Artif Intell Rev (2013) 39:261–283
DOI 10.1007/s10462-011-9272-4

Decision trees: a recent overview

S. B. Kotsiantis

Published online: 29 June 2011

Abstract Decision tree techniques have been widely used to build classification models as such models closely resemble human reasoning and are easy to understand. This paper describes basic decision tree issues and current research points. Of course, a single article cannot be a complete review of all algorithms (also known induction classification trees), yet we hope that the references cited will cover the major theoretical issues, guiding the researcher in interesting research directions and suggesting possible bias combinations that have yet to be explored.

Keywords Machine learning · Decision trees · Classification algorithms

1 Introduction

Decision trees are sequential models, which logically combine a sequence of simple tests; each test compares a numeric attribute against a threshold value or a nominal attribute against a set of possible values. Such symbolic classifiers have an advantage over "black-box" models, such as neural nets, in terms of comprehensibility. The logical rules followed by a decision tree are much easier to interpret than the numeric weights of the connections between the nodes in a neural network. Decision makers tend to feel more comfortable to use models that they can understand.

When a data point falls in a partitioned region, a decision tree classifies it as belonging to the most frequent class in that region. The error rate is the total number of misclassified points divided by the total number of data points; and the accuracy rate is one minus the error rate.

Many programs have been developed that perform automatic induction (creation) of decision trees. These programs require a set of labeled instances, that is, a collection of previously

S. B. Kotsiantis (✉)
Educational Software Development Laboratory, Department of Mathematics, University of Patras,
P.A. Box: 1399, Rio 26 500, Patras, Greece
e-mail: sotos@math.upatras.gr

acquired data for which the class label of each instance has been determined. The algorithm attempts to generalize, or find patterns in, the data. It does so by determining which tests (questions) best divide the instances into separate classes, forming a tree. This procedure can be conceived as a greedy search through the space of all possible decision trees by scanning through the instances in a given node to determine the gain from each split and picking the single split that provides the greatest gain. Then the instances are partitioned based on the split, and this procedure is applied recursively until all the instances in a node are of the same class. As with other pattern classification paradigms, more complex models (larger decision trees) tend to produce poorer generalization performance. Not surprisingly then, a large amount of effort has gone into producing decision trees of smaller size.

Numerous decision tree algorithms have been developed over the years, e.g. C4.5 (Quinlan 1993), CART (Breiman et al. 1984), SPRINT (Shafer et al. 1996), SLIQ (Mehta et al. 1996). One of the latest studies that compare decision trees and other learning algorithms has been done by Tjen-Sien et al. (2000). The study shows that C4.5 has a very good combination of error rate and speed. C4.5 assumes that the training data fits in memory, thus, Gehrke et al. (2000) proposed Rainforest, a framework for developing fast and scalable algorithms to construct decision trees that gracefully adapt to the amount of main memory available.

We have limited our references to recent refereed journals, published books and conferences. In addition, we have added some references regarding the original work that started the particular line of research under discussion. A previous review of decision trees can be found in Murthy (1998). The reader should be cautioned that a single article cannot be a comprehensive review of all classification learning algorithms. Instead, our goal has been to provide a representative sample of existing lines of research in decision trees. In each of our listed areas, there are many other papers that more comprehensively detail relevant work.

Our next section covers basic issues of decision trees. In Sect. 1, we are referred to methods how the decision trees handle special problems such as imbalance data, very large datasets, ordinal classification, concept drift etc. Section 4 deals with hybrid decision tree techniques such fuzzy decision trees. In Sect. 5, ensembles of decision trees are presented. Finally, the last section concludes this work.

2 Basic issues

Figure 1 is an example of a decision tree for the training set of Table 1.

Using the decision tree depicted in Fig. 1 as an example, the instance ⟨at1 = a1, at2 = b2, at3 = a3, at4 = b4⟩ would sort to the nodes: at1, at2, and finally at3, which would classify the instance as being positive (represented by the values "Yes"). The problem of constructing optimal binary decision trees is an NP-complete problem and thus theoreticians have searched for efficient *heuristics* for constructing near-optimal decision trees.

There are two major phases of the DT induction process: the growth phase and the pruning phase. The growth phase involves a recursive partitioning of the training data resulting in a DT such that either each leaf node is associated with a single class or further partitioning of the given leaf would result in at least its child nodes being below some specified threshold. The pruning phase aims to generalize the DT that was generated in the growth phase by generating a sub-tree that avoids over-fitting to the training data. The actions of the pruning phase are often referred to as post-pruning in contrast to the pre-pruning that occurs during the growth phase and which aims to prevent splits that do not meet certain specified threshold (e.g. minimum number of observations for a split search, minimum number of observations for a leaf).

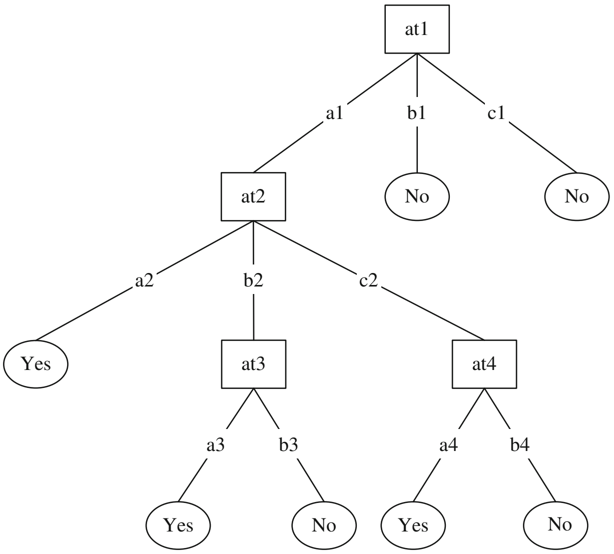

Fig. 1 A decision tree

Table 1 Training set

at1	at2	at3	at4	Class
a1	a2	a3	a4	Yes
a1	a2	a3	b4	Yes
a1	b2	a3	a4	Yes
a1	b2	b3	b4	No
a1	c2	a3	a4	Yes
a1	c2	a3	b4	No
b1	b2	b3	b4	No
c1	b2	b3	b4	No

In each iteration, the algorithm considers the partition of the training set using the outcome of a discrete function of the input attributes. The selection of the most appropriate function is made according to some splitting measures. After the selection of an appropriate split, each node further subdivides the training set into smaller subsets, until no split gains sufficient splitting measure or a stopping criteria is satisfied. Figure 2 presents a general pseudo-code for building decision trees.

The central choice in tree algorithms is finding the best split. Each split partitions the sample into two or more parts and each subset of the partition has one or more classes in it. If there is only one class in a subset, then it is *pure*, else it is *impure*. The purer the partition is, the better it is.

2.1 Splitting measures

Our intent is not to provide an exhaustive review but rather is to provide an overview of Splitting measures that are required to make the paper self-contained.

```
DT(Instances,Target_feature,Features)
If all instances at the current node belong to the same category
  then create a leaf node of the corresponding class
 else
  {
   Find the features A that maximizes the goodness measure
   Make A the decision feature for the current node
    for each possible value v of A
     {
      add a new branch below node testing for A = v
      Instances_v := subset of Instances with A = v
       if Instances_v is empty
        then add a leaf with label the most common value of
            Target_feature in Instances;
       else
       {
        below the new branch add subtree
        DT(Instances_v,Target_feature,Features - {A})
       }
     }
    }
```

Fig. 2 Pseudo-code for building a decision tree

(i) Information gain:

$$\text{Gain}\ (\text{S},\ \text{A}) = \text{Entropy}\ (\text{S}) - \sum_{\text{v}\text{€Values}\ (\text{A})} (|\ \text{S}_{\text{v}}|\ /|\ \text{S}\ |)\ \text{Entropy}\ (\text{S}_{\text{v}})$$

(ii) Gain Ratio:

$$Gain\quad Ratio\ (S, A) = \text{Gain(S, A)/Split Information (S, A)}$$

$$SplitInformation\ (S, A) = \sum_{i=1}^{c} (|\text{S}_{\text{i}}|\ /|\text{S}|)\ \log_2\ (|\text{S}_{\text{i}}|\ /|\text{S}|)$$

(iii) Gini value:

$$Gini\ (\text{D}) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is relative frequency of class j in D.

