone to overfit the training data, some techniques of improving generalization abilities had to be worked out. Simple stop criteria have proved to be far from satisfactory and in many publications they are referred to as significantly worse than numerous post-pruning methods. Therefore, most approaches build complete trees, possibly overfitting the training data and then prune them with methods of two groups:

- algorithms based on statistical tests (Pessimistic Error Pruning, Error-Based Pruning, Minimum Error Pruning) or on the Minimum Description Length principle,
- validation based methods.

The techniques belonging to the first group are in general much faster then those of the second group, because they do not test on external data to determine which nodes to prune, but just analyze the training process with statistical tests. Unfortunately, they often result in less accurate trees in comparison to the methods of higher computational cost.

The second group of methods may be further split to the ones that use a single validation dataset (Reduced Error Pruning is the most common technique, but in fact, all the methods of the second subgroup can be applied also in this way) and those, performing multiple training and test cycles (Cost-Complexity Pruning, Critical Value Pruning and degree based pruning of SSV). The last ones are naturally the most time consuming.

The unified framework described here includes all the approaches mentioned above as instances of the two (or three, if those with single validation pass are regarded as separate) general schemes.

3.3 Other Details

Some DT algorithms use additional techniques which are not visible in the figure. These include different ways of dealing with missing data or reflecting classification error costs in the tree building processes and in final decision making. Such techniques are easily incorporated into data transformations (missing value imputations), split quality measures (error costs), tree decision making (surrogate splits of CART), etc. The space limitations do not let us go into more detail here.

4 Well Known Algorithms as Instances of the Uniform Approach

All the algorithms listed in section 2 perfectly fit the unified view presented in section 3. They have been decomposed into suitable modules. Table 1 presents general information about the types of modules required to realize the goals of particular algorithms. It is not possible to show all the details in such compact visualization.

Table 1 Most popular DT algorithms in terms of the unified model.

	ID3 [20]	C4.5 [22]	CART [2]	SSV [7, 8]	Cal5 [19]	FACT [14]	QUEST [15]	CRUISE [12]	OC1 [18]	LMDT [25]	LDT [26]	LTree family [5, 6]	DT-SE family [11]	Dipolar criterion [1]
Initial data transf.	❶					❷								
Search method	❶	❶	❶	❶❷❸	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶
Split perspective m.	❶	❶	❶	❶❷	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶
Stop criterion	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶
Node data transf.						❷	❺					❹		
Node splitter	❶	❶	❶	❶	❷❸	❸	❸	❸	❹	❹	❹	❶	❹	❹
Split quality m.	❷	❸	❹	❶❺	❷							❸		
Whole feature splitter					❸	❶	❷	❶❷						
Split feature selector					❷	❶	❶	❶						
Tree decision	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶	❶	❷	❶	❶

Some table cells contain more than one numbered bullet indicating that the methods have several variants. Some options are deliberately omitted to keep the table readable, for example, all but one algorithms are assigned ❶ in the row of tree decision, while in fact, many algorithms were tried also with Laplace correction or m-estimates of probabilities.

5 Framework Facilities

The preceding section illustrates how different popular DT induction methods can be constructed from components. Such architecture is very beneficial, when an analysis on the meta-level needs to be performed. It is especially useful in meta-learning, which seems to be the future of data mining.

The framework is being successfully used in research activities concerning different aspects of DT induction. It facilitates as just comparisons of different components in action, as possible. For example, to perform a reliable comparative test of different split criteria, we embed each competing component into the same surroundings consisting of a repeated cross-validation of a DT induction process specified by the search method, validation method, decision module etc. Thanks to providing the same environment to all the competing modules we guarantee the same training and test data in corresponding passes, even in the case of inner validation if required by the test scenario. After collecting the results from such test procedures, full information about corresponding results is available, so statistical tests like paired t test, Wilcoxon test or even McNemar test (which requires the information about correspondence between single classification decisions, not only between the mean accuracies for the whole test datasets), can be applied. In the same way, we compare other types of components like data transformations, stop criteria, validation methods etc.

Conducting the test is quite easy with such framework at hand implemented within as flexible machine learning environment as Intemi.

The information from such tests provides very precious meta-knowledge, to be used in further meta-learning approaches and eventually to compose more accurate DT induction methods.

6 Summary

The unified model of decision trees, presented here, generalizes all the popular approaches to decision tree induction. All types of components have been implemented in Intemi, a general data mining framework designed with special emphasis on meta-learning possibilities [9, 10]. Such implementation opens the gates to advanced research on meta-level analysis of decision tree algorithms and their particular components. Now, we are supplied with a versatile and efficient tool facilitating reliable comparisons of different components by testing them in the same environment (keeping all the remaining components the same). The meta-level analysis of the algorithm will certainly bring many interesting conclusions about particular components advantages and drawbacks, will help eliminate unsuccessful methods and build meta-learners capable of adjusting all the components to given data in an optimal or close to optimal way.

