Department of Computer Science

The University of Waikato

Te Whare Wānanga o Waikato

Hamilton, NewZealand

Best-first Decision Tree Learning

Haijian Shi

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science at The University of Waikato.

2006

Abstract

Decision trees are potentially powerful predictors and explicitly represent the structure of a dataset. Standard decision tree learners such as C4.5 expand nodes in depth-first order (Quinlan, 1993), while in best-first decision tree learners the "best" node is expanded first. The "best" node is the node whose split leads to maximum reduction of impurity (e.g. Gini index or information in this thesis) among all nodes available for splitting. The resulting tree will be the same when fully grown, just the order in which it is built is different. In practice, some branches of a fully-expanded tree do not truly reflect the underlying information in the domain. This problem is known as overfitting and is mainly caused by noisy data. Pruning is necessary to avoid overfitting the training data, and discards those parts that are not predictive of future data. Best-first node expansion enables us to investigate new pruning techniques by determining the number of expansions performed based on cross-validation.

This thesis first introduces the algorithm for building binary best-first decision trees for classification problems. Then, it investigates two new pruning methods that determine an appropriate tree size by combining best-first decision tree growth with cross-validation-based selection of the number of expansions that are performed. One operates in a pre-pruning fashion and the other in a post-pruning fashion. They are called best-first-based pre-pruning and best-first-based post-pruning respectively in this thesis. Both of them use the same mechanisms and thus it is possible to compare the two on an even footing. Best-first-based pre-pruning stops splitting when further splitting increases the cross-validated error, while best-first-based post-pruning takes a fully-grown decision tree and then discards expansions based on the cross-validated error. Because the two new pruning methods implement cross-validation-based pruning, it is possible to compare the two to another cross-validation-based pruning method: minimal cost-complexity pruning (Breiman et al., 1984). The two main results are that best-first-based pre-pruning is competitive with best-first-based post-pruning if the so-called "one standard error rule" is used. However, minimal cost-complexity pruning is preferable to both methods because it generates smaller trees with similar accuracy.

Acknowledgments

This thesis would not have been possible without the support and assistance of the people in the Department of Computer Science at the University of Waikato, especially the members of machine learning group. These people provides me a very good environment to do my research in New Zealand.

First and foremost, I would like to thank my supervisor Dr. Eibe Frank for guiding me through the whole thesis. He provided a lot of useful materials and helped with implementation problems I encountered during the development. He also helped me revise over the thesis draft. Moreover, he gave me valuable suggestions of how to run the experiments presented in this thesis.

Special thanks to Dr. Mark Hall who told me how to use the rpart package (the implementation of the CART system in the language R). This helped me do a comparison between my basic implementation of the CART system in Java and the rpart package in order to make sure that there is no problem in my implementation.

I would also like to thank all members in the machine learning group. I gained so much useful knowledge from the weekly meetings and e-mails sent to me. These guys are so friendly and helped me solve many problems I met.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

In the early 1990's, the establishment of the Internet made large quantities of data to be stored electronically, which was a great innovation for information technology. However, the question is what to do with all this data. *Data mining* is the process of discovering useful information (i.e. patterns) underlying the data. Powerful techniques are needed to extract patterns from large data because traditional statistical tools are not efficient enough any more. *Machine learning* algorithms are a set of these techniques that use computer programs to automatically extract models representing patterns from data and then evaluate those models. This thesis investigates a *machine learning* technique called best-first decision tree learner. It evaluates the applicability of best-first decision tree learning on real-world data and compares it to standard decision tree learning.

The remainder of this chapter is structured as follows. Section 1.1 describes some basic concepts of *machine learning* which are helpful for understanding best-first decision tree learning. Section 1.2 discusses the basic ideas underlying best-first decision trees and compare them to standard decision trees. The pruning methods for best-first decision trees are presented briefly in Section 1.3. The motivation and objectives of this thesis are discussed in Section 1.4. Section 1.5 lists the structure of the rest of this thesis.

1.1 Basic concepts of machine learning

To achieve the goals of *machine learning* described above, we need to organise the input and output first. The input consists of the data used to build models. The input involves *concepts*, *instances*, *attributes* and *datasets*. The thing which is to be learnt is called the *concept*. According to Witten and Frank (2005), there are four different styles of *concepts* in machine learning, *classification learning*, *association learning*, *clustering* and *regression learning*. *Classification learning* takes a set of classified examples (e.g. examples with class values) and uses them to build classification models. Then, it applies those models to classifying unseen examples. *Association learning* takes account of any association between features not just predicting class values. *Clustering* groups a set of similar examples together according to some criteria. *Regression learning* uses a numeric value instead of a class label for the class of each example.

The input of machine learning consists of a set of *instances* (e.g. rows, examples or sometimes observations). Each *instance* is described by a fixed number of *attributes* (i.e. columns), which are assumed to be either nominal or numeric, and a label which is called *class* (when the task is a classification task). The set of instances is called a *dataset*. The output of *machine learning* is a *concept description*. Popular types of *concept descriptions* are *decision tables, decision trees, association rules, decision rules, regression trees* and *instance-based representations* (Witten & Frank, 2005).

The four *concepts* described above can be organised into two categories, *supervised learning* and *unsupervised learning*. *Classification learning* and *regression learning* are supervised learning algorithms. In *supervised classification learning*, the induction algorithm first makes a model for a given set of labelled instances. Then, the algorithm applies the model to unclassified instances to make class predictions. In *supervised regression learning*, the induction algorithm maps each instance to a numeric value, not a class label. *Association learning* and *clustering* are unsupervised learning tasks that deal with discovering patterns for unlabelled instances. The best-first decision tree learner investigated in this thesis is a learning algorithm for *supervised classification learning*.

1.2 Decision trees

A decision tree is a tree in which each internal node represents a choice between a number of alternatives, and each terminal node is marked by a classification. Decision trees are potentially powerful predictors and provide an explicit concept description for a dataset. In practice, decision tree learning is one of the most popular technique in classification because it is fast and produces models with reasonable performance. In the context of this thesis there are two sorts of decision trees. One is constructed in depth-first order and called ”standard” decision tree. The other is constructed in best-first order and called ”best-first” decision tree. The latter type is the focus of this thesis.

1.2.1 Standard decision trees

Standard algorithms such as C4.5 (Quinlan, 1993) and CART (Breiman et al., 1984) for the top-down induction of decision trees expand nodes in depth-first order in each step using the divide-and-conquer strategy. Normally, at each node of a decision tree, testing only involves a single attribute and the attribute value is compared to a constant. The basic idea of standard decision trees is that, first, select an attribute to place at the root node and make some branches for this attribute based on some criteria (e.g. information or Gini index). Then, split training instances into subsets, one for each branch extending from the root node. The number of subsets is the same as the number of branches. Then, this step is repeated for a chosen branch, using only those instances that actually reach it. A fixed order is used to expand nodes (normally, left to right). If at any time all instances at a node have the same class label, which is known as a pure node, splitting stops and the node is made into a terminal node. This construction process continues until all nodes are pure. It is then followed by a pruning process to reduce overfittings (see Section 1.3).

1.2.2 Best-first decision trees

Another possibility, which so far appears to only have been evaluated in the context of boosting algorithms (Friedman et al., 2000), is to expand nodes in best-first order instead of a fixed order. This method adds the ”best” split node to the tree in each step. The ”best” node is the node that maximally reduces impurity among all nodes available for splitting (i.e. not labelled as terminal nodes). Although this results in the same fully-grown tree as standard depth-first expansion, it enables us to investigate new tree pruning methods that use cross-validation to select the number of expansions. Both pre-pruning and post-pruning can be performed in this way, which enables a fair comparison between them (see Section 1.3).

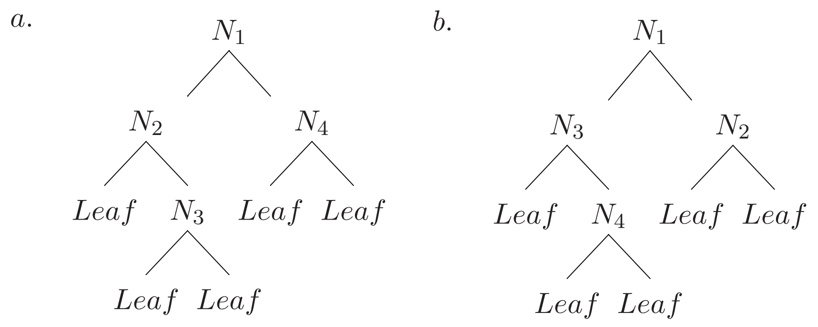

Figure 1.1: Decision trees: (a) a hypothetical depth-first decision tree, (b) a hypothetical best-first decision tree.

Best-first decision trees are constructed in a divide-and-conquer fashion similar to standard depth-first decision trees. The basic idea of how a best-first tree is built is as follows. First, select an attribute to place at the root node and make some branches for this attribute based on some criteria. Then, split training instances into subsets, one for each branch extending from the root node. In this thesis only binary decision trees are considered and thus the number of branches is exactly two. Then, this step is repeated for a chosen branch, using only those instances that actually reach it. In each step we choose the "best" subset among all subsets that are available for expansions. This constructing process continues until all nodes are pure or a specific number of expansions is reached. Figure 1.1 shows the difference in split order between a hypothetical binary best-first tree and a hypothetical binary depth-first tree. Note that other orderings may be chosen for the best-first tree while the order is always the same in the depth-first case.

The problem in growing best-first decision trees is now how to determine which attribute to split on and how to split the data. Because the most important objective of decision trees is to seek accurate and small models, we try to find pure nodes as soon as possible. To measure purity, we can use its opposite, impurity. There are many criteria to measure node impurity. For example, the CART system (Breiman et al., 1984) uses Gini index and C4.5 (Quinlan, 1993) uses information. In this thesis, both the information and the Gini index are used to compare their different performance. The goal is to aim an attribute to split on that can maximally reduce impurity.

Split selection methods used in this thesis are as follows. For a numeric attribute, the split is the same as that for most decision trees such as the CART system

(Breiman et al., 1984): the best split point is found and split is performed in a binary fashion. For example, a split on the *iris* dataset might be $petallength < 3.5$ and $petallength \geq 3.5$. For a nominal attribute, the node is also split into exactly two branches no matter how many values the splitting attribute has. This is also the same as the one used in the CART system. In this case, the objective is to find a set of attribute values which can maximally reduce impurity. Split methods are discussed in detail later in Chapter 3. For binary trees as the ones used in this thesis, it is obvious that each split step increases the number of terminal nodes by only one.

The information and the Gini gain are also used to determine node order when expanding nodes in the best-first tree. The best-first method always chooses the node for expansion whose corresponding best split provides the best information gain or Gini gain among all unexpanded nodes in the tree.

1.3 Pruning in decision trees

Fully-expanded trees are sometimes not as good as smaller trees because of noise and variability in the data, which can result in overfitting. To prevent the problem and build a tree of the right size, a pruning process is necessary. In practice, almost all decision tree learners are accompanied by pruning algorithms. Generally speaking, there are two kinds of pruning methods, one that performs pre-pruning and another one that performs post-pruning. Pre-pruning involves trying to decide to stop splitting branches early when further expansion is not necessary. Post-pruning constructs a complete tree first and prunes it back afterwards based on some criteria. Pre-pruning seems attractive because it avoids expanding the full tree and throwing some branches away afterwards, which can save computation time, but post-pruning is considered preferable because of "early stopping": a significant effect may not be visible in the tree grown so far and pre-pruning may stop too early. However, empirical comparisons are rare.

Because all pruning procedures evaluated in this thesis are based on cross-validation, we now briefly explain how it works. In cross-validation, a fixed number of folds is decided first. Assuming the number is ten, the data is separated into ten approximately equal partitions and each in turn is used for testing and the remainder is used for training. Thus, every instance is used for testing exactly once. This is called *ten-fold cross-validation*. An error estimate is calculated on the test set for each fold and the ten resulting error estimates are averaged to get the overall error estimate. When pruning decision trees based on the methods investigated in this thesis, the final tree size is decided according to this average error estimate.

In the case of best-first decision trees, pre-pruning and post-pruning can be easily performed by selecting the number of expansions based on the error estimate by cross-validation, and this is what makes them different from standard decision trees. Here is the basic idea of how they work. For both pruning methods, the trees in all folds are constructed in parallel (e.g. ten trees for a *ten-fold cross-validation*). For each number of expansion, the average error estimate is calculated based on the temporary trees in all folds. Pre-punning simply stops growing the trees when further splitting increases the average error estimate and chooses the previous number of expansion as the final number of expansions. Post-pruning continues expanding nodes until all the trees are fully expanded. Then it chooses the number of expansions whose average error estimate is minimal. In both cases the final tree is then built based on all the data and the chosen number of expansions.

This thesis compares the two new pruning methods to another cross-validation-based pruning method as implemented in the CART system, called "minimal cost-complexity pruning" (Breiman et al., 1984). This pruning method is another kind of post-pruning technique. The basic idea of this pruning method is that it tries to first prune those branches that relative to their size which leads to the smallest increase in error on the training data. The details of minimal cost-complexity pruning are described in Chapter 2. The pre-pruning and post-pruning algorithms for best-first decision trees are described in Chapter 3.

1.4 Motivation and objectives

Significant research efforts have been invested into depth-first decision tree learners. Best-first decision tree learning has so far only been applied as the base learner in boosting (Friedman et al., 2000), and has not been paid too much attention in spite of the fact that it makes it possible to implement pre-pruning and post-pruning in a very simple and elegant fashion, based on using the number of expansions as a parameter. Thus, this thesis investigates best-first decision tree learning. It investigates the effectiveness of pre-pruning and post-pruning using best-first decision trees. Best-first-based pre-pruning and post-pruning can be compared on an even footing as both of them are based on cross-validation. Minimal cost-complexity pruning is used as the benchmark pruning technique as it is also based on cross-validation and known to perform very well. As mentioned before, the best-first decision trees investigated in this thesis are binary, and the searching time of finding the optimum split for a nominal attribute using exhaustive search is exponential in the number of attribute values of this attribute, so we need to seek an efficient splitting method for nominal attributes. This thesis also lists two efficient search methods for finding binary splits on nominal attributes. One is for two-class problems (Breiman et al., 1984) and the other (i.e. heuristic search) is for multi-class problems (Coppersmith et al., 1999).

In order to fulfil the tasks presented above, the objectives of this thesis are as follows:

1. To evaluate the applicability of best-first decision tree learning to real-world data.

2. To compare best-first-based pre-pruning and post-punning on an even footing using the best-first methodology.

3. To compare the two new best-first-based pruning algorithms to minimal cost-complexity pruning.

4. To compare heuristic search and exhaustive search for binary splits on nominal attributes in multi-class problems.

1.5 Thesis structure

To achieve the objectives described above, the rest of this thesis is organised in four chapters.

The background for the work presented in this thesis is described in Chapter 2. Some known pruning methods, related to cross-validation, are discussed first.

Minimal cost-complexity pruning is discussed in detail as it is involved in the experiments presented in this thesis. Other pruning methods are briefly described. Then, the principles of pre-pruning and post-pruning for standard decision trees are explained and compared. Finally, the paper that introduced best-first decision trees (Friedman et al., 2000), which applied best-first decision trees to boosting, is briefly described.

Chapter 3 discusses the best-first decision trees used in thesis in detail. It presents impurity criteria, splitting methods for attributes, error estimates to determine the number of expansions and the algorithm for constructing best-first decision trees. Pre-pruning and post-pruning algorithms for best-first decision trees are also described in this chapter. We discuss how the one standard error rule (i.e. the 1SE rule) can be used in these two pruning methods.

Chapter 4 provides the experimental results for 38 standard benchmark datasets from the UCI repository (Blake et al., 1998) for best-first decision tree learning presented in this thesis. We evaluate best-first-based pre-pruning and post-pruning using different splitting criteria and different error estimates (determine the number of expansions) to find which splitting criterion and error estimate is better. The performance of the two new pruning methods is compared in terms of classification accuracy, tree size and training time. Then, we compare their performance to minimal cost-complexity pruning in the same way. Experiments are also evaluated to see whether tree size obtained by best-first-based pre-pruning and post-pruning is influenced by training set size. Results for heuristic search and exhaustive search for finding binary splits on nominal attributes are also discussed.

Chapter 5 summarises material presented in this thesis, draws conclusions from the work and describes some possibilities for future work.

Chapter 2
Background

This chapter describes the background for the work presented in this thesis: pruning methods related to cross-validation, work on pre-pruning and post-pruning, and work on best-first decision trees in boosting. The chapter is structured as follows. Section 2.1 describes some known pruning methods related to cross-validation. Minimal cost-complexity pruning (Breiman et al., 1984) is described in detail because it is involved in the experiments of this thesis. Other pruning methods are briefly discussed. Section 2.2 explains the principles of pre-pruning and post-pruning for standard decision trees. The comparison of the two pruning is also discussed. Some work on best-first decision trees, which applies best-first decision trees to boosting (Friedman et al., 2000), is discussed in Section 2.3. Section 2.4 summarises this chapter.