If dataset D is split on A into two subsets D1, D2 the gini index gini(D) is defined as: $Gini_A(\text{D}) = |\text{D}_1|/(|\text{D}|\text{gini}(\text{D}_1)) + |\text{D}_2|/(|\text{D}|\text{gini}(\text{D}_2))$

More extensive review of splitting measures appear in Saffavian and Landgrebe (1991).

Comparative studies of the splitting criteria described above, and others, have been conducted by several researchers during the last years, such as Loh and Shih (1999).

Node splitting measures are primary amongst the techniques that can be implemented during the construction of decision trees and represent one aspect of a multi-part approach for producing compact decision trees with improved generalization abilities.

The distribution of attributes with respect to information gain is very sparse because only a few attributes are major discriminating attributes where a discriminating attribute is an attribute, by whose value we are likely to distinguish one tuple from another. Lo et al. (2003) propose an efficient decision tree classifier for categorical attribute of sparse distribution.

In Robust C4.5 (Yao et al. 2005) algorithm, Gain Ratio is computed using attributes having Gain greater than Average Gain. This takes care of overfitting.

Wang et al. (2006) presents a new approach for inducing decision trees by combining information entropy criteria with Variable Precision Rough Set Model (VPRSM) based methods. From the angle of rough set theory, when inducing decision trees, entropy based methods emphasize the effect of class distribution. Whereas the rough set based approaches emphasize the effect of certainty. The presented approach takes the advantages of both criteria for inducing decision trees.

A new node splitting measure termed as distinct class based splitting measure (DCSM) for decision tree induction giving importance to the number of distinct classes in a partition has been proposed by Chandra et al. (2010). The measure is composed of the product of two terms. The first term deals with the number of distinct classes in each child partition. As the number of distinct classes in a partition increase, this first term increases and thus Purer partitions are thus preferred. The second term decreases when there are more examples of a class compared to the total number of examples in the partition. The combination thus still favors purer partition. It is shown that the DCSM satisfies two important properties that a split measure should possess viz. convexity and well-behavedness. Chandra and Varghese (2009b) propose a slight different node splitting measure for decision tree construction. They show that the proposed measure is convex and cumulative and utilize this in the construction of decision trees for classification.

2.2 Multivariate splits

Multivariate decision trees mainly differ from univariate decision trees in the way they test the attributes. Univariate decision trees test only one attribute at a node whereas multivariate decision trees test more than one attribute (generally a linear combination of attributes) at a node. Most of the work on multivariate splits considers linear (oblique) trees (Ittner and Schlosser 1996).

The multivariate decision tree-constructing algorithm selects not the best attribute but the best linear combination of the attributes: $\sum_{i=1}^{f} w_i x_i > w_0 . w_i$ are the weights associated with each feature $x_i$and $w_0$is the threshold to be determined from the data. So there are basically two main operations in multivariate algorithms: Feature Selection determining which features to use and finding the weights w_i of those features and the threshold w_0.

In designing decision tree classifiers, it is generally assumed that the decision complexity is the same at all nodes. However, this is not true in general. It is argued that nonlinear models may be more appropriate for the nodes that are close to the root (Yıldız and Alpaydın 2001). Since the complexity of the decision decreases as we go down the tree, linear models may generalize better for lower level nodes. The use of nonlinear models may lead to overfitting at these nodes where the training data are limited and classification problems are easier.

Altınçay (2007) proposes the use of model ensemble-based nodes where a multitude of models are considered for making decisions at each node. The ensemble members are generated by perturbing the model parameters and input attributes.

In the case where two or more features equally (or almost equally) satisfy the considered criterion of branching, the choice between these features is made almost randomly and, depending on the feature that has been chosen, the resulting trees may essentially differ both in the combination of the features employed and in their recognition qualities. Djukova and Peskov (2007) consider the following approach to the solution of the aforementioned problem. In the situation where two or more features equally satisfy the criterion, the authors suggest branching independently by each of these features. The resulting construction is called the complete decision tree.

2.3 Controlling decision tree complexity

Li and Belford (2002) discussed the instability inherent in decision tree classifications, showing that slight changes in the training set could require dramatic changes in the tree topology.

It is important to obtain decision trees of small size. The minimization problem for decision trees is known to be NP-hard. In the paper of Sieling et al. (2008) the problem is shown to be even hard to approximate up to any constant factor under the assumption $P \neq NP$.

The tree complexity is explicitly controlled by the stopping criteria used and the pruning method employed. Usually the tree complexity is measured by one of the following metrics: the total number of nodes, total number of leaves, tree depth and number of attributes used.

The growing phase continues until a stopping criterion is triggered. The following conditions are common stopping rules:

1. All instances in the training set belong to a single value of y.
2. The maximum tree depth has been reached.
3. The number of cases in the terminal node is less than the minimum number of cases for parent nodes.
4. If the node were split, the number of cases in one or more child nodes would be less than the minimum number of cases for child nodes.
5. The best splitting criteria is not greater than a certain threshold

If a decision tree induction algorithm generates a decision tree that depends too much on irrelevant features of the training instances, and performs well only on training data but poorly on unseen data, the data over-fitting happens. Many previous works have been done to tackle the over-fitting problem in traditional decision tree learning. These works fall into the following three main categories:

- Pre-pruning—this involves a "termination condition" to determine when it is desirable to terminate some of the branches prematurely when a decision tree is generated.
- Post-pruning—this approach generates a complete decision tree and then removes some of the branches with an aim of improving the classification accuracy on unseen data.
- Data pre-processing—this approach does not try to simplify the resulting decision tree directly. It indirectly tries to reach a simplification through the training data used by the learning algorithm. The aim of the data pre-processing is to find an optimal number of characteristics in order to build a simpler decision tree.

2.3.1 Pruning

There are various techniques for pruning decision trees. Most of them perform top-down or bottom-up traversal of the nodes. A node is pruned if this operation improves a certain criteria.

Several studies aim to compare the performance of different pruning techniques (Esposito et al. 1997). The results indicate that some methods (such as cost-complexity pruning, reduced error pruning) tend to over-pruning, i.e. creating smaller but less accurate decision trees. Other methods (like error-based pruning, pessimistic error pruning and minimum error pruning) bias toward under-pruning. There is no pruning method that in any case outperforms other pruning methods.

A pruning method (called DI pruning) is presented by Fournier and Crémilleux (2002). It takes into account the complexity of sub-trees and is able to keep sub-trees with leaves yielding to determine relevant decision rules, although they do not increase the classification efficiency.

Ferri et al. (2003) analysed several pruning algorithms for estimation trees, leading to a determination of the best algorithm to be applied in a specific situation.

Most post-pruning methods essentially address post-pruning as if it were a single objective problem (i.e. maximize validation accuracy), and address the issue of simplicity (in terms of the number of leaves) only in the case of a tie. However, it is well known that apart from accuracy there are other performance measures (e.g. stability, simplicity, interpretability) that are important for evaluating DT quality. Muata and Bryson (2007) propose that multi-objective evaluation be done during the post-pruning phase in order to select the best sub-tree, and propose a procedure for obtaining the optimal sub-tree based on user provided preference and value function information.

2.3.2 Feature selection

Another way of reducing the risk of overfitting is by selecting the best feature subspace at each node. The main idea is that certain dimensions may not vary in the instance subspace reaching at a particular node and can be considered as redundant. Avoiding these features may increase the generalization ability and reduce node complexity.

Feature selection algorithms in general have two components: a selection algorithm that generates proposed subsets of features and attempts to find an optimal subset; and an evaluation algorithm that determines how ‘good’ a proposed feature subset is, returning some measure of goodness to the selection algorithm. However, without a suitable stopping criterion the feature selection process may run exhaustively or forever through the space of subsets. Stopping criteria can be: (i) whether addition (or deletion) of any feature does not produce a better subset; and (ii) whether an optimal subset according to some evaluation function is obtained.

Ideally, feature selection methods search through the subsets of features, and try to find the best one among all the competing candidate subsets according to some evaluation function. However, this procedure is exhaustive as it tries to find only the best one. It may be too costly and practically prohibitive, even for a medium-sized feature set size. Other methods based on heuristic or random search methods, attempt to reduce computational complexity by compromising performance.