Acknowledgements. The research is supported by the Polish Ministry of Science with a grant for years 2010–2012.

References

[1] Bobrowski, L., Krętowski, M.: Induction of multivariate decision trees by using dipolar criteria. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 331–336. Springer, Heidelberg (2000)

[2] Breiman, L., Friedman, J.H., Olshen, A., Stone, C.J.: Classification and regression trees. Wadsworth, Belmont (1984)

[3] Buntine, W., Niblett, T.: A further comparison of splitting rules for decision-tree induction. Machine Learning 8, 75–85 (1992), 10.1007/BF00994006

[4] Esposito, F., Malerba, D., Semeraro, G.: A comparative analysis of methods for pruning decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(5), 476–491 (1997)

[5] Gama, J.: Probabilistic linear tree. In: ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 134–142. Morgan Kaufmann Publishers Inc., San Francisco (1997)

[6] Gama, J.: Discriminant trees. In: ICML 1999: Proceedings of the Sixteenth International Conference on Machine Learning, pp. 134–142. Morgan Kaufmann Publishers Inc., San Francisco (1999)

[7] Grąbczewski, K., Duch, W.: A general purpose separability criterion for classification systems. In: Proceedings of the 4th Conference on Neural Networks and Their Applications, Zakopane, Poland, pp. 203–208 (June 1999)
[8] Grąbczewski, K., Duch, W.: The Separability of Split Value criterion. In: Proceedings of the 5th Conference on Neural Networks and Their Applications, Zakopane, Poland, June 2000 , pp. 201–208 (2000)
[9] Grąbczewski, K., Jankowski, N.: Versatile and efficient meta-learning architecture: Knowledge representation and management in computational intelligence. In: IEEE Symposium Series on Computational Intelligence (SSCI 2007), pp. 51–58. IEEE, Los Alamitos (2007)
[10] Grąbczewski, K., Jankowski, N.: Efficient and friendly environment for computational intelligence. Knowledge-Based Systems, 41p. (2011) (accepted)
[11] John, G.H.: Robust linear discriminant trees. In: AI & Statistics 1995 [7], pp. 285–291. Springer, Heidelberg (1995)
[12] Kim, H., Loh, W.Y.: Classification trees with bivariate linear discriminant node models. Journal of Computational and Graphical Statistics 12, 512–530 (2003)
[13] Kohavi, R., Sommerfield, D., Dougherty, J.: Data mining using MLC++: A machine learning library in C++. In: Tools with Artificial Intelligence, pp. 234–245. IEEE Computer Society Press, Los Alamitos (1996), `http://www.sgi.com/tech/mlc`
[14] Loh, W.Y., Vanichsetakul, N.: Tree-structured classification via generalized discriminant analysis (with discussion). Journal of the American Statistical Association 83, 715–728 (1988)
[15] Loh, W.Y., Shih, Y.S.: Split selection methods for classification trees. Statistica Sinica 7, 815–840 (1997)
[16] Mingers, J.: An empirical comparison of selection measures for decision-tree induction. Machine Learning 3, 319–342 (1989)
[17] Mingers, J.: An empirical comparison of pruning methods for decision tree induction. Machine Learning 4(2), 227–243 (1989)
[18] Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research 2, 1–32 (1994)
[19] Müller, W., Wysotzki, F.: The decision-tree algorithm CAL5 based on a statistical approach to its splitting algorithm. In: Machine Learning and Statistics: The Interface, pp. 45–65 (1997)
[20] Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 81–106 (1986)
[21] Quinlan, J.R.: Simplifying decision trees. Int. J. Man-Mach. Stud. 27(3), 221–234 (1987)
[22] Quinlan, J.R.: Programs for machine learning (1993)
[23] Rokach, L., Maimon, O.: Top-down induction of decision trees classifiers – a survey. IEEE Transactions on Systems, Man and Cybernetics: Part C 1(11), 1–12 (2002)
[24] Rokach, L., Maimon, O.: Data Mining with Decision Trees: Theory and Applications. World Scientific, Singapore (2008)
[25] Utgoff, P.E., Brodley, C.E.: Linear machine decision trees. Technical Report UM-CS-1991-010, Department of Computer Science, University of Massachusetts (1991)
[26] Yildiz, O.T., Alpaydin, E.: Linear discriminant trees. International Journal of Pattern Recognition and Artifficial Intelligence 19(3), 323–353 (2005)