2.1 Pruning methods related to cross-validation

In practice, cross-validation can reduce sample variance and overfitting, especially for small data sample. This section discusses some pruning methods related to cross-validation. We explain the principles of the pruning methods and how cross-validation can be used in the pruning methods. First, minimal cost-complexity pruning is discussed in detail. The 1SE rule in the pruning procedure is depicted as it is also used in best-first-based pre-pruning and post-pruning in this thesis. Then, the principles of the other three pruning methods, critical value pruning (Mingers, 1987), reduced-error pruning (Quinlan, 1987) and the wrapper approach (Kohavi, 1995a), and how cross-validation can be used in them, are briefly described.

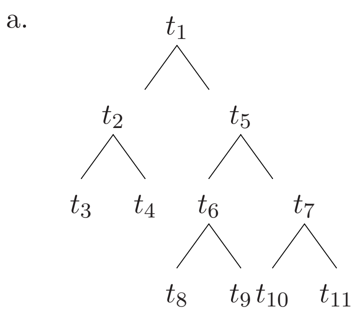

Figure 2.1: (a) The tree T, (b) a branch of T T_{t_5}, (c) the pruned subtree $T' = T - T_{t_5}$.

2.1.1 Minimal cost-complexity pruning

Minimal cost-complexity pruning was introduced in the CART system (Breiman et al., 1984) for inducing decision trees. This is why it is also called CART pruning. Minimal cost-complexity pruning is a kind of post-pruning method. Post-pruning prunes off those branches that are not predictive after a decision tree has been fully expanded. When minimal cost-complexity pruning prunes a fully-grown tree back, it considers not only the misclassification cost but also the complexity cost of the tree. In other words, minimal cost-complexity pruning seeks decision trees that achieve a compromise between misclassification cost and tree complexity. Most other pruning methods such as reduced-error pruning (Quinlan, 1987) only consider misclassification cost. Thus, the trees generated by minimal cost-complexity are normally smaller than those generated by other pruning methods.

The principle of minimal cost-complexity pruning

To understand how minimal cost-complexity pruning works, the concept of *pruned subtrees* needs to be described first. According to Breiman et al. (1984), a pruned subtree T' is obtained by pruning off a branch T_t from a tree T (i.e. T'=$T - T_t$), which replaces T_t with only its root node. The pruned subtree is denoted by $T' \prec T$. Figure 2.1 demonstrates how to obtain the pruned subtree T' from T. Figure 2.1(a) shows the original tree T. Figure 2.1(b) shows a branch of T T_{t_5}. Figure 2.1(c) shows the pruned subtree $T' = T - T_{t_5}$, which is obtained by pruning off T_{t_5} from T.

The underlying idea of minimal cost-complexity pruning is the following definition:
DEFINITION (Breiman et al., 1984) *For any subtree $T \preccurlyeq T_{max}$, define its complexity as $|\widetilde{T}|$, the number of terminal nodes in T. Let $\alpha \geq 0$ be a real number called the*

complexity parameter and define the cost-complexity measure $R_a(T)$ as

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}|.$$

In this definition, T_{max} stands for a fully-grown tree. T stands for a pruned subtree. $R(T)$ represents the misclassification cost of T on the training data. Thus, the cost-complexity measure is formed by the combination of the misclassification cost and a cost penalty for the tree complexity.

The next step of minimal cost-complexity pruning is to find the pruned subtree $T(\alpha) \preccurlyeq T_{max}$ which minimises $R_\alpha(T)$ for each value of α (Breiman et al., 1984), i.e.,

$$R_\alpha(T(\alpha)) = \min_{T \preccurlyeq T_{max}} R_\alpha(T).$$

However, T_{max} is sometimes not a good starting point to seek α values because a pruned subtree T_b may have the same misclassification cost on the training data as the T_{max} while the complexity of T_b is smaller (Breiman et al., 1984). Under these circumstances, it is obvious that T_b is a better starting point than T_{max}. Suppose that T_1 is the smallest pruned subtree of T_{max} whose classification cost is equal to the one of T_{max}. Then T_1 is treated as the starting point of the whole pruning process.

For any nonterminal node t of T_1, denote by T_t the branch of T_1 whose root node is t. If T_t is replaced by t, the pruned subtree T-T_t normally has E_e more misclassified training instances than T_1. Denote the total number of the training instances as N and define a function (Breiman et al., 1984):

$$g(t) = \frac{E_e}{N(|\widetilde{T}_t| - 1)}.$$

The heart of minimal cost-complexity pruning is to calculate each value of α, relative to each pruned subtree. The key to calculating each value of α is to understand that it works by *weakest-link cutting* (Breiman et al., 1984). *Weakest-link cutting* works here by considering the *weakest link* $\bar{t}_1$ as the node such that

$$g(\bar{t}_1) = \min_{t \in T} g(t).$$

T_k	$\widetilde{T_k}$	α_k	T_k	$\widetilde{T_k}$	α_k
T_1	61	0.0	T_9	10	0.0096
T_2	58	0.0005	T_{10}	8	0.0120
T_3	50	0.0012	T_{11}	7	0.0144
T_4	45	0.0013	T_{12}	5	0.0160
T_5	17	0.0016	T_{13}	4	0.0240
T_6	14	0.0021	T_{14}	3	0.0336
T_7	13	0.0048	T_{15}	2	0.0496
T_8	11	0.0072	T_{16}	1	0.1744

Table 2.1: The sequence of pruned subtrees from T_1, accompanied by their corresponding α values on the *balance-scale* dataset.

Then the value of $g(\bar{t}_1)$ is the value of complexity parameter for the pruned subtree T_1-$T_{\bar{t}_1}$ and denoted as α_2. The pruned subtree T_1-$T_{\bar{t}_1}$ is denoted T_2.

Now, by using T_2 as a starting tree instead of T_1, the next step is to find $\bar{t}_2$ in T_2 by *weakest link cutting* and calculate the corresponding value of α α_3. This process is similar to the previous calculation of $\bar{t}_1$. After this step, the second pruned subtree T_3, T_2 - $T_{\bar{t}_2}$ is obtained. This process is repeated recursively until the pruned subtree to be pruned only has the root node. Thus, a decreasing sequence of pruned subtrees $T_1 \succ T_2 \succ T_3 \ldots T_n$ and an increasing sequence of their corresponding α values $\alpha_1 < \alpha_2 < \alpha_3 \ldots \alpha_n$ are formed. T_n stands for the pruned subtree from T_1 that only has the root node. α_n stands for the α value corresponding to T_n. Because T_1 is an unpruned tree, its corresponding value of α α_1 is 0. The reason why the values of α are strictly in increasing order is shown in the CART book (Breiman et al., 1984).

Table 2.1 shows example values generated in the process of minimal cost-complexity pruning in the CART decision tree on the *balance-scale* dataset from the UCI repository (Blake et al., 1998). The sequence of pruned subtrees, their size in terminal nodes and corresponding α values are listed in the table. As discussed by Breiman et al., the pruning algorithm tends to prune off large subbranches first. For example, it prunes 28 nodes from T_4 to T_5 in Table 2.1. When the trees are getting smaller, it only prunes off small subbranches, which is also illustrated in Table 2.1, e.g. from T_{11} to T_{12}, to T_{13}, and so on. The problem of minimal cost-complexity pruning is now reduced to selecting which pruned subtree is optimum from the sequence. This can be done by cross-validation.

Cross-validation in minimal cost-complexity pruning

Cross-validation is the preferred method for error estimation when the data sample is not large enough to form one separated large hold-out set. It can significantly reduce sample variance for small data. In V-fold cross-validation, the original data, denoted by L, is randomly divided into V subsets, L_v, $v = 1, 2, \ldots, V$. In our experiments, in order to make all subsets representative in both training and test sets L, the original data is stratified first in this thesis. Each subset L_v is held out in turn and the remaining V-1 subsets are used to grow a decision tree and fix the values of the complexity parameter α for a sequence of pruned subtrees. The holdout subset (e.g. test set) is then used to estimate misclassification cost. Thus, the learning procedure is executed V times on different training sets and test sets.

Each instance in the original data L is used for testing only once in the cross-validation. Thus, if the number of instances of class j misclassified in the vth fold is denoted by N_{vj}, the total number of misclassified instances for class j on all test sets (i.e. in all folds) is the sum of N_{vj}, $v = 1, 2, \ldots, V$, denoted by N_j. The idea now is to measure the number of misclassified instances of all classes on L in the cross-validation. Assuming T_1 is built on all data L and $T(\alpha)$ is a pruned subtree from T_1, the cross-validated misclassification cost of $T(\alpha)$ can be written as:

$$R^{cv}(T(\alpha)) = 1/N \sum_j R^{cv}(j),$$

where N is the total number of test instances. Suppose a sequence of pruned subtrees T_k is grown from the full dataset L and their corresponding complexity parameter values α_k, $k = 1, 2, \ldots K$, are then obtained. The idea of minimal cost-complexity pruning is to find the value of the complexity parameter, α_{final}, corresponding to the pruned subtree T_{final} from the sequence T_k which has the minimal expected misclassification cost according to the cross-validation. As mentioned before, α_k is an increasing sequence of α values. Define (Breiman et al., 1984)

$$\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$$

so that α'_k is the geometric midpoint of the interval $T_\alpha = T_k$. For each α'_k we then find the corresponding tree from each of the training sets in the cross-validation (i.e. the tree that would have been generated by pruning based on that value α'_k). This tree's

k	$\widetilde{T}_k$	α_k	$R^{cv}(T_k)$ seed=1	$R^{cv}(T_k)$ seed=2
1	61	0.0	0.2291	0.2116
2*	58	0.0005	0.2292	0.2100
3	50	0.0012	0.2260	0.2148
4	45	0.0128	0.2178	0.2164
5	17	0.016	0.21629	0.2164
6	14	0.0021	0.2195	0.2131
7*	13	0.0048	0.21628	0.2163
8	11	0.0072	0.2322	0.2340
9	10	0.0096	0.2548	0.2483
10	8	0.012	0.2692	0.2708
11	7	0.0144	0.2836	0.2804
12	5	0.016	0.2868	0.3013
13	4	0.024	0.3317	0.3413
14	3	0.0336	0.3429	0.3541
15	2	0.0496	0.4134	0.4134
16	1	0.1744	0.5433	0.5417

Table 2.2: $R^{cv}(T_k)$ for different seeds without 1SE on the *balance-scale* dataset.

error is used to compute $R^{cv}(T(\alpha'_k))$. The value of α'_k for which $R^{cv}(T(\alpha'_k))$ is minimal is used as α_{final}. The tree T_{final} is the one that corresponds to the upper bound of the interval from the sequence of pruned subtrees T_k that was used to compute α_{final}.

The 1SE rule in minimal cost-complexity pruning

Because the 1SE rule is also used in pre-pruning and post-pruning for best-first decision trees later in this thesis, it is discussed in a bit more detail here. The reason why the 1SE rule used in minimal cost-complexity pruning is that, for some datasets the pruned subtree that minimises $R^{cv}(T_k)$ is unstable. Small changes in parameter values or even the seed used for randomly selecting the data for each fold of the cross-validation may result in very different T_{final} (Breiman et al., 1984). For example, Table 2.2 shows the effects between different seeds, 1 and 2, in terms of the estimated misclassification cost of the pruned subtrees on the *balance-scale* dataset from the UCI repository (Blake et al., 1998).

From the table, for seed 1, T_7 with 13 terminal nodes is selected as the final pruned subtree. For seed 2, T_2 with 58 terminal nodes is selected. The selected subtrees are significantly different. The 1SE rule can be used to avoid this problem and choose similar pruned subtrees. It has two objectives in minimal cost-complexity pruning (Breiman et al., 1984). One is to reduce the instability of choosing T_{final}. The other is

k	$\widetilde{T}_k$	α_k	$R^{cv}(T_k)$ seed=1	$R^{cv}(T_k)$ seed=2
1	61	0.0	0.2291±0.0168	0.2116±0.0163
2	58	0.0005	0.2292±0.0168	0.2100±0.0163
3	50	0.0012	0.2260±0.0167	0.2148±0.0164
4	45	0.0128	0.2178±0.0165	0.2164±0.0165
5	17	0.016	0.21629±0.0165	0.2164±0.0165
6	14	0.0021	0.2195±0.0166	0.2131±0.0164
7*	13	0.0048	0.21628±0.0165	0.2163±0.0165
8*	11	0.0072	0.2322±0.0169	0.2340±0.0169
9	10	0.0096	0.2548±0.0174	0.2483±0.0173
10	8	0.012	0.2692±0.0177	0.2708±0.0178
11	7	0.0144	0.2836±0.0180	0.2804±0.0180
12	5	0.016	0.2868±0.0181	0.3013±0.0184
13	4	0.024	0.3317±0.0188	0.3413±0.0190
14	3	0.0336	0.3429±0.0190	0.3541±0.0191
15	2	0.0496	0.4134±0.0197	0.4134±0.0197
16	1	0.1744	0.5433±0.0199	0.5417±0.0199

Table 2.3: $R^{cv}(T_k)$ for different seeds with 1SE on the *balance-scale* dataset.

to find the simplest pruned subtree whose performance is comparable to the minimal value of $R^{cv}(T_k)$ $R^{cv}(T_{k_0})$ in terms of accuracy. Denote by N the total number of instances in the original data. The standard error estimate for $R^{cv}(T_k)$ is defined as (Breiman et al., 1984):

$$SE(R^{cv}(T_k)) = [R^{cv}(T_k)(1 - R^{cv}(T_k))/N]^{1/2}.$$

Then the selection of T_{final}, according to the 1SE rule (Breiman et al., 1984), is the smallest pruned subtree T_{k_1} satisfying

$$R^{cv}(T_{k_1}) \leq R^{cv}(T_{k_0}) + SE(R^{cv}(T_{k_0})).$$

Continuing the example shown in Table 2.2, Table 2.3 presents $R^{cv}(T_k)$ with 1SE. According to the 1SE rule, the selection of T_{final} is now different from the previous selection. T_8 with 11 terminal nodes and T_7 with 13 terminal nodes are selected respectively. The T_{final} trees are now stable.

Overfitting in minimal cost-complexity pruning

Oates and Jensen (1997) investigated the effects of training set size on tree complexity for four pruning methods based on the trees generated by C4.5 (Quinlan, 1993) on 19 UCI datasets. These four pruning methods are error-based pruning (Quinlan, 1993), reduced-error pruning (Quinlan, 1987), minimum description length pruning (Quinlan & Rivest, 1989) and minimal cost-complexity pruning. The experimental results indicate that the increase of training set size often results in the increase in tree size, even when that additional complexity does not improve the classification accuracy significantly. That is to say, the additional complexity is redundant and it should be removed. Oates and Jensen found that minimal cost-complexity pruning appropriately limited tree growth much more frequently than the other three pruning methods.

A special example was given by Oates and Jensen considering four pruning algorithms on the *australian* dataset. Minimal cost-complexity pruning is compared both with and without the 1SE rule. According to the paper, accuracy peaks with a small number of training instances. After that, it remains almost constant. However, tree size continues to increase nearly linearly in error-based pruning, reduced-error pruning and minimum description length pruning. Said differently, the trees pruned by these pruning methods are overfitted. Minimal cost-complexity pruning does not suffer from this problem on the dataset. If the 1SE rule is applied, tree size stays almost constant after a small number of the training instances is used. If the 1SE rule is not applied, tree size stays within a stable range after a small number of training instances is input.

2.1.2 Other pruning methods related to cross-validation

Cross-validation can also be used in critical value pruning (Mingers, 1987), reduced-error pruning (Quinlan, 1987) and the wrapper approach (Kohavi, 1995a). Critical value pruning uses cross-validation to decide the threshold of the pruning, where the threshold is the key to determining tree size. Frank (2000) apply the combination of statistical significance tests and cross-validation in reduced-error pruning to seek an optimal significance level. The wrapper approach searches a subset of features and tune their parameters to obtain the model with the best performance based on cross-validation.

Cross-validation in critical value pruning

Critical value pruning (Mingers, 1987) is a bottom-up technique like minimal cost-complexity pruning. However, it does not use a pruning set. When growing a tree, the pruning procedure considers the splitting information of each node. Recall that at each node of a standard decision tree, splitting happens on the attribute which can maximally reduce impurity, or in other words, maximise the value of the splitting criterion, for example, Gini gain in the CART system. Critical value pruning prunes off branches based on this value. It first sets up a fixed threshold for the splitting criterion. Once the value of the splitting criterion corresponding to a node is less than the threshold, the node is made into a terminal node. One additional constraint is that, if the branch contains one or more nodes whose value is greater than the threshold, it will not be pruned.

The problem is now how to determine the threshold. Clearly, if the threshold is too large, the pruning is aggressive and it results in too small trees. On the contrary, if the threshold is too small, the pruning results in too large trees and can not prevent overfittng effectively. This is why cross-validation is suitable in this situation. The goal of cross-validation here is to find the best value of the threshold which leads to trees with reasonable size and good performance.

Cross-validation in reduced-error pruning

Standard reduced-error pruning is known to be one of the fastest pruning algorithms, producing trees that are both accurate and small (Esposito et al., 1997). The idea of reduced-error pruning is that it divides the original data into a training set and a pruning set (i.e. test set). The tree is fully grown on the training set and then the pruning set is used to estimate the error rate of branches. It prunes off unreliable branches in a bottom-up fashion. If replacing a branch with the root of the branch reduces the error on the pruning set, the branch is pruned. However, this pruning method suffers from overfitting the pruning data (Oates & Jensen, 1997), which results in an overly complex tree because the pruning data is used for multiple pruning decision. This problem is similar to the overfitting problem on the training data.