Feature selection is, in general, implemented in a sequential manner where features are added or eliminated one by one. For instance, LMDT (Yıldız and Alpaydın 2005) initially considers the whole feature space to train a linear model. The irrelevant variables are then eliminated by taking into account the weights assigned to each feature. Smaller weights correspond to less contribution to the discriminant and corresponding variables are considered as candidates for elimination.

2.4 Decision tree characteristics

One of the most useful characteristics of decision trees is their comprehensibility. People can easily understand why a decision tree classifies an instance as belonging to a specific class. Since a decision tree constitutes a hierarchy of tests, an unknown feature value during classification is usually dealt with by passing the example down all branches of the node where the unknown feature value was detected, and each branch outputs a class distribution. The output is a combination of the different class distributions that sum to 1. The assumption made in the decision trees is that instances belonging to different classes have different values in at least one of their features. Decision trees tend to perform better when dealing with discrete/categorical features.

Even though the divide-and-conquer algorithm is quick, efficiency can become important in tasks with hundreds of thousands of instances. The most time-consuming aspect is sorting the instances on a numeric feature to find the best threshold t. This can be expedited if possible thresholds for a numeric feature are determined just once, effectively converting the feature to discrete intervals, or if the threshold is determined from a subset of the instances (Liu et al. 2009). Efficient C4.5 (Ruggieri 2002) is an improvement over C4.5 in which various strategies are suggested. One of the strategies proposed is to sort the data using quick sort or counting sort, and the other strategy is to compute the local threshold of C4.5 using main memory version of the Rainforest algorithm, which does not need sorting.

Provost and Domingos (2003) showed that decision tree class probability estimates can be improved by skipping the pruning phase and smoothing the distributions by applying Laplace correction. Furthermore, Niculescu-Mizil and Caruana (2005) experimented other ways of correcting the poor probability estimates predicted by decision tree classifiers.

Piramuthu et al. (2008) considers some characteristics of input data and their effect on the learning performance of decision trees. Preliminary results indicate that the performance of decision trees can be improved by considering the effects due to non-linearity, outliers, heteroschedasticity, and multicollinearity in input data as well as data reduction. Data where the error term variance is not constant has Heteroschedasticity. Data where the independent variables are highly correlated is said to have multicollinearity. Both non-linearity and the presence of outliers did affect the classification performance of decision trees. The presence of heteroschedasticity did not affect the classification performance of decision trees significantly. And, the presence of multicollinearity is not of concern for decision trees. The attempt to remove multicollinearity resulted in poor classification performance. Data reduction resulted in improved performance both in terms of the resulting tree-size and classification.

3 Handling special problems with decision trees

One challenge is how to design classifiers to handle ultra-high dimensional classification problems. There is a strong need now to build useful classifiers with hundreds of millions or billions of features, for applications such as text mining and drug safety analysis. Such problems often begin with tens of thousands of features and also with interactions between the features, so the number of implied features gets huge quickly. In Sect. 3.1 we are referred to methods how decision trees can handle very large datasets.

Another related issue is how to deal with unbalanced and cost-sensitive data, a major challenge in research. Furthermore, the costs of different outcomes are dependent on the examples; for example, the false negative cost of direct marketing is directly proportional to the amount of a potential donation. Traditional methods for obtaining these costs relied on sampling methods. In Sect. 3.2 we are referred to methods how decision trees can handle cost-sensitive problems.

An important issue is that the learned models should incorporate time because data is not static and is constantly changing in many domains. Historical actions in sampling and model building are not optimal, but they are not chosen randomly either. In Sect. 3.3 we are referred to methods how decision trees can handle concept drift.

In many real-world problems, classes of examples in the training set may be partially defined and even missing. For example, for some instances, an expert may be unable to give the exact class value. A doctor who cannot specify the exact disease of a patient, a banker who cannot decide whether to give or not a loan for a client, a network administrator who is not able to decide about the exact signature of a given connection, etc. Hence, in these different examples, the expert can provide imprecise or uncertain classifications expressed in the form of a ranking on the possible classes. Ignoring the uncertainty may affect the classification results and even produce erroneous decisions. Consequently, ordinary classification techniques such as decision trees should be adequately adapted to take care of this problem. In Sect. 3.4 we are referred to methods how decision trees can handle uncertain data.

Another special problem is the ordinal classification, in which a value set of the decision attribute (output, dependent variable) is finite and ordered. This problem shares some characteristics of multi-class classification and regression, however, in contrast to the former, the order between class labels cannot be neglected, and, in the contrast to the latter, the scale of the decision attribute is not cardinal. In Sect. 3.5 we are referred to methods how decision trees can handle ordinal classification.

Multiple Instance Learning (MIL) is a variation of supervised learning for problems with incomplete knowledge about labels of training examples. In supervised learning, every training instance is assigned with a discrete or real-valued label. In comparison, in MIL the labels are only assigned to bags of instances. In the binary case, a bag is labeled positive if at least one instance in that bag is positive, and the bag is labeled negative if all the instances in it are negative. There are no labels on the individual instances. The goal of MIL is to classify unseen bags or instances based on the labeled bags as the training data. In Sect. 3.5 we are referred to methods how decision trees can handle Multiple Instance data.

3.1 Handling very large datasets with decision trees

Recently, as the data mining applications became more widespread and the processing power of computers increased significantly, the expectations from data mining algorithms grew high accordingly.

One of the most challenging problems in data mining is to develop scalable algorithms capable of mining massive data sets whose sizes exceed the capacity of a computer's memory. Li (2005) proposed a decision tree algorithm, named SURPASS (for Scaling Up Recursive Partitioning with Sufficient Statistics), that is effective in handling such large data. SURPASS incorporates linear discriminants into decision trees' recursive partitioning process. In SURPASS, the information required to build a decision tree is summarized into a set of sufficient statistics, which can be gathered incrementally from the data, by reading a subset of the data from storage space to main memory one at a time. As a result, the data size that can be handled by this algorithm is independent of memory size.

It takes very long time to generate decision trees for the data mining of very large databases that contain many continues data values, and size of decision trees has the tendency of dependency on the size of training data. Sug (2005) proposes to use samples of confidence in proper size as the training data to generate comprehensible trees as well as to save time. Gill et al. (2004) show that, by using some very weak knowledge in the sampling stage, an improved performance is achieved by a decision tree classifier. Gill et al. (2004) apply cluster-based stratified sampling using the k-means algorithm, from which the training and test sets are randomly sampled.

Decision tree algorithms need to recursively partition dataset into subsets according to some splitting criteria i.e. they still have to repeatedly compute the records belonging to a node and then compute the splits for the node. For large data sets, this requires multiple passes of original dataset and therefore is often infeasible in many applications. Fu (2006) presents a new approach to constructing decision trees using pre-computed data cube.

Classification decision tree construction algorithms have natural concurrency, as once a node is generated, all of its children in the classification tree can be generated concurrently. Furthermore, the computation for generating successors of a classification tree node can also be decomposed by performing data decomposition on the training data. Nevertheless, parallelization of the algorithms for construction the classification tree is challenging for the following reasons. First, the shape of the tree is highly irregular and is determined only at runtime. Furthermore, the amount of work associated with each node also varies, and is data dependent. Hence any static allocation scheme is likely to suffer from major load imbalance. Second, even though the successors of a node can be processed concurrently, they all use the training data associated with the parent node. If this data is dynamically partitioned and allocated to different processors that perform computation for different nodes, then there is a high cost for data movements. If the data is not partitioned appropriately, then performance can be bad due to the loss of locality.

Theoretically, feature based parallelization would have a high speedup on a data set with high dimensionality, data based parallelization on a data set with too many instances and node based parallelization on a data set which has a tree with a high number of nodes when discriminated by the serial decision tree. But, this requires perfect load balancing. Srivastava et al. (1999) present parallel formulations of classification decision tree learning algorithm based on induction.

Jin and Agrawal (2003) proposed a decision tree construction algorithm called SPIES, in which the number of possible split points is limited by taking a sample from the data set, partitioning the values into intervals and computing the class histograms for candidate split points. This reduces the space complexity of the algorithm and the communication cost between processors. They parallelized this algorithm using the FREERIDE framework and obtained nearly linear speedups. Yıldız (2007) show how to parallelize two different univariate decision tree algorithms C4.5 and univariate linear discriminant tree in three ways: (i) feature based; (ii) node based; (iii) data based manners. Experimental results show that performance of the parallelizations highly depend on the dataset and the node based parallelization demonstrate good speedups.

Most decision tree induction methods assume training data being present at one central location. Given the growth in distributed databases at geographically dispersed locations, the methods for decision tree induction in distributed settings are gaining importance. The work of Caragea et al. (2004) divides the learning task into two components: (1) hypothesis generation and (2) information extraction. This decomposition enables the distributed exact learning by extending the information extraction to distributed settings while keeping the hypothesis generation unchanged.

Bar-Or et al. (2005) suggested a distributed decision tree algorithm for data sites organized in a hierarchical structure. The hierarchical structure usually corresponds to structure of management or abstraction. In such structures, statistical information is moved from the leaf data site to the root data site. The root is responsible for building the decision tree. The algorithm is based on building the feature cross-table and computing the information gain. Their contribution defines the lower and upper bounds of information gain for a dataset in a recursive manner.

Ouyang et al. (2009) describe one distributed method that generates compact trees using multifeature splits in place of single feature split decision trees generated by most existing methods for distributed data.

3.2 Handing cost-sensitive problems with decision trees

Cost-sensitive learning is an extension of traditional non-cost-sensitive data mining. It is an important research area with many real world applications. For example, in medical diagnosis domain, diseases are not only very expensive but also rare; for a bank, an error of approving a home loan to a bad customer is more costly than an error of rejecting a home loan to a good customer.

Since different decision trees may excel under different cost settings, a set of non-dominated decision trees should be developed and presented to the decision maker for consideration, if the costs of different types of misclassification errors are not precisely determined. Zhao (2007) proposes a multi-objective genetic programming approach to developing such alternative Pareto optimal decision trees. It also allows the decision maker to specify partial preferences on the conflicting objectives, such as false negative vs. false positive, sensitivity vs. specificity, and recall vs. precision, to further reduce the number of alternative solutions.

Classifiers are faced with imbalanced data sets in many applications, which can cause the classifier to be biased towards majority class (Hulse and Khoshgoftaar 2009). It can be attributed to the way in which classifiers are designed. Inductive classifiers are typically designed to minimize errors over the training instances. Decision tree classifiers, usually, employ post-pruning techniques that evaluate the performance of decision trees as they are pruned using a validation set. Any node can be removed and assigned the most common class of the training instances that are sorted to the node in question. Thus, if a class is rare, decision tree algorithms often prune the tree down to a single node that classifies all instances as members of the common class leading to poor accuracy on the instances of minority class. For a number of application domains, a huge disproportion in the number of cases belonging to each class is common such as customer churn prediction and text classification. Moreover, in direct marketing, it is frequent to have a small response rate (about 1%) for most marketing campaigns. Other examples of domains with intrinsic imbalance can be found in the literature such as rare medical diagnoses and oil spills in satellite images. Therefore, learning with skewed class distributions is an important issue in supervised learning.

A simple method data level that can be used to imbalanced datasets is to reweigh training instances according to the total cost assigned to each class. In case of decision trees, the class probability distribution of an example can be estimated by the class distribution of the examples in the leaf that is reached by that example. Using these probability estimates we can directly compute the optimal class label for each test example using the cost matrix. This cost-sensitive learning method is called direct cost-sensitive learning (Zadrozny and Elkan 2001).

On several experiments performed in Provost and Fawcett (2001), decision tree classifiers generated from balanced distributions obtained results that were, frequently, better than those obtained from the naturally occurring distributions. Especially, these experiments were conducted with no pruning. Many of the results can be explained by understanding the role of small disjuncts in learning. Decision tree algorithms tend to form large disjuncts to cover general cases and small disjuncts to cover rare cases. Concepts with many rare cases are harder to learn than those with few, since general cases can be accurately sampled with less training data.

Like other learning algorithms, cost-sensitive learning algorithms must face a significant challenge, over-fitting, in an applied context of cost-sensitive learning. Specifically speaking, they can generate good results on training data but normally do not produce an optimal model when applied to unseen data in real world applications. It is called data over-fitting. The paper of Wang et al. (2010) deals with the issue of data over-fitting by designing three simple and efficient strategies, feature selection, smoothing and threshold pruning, against the TCSDT (test cost-sensitive decision tree) method. The feature selection approach is used to pre-process the data set before applying the TCSDT algorithm. The smoothing and threshold pruning are used in a TCSDT algorithm before calculating the class probability estimate for each decision tree leaf.

It must be mentioned that the classification of new cases using a predictive model incurs two types of costs—testing costs and misclassification costs. In many real life scenarios, however, we cannot afford to conduct all the tests required by the predictive model. For example, a medical center might have a fixed predetermined budget for diagnosing each patient. For cost bounded classification, decision trees are considered attractive as they measure only the tests along a single path.

Recently, several research efforts have been invested in developing tree-learners that consider both test costs and misclassification costs, and look for a tree that minimizes their sum. In DTMC (Decision Trees with Minimal Cost), a greedy method that attempts to minimize both types of costs simultaneously, a tree is built top-down, and a greedy split criterion that takes into account both testing and misclassification costs is used (Ling et al. 2004; Sheng et al. 2005). Freitas et al. (2007) presented a greedy algorithm for decision tree induction that considers misclassification costs, testing costs, delayed costs, and costs associated with risk, and applied on several medical tasks. Chen et al. (2009a,b) also consider a cost-sensitive decision tree construction problem. They assume that there are test costs that must be paid to obtain the values of the decision attribute and that a record must be classified without exceeding the spending cost threshold. Esmeir and Markovitch (2010) present an anytime framework for producing decision-tree based classifiers that can make accurate decisions within a strict bound on testing costs.

3.3 Handling concept drift with decision trees

In real world situations, explanations for the same observations may be different depending on perceptions or contexts. They may change with time especially when concept drift occurs. This phenomenon incurs ambiguities. It is useful if an algorithm can learn to reflect ambiguities and select the best decision according to context or situation.

Hulten et al. (2001) considered the problem of concept drift in decision trees leading to CVFDT (Concept-adapting Very Fast Decision Tree). CVFDT is one of the best-known systems that can efficiently classify streaming data. To further improve VFDT, Gama et al. (2003) extended the VFDT system in two directions: the ability to deal with continuous data and the use of more powerful classification techniques at tree leaves.

To build decision trees on streaming data, the first examples must be used to choose the root test; once the root attribute is chosen, the succeeding examples will be passed down to the corresponding leaves and used to choose the appropriate attributes there, and so on, recursively. Liu et al. (2009) study the problem of deriving ambiguous decision trees from data streams to cope with concept drift.

Bach-based algorithms require an amount of memory which is linearly proportional to the size of the data set. When trained on large data sets and possibly infinite data streams, it is desirable to have an online algorithm that would converge to the batch algorithm with memory constraints that are independent of the size of the training set Gama et al. (2006).

In real-world applications, such as credit fraud detection, network intrusion detection, huge volume of data arrives continuously with high speed. Such applications could be modeled as data stream classification problems. Data streams pose several challenges on data mining algorithm design. First, algorithms must make use of limited resources (time and memory). Second, by necessity they must deal with data whose nature or distribution changes over time. The following constraints apply in the Data Stream model:

- Data arrives as a potentially infinite sequence. Thus, it is impossible to store it all. Therefore, only a small summary can be computed and stored.
- The speed of arrival of data is fast, so each particular element has to be processed essentially in real time, and then discarded.
- The distribution generating the items may change over time. Thus, data from the past may become irrelevant (or even harmful) for the current prediction.

Liang et al. (2010) focus on uncertain data stream classification with decision trees.

3.4 Handling uncertain data with decision trees

Jenhani et al. (2008) investigate the issue of building decision trees from data with uncertain class values by developing a non-specificity based gain ratio as the attribute selection measure which, is more appropriate than the standard gain ratio based on Shannon entropy.

Wei et al. (2007) present another approach for inducing decision trees based on Variable Precision Rough Set Model. The presented approach is aimed at handling uncertain information during the process of inducing decision trees and generalizes the rough set based approach to decision tree construction by allowing some extent misclassification when classifying objects. Twala et al. (2008) propose a simple and effective method for dealing with missing data in decision trees used for classification.

The problem of uncertain data has remained a great challenge for classification algorithms. In this setting, each training instance is labeled by a possibility distribution over the different possible classes. Hence, instead of just giving a subset of potential labels (the case of ambiguous-classification), an expert can express his opinion by giving an order on the possible labels. Thus, a possibility degree of a given label expresses the degree of confidence of the expert that the label is the true one.