Frank (2000) investigated standard reduced-error pruning accompanied with statistical significance tests based on cross-validation to avoid the problem of overfitting the pruning set. In his experiments, the significant level is set as a parameter to optimise the performance of the pruned trees. The value of optimised significance level is obtained based on cross-validation. Experiments on 27 datasets from the UCI repository (Blake et al., 1998) shows that the tree size in 25 datasets is significantly smaller than before and the accuracy only significantly decreases in 5 datasets.

Cross-validation in the wrapper approach

For an induction algorithm, different parameter settings result in different performances in most cases. In fact, every possible parameter setting of an algorithm can be treated as a different model. The wrapper approach (Kohavi, 1995a) is based on this observation. Given an algorithm, the goal of the wrapper approach is to search a subset of features and tune their parameters to obtain the model with the best performance. Cross-validation is used as the evaluation function in the tuning of parameters.

Kohavi and John (1995) compared C4.5 (Quinlan, 1993) with default parameter setting to C4.5 with automatic parameter tuning on 33 datasets. In the automatic parameter tuning, four parameters are adjusted, namely, the minimal number instances at terminal nodes, confidence level, splitting criterion and the number of branches for a node. They pointed that if the confidence level is set to 0.9, C4.5 with automatic parameter tuning is significantly better than C4.5 with default parameter setting on nine datasets, and the latter one is significantly better on only one dataset. In the case where C4.5 with default parameter setting is better, the entire dataset is very small with only 57 instances. If the confidence level is set to 0.95, C4.5 with automatic parameter tuning outperforms C4.5 with default parameter setting on six datasets and is never outperformed by the latter one.

2.2 Pruning in standard decision trees

Pruning algorithms are designed to avoid overfitting and sample variance. They stop splitting early or discard those branches which have no improvement in performance (e.g. the accuracy in most cases). Generally speaking, there are two sorts of pruning, pre-pruning and post-pruning. Pre-pruning stops splitting when information becomes unreliable (e.g. no statistical significance in the most popular cases). Post-pruning takes a fully-grown tree and pruning off those branches that are not predictive. This section briefly describes the principles of pre-pruning and post-pruning algorithms for standard decision trees. Then, comparisons of pre-pruning and post-pruning are discussed.

2.2.1 Pre-pruning and post-pruning

As mentioned before, pre-pruning stops splitting when the splitting cannot improve predictive performance. In decision trees, pre-pruning is actually a problem of attribute selection (Frank, 2000). In other words, it selects those attributes which are predictive for the class. Pre-pruning only use *local* attribute selection: given a set of attributes in each node of a decision tree, find those attributes that are relevant to predict the class to split on. If no relevant attribute is found, this split stops and the current node is made into a terminal node. Otherwise, the node is split based on one of the predictive attributes. A predictive attribute means that there is a significant association between this attribute and the class. Thus, the goal of pre-pruning is to find whether a attribute is significantly correlated to the class. Statistical significance tests such as the chi-squared test, are designed for this kind of situation. Statistical significance tests can determine whether an observed association is a reflection of the real representation underlying the data sample or it is generated by random chance.

The problem now becomes, given a set of attributes at each node of a decision tree, finding whether there is any statistical significance; if statistical significance exists, how to select an attribute to split the training set into subsets. Otherwise, stop splitting the training set and make the node into a terminal node. The chi-squared test is traditionally used as a statistical significant test. For example, Quinlan's ID3 decision tree (1986) uses it to decide when to stop splitting the training set. The same technique is also used by the decision tree inducer CHAID (Kass, 1980). The idea behind the chi-squared test is that information in a small data sample may not be reliable. The question is now simplified to find the attribute with the optimum value of the splitting criterion to split on among all attributes for which the chi-squared test shows a significant association with the class. Potential problems with pre-pruning are discussed below.

Post-pruning takes a fully-grown tree and prunes off those branches that are not predictive such as the pruning methods discussed in the previous section. Most post-pruning procedures prune branches according to classification error. In other words, post-pruning prunes off branches which do not improve accuracy. Although, in theory, statistical significance tests can be used to determine whether a branch should be split in both pre-pruning and post-pruning, most standard post-pruning

a	b	class
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.4: A parity problem.

algorithms do not involve significance tests. Moreover, most post-pruning algorithms prune trees use a bottom-up fashion such as reduced-error pruning.

2.2.2 Comparisons of pre-pruning and post-pruning

Historically, pre-pruning techniques were investigated before post-pruning techniques. The pre-pruning methods of the earliest decision tree algorithms were introduced to deal with noisy data. Since then, pre-pruning had largely been discarded when two influential books by Breiman et al. (1984) and Quinlan (1993) were brought forth. Clearly, pre-pruning is faster than post-pruning because it stops splitting earlier. However, pre-pruning is assumed to be inferior to post-pruning because it might encounter the problem of *interaction effects*. The problem is that, in univariate decision trees each test is only based on a single attribute, so pre-pruning might overlook the effect of several interacting attributes and stop growing trees too early. Post-pruning does not suffer from this problem because the *interaction effects* are visible in fully-grown trees.

The *parity problem* is a typical example of *interaction effects* at work. A dataset shown in Table 2.4 is an example of this problem. The dataset has two binary attributes, *a* and *b*. Each attribute has two values, 0 and 1. The class is also binary and it has two labels 0 and 1. Given any single attribute, both classes are equally likely. Clearly, no individual attribute exhibits any significant association to the class, so no further split will be performed. Post-pruning is not influenced because the pruning starts with a fully-expanded tree and it can retain fully expanded branches. The fully-grown tree for this data is shown in Figure 2.2.

However, Frank (2000) found that, on the real-world datasets, parity problems have very little influence on performance differences between pre-pruning and post-pruning. Frank attempted a fair comparison of pre-pruning and post-pruning using the same

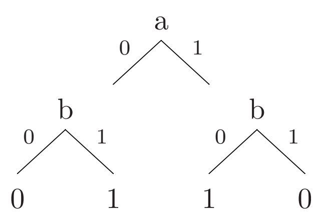

Figure 2.2: The tree for the parity problem.

statistical significance criteria. He executed his investigation on 27 datasets from the UCI repository (Blake et al., 1998). According to his research, the performance differences can be eliminated by adjusting the significance level for the two pruning individually and the performance differences only happen if a significance level is fixed. Moreover, he showed that, on the real-world datasets he investigated, parity problems have very little influence on performance differences between pre-pruning and post-pruning.

The question is now which pruning procedure should be chosen. The suggestion is that, for very large datasets, pre-pruning methods may be preferable because they are much faster. For small datasets or large datasets where time is not a factor, post-pruning methods may be preferable because they could guarantee that the parity problems cannot cause problems.

2.3 Best-first decision trees in boosting

Best-first decision trees have so far been only evaluated in the context of boosting algorithms (Friedman et al., 2000) and no pruning methods were applied to them. Boosting is a supervised machine learning technique that combines the performance of many ”weak” classifiers (i.e. those that have an accuracy greater than chance) to produce a powerful ”committee”. The basic idea underlying boosting is that, given a dataset, each training instance is first assigned an equal weight. Then, the dataset is used to form ”weak” models repeatedly. New models are influenced by the performance of previously built ones by re-weighting the training instances. Finally, all models are averaged or voted to build the final model and predict class values. The final model is actually a combination of all the models produced in the iterations of the boosting algorithm. There are several different boosting algorithms, depending on the exact way of weighting instances and models.

Friedman et al. (2000) investigated four boosting algorithms, *LogitBoost, Real AdaBoost, Gentle AdaBoost and Discrete AdaBoost* using decision trees as base learners. Generally, when decision trees are applied to boosting, the growing and pruning of trees are normally the same as trees are built in isolation. However, pruning is sometimes not necessary as the trees can be restricted to a fixed size instead in boosting. According to Friedman et al. (2000), although building large trees decreases the error rate of each individual model, it increases the error rate of the final model in all four boosting algorithms. Friedman et al. (2000) also found that after enough iterations of boosting were performed, the stump-based model (a tree that only has two terminal nodes) produced superior accuracy.

The task is thus to find a tree which can improve accuracy while restricting the depth of the tree to be not much larger than the actual tree. The depth of the tree is called a ”meta-parameter” (Friedman et al., 2000). It is clear that the ”meta-parameter” is unknown in advance for different datasets. To achieve the goal, one possibility is to first grow a fully expanded tree and then use standard pruning methods to prune it back. However, it is computationally wasteful because the iterations of boosting normally are executed many times.

Another possibility is to stop growing the tree early at the specified size, which can reduce computation time significantly. Under these circumstances, it is necessary to define the order of splitting nodes so that maximum use is made of the limited size. This is how the best-first tree works. The best-first tree first splits the node which maximally reduces the impurity among all nodes available to split. It continues this step until a fixed number M of terminal nodes is reached. Because M is the parameter for all trees in the boosting algorithm, it is also the parameter for the boosting algorithm itself. Thus, its value can be obtained by parameter selection based on cross-validation. This combination of truncated best-first decision trees with boosting shows excellent performance in the experiments by Friedman et al. (2000). They compared their experimental results to the corresponding results for boosting reported by Dietterich (1998) and pointed out that, the error rates with small truncation values are quite favourable to other committee approaches with much larger trees at each iteration. Even if the accuracy is the same, the computation time is dramatically smaller. In this thesis, the same best-first decision tree growing strategy is evaluated in the context of pruning stand-alone trees.

2.4 Summary

This chapter concerns the background of this thesis. We discussed some pruning methods based on cross-validation. Minimal cost-complexity pruning was discussed in detail because it would be compared to pre-pruning and post-pruning with best-first decision trees in this thesis. Minimal cost-complexity pruning is a pruning procedure which considers not only misclassification cost but also complexity cost for decision trees. In the description, the 1SE rule was also presented, which can guarantee the stability of the final selected pruned subtree. Then, cross-validation in critical value pruning, reduced-error pruning and the wrapper approach was briefly discussed.

Then, the principles of pre-pruning and post-pruning for standard decision trees were discussed. Pre-pruning stops splitting a node of a decision tree if there is no statistical significance. Post-pruning prunes off branches of a fully-grown tree back which cannot improve performance. The advantages and disadvantages of pre-pruning and post-pruning in standard decision trees were also discussed. The suggestion is that, for very large datasets, pre-pruning may be preferable because it is faster. For small datasets or large datasets where time is not a factor, post-pruning may be the better choice because it guarantees that parity problems cannot cause problems.

Lastly, the work related to best-first decision trees was discussed. In this work, best-first decision trees are applied to boosting. This work shows that, if the number of iterations is large enough, boosting with small decision trees can produce good results. Thus, it is a good idea to decide the order of nodes to split. Best-first decision trees fulfil this requirement. Error rates of ensembles with small trees are quite favourable to other committee approaches with much larger trees built in each iteration. And even if the accuracy is the same, the computation time can be dramatically smaller.

Chapter 3
Best-first decision tree learning

Best-first decision tree learning is a kind of decision tree learning, and thus it has almost all properties of standard decision learning. Decision tree learning is one of the most popular learning approaches in classification because it is fast and produces models with good performance. Generally, decision tree algorithms are especially good for classification learning if the training instances have errors (i.e. noisy data) and attributes have missing values. A decision tree is an arrangement of tests on attributes in internal nodes and each test leads to the split of a node. Each terminal node is then assigned a classification. In practice, in most decision tree algorithms, each internal node tests only one attribute such as in ID3 (Quinlan, 1986) although a test can be based on a set of attributes with a corresponding loss of interpretability. The best-first decision tree algorithm presented in this thesis only uses tests on one attribute. Thus, from now on, all tests we talk about are based on a single attribute. A simple example tree is shown in Figure 3.1, which has been obtained by applying the best-first decision tree algorithm on the *iris* dataset. The minimal number of instances at the terminal nodes was set to two. In general, decision trees represent a disjunction of conjunctions of constraints on the attribute-values of instances. Each path from the root to a terminal node corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions (Mitchell, 1997).

When building models, decision tree algorithms separate instances down the tree from the root node to the terminal nodes. Each terminal node provides a classification. Each internal node in the tree specifies a test of an attribute of the instances, and each branch is based on a subset of the values of the instances for this attribute. When classifying an instance, the decision tree algorithms start at the root node, test the attribute specified by this node, and then move down to the tree branch corresponding to the value of the attribute. This process is then repeated until a terminal node is reached. The classification of the terminal node is

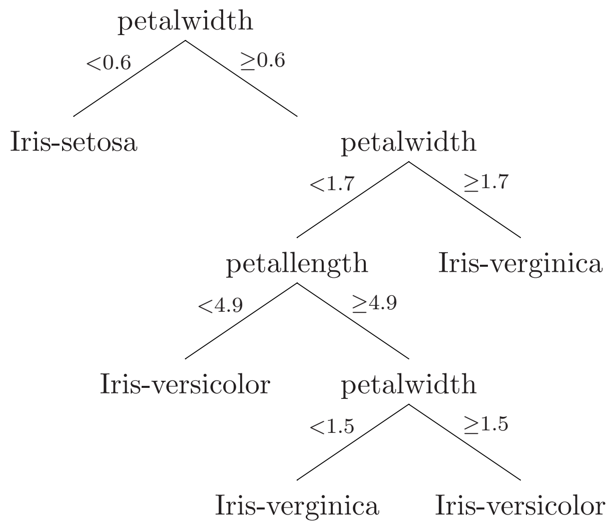

Figure 3.1: The best-first decision tree on the *iris* dataset.

the predicted value for the instance.

Trees generated by best-first decision tree learning have all properties described above. The only difference is that, standard decision tree learning expands nodes in depth-first order, while best-first decision tree learning expands the "best" node first. Standard decision tree learning and best-first decision tree learning generate the same fully-expanded tree for a given data. However, if the number of expansions is specified in advance, the generated trees are different in most cases. For example, Figure 3.2 shows a hypothetical standard decision tree and a hypothetical best-first decision tree with three expansions on the same data. The first tree in the figure is the fully-expanded tree generated by best-first decision tree learning and the second tree is the fully-expanded tree generated by standard decision tree learning. In this example, considering the fully-expanded best-first decision tree the benefit of expanding node N_2 is greater than the benefit of expanding N_3.

The rest of this chapter is organised as follows. Section 3.1 describes two splitting criteria to measure impurity in best-first decision trees that are investigated in this thesis: information and Gini index. The methods of calculating reduction of impurity, namely, information gain and Gini gain, are also discussed and examples are given on the *weather* dataset. Section 3.2 discusses splitting rules used in best-decision decision trees presented in this thesis. The goal of the splitting rules is to find the "best" binary split for both numeric attributes and nominal attributes. The "best"

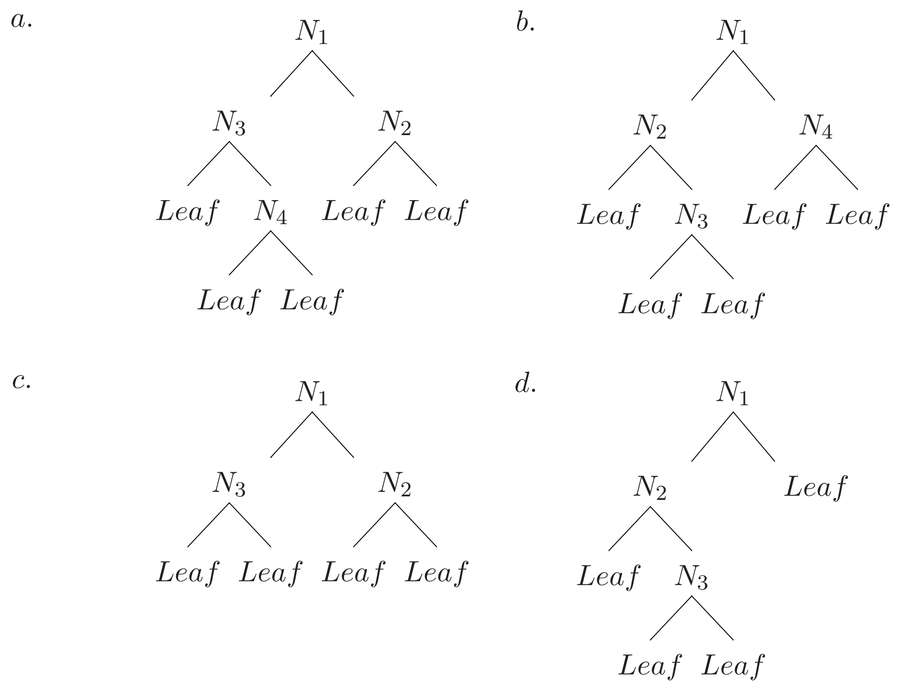

Figure 3.2: (a) The fully-expanded best-first decision tree; (b) the fully-expanded standard decision tree; (c) the best-first decision tree with three expansions from (a); (d) the standard decision tree with three expansions from (b).

split is the split with the maximal reduction of impurity. Section 3.3 presents the algorithm of best-first decision tree learning. The method of dealing with missing values is also discussed in this section. Section 3.4 discusses two pruning methods for best-first decision trees. We explain how the 1SE rule can be used in the pruning process. Section 3.5 discusses complexity of best-first decision tree induction. Section 3.6 summarises this chapter.

3.1 Splitting criteria

In order to find the "best" node to split at each step of best-first decision trees, splitting criteria must be addressed. There are many criteria for decision trees and two of them are most widely used, the information and the Gini index. For example, the information is used in ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993), and the Gini index is used in the CART system (Breiman et al., 1984). Best-first decision trees can also use these two criteria.

Splitting criteria are designed to measure node impurity. The node impurity is

outlook	temperature	humidity	windy	play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
rainy	81	75	false	yes
rainy	71	91	true	no

Table 3.1: The instances of the *weather* dataset.

based on the distribution of classes. Recall that the main objective of decision tree learning is to obtain accurate and small models. Thus, when splitting a node, we should find pure successor nodes as early as possible. In other words, the goal of splitting is to find the maximal decrease of impurity at each node. The decrease of impurity is calculated by subtracting the impurity values of successor nodes from the impurity of the node. When the subtraction is performed, the impurity values of the successor nodes are weighted by the size of each node: the number instances reaching each node. If the splitting criterion is the information, the decrease in impurity is measured by the information gain. Similarly, if the splitting criterion is the Gini index, the decrease in impurity is measured by the Gini gain.

When describing the splitting criteria, we use the *weather* dataset as an example to explain the calculation of the information and the Gini index in best-first decision trees. The reduction of impurity, namely, the Gini gain or the information gain, is also discussed. The *weather* dataset has fourteen instances. Every instances has four attributes, *outlook, temperature, humidity* and *windy*, and the class *play.* Attribute *outlook* is nominal and it has three values, *sunny, overcast* and *rainy.* Attributes *temperature* and *humidity* are numeric. Attribute *windy* is binary and it has two values, *true* and *false.* The class *play* is binary. The fourteen instances of the *weather* data are shown in Table 3.1.

3.1.1 Information

Information is the most widely used splitting criterion to measure impurity and it is expressed in units called *bits*. The *bits* in information calculation are fractions and they are normally less than 1. The information is based on the distribution of classes as mentioned before. The information value is called *entropy*. For a general dataset which has n classes, the *entropy* is defined as (Quinlan, 1986):

$$entropy(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n,$$

where p_k, $k = 1, 2, \ldots, n$, is the probability of each class and the sum of the p_k is 1. Normally, the base of the logarithms is two and this is the reason why the result can be viewed "bits". The reason for the minus signs is that logarithms of the fractions p_k are negative. Thus, the *entropy* is positive. Note that if p_k is 0 $\log p_k$ is set to 0 in the calculation. Then, the information value can be estimated by the formula:

$$info([P_1, P_2, \ldots, P_n]) = entropy(p_1, p_2, \ldots, p_n),$$

where P_k, $k = 1, 2, \ldots, n$, is the number of instances for each class. The estimated value of p_k is the value of P_k divided by the sum of all P_k.

According to the formula, the purer a node is, the smaller the information value will be. For a absolutely pure node, namely, for which one class probability is 1 and the other class probabilities are 0, the information value is 0. For a two-class problem, the *entropy* is $-p_1 \log p_1 - p_2 \log p_2$, where the sum of p_1 and p_2 is 1. The relationship between the information value and the class probability of the first class in the two-class problem is shown in Figure 3.4.

Recall that this thesis investigates binary decision trees. Figure 3.3 lists four possible splits as examples to calculate information values, one for each attribute in the *weather* dataset. Let us evaluate the first split in Figure 3.3 now. The class distributions (i.e. yes/no) for the two successor nodes are 2/3 and 7/2 respectively. The information values for the root node and the two successors nodes are thus

$$info([9,5]) = entropy(\tfrac{9}{14}, \tfrac{5}{14}) = -\tfrac{9}{14} \log \tfrac{9}{14} - \tfrac{5}{14} \log \tfrac{5}{14} = 0.940 \ bits$$

$$info([2,3]) = entropy(\tfrac{2}{5}, \tfrac{3}{5}) = -\tfrac{2}{5} \log \tfrac{2}{5} - \tfrac{3}{5} \log \tfrac{3}{5} = 0.971 \ bits$$

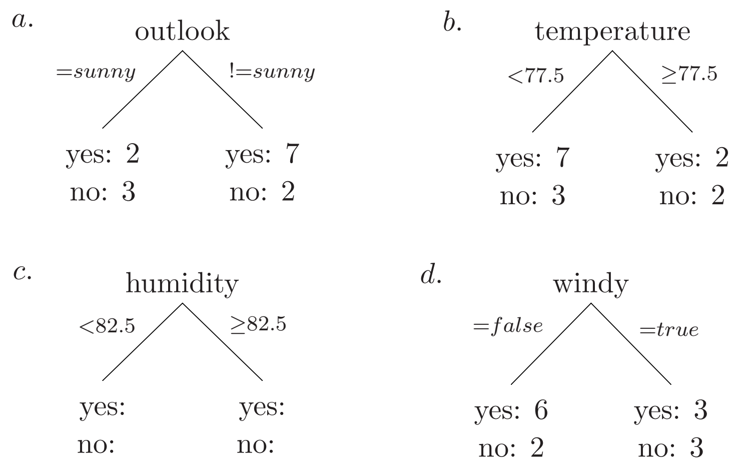

Figure 3.3: Possible splits for the four attributes of the *weather* dataset.

$$info([7,2]) = entropy(\tfrac{7}{9}, \tfrac{2}{9}) = -\tfrac{7}{9}\log\tfrac{7}{9} - \tfrac{2}{9}\log\tfrac{2}{9} = 0.764\ bits.$$

Taking into account the number of instances that go down each branch, five down the left and nine down the right, the information value for the two successor nodes is

$$info([2,3][7,2]) = \tfrac{5}{14} \times 0.971 + \tfrac{9}{14} \times 0.764 = 0.838\ bits.$$

Thus, the information gain for the first split is,

$$infoGain(outlook{=}sunny) = info([9,5]) - info([2,3][7,2]) = 0.102\ bits.$$

Similarly, the information gains for the other three splits in Figure 3.3 can be calculated as (Note that "*temperature*<77.5" means that if the value of an instances at the attribute *temperature* is less than 77.5, the instances goes down the left branch. Otherwise, it goes down the right branch.):

$$infoGain(temperature{<}77.5) = info([9,5]) - info([7,3][2,2]) = 0.025\ bits$$

$$infoGain(humidity{<}82.5) = info([9,5]) - info([6,1][3,4]) = 0.152\ bits$$

$$infoGain(windy{=}false) = info([9,5]) - info([6,2][3,3]) = 0.048\ bits$$

Note that the above four splits may not be the best splits for each attribute using the information as the splitting criterion; the selection of the best split for a particular attribute is described in the next section (i.e. splitting rules).

3.1.2 Gini index

Gini index is another criterion to measure the impurity of a node, which is used in the CART system (Breiman et al., 1984). If p_i stands for the probability that an instance

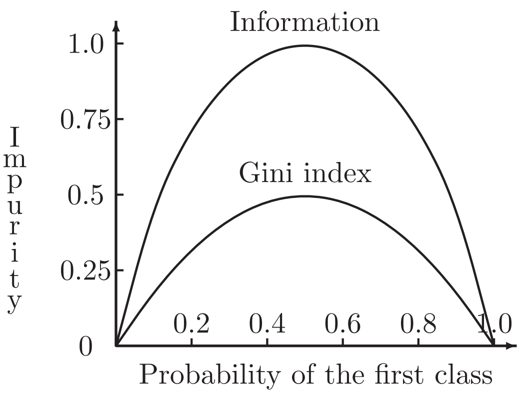

Figure 3.4: The information value and the Gini index in a two-class problem.

is in class i and p_j is the probability that the instance is in class j, the Gini index has the form (Breiman et al., 1984):

$$gini(p_1, p_2, \ldots, p_n) = \sum_{j \neq i} p_i p_j.$$

The Gini index has an interesting interpretation in terms of sample variances (Breiman et al., 1984). If all instances at a node that are in class j are assigned the value 1 and the other instances are assigned the value 0, the sample variance of these values is $p_j(1 - p_j)$, i.e. $p_j - p_j^2$. Repeating this for all classes and summing the variances, as the sum of the p_j over all classes is 1, it can be shown that the Gini index can be written as

$$gini(p_1, p_2, \ldots, p_n) = \sum_j p_j(1 - p_j) = 1 - \sum_j p_j^2.$$

When only two classes are presented, the Gini index can be simplified to $2p_1p_2$. Figure 3.4 shows the relationship between the Gini index and the class probability of the first class in a two-class problem, accompanied by the relationship between the information value and the same class probability. From the figure, we can see that in both cases, the impurity is maximal if the two classes have equal probability (i.e. 0.5 for each class). And if either class probability is 1, the impurity is 0. This means the node is absolutely pure.

Let us evaluate the splits in Figure 3.3 based on the Gini index now. The data has nine instances in class *yes* and five instances in class *no*, so the Gini index of the root node is

$$gini([\tfrac{9}{14}, \tfrac{5}{14}]) = 1 - (\tfrac{9}{14})^2 - (\tfrac{5}{14})^2 = 0.4591.$$

Evaluating the first split in the figure and calculating the Gini indexes for the two successor nodes result in:

$$gini([\tfrac{2}{5},\tfrac{3}{5}]) = 1-(\tfrac{2}{5})^2-(\tfrac{3}{5})^2 = 0.4800$$

$$gini([\tfrac{7}{9},\tfrac{2}{9}]) = 1-(\tfrac{7}{9})^2-(\tfrac{2}{9})^2 = 0.3457.$$

Taking into account the number of instances that go down each branch, the Gini index of the split and its Gini gain are

$$gini([\tfrac{2}{5},\tfrac{3}{5}],[\tfrac{7}{9},\tfrac{2}{9}]) = \tfrac{5}{14}\times 0.48+\tfrac{9}{14}\times 0.3457 = 0.3937$$

$$giniGain(outlook{=}sunny) = gini([\tfrac{9}{14},\tfrac{5}{14}]) - gini([\tfrac{2}{5},\tfrac{3}{5}],[\tfrac{7}{9},\tfrac{2}{9}]) = 0.0655.$$

Similarly, the Gini gains of the other three splits in Figure 3.3 can be calculated:

$$giniGain(temperature{<}77.5) = gini([\tfrac{9}{14},\tfrac{5}{14}]) - gini([\tfrac{7}{10},\tfrac{3}{10}],[\tfrac{2}{4},\tfrac{2}{4}]) = 0.0163$$

$$giniGain(humidity{<}82.5) = gini([\tfrac{9}{14},\tfrac{5}{14}]) - gini([\tfrac{6}{7},\tfrac{1}{7}],[\tfrac{3}{7},\tfrac{4}{7}]) = 0.0918$$

$$giniGain(windy{=}false) = gini([\tfrac{9}{14},\tfrac{5}{14}]) - gini([\tfrac{6}{8},\tfrac{2}{8}],[\tfrac{3}{6},\tfrac{3}{6}]) = 0.0306.$$

As before these four splits are just same possible splits and they may not be the best splits for each attribute if the Gini index is used. How to find the best split for each attribute base on the Gini index is described in the next section.

3.2 Splitting rules

Recall that the goal of decision tree learning is to find accurate and small models. To get the smallest trees, heuristically, we need to test all attributes and choose the one that produces the purest nodes to split at each step. In other words, the goal of splitting rules is to find the split which maximally reduces the impurity. Thus, if the information or the Gini index is used as the splitting criterion, the task of splitting rules is to find the split which leads to maximal information gain or Gini gain. In fact, finding the maximal Gini gain or information gain for a split at a node is to find the minimal values of the weighted sum of the information values or the Gini index values of its successor nodes.

This section describes splitting rules for numeric attributes and nominal attributes in best-first decision trees. For numeric attributes, the splitting rule is the same as the one used in C4.5 (Quinlan, 1993) and the CART system (Breiman et al., 1984). For nominal attributes, in multi-class problems both exhaustive search and heuristic search are discussed. The exhaustive search used is the same as in the CART system. The computation time of the exhaustive search is exponential in the number of values of a nominal attribute. Heuristic search can reduce the search time to linear. It is obvious that for an attribute that has many values heuristic search is the better choice because it can reduce computation time significantly.

3.2.1 Numeric attributes

A numeric attribute can be viewed as a sequence of ordered values. Thus, there can be many potential split points to divide training instances into two subsets, one for each pair of adjacent values. The reduction of impurity must be computed for each split point. When choosing split points, all the numeric values of training instances for the attribute concerned are first sorted in ascending order. Then the split points are set to the midpoints of two different adjacent values. The goal of the splitting rule for the numeric attribute is to find the midpoint that leads to the maximal reduction of impurity.

Suppose that the sequence of distinctive values of the attribute is sorted in ascending order: b_1, b_2, … b_m, where m is the number of distinctive values. The splitting rule evaluates the reduction of impurity of the midpoints of (b_k and b_{k+1}) where k is 1, 2, …, m-1. Thus, there are m-1 possible split points to be evaluated. When separating instances into two subsets (i.e. branches), if the attribute value of an instance is less than the split point, it is placed into the left subset. Otherwise it is placed into the right subset. The reduction of impurity of each split point is computed in order and the split point with the maximal value is selected as the split point of the attribute. The corresponding maximal reduction of impurity is the reduction of impurity of this attribute.

Now, as an example let us evaluate the splitting of the two numeric attributes in the *weather* dataset. Considering the attribute *temperature*, its sequence of numeric values in sorted order is

64 65 68 69 70 71 72 72 75 75 80 81 83 85.

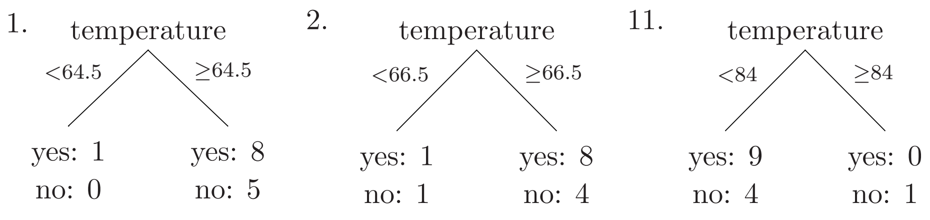

Figure 3.5: Possible binary splits on the attribute *temperature*.

Following the splitting rule described above, all midpoints between two different adjacent values in the sequence are computed. There are eleven split points, 64.5, 66.5, 68.5, 69.5, 70.5, 71.5, 73.5, 77.5, 80.5, 82 and 84. Figure 3.5 shows the situation for three split points and the other split points are illustrated in Appendix A. Based on the distributions given in the figure, we can compute the reduction of impurity for each split point and then find the best split point and its corresponding reduction of impurity. Thus, if information is used, the best split point is 84 and the corresponding information gain for this attribute is

$$infoGain(\textit{temperature}{<}84) = info([9,5]) - info([9,4][0,1]) = 0.113\ \textit{bits}$$

Similarly, the best split point for the attribute *humidity* and its corresponding information gain can be calculated as:

$$infoGain(\textit{humidity}{<}82.5) = 0.152\ \textit{bits}.$$

If the Gini index is used as the splitting criterion, the best split points and Gini gains for the two attributes are calculated as:

$$\textit{giniGain}(\textit{temperature}{<}84) = 00636$$

$$\textit{giniGain}(\textit{humidity}{<}82.5) = 0.0918.$$

We can see that, considering the *weather* dataset, no matter which splitting criterion is chosen, the split points for the two attributes are the same and the gain of the attribute *humidity* is bigger than the gain of the attribute *temperature* in both cases. Note that finding the best split point for a numeric attribute can be done in time linear in the number of instances, by simply updating the counts for distributions as the split point candidates are considered in order.

3.2.2 Nominal attributes

The splitting rule for nominal attributes is quite different from the one for numeric attributes. The goal of the splitting rule for a nominal attribute is to find a subset of attribute values that can maximally reduce impurity. When separating instances into two branches, if the value of an instance at the attribute is in the subset of values, the instance is placed into the left subset. Otherwise, it is placed into the right subset. Suppose that the nominal attribute A takes a set of values $\{a_1, a_2, \ldots, a_n\}$ at a node, where n is the number of attribute values. The goal of the splitting rule is to search for a subset of values

$$A^* = \{a_{i_1}, \ldots\} \subset \{a_1, a_2, \ldots, a_n\},$$

where the split based on this subset can maximally reduce impurity. Note that this subset can be more than one due to ties. To find such subset, 2^{n-1}-1 subsets need to be looked in the exhaustive search case as there is no difference in placing a set of values into the left branch or the right branch in a binary tree. The rest of this section introduces two search methods that can reduce the search space to linear. One is for two-class problems (Breiman et al., 1984) and the other is for multi-class problems.

The two-class problem

In a two-class problem, Breiman et al. (1984) introduced a search method that can reduce the binary search space from 2^{n-1}-1 to n-1. The principle is as follows. If the probability of being in class 1 for a given single value a_i at a node is denoted $p(1|x = a_i)$, for all single attribute values we sorted their class probabilities for class 1 as:

$$p(1|x = a_{i_1}) \leq p(1|x = a_{i_2}) \leq \ldots \leq p(1|x = a_{i_n}).$$

Then, it can be shown that one of the subsets $\{a_{i_1}, \ldots, a_{i_m}\}$ (m=1,...,$n-1$) is the subset that maximally reduces the impurity if the Gini index is used as impurity measure. We also use this strategy for the information gain. The underlying idea is that the best split should put all those attribute values leading to high probabilities in class 1 into one branch and the attribute values leading to low probabilities in class 1 into another branch.