Jenhani et al. (2009) present an extension of C4.5 algorithm. This extension allows the C4.5 algorithm to handle uncertain labeled training data where uncertainty is modeled within the possibility theory framework. The extension mainly concerns the attribute selection measure in which a clustering of possibility distributions of a partition is performed in order to assess the homogeneity of that partition.

3.5 Handling ordinal classification with decision trees

Settings in which it is natural to rank or rate instances arise in many fields such as information retrieval, visual recognition, collaborative filtering, econometric models and classical statistics. For instance, the rating that a customer gives on a movie might be one of do-not-bother, only-if-you-must, good, very-good, and run-to-see. The ratings have a natural order, which distinguishes ordinal regression from general multiclass classification. Moreover, ordinal text classification involves classifying a document into an ordinal scale consisting of three or more classes. This type of problem is characterised by classes that posses a similarity that decays with the ordinal distance between them.

As such, a decision tree for ordinal classification should aim at producing an ordering which is most consistent with the implicit ordering in the input data. Ordinal classification problems are often dealt with by treating ordinal classes as nominal classes, or by representing the classes as values on a quantitative scale. Such approaches may not lead to the most desirable results since the methods do not fit the type of data, viz. ordinal data, concerned. Lee and Liu (2002) propose a new measure for assessing the quality of output from an ordinal classification approach.

Another approach that enables decision tree algorithms to make use of ordering information in ordinal class attributes is presented in Frank and Hall (2001). This method converts the original ordinal class problem into a set of binary class problems that encode the ordering of the original classes. However, to predict the class value of an unseen instance this algorithm needs to estimate the probabilities of the m original ordinal classes using $m - 1$ models. For example, for a three class ordinal problem, estimation of the probability for the first ordinal class value depends on a single classifier: Pr(Target < first value) as well as for the last ordinal class: Pr(Target > second value). Whereas, for class value in the middle of the range, the probability depends on a pair of classifiers and is given by

$$\Pr(\text{Target} > \text{first value}) \times (1 - \Pr(\text{Target} > \text{second value})).$$

Kotsiantis and Kanellopoulos (2010) proposed a cascade generalization technique, which combines the predictions of a classification tree and a model tree algorithm. The proposed technique can be a more robust solution to the ordinal problems since it minimizes the distance between the actual and the predicted class and improves the classification accuracy.

3.6 Handling multi-instance data with decision trees

Multi-label decision trees (Blockeel et al. 2005) deal with problems where training data can be labeled by more than one class (not mutually exclusive). In Multi-instance decision tree learning (Blockeel et al. 2005), the objective is to build a classifier from a set of labeled bags rather than a set of labeled instances. A bag is a collection of instances that is labeled positive if all its instances are positive otherwise it is labeled negative.

Ambiguous-label decision trees (Hüllermeier and Beringer 2005) deal with training instances that are labeled by a subset of candidate classes.

Multi-valued attribute means that the value of attribute is a set of values rather than a single field. Multi-labeled data means that a record of data may belong to several classes, that is, with several class labels as classification (Chen et al. 2003). The MMDT(multi-valued and multi-labeled decision tree) algorithm (Chou and Hsu 2005) uses the scoring of similarity and appropriateness of label-sets of child nodes to determine the goodness of attributes, and the scoring approach ascribes to the measuring of similarity of label-sets which is the essential index. In MMDT, the measuring of similarity starts from one single side of the same feature or consistent feature of label-sets separately, neglecting the comprehensive contributions of same and consistent feature to the similarity of label-sets.

Li et al. (2006) propose a new approach of measuring similarity considering not only the same feature but also the consistent feature of label-sets, in which the calculation proportion of the two features is adjusted dynamically, and develops a new decision tree classifier SSC (Similarity of Same and Consistent) for multi-valued and multi-labeled data.

4 Hybrid methods

Hybrid methods try to increase prediction accuracy by combining different supervised learning methods.

The alternating decision tree (ADTree) is a generalization of decision trees, voted decision trees and voted decision stumps (Freund and Mason 1999, Pfahringer et al. 2001). A general alternating tree defines a classification rule as follows. An instance defines a set of paths in the alternating tree. As in standard decision trees, when a path reaches a decision node it continues with the child which corresponds to the outcome of the decision associated with the node. However, when reaching a prediction node, the path continues with all of the children of the node. More precisely, the path splits into a set of paths, each of which corresponds to one of the children of the prediction node. We call the union of all the paths reached in this way for a given instance the "multi-path" associated with that instance. The sign of the sum of all the prediction nodes that included in a multi-path is the classification which the tree associates with the instance.

Zhou and Chen (2002) generated a binary hybrid decision tree according to the binary information gain ratio criterion. If attributes cannot further distinguish training examples falling into a leaf node whose diversity is beyond the diversity threshold, then the node is marked as a dummy node and a feed-forward neural network named FANNC is then trained in the instance space defined by the used attributes.

Carvalho and Freitas (2004) developed two genetic algorithms (GA) specifically designed for discovering rules covering examples belonging to small disjuncts, whereas a conventional decision tree algorithm is used to produce rules covering examples belonging to large disjuncts.

Gama (2004) combines a univariate decision tree with a linear function by means of constructive induction. Decision trees derived from the framework are able to use decision nodes with multivariate tests, and leaf nodes that make predictions using linear functions. Multivariate decision nodes are built when growing the tree, while functional leaves are built when pruning the tree.

Ling et al. (2004) presented Flexible NBTree: a decision tree learning algorithm in which nodes contain univariate splits as do regular decision trees, but the leaf nodes contain General Naive Bayes, which is a variant of the standard Naive Bayesian classifier.

Tree induction methods and linear models are popular techniques for supervised learning tasks, both for the prediction of nominal classes and numeric values. For predicting numeric quantities, there has been work on combining these two schemes into 'model trees', i.e. trees that contain linear regression functions at the leaves. In this paper, we present an algorithm that adapts this idea for classification problems, using logistic regression instead of linear regression. Landwehr et al. (2005) use a stagewise fitting process to construct the logistic regression models that can select relevant attributes in the data in a natural way, and show how this approach can be used to build the logistic regression models at the leaves by incrementally refining those constructed at higher levels in the tree.

Zhou et al. (2006) apply Bayesian approach to DT and seek to incorporate the Bayesian inference into the learning procedure of the model. An evolutionary method is presented by Aitkenhead MJ (2008) which allows decision tree flexibility through the use of co-evolving competition between the decision tree and the training data set.

Nijssen and Fromont (2007) presented DL8, an exact algorithm for finding a decision tree that optimizes a ranking function under size, depth, accuracy and leaf constraints. The key idea behind DL8 is that constraints on decision trees are simulated as constraints on itemsets. They show that optimal decision trees can be extracted from lattices of itemsets in linear time.

Chandra et al. (2010) present a hybrid classification model by integrating a case-based reasoning technique, a fuzzy decision tree (FDT), and genetic algorithms (GAs) to construct a decision-making system for data classification in various database applications. The model is major based on the idea that the historic database can be transformed into a smaller case base together with a group of fuzzy decision rules. As a result, the model can be more accurately respond to the current data under classifying from the inductions by these smaller case-based fuzzy decision trees.

Kumar and Gopal (2010) proposed a hybrid SVM based decision tree to speedup SVMs in its testing phase for binary classification tasks. The central idea is to approximate the decision boundary of SVM using decision trees. The resulting tree is a hybrid tree in the sense that it has both univariate and multivariate (SVM) nodes. The hybrid tree takes SVM's help only in classifying crucial data points lying near decision boundary; remaining less crucial data points are classified by fast univariate nodes.

Subramanian et al. (2010) focus on improving decision tree induction algorithms when a kind of tie appears during the rule generation procedure for specific training datasets. The tie occurs when there are equal proportions of the target class outcome in the leaf node's records that leads to a situation where majority voting cannot be applied. To solve the above mentioned exception, Subramanian et al. (2010) propose to base the prediction of the result on the naive Bayes (NB) estimate, k-nearest neighbour (k-NN) and association rule mining (ARM). The other features used for splitting the parent nodes are also taken into consideration.