Let us consider the attribute *outlook* in the *weather* data. If exhaustive search is used, the attribute has three possible splitting subsets. The subsets are $\{$*sunny*$\}$,

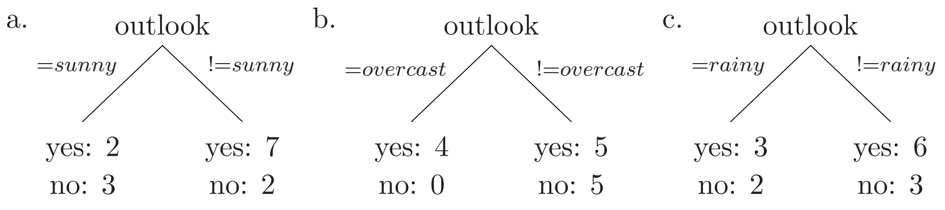

Figure 3.6: Possible binary splits on the attribute *outlook*.

$\{sunny, rainy\}$ and $\{sunny, overcast\}$ which are the same as $\{sunny\}$, $\{overcast\}$ and $\{rainy\}$ since there are only three values. The splits are indicated in Figure 3.6. The information gains for the three splits are

$$infoGain(outlook{=}sunny) = info([9,5]) - info([2,3][7,2]) = 0.102\ bits$$

$$infoGain(outlook{=}overcast) = info([9,5]) - info([5,5][4,0]) = 0.226\ bits$$

$$infoGain(outlook{=}rainy) = info([9,5]) - info([6,3][3,2]) = 0.003\ bits.$$

Thus, the subset $\{outlook{=}overcast\}$, is chosen as the split for the attribute. Let us now use Breiman's method to evaluate this attribute. The class *no* is chosen to measure probabilities for single attribute values

$$p(no|outlook{=}sunny) = \tfrac{3}{5}$$

$$p(no|outlook{=}overcast) = 0$$

$$p(no|outlook{=}rainy) = \tfrac{2}{5}$$

The sorted class probabilities are

$$p(yes|outlook{=}overcast) < p(yes|outlook{=}rainy) < p(yes|outlook{=}sunny)$$

The search space is now reduced to two subsets, $\{overcast\}$ and $\{overcast, rainy\}$. The subset $\{overcast\}$ is found to be the better split. This subset is the same as what is obtained in the exhaustive search.

Let us evaluates the splits for the attribute *outlook* again using the Gini index as the splitting criterion now. The Gini gains for the three splits in Figure 3.6 are

$$giniGain(outlook{=}sunny) = 0.066$$

$$giniGain(outlook{=}overcast) = 0.102$$

$$giniGain(outlook{=}rainy) = 0.002.$$

We still only need to evaluate the two subsets $\{overcast\}$ and $\{overcast, rainy\}$ because the probabilities for the class *no* of each attribute are not changed. The Gini gain for $\{overcast\}$ is bigger than the one for $\{overcast, rainy\}$. Thus, the subset *overcast* is still chosen as the split subset.

The multi-class problem

In a multi-class problem, Coppersmith et al. (1999) introduced a heuristic search method that achieves a compromise between reduction of impurity and search speed. For the optimal reduction of impurity, the method searches for a partition based on a separating hyperplane in the class probability space. For search speed, it assigns a scalar value to each attribute value and forms a sequence of sorted attribute values to split. Assuming that the attribute A has n values $\{a_1, a_2, \ldots, a_n\}$ and this is a k-class problem, the basic idea is as follows. First, a particular direction in class probability space is chosen. This direction defines hyperplanes that are perpendicular to this direction. The direction is selected to be the first principal component of the n points p^a, $a \in A$, where each point is weighted according to the number of instances and p^a is the vector of the class probabilities for attribute value a. The first principal component should capture as much as possible of the variation in the point in the class probability space (Coppersmith et al., 1999).

The problem is now how to compute the first principal component. Let N denote the n by k class frequency matrix whose rows are n^a: the number of instances of each class for attribute value a. And let D be the number of all instances in the case. Denote P the corresponding n by k class probability matrix whose rows are p^a. Note that if the relative frequencies for two attribute values are equal, they are combined into a single attribute value (Coppersmith et al., 1999). Let the number of instances taking value a be $D_a = \sum_{i=1}^{k} n_i^a$. The vector of mean class probabilities can be calculated as (Coppersmith et al., 1999):

$$\overline{p} = \frac{1}{D} \sum_{a \in A} n^a.$$

Then, the weighted covariance matrix of the p_a points M is calculated by (Coppersmith et al., 1999)

$$M = \frac{1}{D-1} \sum_{a \in A} D_a (p^a - \overline{p})(p^a - \overline{p})^T.$$

Attribute Value	Class		
	c_1	c_2	c_3
α_1	40	10	10
α_2	10	40	10
α_3	20	30	10
α_4	20	15	25
α_5	10	5	45

Table 3.2: The class frequencies for the artificial attribute A.

The eigenvector corresponding to the largest eigenvalue of M is the first principal component, denoted as v. Now, the direction that captures as much as possible of the variation on the points in class probability space is found. The next step is to assign scalar values to each single attribute value, which can be done by calculating the first principal component score. Let S_a=$v \cdot p_a$ denote the first principal component score for the attribute value a. Sorting the scores S_a in ascending order as $S_{a_{i_1}} \leq S_{a_{i_2}} \ldots \leq S_{a_{i_n}}$, one of the subsets of attribute values $\{a_{i_1}, \ldots, a_{i_m}\}$ (m=1,. . .,$n-1$) is chosen as the splitting subset for the attribute.

According to the paper, this procedure usually finds the splitting subset which achieves an optimal split or a split very near to the optimal split. However, it only requires n-1 total impurity evaluation time instead of 2^{n-1}-1.

Let us consider an example. Table 3.2 lists the class frequencies for an artificial attribute A that has five attribute values a_1, a_2, a_3, a_4 and a_5, and 3 classes c_1, c_2 and c_3. Then, n is 5 and k is 3. According to the formulae we described above, the class frequency matrix for this example is

$$N = \begin{vmatrix} 40 & 10 & 10 \\ 10 & 40 & 10 \\ 20 & 30 & 10 \\ 20 & 15 & 25 \\ 10 & 5 & 45 \end{vmatrix}.$$

The class probability matrix is

$$P = \begin{vmatrix} 0.667 & 0.167 & 0.167 \\ 0.167 & 0.667 & 0.167 \\ 0.333 & 0.5 & 0.167 \\ 0.333 & 0.25 & 0.417 \\ 0.167 & 0.083 & 0.75 \end{vmatrix}.$$

The vector of mean class probabilities is

$$\overline{p} = [0.333\ 0.333\ 0.333].$$

The weighted covariance matrix is

$$M = \begin{vmatrix} 10.0 & -4.167 & -5.833 \\ -4.167 & 14.167 & -10.0 \\ -5.833 & -10.0 & 15.833 \end{vmatrix}.$$

The eigenvalues of this matrix are 0, 14.796, 25.204. The first principal component, namely, the eigenvector corresponding to the largest eigenvalue 25.204 is

$$v = [-0.114\ -0.643\ 0.757].$$

The first principal component scores S_1, S_2, S_3, S_4 and S_5 for the corresponding attribute values a_1, a_2, a_3, a_4 and a_5 are

$$-0.057\ -0.321\ -0.233\ 0.117\ 0.495.$$

According to these S_a values, the sorted attribute values are a_2, a_3, a_1, a_4 and a_5. Then the n-1 (i.e. 4) subsets to be evaluated are $\{a_2\}$, $\{a_2, a_3\}$, $\{a_2, a_3, a_1\}$ and $\{a_2, a_3, a_1, a_4\}$. The Gini gains and information gains for these four subsets can be found in Table 3.3. Considering exhaustive search on this example, there are $2^4 - 1$ (i.e. 15) possible subsets. Their corresponding Gini gains and information gains are shown in Table 3.3. The maximal Gini gain and information gain is marked in bold. We can see that if the Gini index is used, the heuristic search method finds the subset $\{a_5\}$ (i.e. $\{a_2, a_3, a_1, a_4\}$) that leads to the optimal split. If the information is used, the heuristic search method also finds the subset $\{a_2, a_3\}$ that leads to the optimal split.

3.2.3 The selection of attributes

The splitting criteria and splitting rules have been described above. The next thing is to find the "best" attribute among all attributes to split on at a node. The "best"

No	a_1	a_2	a_3	a_4	a_5	Gini gain	info gain
1	0	0	0	0	1	**0.0691**	0.1467
2	0	0	0	1	0	0.0035	0.0078
3	0	0	0	1	1	0.0671	0.1482
4	0	0	1	0	0	0.0143	0.0329
5	0	0	1	0	1	0.0164	0.0352
6	0	0	1	1	0	0.0024	0.0053
7	0	0	1	1	1	0.0276	0.0630
8	0	1	0	0	0	0.0446	0.0920
9	0	1	0	0	1	0.0293	0.0666
10	0	1	0	1	0	0.0177	0.0383
11	0	1	0	1	1	0.0352	0.0788
12	0	1	1	0	0	0.0689	**0.1528**
13	0	1	1	0	1	0.0287	0.0626
14	0	1	1	1	0	0.0468	0.1101
15	0	1	1	1	1	0.0372	0.0765

Table 3.3: Splitting subsets and their corresponding Gini gains and information gains on the artificial attribute A.

attribute is the attribute that leads to the split of maximal reduction of impurity. Note that this attribute is sometimes not unique.

Let us consider the *weather* data example to find the attribute to split at the root node and its corresponding splitting value. If the information is used as the splitting criterion, the best splits for each attribute and the corresponding information gains are

$$infoGain(outlook = overcast) = 0.226\ bits$$

$$infoGain(temperature{<}84) = 0.113\ bits$$

$$infoGain(humidity{<}82.5) = 0.152\ bits$$

$$infoGain(windy = false) = 0.048\ bits.$$

If the Gini index is used, the best splits for each attribute and the corresponding Gini gains are

$$giniGain(outlook = overcast) = 0.102$$

$$giniGain(temperature{<}84) = 0.064$$

$$giniGain(humidity{<}82.5) = 0.0918$$

$$giniGain(windy = false) = 0.031.$$

In both cases, the attribute *outlook* with the split subset {*overcast*} maximally reduces impurity. Thus, it is selected to split the instances at the root node in both cases.

3.3 Best-first decision trees

Like standard decision trees, best-first decision trees are constructed in the divide-and-conquer fashion. Each nonterminal node in a best-first decision tree tests an attribute and each terminal node is assigned a classification. During the process of construction, three important things must be considered. The first one is to find the best attribute to split on at each node. The second one is to find which node in the node list (i.e. all nodes that are candidates for splitting) is to be expanded next. The third one is to make the decision when to stop growing trees.

The selection of the best attribute and the corresponding splitting value at a node have been discussed in Section 3.2: the attribute that leads to the maximal reduction of impurity is chosen to split on among all attributes. For a numeric attribute, the splitting point that achieves maximal reduction of impurity is the splitting point of the attribute, and the corresponding reduction of impurity is the reduction of impurity of the attribute. For a nominal attribute, the subset of attribute values that leads to the maximal reduction of impurity is the splitting subset of the attribute, and the corresponding reduction of impurity is the reduction of impurity of the attribute.

Best-first decision tree learning chooses the "best" node to split at each step. The "best" node is the node that has the maximal reduction of impurity among all nodes in the node list. This node can be any one in the list while it is always the same one in standard decision tree learning (as determined by the depth-first search order). To save searching time, it is a good idea to sort all nodes in the list in descending order according to Gini gain or information gain (i.e. to keep a priority queue). After sorting, the first node in the list is always the one to be expanded next. If the reduction of impurity of the first node is zero, the reduction of all nodes in the list is zero and all nodes cannot be split any more.

Regarding the stopping criteria, standard decision tree learning stops expanding a tree when all nodes are pure or the impurity of all nodes cannot be reduced by further splitting. Sometimes a minimal number of instances is required. However, besides these stopping criteria, in best-first decision tree learning we can specify a fixed number of expansions. A tree stops expanding when a fixed number of expansions is reached. In standard decision tree learning, specifying a number of expansions is not meaningful as the order of node splitting is fixed. This stopping criterion enables us to investigate new pre-pruning and post-pruning methods by choosing the fixed number of expansions based on cross-validation. This is discussed in Section 3.4.

3.3.1 The best-first decision tree learning algorithm

Figure 3.7 presents the best-first decision tree learning algorithm investigated in this thesis. Given a set of training instances E, its set of attributes A, the fixed number of expansions N and the fixed minimal number of instances at a terminal node M, the algorithm can be divided into two stages. In the first stage, the algorithm starts at the root node RN and finds the best splitting attribute A_b in A according to the reduction of impurity. E and A_b are kept in RN. Then, RN is added into the empty node list NL.

In the second stage, the first node FN in NL, and its corresponding best splitting attribute A_b and the instances reaching the node E are retrieved first. If the reduction of impurity of FN is 0 or N is reached, all nodes in the list NL are made into terminal nodes and the process of constructing the best-first decision tree is finished. Otherwise, if the split of FN on A_b would lead to a successor node with fewer than M instances, FN should not be split and it is removed from NL. Then, the algorithm executes this stage again with the new node list NL. If the split does not leads to this circumstance, FN is split into two successor nodes SN_1 and SN_2 (i.e. branches) based on its best splitting attribute A_b, and its training instances E are separated into two corresponding subsets E_1 and E_2, one for each branch. Then, the best reduction of impurity for SN_1 and SN_2, and the corresponding best splitting attributes A_{b_1} and A_{b_2}, are calculated. In the next step SN_1 with A_{b_1} and E_1, and SN_2 with A_{b_2} and E_2 are added into NL according to the reduction of impurity (i.e. NL is kept in sorted order). The number of expansions of the tree is incremented

```
function BFTree ($A$: a set of attributes,
            $E$: the training instances,
            $N$: the number of expansions,
            $M$: the minimal number of instances at a terminal node
) return a decision tree
    begin
        If $E$ is empty, return failure;
        Calculate the reduction of impurity for each attribute
              in $A$ on $E$ at the root node $RN$;
        Find the best attribute $A_b$ in $A$;
        Initialise an empty list $NL$ to store nodes;
        Add $RN$ (with $E$ and $A_b$) into $NL$;
        expandTree($NL$, $N$, $M$);
        return a tree with the root $RN$;
    end


expandTree($NL$, $N$, $M$)
    begin
        If $NL$ is empty, return;
        Get the first node $FN$ from $NL$;
        Retrieve training instances $E$ and the best splitting attribute $A_b$ of $FN$;
        If $E$ is empty, return failure;
        If the reduction of impurity of $FN$ is 0 or $N$ is reached,
            Make all nodes in $NL$ into terminal nodes;
            return;
        If the split of $FN$ on $A_b$ would result in a successor node
                with less than $M$ instances,
            Make $FN$ into the terminal node;
            Remove $FN$ from $NL$;
            expandTree($NL$, $N$, $M$);
        Let $SN_1$ and $SN_2$ be the successor nodes generated by
            splitting $FN$ on $A_b$ on $E$;
        Increment the number of expansions by one;
        Let $E_1$ and $E_2$ be the subsets of instances corresponding to
            $SN_1$ and $SN_2$;
        Find the corresponding best attributes $A_{b_1}$ for $SN_1$;
        Find the corresponding best attributes $A_{b_2}$ for $SN_2$;
        Put $SN_1$ (with $E_1$ and $A_{b_1}$) and $SN_2$ (with $E_2$ and $A_{b_2}$)
            into $NL$ according to the reduction of impurity;
        Remove $FN$ from $NL$;
        expandTree($NL$, $N$, $M$);
    end
```

Figure 3.7: The best-first decision tree learning algorithm.

by one. Next, FN is removed from NL. Finally, this stage is repeated with the new node list NL.

When building decision trees, for a numeric attribute, it is a good idea to sort the training instances by the values of the attribute at the root node and then every descendant node can use the sort order from its parent node. The reason is that, for a very large dataset, if we sort instances at each node the sorting time is very expensive. To achieve the goal, the only thing that needs to be done is to keep the sorted indexes of the parent node, and the successor nodes can then derive the indexes from the parent node. The time of the derivation is linear in the number of instances while the time of re-sorting instances is log-linear. For a nominal attribute, as mentioned before, if exhaustive search is used, the computation time is exponential in the number of values of the attribute. If heuristic search is used, the computation time grows linearly in the number of the attribute values.