Some user-centered manual (i.e. interactive or non-automatic) algorithms inducing decision trees have also appeared: Decision Tree Visualization (DTViz) (Ware et al. 2001) or CIAD (Poulet 2002). All of them try to involve the user more intensively in the data-mining process. They are intended to be used by a domain expert and not the usual statistician or data-analysis expert. Poulet and Do (2008) deal with a higher-level representation of the data. This allows the authors to treat potentially very large dataset because we do not deal with the original data, but this higher-level representation of the data.

4.1 Fuzzy decision tree algorithms

Comprehensibility is an important aspect of models used in medical data mining as it determines model credibility and even acceptability. In the practical sense though, inordinately long decision trees compounded by replication problems detracts from comprehensibility. This demerit can be partially attributed to their rigid structure that is unable to handle complex non-linear or/and continuous data.

Fuzzy ID3 (Wang et al. 2000) is a popular and efficient method of making fuzzy decision trees from a group of training examples, which is very similar to ID3. The probability of fuzzy event is used to replace the probability of crisp value for numerical attribute values. The average fuzzy classification entropy is used to measure the disorder of the fuzzified data. According to the average fuzzy classification entropy of each attribute, the most suitable attribute is selected for branching.

Olaru and Wehenkel (2003) presented a fuzzy decision trees called soft decision trees (SDT). This approach combines tree-growing and pruning, to determine the structure of the soft decision tree, with refitting and backfitting, to improve its generalization capabilities.

Mugambi et al. (2004) introduce a novel hybrid multivariate decision tree composed of polynomial, fuzzy and decision tree structures. The polynomial nature of these multivariate trees enable them to perform well in non-linear territory while the fuzzy members are used to squash continuous variables. By trading-off comprehensibility and performance using a multi-objective genetic programming optimization algorithm, Mugambi et al. (2004) can induce polynomial-fuzzy decision trees (PFDT) that are smaller, more compact and of better performance than their linear decision tree (LDT) counterparts.

Algorithms that prefuzzify the attribute and then use ID3/C4.5 approach to build the decision tree and using entropy, information gain or gain ratio as the attribute selection measure (Chen et al. 2007; Chengming 2007; Qin and Lawry 2005; Shyi-Ming and Fu-Ming 2007).

A fuzzy decision tree algorithm Gini Index based (G-FDT) is proposed by Chandra and Varghese (2009a) to fuzzify the decision boundary without converting the numeric attributes into fuzzy linguistic terms. Gini Index is used as split measure for choosing the most appropriate splitting attribute at each node. The performance of G-FDT algorithm is compared with Gini Index based crisp decision tree algorithm (SLIQ) using several real life datasets taken from the UCI machine learning repository. G-FDT algorithm outperforms its crisp counterpart in terms of classification accuracy. The size of the G-FDT is significantly less compared to SLIQ.

More analysis of fuzzy decision trees are out of the scope of this article. Recently, Chen et al. (2009a,b) present a survey of current methods for building Fuzzy Decision Trees (FDT). Fuzzy sets and approximate reasoning allow for processing of noisy, inconsistent and incomplete data. It is more accurate than standard decision trees. However, from a computational point of view, the FDT methods are intrinsically slower than crisp tree induction.

5 Ensembles of decision trees

It is generally recognised that recursive partitioning, as used in the construction of decision trees, is inherently unstable, particularly for small data sets. Classification accuracy and, by implication, tree structure, are sensitive to changes in the training data (Zhang et al. 2010). Successful approaches to counteract this effect include multiple classifiers (Banfield et al. 2007), e.g. bagging and boosting.

Bagging is a method for building ensembles that uses different subsets of training data with a single learning method (Breiman 1996). Given a training set of size t, bagging draws t random instances from the dataset with replacement (i.e. using a uniform distribution). These t instances are learned, and this process is repeated several times. Since the draw is with replacement, usually the instances drawn will contain some duplicates and some omissions, as compared to the original training set. Each cycle through the process results in one classifier. After the construction of several classifiers, taking a vote of the predictions of each classifier produces the final prediction.

Another method that uses different subsets of training data with a single learning method is the boosting approach (Freund and Schapire 1997). Boosting is similar in overall structure to bagging, except that it keeps track of the performance of the learning algorithm and concentrates on instances that have not been correctly learned. Instead of choosing the t training instances randomly using a uniform distribution, it chooses the training instances in such a manner as to favor the instances that have not been accurately learned. After several cycles, the prediction is performed by taking a weighted vote of the predictions of each classifier, with the weights being proportional to each classifier's accuracy on its training set. AdaBoost is a practical version of the boosting approach (Freund and Schapire 1997). Adaboost requires less instability than bagging, because Adaboost can make much larger changes in the training set.

MultiBoosting (Webb 2000) is another method of the same category. It can be conceptualized wagging committees formed by AdaBoost. Wagging is a variant of bagging: bagging uses resampling to get the datasets for training and producing a weak hypothesis, whereas wagging uses reweighting for each training instance, pursuing the effect of bagging in a different way. Webb (2000), in a number of experiments, showed that MultiBoost achieved greater mean error reductions than any of AdaBoost or bagging decision trees in both committee sizes that were investigated (10 and 100).

Random subspace method (Ho 1998) is used to select a feature subspace for each component model. The proposed approach allows representation of the instance subset reaching at a node by several feature subspaces eliminating the need for explicit feature selection. Using different training conditions and models as described above, the input of each node is represented by several formalisms aiming at effective modeling of decision boundaries having varying complexities at different nodes.

Random forests (Breiman 2001) are one of the best performing methods for constructing ensembles. They derive their strength from two aspects: using random subsamples of the training data (as in bagging) and randomizing the algorithm for learning base-level classifiers (decision trees). The base-level algorithm randomly selects a subset of the features at each step of tree construction and chooses the best among these.

It may be better to ensemble some instead of all of the learners. In Zhou and Tang (2003) the problem of finding a globally optimal subset of decision trees is solved approximately by means of a genetic algorithm.

Melville and Mooney (2005) presented a new meta-learner DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples), that uses a decision tree learner to build an effective diverse committee in a simple, straightforward manner. This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that disagree with the current decision of the committee, thereby easily and directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

Based on Principal Component Analysis (PCA), Rodríguez et al. (2006) proposed a new ensemble classifier generation technique Rotation Forest. Its main idea is to simultaneously encourage diversity and individual accuracy within an ensemble classifier. Specifically, diversity is promoted by using PCA to do feature extraction for each base classifier and accuracy is sought by keeping all principal components and also using the whole data set to train each base classifier.

Wang et al. (2008) presents a novel induction of multiple fuzzy decision trees based on rough set technique. The induction consists of three stages. First several fuzzy attribute reducts are found by a similarity based approach, and then a fuzzy decision tree for each fuzzy attribute reduct is generated according to the fuzzy ID3 algorithm. The fuzzy integral is finally considered as a fusion tool to integrate the generated decision trees, which combines together all outputs of the multiple fuzzy decision trees and forms the final decision result.

6 Conclusion

The majority of decision trees is made up of two major procedures: the building (induction) and the classification (inference) procedures.

- Building procedure: Given a training set, building a decision tree is usually done by starting with an empty tree and selecting for each decision node the 'appropriate' test attribute using an attribute selection measure. The principle is to select the attribute that maximally diminish the mixture of classes between each training subset created by the test, thus, making easier the determination of object's classes. The process continues for each sub decision tree until reaching leaves and fixing their corresponding classes.
- Classification procedure: To classify a new instance, having only values of all its attributes, we start with the root of the constructed tree and follow the path corresponding to the observed value of the attribute in the interior node of the tree. This process is continued until a leaf is encountered. Finally, we use the associated label to obtain the predicted class value of the instance at hand.

Many DM software packages (e.g. C5.0, SAS Enterprise Miner, IBM Intelligent Miner) provide facilities that make the generation of DTs a relatively easy task. However, in using these DM software the DM analyst has to make decisions on various parameter values (e.g. Minimum Number of Cases per Leaf, Splitting Criterion, Minimum Number of Cases for a Split, Maximum Number of Branches from a Node, Maximum Depth of Tree) that could determine the DT that is generated. Even in those situations when some major objectives of the DM project are known (e.g. accuracy of top events in the 3rst quartile), the choice of the parameter values may not be obvious. The DM analyst may thus have to experiment with many different sets of parameter values thus resulting in a significant number different DTs that must be evaluated. Although one may be concerned that the selected DT should give the best performance with regards to Accuracy, there are other criteria (e.g. Simplicity, Stability, Lift) could also be important in determining the most appropriate DT. Muata and Bryson (2004) propose a multi-criteria decision analysis (MCDA) based process to provide support to the DM project team in its effort to select the most appropriate DT.