3.3.2 Missing values

Missing values are endemic in real-word datasets for reasons such as malfunctioning equipment or missing measurements. One way of dealing with this problem is to simply treat them as another possible value of the attribute, which is most applicable if the fact that a value is missing plays a significant role in the decision. However, often the fact is that a value missing has no special significance. Under these circumstances, instances with missing values are similar to other instances. Witten and Frank (2005) outlined a solution by splitting instances into pieces, using a numeric weighting scheme. We use this method to deal with missing values in this thesis. The idea of this method is as follows. When building a best-first decision tree, first of all, split an attribute with the instances without missing values at a node. Then, split each instance with missing values into pieces and send part of it down each branch. Each part of the instance is weighted according to the proportion of instances without missing values going down that branch. Fractional counts are also used to estimate the class probabilities for the splitting criteria. Then, the reduction of impurity is computed by considering the combination of the class probabilities of the instances with (i.e. fractional counts) and without missing values. This process is repeated recursively. Then the various parts of the instance with missing values each reach a terminal node. When classifying an instance with missing values, the

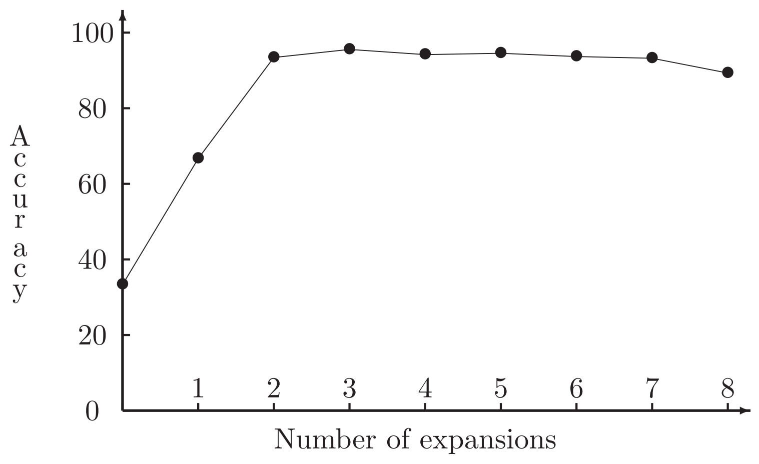

Figure 3.8: The accuracy for each number of expansions of best-first decision tree learning on the *iris* dataset.

instance is also split into pieces and each piece is put down its corresponding branch. Each part gets a new weight by multiplying its old weight by the proportion for that branch. Once all pieces of the instance reach terminal nodes, the class distributions for the instance at these terminal nodes are calculated. Then these class distributions are summed according to their weights to form the class distribution for the instance.

3.4 Pruning

As in standard decision trees, using an unpruned decision tree for classification potentially overfits the training data because of noise and variability in the data. Figure 3.8 shows the number of expansions and the corresponding accuracy based on ten-times ten-fold cross-validation for best-first decision tree learning on the *iris* dataset. Note that the minimal number of instances at a terminal node has been set to one in this case. The figure illustrates that, first, the accuracy increases when the expansions start. Then, the accuracy peaks when the number of expansions reaches three and further expansions do not improve performance. Thus, they should not be performed (i.e. pre-pruning) or they should be deleted if they have already performed (i.e. post-pruning).

The goal of pruning is to only retain those parts that truly reflect the underlying information in the data and remove the others. The pruning methods presented in this thesis, namely, best-first-based pre-pruning and post-pruning, try to find the appropriate number of expansions for best-first decision trees. The tree size is decided according to the error estimate obtained from an (internal) cross-validation. We investigate two types of error estimates, the classification error rate and the root mean squared error to see their corresponding performance.

The calculation of the error rate for a decision tree is straightforward. Let M_{error} be the number of instances that are classified incorrectly by the tree and let M be the total number of instances used in the testing set. The error rate is M_{error} divided by M. This value is between 0 and 1 (0 and 1 included).

Let $p_1 \dots p_k$ be the actual class probabilities of an instance in a k-class problem (which are either 0 or 1) and $q_1 \dots q_k$ be the class probabilities of the instance as obtained from the decision tree model. The root mean squared error of all testing instances can be calculated by the formula

$$RMSE(E) = \sqrt{\frac{\sum_M \frac{\sum_k (p_i - q_i)^2}{k}}{M}}.$$

The error rate only considers whether an instance is classified correctly. The root mean squared error considers the classification distance between the actual class and predicted probabilities for an instance.

3.4.1 Best-first-based pre-pruning

Like in standard decision trees, pre-pruning in best-first decision trees stops expanding a tree early when further splitting step appears to increase error estimate (i.e. the error rate or the root mean squared error in this thesis). Recall best-first-based pre-pruning considered here is based on cross-validation. To this end, the trees for all training folds are constructed in a parallel fashion (e.g. ten trees for a *ten-fold cross-validation*). For each number of expansions, the average error estimate is calculated based on the temporary trees in all folds. Best-first-based pre-pruning stops growing the trees when further splitting increases the average error estimate and the previous number of expansion is chosen as the final number of expansions. Then, the final tree is built according to the chosen number of expansions on all the data. Figure 3.9 shows the best-first-based pre-pruning algorithm.

In all folds, each best-first tree expands a node at a time. Note that in each fold,

```
function prePruning ($A$: a set of attributes,
            $E$: the training instances,
            $M$: the minimal number of instances at a terminal node,
            $F$: the number of folds
) return a pre-pruned decision tree
    begin
        Initialise the number of expansions $N = 0$;
        Divide $E$ into $F$ subsets, $E_k$, where $k$=1,...,$F$;
        Repeat
            For $k$ =1 to $F$,
                BFTree($A$, $E - E_k$, $N$, $M$);
                Evaluate the tree to get the error estimate on $E_k$;
            Calculate the average error estimate $\overline{E_e}$ on all folds;
            If the current $\overline{E_e}$ is larger than the previous $\overline{E_e}$,
                break the loop;
            Increment $N$ by one;
        BFTree($A$, $E$, $N$, $M$);
    end
```

Figure 3.9: The best-first-based pre-pruning algorithm.

when expanding a tree, we do not need to re-generate the tree. We only need to keep the previous tree and select the next "best" node from the node list to split. This can significantly save computation time. Like in pre-pruning methods for standard decision trees, the above algorithm also suffers from the problem that the process may stop too early, and further splitting may decrease the error estimate again. This is also known as the *horizon problem*. Table 3.4 presents an example of this problem on the *glass* dataset. The first column N is the number of expansions. The number of folds in the internal cross-validation was set to ten. The minimal number of instances at the terminal nodes was set to two. The minimal error estimates are marked in bold. Note that the error estimates listed are the internal cross-validated error estimates.

If the error rate is used, best-first-based pre-pruning selects five expansions and stops expanding because the error rate increases after five expansions. However, after six expansions, the error rate actually decreases again. In fact, the minimal error rate appears when the number of expansions reaches 24. The error rate of 24 expansions is about 10% smaller than that of five expansions. In this case, best-first-based pre-pruning performs really bad. If the root mean squared error is used, the performance is even worse. With this criterion only one expansion is selected. The problem can be relieved using the new 1SE rule, which is described

N	Error rate	Root mean squared error	N	Error rate	Root mean squared error
0	0.6478	0.3247	14	0.2576	0.2557
1	0.5307	*0.2970	15	0.2487	0.2558
2	0.5210	0.3003	16	0.2719	0.2595
3	0.3846	0.2829	17	0.2671	0.2620
4	0.3519	0.2714	18	0.2721	0.2649
5	* 0.3422	0.2693	19	0.2673	0.2650
6	0.3470	0.2715	20	0.2864	0.2660
7	0.3242	0.2689	21	0.2623	0.2605
8	0.3050	0.2634	22	0.2524	0.2571
9	0.3190	0.2665	23	0.2669	0.2596
10	0.3000	0.2666	24	**0.2462**	**0.2534**
11	0.2859	0.2641	25	0.2489	0.2591
12	0.2859	0.2633	26	0.3182	0.3033
13	0.2716	0.2601	27	0.3182	0.3042

Table 3.4: The error rate and the root mean squared error for each expansion using best-first decision tree learning on the *glass* dataset.

later. Note that this example also illustrates that smaller trees can perform better than complex trees.

3.4.2 Best-first-based post-pruning

For best-first-based post-pruning, trees in all training folds are also constructed in a parallel fashion. For each number of expansions, the average error estimate is calculated based on the temporary trees in all folds. This step is repeated until the trees cannot be expanded any more. Then a sequence of the number of expansions and their corresponding error estimates based on the cross-validation can be calculated. The number of expansions whose average error estimate is minimal is chosen as the final number of expansions. The final tree is then built according to the chosen number of expansions on all the data. Figure 3.10 presents the best-first-based post-pruning algorithm.

Like in best-first-based pre-pruning, in each fold, when expanding a tree, we only need to keep the previous tree and select the next "best" node from the node list to split to save computation time. Best-first-based post-pruning does not suffer from the *horizon problem* described in best-first-based pre-pruning as it considers all possible expansions regardless whether an expansion increases the error estimate or not. Continuing the example in Table 3.4, best-first-based post-pruning selects 24 expansions in both the

```
function postPruning ($A$: a set of attributes,
            $E$: the training instances,
            $M$: the minimal number of instances at a terminal node,
            $F$: the number of folds
) return a post-pruned decision tree
    begin
        Initialise the number of expansions $N = 0$;
        Divide $E$ into $F$ subsets, $E_k$, where $k$=1,. . .,$F$;
        Repeat until all trees cannot be expanded
            For $k$ =1 to $F$,
                BFTree($A$, $E - E_k$, $N$, $M$);
                Evaluate the tree to get the error estimate on $E_k$;
            Calculate the average error estimate $\overline{E_e}$ on all folds;
            Increment $N$ by one;
        Find the number of expansions $N_{final}$ with the minimal error estimate;
        BFTree($A$, $E$, $N_{final}$, $M$);
    end
```

Figure 3.10: The best-first-based post-pruning algorithm.

error rate case and the root mean squared error case. In this case, best-first-based post-pruning performs much better than best-first-based pre-pruning. The horizon effect is the reason why post-pruning is preferred in most cases. However, pre-pruning has the advantage that it can save computation time: the tree is only expanded five times if the error rate is used or one time if the root mean squared error is used.

3.4.3 The 1SE rule in best-first-based pre-pruning and post-pruning

In best-first-based post-pruning, we always choose the optimal tree that yields the minimal error estimate. However, like in minimal cost-complexity pruning that is used in the CART system (Breiman et al., 1984), the optimal tree is not stable due to noise and sample variance in the data and the instability of the cross-validation procedure. The 1SE rule (Breiman et al.,1984) is a good technique to avoid the instability. The formula for calculating the standard error and the principle of the 1SE rule have been discussed in Chapter 2. The idea behind the 1SE rule in best-first-based post-pruning is to sacrifice a statistically insignificant drop in accuracy to decrease tree complexity and improve the stability in the final selection of tree. Continuing the example described in Table 3.4, the error estimates with one standard errors are shown in Table 3.5. According to the 1SE rule, best-first-based post-pruning chooses the smallest tree whose error estimate is less or equal to the minimal error plus 1SE of the minimal error. Thus, in Table 3.5 best-first-based

N	Error rate	Root mean squared error	N	Error rate	Root mean squared error
0	0.6478±0.0327	0.3247±0.0320	14	0.2576±0.0299	0.2557±0.0298
1	0.5307±0.0341	0.2970±0.03120	15	0.2487±0.0295	0.2558±0.0298
2	0.5210±0.03410	0.3003±0.0313	16	0.2719±0.0304	0.2595±0.0300
3	0.3846±0.0333	0.2829±0.0308	17	0.2671±0.0302	0.2620±0.0301
4	0.3519±0.0326	0.2714±0.0304	18	0.2721±0.0304	0.2649±0.0302
5	0.3422±0.0324	0.2693±0.0303	19	0.2673±0.0303	0.2650±0.0302
6	0.3470±0.0325	0.2715±0.0304	20	0.2864±0.0309	0.2660±0.03020
7	0.3242±0.0320	0.2689±0.0303	21	0.2623±0.0301	0.2605±0.0300
8	0.3050±0.0315	0.2634±0.0301	22	0.2524±0.0297	0.2571±0.0299
9	0.3190±0.0319	0.2665±0.0302	23	0.2669±0.0302	0.2596±0.0300
10	0.300±0.0313	0.2666±0.0302	24	0.2462±0.0294	0.2534±0.0297
11	0.2859±0.0309	0.2641±0.0301	25	0.2489±0.0296	0.2591±0.0300
12	0.2859±0.0309	0.2633±0.0301	26	0.3182±0.0318	0.3033±0.0314
13	0.2716±0.0304	0.2601±0.0300	27	0.3182±0.0318	0.3042±0.0314

Table 3.5: The error rate and the root mean squared error for each expansion with 1SE using best-first decision tree learning on the *glass* dataset.

post-pruning chooses 13 expansions instead of 24 expansions if the error rate is used as the error estimate and it chooses three expansions instead of 24 expansions if the root mean squared error is used.

In best-first-based pre-pruning, 1SE can be used to yield more expansions. We call it "the new 1SE rule" in this thesis. The idea of the new 1SE rule is that in the process of building a tree the minimal error estimate is recorded. Note that the minimal error estimate changes during the expanding process. If the current error estimate is larger than the current minimal error plus its 1SE, splitting stops. Then, the "standard" 1SE rule is applied to all expansions obtained so far to decide on the final tree size like the one in best-first-based post-pruning. The underlying idea is that if the current error estimate is larger than the current minimal error estimate plus its 1SE, it is unlikely that further splitting will result in the error estimate smaller than the current minimal error. This method can help with the problem from Table 3.4: when the root mean squared error was used, best-first-based pre-pruning selected only one expansion. However, this method suffers from the problem that in the worst case, it would fully expand the trees. Considering the example of the *glass* dataset described above, according to the new 1SE rule, if the error rate is used, the trees would be expanded 26 times and the selected number of expansions is 13. If the root mean squared error is used, best-first-based pre-pruning would also expand 26 time and chooses three expansions, which is much better than the one without the new 1SE rule.

3.5 Complexity of best-first decision tree induction

The best-first decision tree learning algorithm and the two new pruning algorithms have been discussed. Now, let us consider their computational complexity using the big O notation. The method used here is similar to the one used in Witten and Frank (2005): $O(n)$ stands for a quantity that grows at most linearly with n and $O(n^2)$ stands for a quantity that grows at most quadratically with n. Assume that a tree is built on a dataset which has n training instances and m attributes, and the depth of the tree is on the order of $O(\log n)$. In the building process, if we assume each node requires effort that is linear in the number of instances, as all instances need to be considered at each level, the amount of work for one attribute is $O(n \log n)$. Also, at each node, all attributes are considered, so the time complexity of building the tree is $O(mn \log n)$.

There is a difference between the time complexity of numeric attributes and nominal attributes. In the case of all numeric attributes, the complexity of sorting instances for an attribute at the root node is $O(n \log n)$. If descendant nodes derive the sorting order from the root node, the complexity is still $O(mn \log n)$. In the case of all nominal attributes, at each node we need to find the best subset of an attribute values to split. Assume the number of values of an attribute is k. In the exhaustive search case, the computation time for the best subset search for an attribute is $2^{k-1} - 1$ and in big O notation this is $O(2^k)$. As the tree is binary, the number of nodes is $2n - 1$ at most and in big O notation this is $O(n)$, so the worst-case time complexity for a nominal attribute is $O(2^k n)$. Thus, the time complexity for building the tree is $O(m2^k n)$. In the heuristic search case, as the time required for the best subset search for an attribute becomes $O(k)$, the time complexity for building the tree is $O(kmn)$.

Let F be the number of folds. Let $O(X)$ (where X refers to one of the three possible time complexities described above) be the time complexity for building a tree. If best-first-based pre-pruning or post-pruning are applied, as in each fold a tree needs to be built, the time complexity for building trees in all folds would be FX and in big O notation it is $O(X)$ because F is a small constant compared to n. In each fold, an error estimate is computed for every node. In the worst case, the number of nodes is $2n-1$ and in big O notation this is $O(n)$. Thus the time complexity for the estimates for all folds is $F(2n-1)$ and in big O notation this is $O(n)$. Obviously, there is no difference between best-first-based pre-pruning and post-pruning because in the worst case best-first-based pre-pruning needs to fully expand the tree. Thus, the computation time of the whole process is $O(X)+O(n)$, which is O(X).

3.6 Summary

This chapter discussed the details of best-first decision tree induction. Two splitting criteria used in the induction algorithm were discussed first, namely, the Gini index and the information. These splitting criteria were introduced to measure impurity of a node. We presented the formulae of calculating the Gini index and the information. The value of the Gini index and the information is based on class distributions. For an absolutely pure node, the value of the Gini index and the information is zero. The reduction of impurity, namely, the Gini gain and the information gain, was also discussed. We used the *weather* dataset as the example to illustrate how to calculate the value of the Gini index, the information, the Gini gain and the information gain.

Then, we discussed the splitting rules for numeric attributes and nominal attributes respectively. For a numeric attribute, the values of the attribute are first sorted in an ascending order. Then, the decrease of impurity of all mid-points between two different adjacent values are evaluated and the one that leads to the maximal reduction of impurity is chosen as the split point of the attribute. For a nominal attribute, the goal of the splitting rule is to find the subset of values of the attribute that leads to the maximal reduction of impurity. We discussed not only exhaustive search but also heuristic search to solve this problem. The exhaustive search method is straightforward: it evaluates all possible subsets. Because this is exponential in the number of values of a nominal attribute, effective search methods must be considered. We introduced Breiman's (1984) method to solve two-class problems and Coppersmith's method (1999) to solve multi-class problems. After the explanation of the splitting rules, the method of selecting the best attribute to split on was discussed.

In the third section, we discussed best-first expansion for decision trees. The best-first decision tree learning algorithm was examined first. Simply speaking, a best-first tree splits the "best" node at each step where the "best" node is the node which has maximal reduction of impurity among all nodes available to split. Then, we discussed the method of how to deal with missing values in the best-first decision tree learning algorithm. The method splits instances with missing values into pieces and weights them according to the proportion of instances in branches in both training and testing. When testing an instance, the pieces of the instance with missing values are merged together to classify it.

The last two sections discussed the two new pruning methods and the computational complexity of best-first decision tree induction. The best-first-based pre-pruning and post-pruning algorithms were presented in detail. An example was given to illustrate the selection of the number of expansions. The differences between the two algorithms were briefly discussed. The 1SE rule was used in best-first-based post-pruning in order to find the smallest tree that leads to a statistically insignificant drop in accuracy compared to the tree with the minimal error estimate. The new 1SE rule in best-first-based pre-pruning was used to yield more expansions. An example of selecting the tree with the 1SE rule was presented on the *glass* dataset. The computational complexity of best-first decision tree induction was discussed using big O notation. We discussed the time complexity for building a tree for two different conditions, namely, based on the all numeric attributes case or the all nominal attributes case. In the all nominal attributes case, the complexity of exhaustive search and heuristic search were discussed separately. Then, the time complexities of the best-first-based pre-pruning and post-pruning were presented.

Chapter 4
Experiments

This chapter evaluates best-first decision tree induction on real-world datasets. The chapter is organised as follows. Datasets and methodology used in the experiments are discussed in Section 4.1. Section 4.2 tests whether using different splitting criteria (i.e. the Gini index or the information) results in different performance in terms of accuracy and tree size when best-first-based post-pruning is used. Section 4.3 evaluates whether using different error estimates (i.e. the error rate or the root mean squared error) to determine the number of expansions in the internal cross-validation results in different performance. This is evaluated for both best-first-based pre-pruning and post-pruning. The comparison of the performance between the heuristic search method and the exhaustive search method for finding splits for nominal attributes in multi-class problems is discussed in Section 4.4. The effects of the 1SE rule in best-first-based pre-pruning and post-pruning are discussed in Section 4.5. Section 4.6 compares best-fist-based pre-pruning to best-first-based post-pruning both with and without the 1SE rule. The comparison of the two new pruning methods to minimal cost-complexity pruning is discussed in Section 4.7. Section 4.8 discusses whether training data size influences tree size in best-first-based pre-pruning and post-pruning. Section 4.9 summarises the findings from the experiments.

4.1 Datasets and methodology

In order to compare the performance of the different pruning methods presented in this thesis, 38 standard benchmark datasets from the UCI Machine Learning Repository (Blake et al., 1998) are used in the experiments. The datasets and their properties are listed in Table 4.1. Some datasets only contain nominal attributes or numeric attributes, and others contain a mix of nominal attributes and numeric attributes. Some datasets contains missing values and others do not. About half of the datasets represent two-class domains and the other half represent multi-class domains. Thus, these datasets have a good representation of real-world machine learning problems.

In practice, the accuracy on the training data is not a good indicator of a classifier's future performance. Otherwise, it would be easy to achieve 100% accuracy. Instead, a separate sample of test instances must be used to get an unbiased estimate. One way to achieve this is to divide the original data into two subsets. One subset is used to build the classifier and the other subset is used to estimate the error for the classifier. This is known as the *hold-out* method. However, the performance of the *hold-out* method depends very much on how the original data is divided and it may be significantly different for different divisions, especially if the original data is small. This is known as sample variance. Cross-validation is designed to solve this problem.

The purpose of cross-validation here is a little different from the one described in the last chapter (i.e. internal cross-validation), which is used to determine the number of expansions. The purpose of cross-validation here is to evaluate the performance of the three cross-validation-based pruning methods presented in this thesis by generating an accurate estimate of performance. The basic idea of cross-validation is still the same: in k-fold cross-validation, the original data is divided into k approximately equal partitions and each in turn is used for testing and the remainder is used for training. An error estimate is calculated on the test set for each fold and the k resulting error estimates are averaged to get an overall error estimate. In this thesis, to further reduce sample variance in the data, all datasets are stratified first before they are used in cross-validation.

Usually the number of folds in the cross-validation is set to ten. This number has been found empirically to be a good choice (Kohavi, 1995b) and this idea is supported using a theoretical result by Kearns (1996). This thesis uses ten times stratified ten-fold cross-validation. When randomising the original data each time, the seed is set differently before the data is divided into ten parts. Thus, each time a model is built on different data and classification is also based on different data. This reduces variance further (Kohavi, 1995b).

To measure whether two learning schemes perform differently, we need to make

Dataset	Instances	Missing values	Numeric attributes	Nominal attributes	Classes
anneal	898	no	6	32	5
arrhythmia	452	yes	206	73	13
audiology	226	yes	0	69	24
autos	205	yes	15	10	6
balance-scale	625	no	4	0	3
breast-cancer	286	yes	0	9	2
breast-w	699	yes	9	0	2
horse-colic	368	yes	7	15	2
credit-rating	690	yes	6	9	2
german_credit	1000	no	7	13	2
pima_diabetes	768	no	8	0	2
ecoli	336	no	7	0	8
glass	214	no	9	0	6
heart-c	303	yes	6	7	2
heart-h	294	yes	6	7	2
heart-statlog	270	no	13	0	2
hepatitis	155	yes	6	13	12
hypothyroid	3772	yes	7	22	4
ionosphere	351	no	34	0	2
iris	150	no	4	0	3
kr-vs-kp	3196	no	0	36	2
labor	57	yes	8	8	2
letter	20000	no	16	0	26
lymphography	148	no	3	15	4
mushroom	8124	yes	0	22	2
optdigits	5620	no	64	0	10
pendigits	10992	no	16	0	10
primary-tumor	339	yes	0	17	21
segment	2310	no	19	0	7
sick	3772	yes	7	22	2
sonar	208	no	60	0	2
soybean	683	yes	0	35	19
splice	3190	no	0	61	3
vehicle	846	no	18	0	4
vote	435	yes	0	16	2
vowel	990	no	10	3	11
waveform	5000	no	41	0	3
zoo	101	no	1	16	7

Table 4.1: The 38 UCI datasets and their properties.

them run on the same data using the same ten-fold cross-validation and the same ten randomisation. After this step, the 100 estimates can be used to make a fair comparison. Because the mean of a limited number of cross-validation estimates is approximately normally distributed around the "true" mean (Frank, 2000), the ten-times ten-fold cross-validation described above can output the information used to compare the two schemes. A two-tailed corrected resampled paired t-test (Nadeau & Bengio, 2003) is used in this thesis to determine whether the results of the cross-validation show that there is a difference between the two schemes. The corrected test is used because the samples are not independent. In this thesis, we call a difference in performance "significant" according to the t-test at a specific significance level. The significance level was set to 5% in the experiments.

Note that as the *splice* dataset has a nominal attribute with 3318 values and it is extremely time-consuming to compute the best binary split for the attribute for both heuristic search and exhaustive search, the attribute was deleted in the experiments. During the experiments (except in Section 4.4), the heuristic search method was used to find the split for nominal attributes in multi-class problems when the number of values of a nominal attribute is larger than four. When the number of values of a nominal attribute is less than or equal to four, the exhaustive search method was used because the exhaustive search time for a nominal attribute with four values is not large.

4.2 Gini index versus information

As mentioned before, both the Gini index and the information can be used as the splitting criteria to construct best-first decision trees. This section compares best-first decision tree learning with post-pruning using the Gini index and the information. Only best-first-based post-pruning was used in the experiment because we want to look at the effect of different splitting criteria on fully-expanded trees and best-first-based pre-pruning may stop too early. The performance of the two schemes is compared in terms of classification accuracy and tree size. Both the error rate and the root mean squared error were used as the error estimates in the internal cross-validation for determining the number of expansions. Results are presented for both measures. Except for the splitting criteria, the parameter settings for the two schemes being compared were the same. The minimal number of instances at the terminal nodes was set to two.

The accuracy of the two schemes is presented in Table 4.2 and their corresponding tree size is presented in Table 4.3. Note there are two groups of results in each table. In each table, the first group uses the error rate as the error estimate in the internal cross-validation (consisting of the second column and the third column). The second group uses the root mean squared error as the error estimate (consisting of the fourth column and the fifth column). The third column is compared to the second column and the fifth column is compared to the fourth column. In Table 4.2 the ◦ symbol indicates that the scheme using the information produces significantly less accurate trees than the scheme using the Gini index. In Table 4.3, the ◦ symbol indicates that the scheme using the information produces significantly smaller trees than the scheme using the Gini index. The • symbol indicates that the scheme using the information produces significantly larger trees than the scheme using the Gini index. In each table if there is no ◦ symbol or • symbol, it means that there is no significant difference between the two schemes.

Table 4.2 shows that if the error rate is used as the error estimate in the internal cross-validation, there is no significant difference between the two schemes on all datasets in terms of accuracy. If the root mean squared error is used, there is no significant difference in almost all datasets. The only exception is the *waveform* dataset where the scheme using the information generates a significantly less accurate tree than the scheme using the Gini index.

Table 4.3 illustrates that if the error rate is used as the error estimate, the scheme using the information produces significantly larger trees than the scheme using the Gini index on four datasets: *anneal, letter, mushroom* and *waveform.* The former scheme generates significantly smaller trees on only two datasets: *kr-vs-kp* and *soybean.* On the other datasets there is no significant difference between the two schemes. If the root mean squared error is used, the scheme using the information generates significantly larger trees than the scheme using the Gini index on three datasets: *anneal, letter* and *mushroom.* The tree generated by the former scheme on *kr-vs-kp* and *soybean* is significantly smaller than the one generated by the latter

Dataset	Error rate		Root mean squared error	
	Gini index	information	Gini index	information
anneal	98.32	98.22	98.33	98.19
arrhythmia	67.28	64.16	66.77	63.45
audiology	75.39	74.72	75.09	74.42
autos	75.37	79.52	75.47	79.42
balance-scale	78.87	78.15	78.95	78.04
breast-cancer	69.24	69.09	69.63	69.14
breast-w	94.34	94.22	94.38	93.97
horse-colic	84.04	84.50	84.59	84.67
credit-rating	84.09	83.67	84.70	83.29
german_credit	72.40	72.13	71.53	70.31
pima_diabetes	73.20	72.64	74.23	74.06
ecoli	82.80	82.08	82.59	82.09
glass	70.39	68.24	69.69	66.65
heart-c	76.20	75.18	76.17	74.85
heart-h	77.66	76.32	78.18	77.05
heart-statlog	77.00	75.96	76.22	75.15
hepatitis	77.87	78.45	78.14	78.53
hypothyroid	99.50	99.57	99.49	99.57
ionosphere	88.84	89.35	88.90	89.63
iris	94.20	94.47	94.27	94.40
kr-vs-kp	99.42	99.48	99.41	99.48
labor	84.63	82.30	84.80	81.57
letter	87.08	87.56	87.10	87.57
lymphography	78.01	76.70	78.14	76.85
mushroom	99.96	100.00	99.96	100.00
optdigits	90.35	90.79	90.39	90.79
pendigits	96.14	96.42	96.15	96.42
primary-tumor	39.27	39.80	35.81	35.55
segment	95.78	96.52	95.80	96.54
sick	98.87	98.95	98.86	98.96
sonar	71.64	74.05	72.60	73.65
soybean	91.23	91.33	91.23	91.42
newSplice	94.30	94.38	94.22	94.37
vehicle	70.06	71.98	66.48	67.49
vote	94.82	94.87	94.85	95.12
vowel	79.74	80.68	79.83	80.64
waveform	76.21	75.57	74.60	72.06 ○
zoo	40.31	40.61	40.61	40.61
All	(●/ /○)	(0/38/0)		(0/37/1)

Table 4.2: The accuracy of best-first-based post-pruning using (a) the Gini index and (b) the information.

Dataset	Error rate			Root mean squared error		
	Gini index	information		Gini index	information	
anneal	25.60	28.50	●	25.66	28.38	●
arrhythmia	66.84	75.76		61.42	64.10	
audiology	47.12	46.50		47.22	44.72	
autos	51.56	48.50		51.26	48.36	
balance-scale	160.94	162.14		164.80	165.22	
breast-cancer	28.46	28.50		9.96	6.20	
breast-w	37.80	40.60		41.08	42.26	
horse-colic	152.80	152.74		133.40	130.32	
credit-rating	43.64	46.42		16.88	23.40	
german_credit	94.96	122.96		11.06	10.10	
pima_diabetes	69.90	71.66		22.52	15.84	
ecoli	32.78	36.66		31.66	33.18	
glass	41.98	47.18		36.30	40.56	
heart-c	36.94	46.16		26.38	33.36	
heart-h	31.22	41.98		23.42	24.16	
heart-statlog	37.88	43.82		32.90	29.08	
hepatitis	50.24	47.62		43.68	42.36	
hypothyroid	46.52	51.20		46.56	51.54	
ionosphere	19.08	20.96		17.80	20.00	
iris	9.84	9.46		10.62	10.34	
kr-vs-kp	85.62	73.28	○	85.52	73.40	○
labor	21.18	20.86		21.78	21.42	
letter	2341.30	2416.52	●	2348.24	2423.46	●
lymphography	21.50	27.04		22.26	25.40	
mushroom	13.38	16.50	●	13.40	16.52	●
optdigits	386.04	391.48		390.04	392.62	
pendigits	393.02	388.84		393.98	390.02	
primary-tumor	123.50	127.66		66.98	62.32	
segment	85.84	86.32		86.22	86.88	
sick	85.06	83.44		85.00	83.38	
sonar	19.12	24.66		10.12	17.00	
soybean	284.36	213.88	○	282.38	204.10	○
newSplice	81.58	92.74		80.88	88.92	
vehicle	139.52	146.76		66.28	72.56	
vote	66.40	73.10		57.82	62.80	
vowel	184.16	181.82		184.66	181.84	
waveform	360.40	529.98	●	102.48	141.98	
zoo	1.12	1.60		1.00	1.00	
All	(●/ /○)	(4/32/2)			(3/33/2)	

Table 4.3: The tree size of best-first-based post-pruning using (a) the Gini index and (b) the information.

scheme. On the other datasets there is no significant difference between the two schemes.

Discussion

As mentioned before, the goal of decision tree learning is to find accurate and small models. Considering accuracy, the Gini index appears to be a better choice than the information as the splitting criterion because the scheme using the Gini index is never significantly worse and significantly better in one case. Considering tree size, if the error rate is used as the error estimate, the scheme using the Gini index generates significantly smaller trees on four datasets. On the contrary, the latter scheme produces significantly smaller trees on only two datasets. A similar thing also happens if the root mean squared error is used: three significantly smaller trees for the Gini index and only two significantly smaller trees for the information. Thus, if only tree size is considered regardless of accuracy, the Gini index still appears to be a better choice than the information. To sum up, the Gini index performs better than the information as the splitting criterion for best-first decision tree learning on the UCI datasets used in the experiment. Thus, only the Gini index is applied to best-first decision tree learning in the following experiments.

4.3 Error rate versus root mean squared error

As described in the last chapter, both the error rate and the root mean squared error can be used as the error estimates in the internal cross-validation to determine the number of expansions for best-first decision tree learning. This section discusses the effect of changing the error estimate on the performance of both best-first-based pre-pruning and post-pruning. As before, the performance is measured in terms of classification accuracy and tree size, and the experiment is organised into two groups. In each group, the scheme using the error rate is compared to the scheme using the root mean squared error. The first group considers best-first-based post-pruning and the second group considers best-first-based pre-pruning. The experiment is organised in the same way when tree size is considered. As mentioned in the last section , the Gini index was used in the experiment. The minimal number of instances at the terminal nodes was again set to two.

The accuracy of best-first-based pre-pruning and post-pruning using the two different error estimates is listed in Table 4.4 and their corresponding tree size is listed in Table 4.5. In each table, the first group consists of the second column (post-pruning using the error rate) and the third column (post-pruning using the root mean squared error). And the second group consists of the fourth column (pre-pruning using the error rate) and the fifth column (pre-pruning using the root mean squared error). Note that in both tables the third column is compared to the the second column and the fifth column is compared to the fourth column.

In Table 4.4 the ∘ symbol in the third column indicates that post-pruning using the root mean squared error produces significantly less accurate trees than the one using the error rate. The ∘ symbol in the fifth column indicates that pre-pruning using the root mean squared error generates significantly less accurate trees than the one using the error rate. The • symbol in the fifth column indicates that pre-pruning using the root mean squared error generates significantly more accurate trees than the one using the error rate.

In Table 4.5 the ∘ symbol in the third column indicates that post-pruning using the root mean squared error produces significantly smaller trees than post-pruning using the error rate. The ∘ symbol in the fifth column indicates that pre-pruning using the root mean squared error produces significantly smaller trees than pre-pruning using the error rate. The • symbol in the fifth column indicates that pre-pruning using the root mean squared error generates significantly larger trees than pre-pruning using the error rate. In both tables if there is no ∘ symbol or • symbol, it means that there is no significant difference between the two schemes.