The size and shape of the classification tree varies a lot depending on the application domain and training data set. Some classification trees might be shallow and the others might be deep. Some classification trees could be skinny others could be bushy. Some classification trees might be uniform in depth while other trees might be skewed in one part of the tree. As decision trees use the "divide and conquer" method, they tend to perform well if a few highly relevant attributes exist, but less so if many complex interactions are present. Node splitting measures represent one aspect of decision tree construction and need to be combined with techniques with other approaches that optimize the tree after its construction.

To sum up, there are five main criteria for simplifying decision tree structure:

- Controlling tree size by pre-pruning or post-pruning (editing).
- Modifying the space of states(trees) searched.
- Modifying the search algorithm used.
- Restricting the data either by removing cases or certain features from being considered by the search algorithm.
- Translating the tree structure into another data structure such as a set of rules.

Despite the obvious advantages, ensemble methods of decision trees have at least three weaknesses (Dietterich 2000). The first weakness is increased storage as a direct consequence of the requirement that all component classifiers, instead of a single classifier, need to be stored after training. The total storage depends on the size of each component classifier itself and the size of the ensemble (number of classifiers in the ensemble). The second weakness is increased computation because in order to classify an input query, all component classifiers (instead of a single classifier) must be processed. The last weakness is decreased comprehensibility. With involvement of multiple classifiers in decision-making, it is more difficult for non-expert users to perceive the underlying reasoning process leading to a decision.

References

Aitkenhead MJ (2008) A co-evolving decision tree classification method. Exp Syst Appl 34:18–25

Altınçay H (2007) Decision trees using model ensemble-based nodes. Pattern Recogn 40:3540–3551

Appavu Alias Balamurugan Subramanian, Pramala S, Rajalakshmi B, Rajaram R (2010) Improving decision tree performance by exception handling. Int J Automat Comput 7(3):372–380

Banfield RE, Hall LO, Bowyer KW (2007) A comparison of decision tree ensemble creation techniques. IEEE Trans Pattern Anal Mach Intell 29:173–180

Bar-Or, Wolff ASR, Keren D (2005) Decision tree induction in high dimensional, hierarchically distributed databases. In: Proceedings of 2005 SIAM international conference on data mining SDM'05, Newport Beach, CA

Blockeel H, Page D, Srinivasan A (2005) Multi-instance tree learning. In: Proceedings of the 22nd international conference on Machine learning, Bonn, Germany, pp 57–64

Breiman L (1996) Bagging predictors. Mach Learn 24:123–140

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Breiman L, Friedman JH, Olshen RA, Sotne CJ (1984) Classification and regression trees. Wadsworth, Belmont

Caragea D, Silvescu A, Honavar V (2004) A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. Int J Hybrid Intell Syst 1(2):80–89

Carvalho DR, Freitas AA (2004) A hybrid decision tree/genetic algorithm method for data mining. Inf Sci 163:13–35

Chandra B, Varghese PP (2009a) Fuzzifying Gini index based decision trees. Exp Syst Appl 36:8549–8559

Chandra B, Varghese PP (2009b) Moving towards efficient decision tree construction. Inf Sci 179:1059–1069

Chandra B, Kothari R, Paul P (2010) A new node splitting measure for decision tree construction. Pattern Recogn 43:2725–2731

Chang P-C, Fan C-Y, Dzan W-Y (2010) A CBR-based fuzzy decision tree approach for database classification. Exp Syst Appl 37:214–225

Chen Y, Hsu C, Chou S (2003) Constructing a multi-valued and multi-labeled decision tree. Exp Syst Appl 25(2):199–209

Chen RY, Sheu DD, Liu CM (2007) Vague knowledge search in the design for outsourcing using fuzzy decision tree. Comput Oper Res 34:3628–3637

Chen Y-l, Wang T, Wang B-s, Li Z-j (2009a) A survey of fuzzy decision tree classifier. Fuzzy Inf Eng 2:149–159

Chen Y-L, Wu C-C, Tang K (2009b) Building a cost-constrained decision tree with multiple condition attributes. Inf Sci 179:967–979

Chengming Q (2007) A new partition criterion for fuzzy decision tree algorithm. In: Intelligent information technology application, workshop on 2–3 December 2007, pp 43–46

Chou S, Hsu C (2005) MMDT: a multi-valued and multi-labeled decision tree classifier for data mining. Exp Syst Appl 28(2):799–812

Dietterich T (2000) An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Mach Learn 40:139–157

Djukova EV, Peskov NV (2007) A classification algorithm based on the complete decision tree. Pattern Recogn Image Anal 17(3):363–367

Esmeir S, Markovitch S (2010) Anytime learning of anycost classifiers. Mach Learn, doi:10.1007/s10994-010-5228-1

Esposito F, Malerba D, Semeraro G (1997) A comparative analysis of methods for pruning decision trees. EEE Trans Pattern Anal Mach Intell 19(5):476–492

Ferri U, Flach PA, Hernandez-Orallo J (2003) Improving the AUC of probabilistic estimation trees. Lect Notes Artif Intell 2837:121–132

Fournier D, Crémilleux B (2002) A quality index for decision tree pruning. Knowl Based Syst 15(1-2):37–43

Frank E, Hall M (2001) A simple approach to ordinal prediction. In: De Raedt L, Flach P (eds) ECML 2001, LNAI. Springer, Berlin, vol 2167, pp 145–156

Freitas A, Pereira A, Brazdil P (2007) Cost-sensitive decision trees applied to medical data. Lect Notes Comput Sci 4654:303–312

Freund Y, Mason L (1999) The alternating decision tree learning algorithm. In: Proceedings of the sixteenth international conference on machine learning, Bled, Slovenia, pp 124–133

Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139

Fu L (2006) Construction of decision trees using data cube. In: Chen CS et al (eds) Enterprise information systems, vol VII, pp 87–94

Gama J (2004) Functional trees. Mach Learn 55(3):219–250

Gama J, Rocha R, Medas P (2003) Accurate decision trees for mining high-speed data streams. In: Proceedings of 9th ACM SIGKDD international conference on knowledge discovery and data mining, pp 523–528

Gama J, Fernandes R, Rocha R (2006) Decision trees for mining data streams. Intell Data Anal 1:23–45

Gehrke J, Ramakrishnan R, Ganti V (2000) RainForest—a framework for fast decision tree construction of large datasets. Data Mining Knowl Discovery 4(2–3):127–162

Gill Abdul A, Smith George D, Bagnall Anthony J (2004) Improving decision tree performance through induction- and cluster-based stratified sampling. LNCS 3177:339–344

Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844

Hulse J, Khoshgoftaar T (2009) Knowledge discovery from imbalanced and noisy data. Data Knowl Eng 68(12):1513–1542

Hüllermeier E, Beringer J (2005) Learning from ambiguously labeled examples. Intell Data Anal 168–179

Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceedings of 7th ACM SIGKDD international conference on knowledge discovery and data mining, pp 97–106

Ittner, Schlosser M (1996) Discovery of relevant new features by generating non-linear decision trees. In: Proceedings of second international conference on knowledge discovery and data mining. AAAI Press, Menlo Park, pp 108–113

Jenhani I, Amor Nahla B, Elouedi Z (2008) Decision trees as possibilistic classifiers. Int J Approx Reason 48:784–807

Jenhani I, Benferhat S, Elouedi Z (2009) On the Use of clustering in possibilistic decision tree induction. LNAI 5590:505–517

Jin R, Agrawal G (2003) Communication and memory efficient parallel decision tree construction. In: Proceedings of third SIAM conference on data mining

Kotsiantis S, Kanellopoulos D (2010) Cascade generalization of ordinal problems. Int J Artif Intell Soft Comput (IJAISC) 2(1/2):46–57

Kumar MA, Gopal M (2010) A hybrid SVM based decision tree. Pattern Recogn 43:3977–3987

Landwehr N, Hall M, Frank E (2005) Logistic model trees. Mach Learn 59(1–2):161–205

Lee JWT, Liu D-Z (2002) Induction of ordinal decision trees. In: International conference on machine learning and cybernetics, pp 2220–2224

Li X-B (2005) A scalable decision tree system and its application in pattern recognition and intrusion detection. Decis Support Syst 41:112–130

Li RH, Belford GG (2002) Instability of decision tree classification algorithms. In: Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining, Edmonton, pp 570–575

Li H, Zhao R, Chen J, Xiang Y (2006) Research on multi-valued and multi-labeled decision trees. LNAI 4093:247–254

Li C-G, Wang M, Sun Z-G, Wang X-R, Zhang Z-F (2009) Decision tree algorithm using attribute frequency splitting and information entropy discretization. Comput Eng Appl 45(12):153–156

Liang C, Zhang Y, Song Q (2010) Decision tree for dynamic and uncertain data streams. In: JMLR: workshop and conference proceedings, vol 13, pp 209–224, 2nd Asian conference on machine learning (ACML2010), Tokyo, Japan

LiMin W, SenMiao Y, Ling L, HaiJun L (2004) Improving the performance of decision tree: a hybrid approach. Lect Notes Comput Sci 3288:327–335

Ling CX, Yang Q, Wang J, Zhang S (2004) Decision trees with minimal costs. In: Proceedings of the 21st international conference on machine learning (ICML-2004), Banff, pp 69–77

Liu J, Li X, Zhong W (2009) Ambiguous decision trees for mining concept-drifting data streams. Pattern Recogn Lett 30:1347–1355

Lo S-H, Ou J-C, Chen M-S (2003) Inference based classifier: efficient construction of decision trees for sparse categorical attributes. LNCS 2737:182–191

Loh WY, Shih X (1999) Families of splitting criteria for classification trees. Stat Comput 9:309–315

Mehta M, Agrawal R, Riassnen J (1996) SLIQ: a fast scalable classifier for data mining. Extending database technology. Springer, Avignon 18–32

Melville P, Mooney R (2005) Creating diversity in ensembles using artificial data. Inf Fus 6:99–111

Muata K, Bryson O (2004) Evaluation of decision trees: a multi-criteria approach. Comput Oper Res 31:1933–1945

Muata K, Bryson O (2007) Post-pruning in decision tree induction using multiple performance measures. Comput Oper Res 34:3331–3345

Mugambi EM, Hunter A, Oatley G, Kennedy L (2004) Polynomial-fuzzy decision tree structures for classifying medical data. Knowl Based Syst 17:81–87

Murthy SK (1998) Automatic construction of decision trees from data: a multi-disciplinary survey. Data Mining Knowl Discovery 2(4):345–389

Niculescu-Mizil A, Caruana R (2005) Predicting good probabilities with supervised learning. Association for Computing Machinery Inc., New York

Nijssen S, Fromont E (2007) Mining optimal decision trees from itemset lattices. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2007), San Jose, pp 530–539

Olaru C, Wehenkel L (2003) A complete fuzzy decision tree technique. Fuzzy Sets Syst 138(2):221–254

Ouyang J, Patel N, Sethi I (2009) Induction ofmulticlassmultifeature split decision trees fromdistributed data. Pattern Recogn 42:1786–1794

Pfahringer B, Holmes G, Kirkby R (2001) Optimizing the induction of alternating decision trees. In: Proceedings of the fifth Pacific-Asia conference on advances in knowledge discovery and data mining, pp 477–487

Piramuthu S (2008) Input data for decision trees. Exp Syst Appl 34:1220–1226

Poulet F (2002) Cooperation between automatic algorithms, interactive algorithms and visualization tools for visual data mining. In: Proceedings of VDM@ECML/PKDD 2002, international workshop on visual data mining, Helsinki, pp 67–80

Poulet F, Do TN (2008) Interactive decision tree construction for interval and taxonomical data. LNCS 4404:123–135

Provost F, Domingos P (2003) Tree induction for probability-based ranking. Mach Learn 52:199–215

Provost F, Fawcett T (2001) Robust classification for imprecise environments. Mach Learn 42:203–231

Qin Z, Lawry J (2005) Decision tree learning with fuzzy labels. Inf Sci 172:91–129

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Francisco

Rodríguez JJ, Kuncheva LI, Alonso CJ (2006) Rotation forest: a new classifier ensemble method. IEEE Trans Pattern Anal Mach Intell 28(10):1619–1630

Ruggieri S (2002) Efficient C4.5 [classification algorithm]. IEEE Trans Knowl Data Eng 14(2):438–444

Saffavian SR, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern 21(3):660–674

Shafer J, Agrawal R, Mehta M (1996) SPRINT: a scalable parallel classifier for data mining. In Proceedings of the VLDB conference, Bombay

Sheng S, Ling CX, Yang Q (2005) Simple test strategies for cost-sensitive decision trees. In: Proceedings of the 9th European conference on machine learning (ECML-2005), Porto, pp 365–376

Shyi-Ming C, Fu-Ming T (2007) Generating fuzzy rules from training instances for fuzzy classification systems. Exp Syst Appl, doi:10.1016/j.eswa.2007.07.013

Sieling D (2008) Minimization of decision trees is hard to approximate. J Comput Syst Sci 74:394–403

Srivastava A, Han E-H, Kumar V, Singh V (1999) Parallel formulations of decision-tree classification algorithms. Data Mining Knowl Discovery 3:237–261

Sug H (2005) A comprehensively sized decision tree generation method for interactive data mining of very large databases. LNAI 3584:141–148

Tjen-Sien L, Wei-Yin L, Yu-Shan S (2000) A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Mach Learn 40:203–228

Twala BETH, Jones MC, Hand DJ (2008) Good methods for coping with missing data in decision trees. Pattern Recogn Lett 29:950–956

Wang X, Chen B, Qian G, Ye F (2000) On the optimization of fuzzy decision trees. Fuzzy Sets Syst 112:117–125

Wang S, Wei J, You J, Liu D (2006) ComEnVprs: a novel approach for inducing decision tree classifiers. LNAI 4093:126–134

Wang X-Z, Zhai J-H, Lu S-X (2008) Induction of multiple fuzzy decision trees based on rough set technique. Inf Sci 178:3188–3202

Wang T, Qin Z, Jin Z, Zhang S (2010) Handling over-fitting in test cost-sensitive decision tree learning by feature selection, smoothing and pruning. J Syst Softw 83:1137–1147

Ware M, Franck E, Holmes G, Hall M, Witten I (2001) Interactive machine learning: letting users build classifiers. Int J Hum Comput Stud 55:281–292

Webb GI (2000) MultiBoosting: a technique for combining boosting and wagging. Mach Learn 40:159–196

Wei J-M, Wang S-Q, Wang M-Y, You J-P, Liu D-Y (2007) Rough set based approach for inducing decision trees. Knowl Based Syst 20:695–702

Yao Z, Liu P, Lei L, Yin J (2005) R_C4.5 decision tree model and its applications to health care dataset. In: Proceedings of international conference on services systems and services management—ICSSSM'05, vol 2, pp 13–15, IEEE, 2005, pp 1099–1103

Yıldız OT, Alpaydın E (2001) Omnivariate decision trees. IEEE Trans Neural Netw 12(6):1539–1546

Yıldız OT, Alpaydın E (2005) Linear discriminant trees. Int J Pattern Recogn 19(3):323–353

Yıldız OT, Dikmen O (2007) Parallel univariate decision trees. Pattern Recogn Lett 28:825–832

Zadrozny B, Elkan C (2001) Learning and making decisions when costs and probabilities are both unknown. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, San Francisco, pp 204–213

Zhang D, Zhou X, Leung S, Zheng J (2010) Vertical bagging decision trees model for credit scoring. Exp Syst Appl 37:7838–7843

Zhao H (2007) A multi-objective genetic programming approach to developing Pareto optimal decision trees. Decis Support Syst 43:809–826

Zhou Z-H, Chen Z-Q (2002) Hybrid decision tree. Knowl Based Syst 15(8):515–528

Zhou Z-H, Tang W (2003) Selective ensemble of decision trees. In: Lecture notes in artificial intelligence. Springer, Berlin, vol 2639, pp 476–483

Zhou Y, Zhang T, Chen Z (2006) Applying Bayesian approach to decision tree. LNAI 4114:290–295