From table 4.4, we can see that post-pruning using the root mean squared error as the error estimate in the internal cross-validation produces significantly less accurate trees than the one using the error rate on two datasets: *vehicle* and and *waveform*. On the other datasets there is no significant difference between the two schemes. Moreover, pre-pruning using the root mean squared error generates significantly less accurate trees than the one using the error rate on one dataset: *glass*. It also generates significantly more accurate trees on one dataset: *kr-vs-kp*. On the other datasets there is no significant difference between the two schemes.

Dataset	Post-pruning			Pre-pruning		
	error rate	RMSE		error rate	RMSE	
anneal	98.32	98.33		96.19	97.22	
arrhythmia	67.28	66.77		59.22	58.85	
audiology	75.39	75.09		65.05	61.52	
autos	75.37	75.47		60.47	56.26	
balance-scale	78.87	78.95		66.70	66.20	
breast-cancer	69.24	69.53		69.46	69.80	
breast-w	94.34	94.38		94.15	94.28	
horse-colic	84.04	84.59		85.45	85.65	
credit-rating	84.09	84.70		85.13	85.14	
german_credit	72.40	71.53		71.23	70.74	
pima_diabetes	73.20	74.23		74.25	74.40	
ecoli	82.80	82.59		81.52	81.49	
glass	70.39	69.69		66.34	56.51	○
heart-c	76.20	76.17		74.21	76.33	
heart-h	77.66	78.18		79.75	79.34	
heart-statlog	77.00	76.22		73.19	73.85	
hepatitis	77.87	78.14		78.40	78.02	
hypothyroid	99.50	99.49		99.44	99.34	
ionosphere	88.84	88.90		89.32	89.32	
iris	94.20	94.27		94.53	94.60	
kr-vs-kp	99.42	99.41		94.90	98.13	●
labor	84.63	84.80		79.70	78.33	
letter	87.08	87.10		11.17	10.97	
lymphography	78.01	78.14		78.02	77.96	
mushroom	99.96	99.96		99.96	99.88	
optdigits	90.35	90.39		50.10	52.88	
pendigits	96.14	96.15		87.82	86.76	
primary-tumor	39.27	35.81		25.55	28.73	
segment	95.78	95.80		93.34	93.40	
sick	98.87	98.86		98.17	98.15	
sonar	71.64	72.60		72.11	71.91	
soybean	91.23	91.23		87.93	82.13	
newSplice	94.30	94.22		88.87	89.55	
vehicle	70.06	66.48	○	62.91	63.47	
vote	94.82	94.85		95.31	95.49	
vowel	79.74	79.83		36.59	38.03	
waveform	76.21	74.60	○	70.01	69.90	
zoo	40.31	40.61		40.61	40.61	
All	(●/ /○)	(0/36/2)			(1/36/1)	

Table 4.4: The accuracy of best-first-based pre-pruning and post-pruning using (a) the error rate and (b) the root mean squared error.

Dataset	Post-pruning			Pre-pruning		
	error rate	RMSE		error rate	RMSE	
anneal	25.60	25.66		12.98	18.26	●
arrhythmia	66.84	61.42		5.04	4.36	
audiology	47.12	47.22		22.16	16.50	
autos	51.16	51.26		18.88	14.82	
balance-scale	160.94	164.80		13.64	12.32	
breast-cancer	28.46	9.96	○	2.28	3.76	
breast-w	37.80	41.08		10.92	11.24	
horse-colic	152.80	133.40		7.14	6.70	
credit-rating	43.64	16.88	○	5.52	5.80	
german_credit	94.96	11.06	○	7.06	5.20	
pima_diabetes	69.90	25.22	○	6.66	6.48	
ecoli	32.78	31.66		14.08	14.72	
glass	41.98	36.30		13.56	8.22	○
heart-c	36.94	26.38		5.66	6.90	
heart-h	31.22	23.42		4.76	6.44	●
heart-statlog	37.88	32.90		5.60	6.44	
hepatitis	50.24	43.68		2.86	2.98	
hypothyroid	46.52	46.56		22.86	18.10	
ionosphere	19.08	17.80		6.16	5.54	
iris	9.84	10.62		8.12	9.00	
kr-vs-kp	85.62	85.52		15.90	45.34	●
labor	21.18	21.78		7.92	7.22	
letter	2341.30	2348.24		42.68	36.54	
lymphography	21.50	22.26		5.66	5.68	
mushroom	13.38	13.40		13.40	11.70	
optdigits	386.04	390.04		76.02	85.42	
pendigits	393.02	393.98		169.30	194.32	
primary-tumor	123.50	66.98	○	4.26	8.12	
segment	85.84	86.22		31.30	33.36	
sick	85.06	85.00		12.40	14.30	
sonar	19.12	10.12	○	3.92	3.74	
soybean	284.36	282.38		80.96	72.98	
newSplice	81.58	80.88		13.76	21.06	●
vehicle	139.52	66.28	○	15.40	16.34	
vote	66.40	57.82		7.48	7.44	
vowel	184.16	184.66		41.20	36.18	
waveform	360.40	102.48	○	21.12	18.46	
zoo	1.12	1.00		1.00	1.00	
All	(●/ /○)	(0/30/8)			(4/33/1)	

Table 4.5: The tree size of best-first-based pre-pruning and post-pruning using (a) the error rate and (b) the root mean squared error.

Table 4.5 shows that post-pruning using the root mean squared error generates significantly smaller trees than post-pruning using the error rate on eight datasets. On the other datasets there is no significant difference between the two schemes. Furthermore, trees generated by pre-pruning using the root mean squared error are significantly larger than those generated by pre-pruning using the error rate on four datasets. The latter scheme generates significantly larger trees on only one datasets: *newSplice*. On the other datasets there is no significant difference between the two schemes.

Discussion

To sum up, if only accuracy is considered, for best-first-based post-pruning, the error rate appears to be a better choice than the root mean squared error because on all datasets the accuracy of the scheme using the error rate is better than or equal to the accuracy of the scheme using the root mean squared error. Considering best-first-based pre-pruning, there is no difference between the two error estimates (i.e. each one generates a significantly more accurate tree). If only tree size is considered, the root mean squared error is preferred for post-pruning because its corresponding scheme generates more significantly smaller trees than the scheme using the error rate. The error rate is preferred for pre-pruning because of the same reason. As the most important goal of decision tree learning is to find accurate models, the error rate appears to overall be a better choice than the root mean squared error. The latter measure appears to result in too much pruning on several datasets. Furthermore, this thesis shows that we can use the 1SE rule to reduce tree size in best-first-based post-pruning without significantly losing accuracy (see section 4.5). Thus, we choose the error rate as the error estimate in the internal cross-validation in order to obtain more accurate models in the following experiments.

4.4 Heuristic search versus exhaustive search

The experiments in this section evaluate the performance of heuristic search and exhaustive search for nominal attributes on multi-class datasets. We only test heuristic search and exhaustive search in conjunction with best-first-based post-pruning because the differences between the two search methods are much more obvious in

Dataset	Max num of values	Datasets	Max num of values
anneal	8	primary-tumor	3
audiology	6	soybean	7
autos	22	splice	3178
hypothyroid	5	vowel	15
lymphography	8	zoo	100

Table 4.6: The multi-class datasets used for comparing heuristic search and exhaustive search.

fully-expanded trees. As we have found that the performance of best-first-based post-pruning using the Gini index and the error rate is better than that using the information and the root mean squared error in terms of accuracy, the experiments in this section used the Gini index and the error rate. The minimal number of instances at the terminal nodes was again set to two. Note that, in this section for nominal attributes, the experiments were run using heuristic search and exhaustive search separately while in other sections we use the combination of heuristic search and exhaustive search (i.e. as described in Section 4.1).

As the experiments in this section evaluate the search methods for those datasets that have multiple classes and nominal attributes, there are only 12 datasets that meet these conditions. Moreover, if all nominal attributes in a dataset only have two values, the heuristic search method is not needed. Thus, only ten datasets were selected for the experiments. Some properties of these ten datasets are listed in Table 4.6. The second column and the fourth column in the table indicate the number of values of the attribute that has the largest number of values among all attributes in a dataset.

Table 4.7 lists the accuracy for heuristic search and exhaustive search using best-first-based post-pruning. In the table, cells with question marks indicate that the training time of the search method is extremely expensive on the dataset because the number of values for one or more attributes is too large. There are seven datasets that can be run using both heuristic search and exhaustive search. Considering accuracy, there is no significant difference between the heuristic search method and the exhaustive search method on these seven datasets.

Table 4.8 presents experimental results obtained after deleting attributes that are

Dataset	Heuristic	Exhaustive
anneal	98.42	98.48
audiology	76.41	75.44
autos	75.37	?
hypothyroid	99.51	99.50
lymphography	77.05	77.94
primary-tumor	39.65	39.27
soybean	92.23	90.86
splice	?	?
vowel	79.73	81.53
zoo	40.31	?

Table 4.7: The accuracy for heuristic search and exhaustive search.

Dataset	Heuristic	Exhaustive
newAutos	75.19	74.12
newSplice	94.07	94.17
newZoo	92.11	92.11

Table 4.8: The accuracy for heuristic search and exhaustive search on three datasets whose attribute with maximum number of values has been deleted.

extremely expensive to run (i.e. *make* in *autos*, *instance_name* in *splice* and *animal* in *zoo*). The table shows that the accuracy for the two search methods is not significantly different on the three datasets.

4.5 The effects of the 1 SE rule in best-first-based pre-pruning and post-pruning

This section discusses the effects of the 1 SE rule in best-first-based post-pruning and pre-pruning regarding accuracy and tree size. It presents experimental results for the two pruning methods both with and without the 1SE rule. Recall that the 1SE rule in best-first-based pre-pruning is used to yield more expansions, which is different from the standard 1SE rule. The 1SE rule in post-pruning is the same as the one in minimal cost-complexity pruning: it chooses the smallest tree whose error estimate is less or equal to the minimal error plus 1SE of the minimal error. Again, the experiment in this section used the Gini index as the splitting criterion and the error rate as the error estimate in the internal cross-validation. The minimal number of instances at the terminal nodes was set to two.

Table 4.9 and Table 4.10 list the accuracy and tree size of the two pruning methods both with and without the 1 SE rule. In each table, there are two groups of results as before. The first group consists of the second column (i.e. post-pruning without the 1SE rule) and the third column (i.e. post-pruning with the 1SE rule). The second group consists of the fourth column (i.e. pre-pruning without the new 1SE rule) and the fifth column (i.e. pre-pruning with the new 1SE rule). Note that in both tables the third column is compared to the the second column and the fifth column is compared to the fourth column.

4.5.1 Accuracy

In Table 4.9, the • symbol in the fifth column indicates that best-first-based pre-pruning with the new 1SE rule produces significantly more accurate trees than the pre-pruning without the new 1SE rule. If there is no • symbol, it means that there is no significant difference between the two schemes. The table shows that considering best-first-based post-pruning, there is no significant difference between the scheme with the 1SE rule and the scheme without the 1SE rule on all datasets in terms of accuracy. Considering best-first-based pre-pruning, the scheme with the new 1SE rule produces significantly more accurate trees on 15 datasets. On the other datasets, there is no significant difference between the two schemes.

4.5.2 Tree size

In Table 4.10, the ◦ symbol in the third column indicates that best-first-based post-pruning with the 1SE rule generates significantly more smaller trees than the post-pruning without the 1SE rule. The • symbol in the fifth column indicates that best-first-based pre-pruning with the new 1SE rule generates significantly larger trees than the pre-pruning without the new 1SE rule. The table shows that, considering best-first-based post-pruning, the scheme with the 1SE rule produces significantly smaller trees than the scheme without the 1SE rule on 26 datasets. On the other datasets there is no significant difference between the two schemes. Regarding best-first-based pre-pruning, the scheme with the new 1SE rule generates significantly larger trees than the scheme without the new 1SE rule on 23 datasets. On the other datasets there is no significant difference between the two schemes.

Dataset	Post-pruning		Pre-pruning	
	without 1SE	with 1SE	without 1SE	with 1SE
anneal	98.32	98.32	96.19	98.26 ●
arrhythmia	67.28	66.68	59.22	66.11 ●
audiology	75.39	74.07	65.05	73.90 ●
autos	75.37	74.16	60.47	73.78 ●
balance-scale	78.87	78.47	66.70	77.82 ●
breast-cancer	69.24	68.58	69.46	68.51
breast-w	94.34	94.28	94.15	94.22
horse-colic	84.04	84.72	85.45	85.56
credit-rating	84.09	84.61	85.13	85.29
german_credit	72.40	72.54	71.23	72.45
pima_diabetes	73.20	73.60	74.25	74.23
ecoli	82.80	82.21	81.52	81.89
glass	70.39	69.45	66.34	68.84
heart-c	76.20	75.96	74.21	75.59
heart-h	77.66	79.49	79.75	79.41
heart-statlog	77.00	76.04	73.19	74.59
hepatitis	77.87	78.29	78.40	78.20
hypothyroid	99.50	99.50	99.44	99.50
ionosphere	88.84	88.75	89.32	89.00
iris	94.20	94.27	94.53	94.27
kr-vs-kp	99.42	99.37	94.90	97.51 ●
labor	84.63	83.43	79.70	82.13
letter	87.08	86.94	11.17	86.83 ●
lymphography	78.01	77.60	78.02	77.00
mushroom	99.96	99.96	99.96	99.96
optdigits	90.35	90.20	50.10	90.14 ●
pendigits	96.14	96.08	87.82	96.05
primary-tumor	39.27	38.32	25.55	36.79 ●
segment	95.78	95.63	93.34	95.54 ●
sick	98.87	98.79	98.17	98.74 ●
sonar	71.64	71.88	72.11	71.45
soybean	91.23	90.56	87.93	90.92
newSplice	94.30	93.75	88.87	93.92 ●
vehicle	70.06	69.52	62.91	69.01 ●
vote	94.82	95.01	95.31	95.38
vowel	79.74	79.08	36.59	78.19 ●
waveform	76.21	76.01	70.01	75.89 ●
zoo	40.31	40.61	40.61	40.61
All	(●/ /○)	(0/38/0)		(15/23/0)

Table 4.9: The accuracy of best-first-based pre-pruning and post-pruning both with and without the 1SE rule.

Dataset	Post-pruning			Pre-pruning		
	without 1SE	with 1SE		without 1SE	with 1SE	
anneal	25.60	25.14		12.98	24.56	●
arrhythmia	66.84	49.94	○	5.04	47.74	●
audiology	47.12	43.40	○	22.16	42.74	●
autos	51.16	47.50	○	18.88	46.10	●
balance-scale	160.94	139.76		13.64	125.78	●
breast-cancer	28.46	14.84		2.28	3.60	
breast-w	37.80	28.28	○	10.92	21.72	
horse-colic	152.80	93.06		7.14	9.72	
credit-rating	43.64	19.12	○	5.52	4.02	
german_credit	94.96	43.76	○	7.06	16.82	
pima_diabetes	69.90	35.38	○	6.66	7.20	
ecoli	32.78	25.42	○	14.08	23.18	●
glass	41.98	32.70	○	13.56	27.94	●
heart-c	36.94	24.12	○	5.66	11.96	
heart-h	31.22	18.60	○	4.76	8.20	
heart-statlog	37.88	28.40	○	5.60	16.04	●
hepatitis	50.24	32.92	○	2.86	9.40	
hypothyroid	46.52	40.90		22.86	38.96	●
ionosphere	19.08	14.50	○	6.16	8.20	
iris	9.84	8.36	○	8.12	8.36	
kr-vs-kp	85.62	82.36		15.90	56.70	●
labor	21.18	18.28		7.92	16.06	●
letter	2341.30	2277.28	○	42.68	2241.92	●
lymphography	21.50	17.14		5.66	12.18	●
mushroom	13.38	13.38		13.40	13.36	
optdigits	386.04	361.08	○	76.02	351.06	●
pendigits	393.02	378.94	○	169.30	370.66	●
primary-tumor	123.50	93.60	○	4.26	75.24	●
segment	85.84	80.58	○	31.30	77.72	●
sick	85.06	73.74	○	12.40	65.78	●
sonar	19.12	13.62	○	3.92	7.86	
soybean	284.36	202.10	○	80.96	174.90	●
newSplice	81.58	70.88		13.76	53.40	●
vehicle	139.52	119.98	○	15.40	103.02	●
vote	66.40	50.60		7.48	6.54	
vowel	184.16	176.22	○	41.20	169.82	●
waveform	360.40	240.48	○	21.12	172.08	●
zoo	1.12	1.00		1.00	1.00	
All	(●/ /○)	(0/12/26)			(23/15/0)	

Table 4.10: The tree size of best-first-based pre-pruning and post-pruning both with and without the 1SE rule.

4.5.3 Discussion

Recall the goal of decision tree learning is to find accurate and small trees. The 1SE rule is a good choice for best-first-based post-pruning because it can significantly reduce tree size on most datasets without significantly losing accuracy on any dataset. Regarding best-first-based pre-pruning, although the scheme with the new 1SE rule produces significantly larger trees on 23 datasets, it generates significantly more accurate trees for 15 datasets. As mentioned before, accuracy is the most important criterion for decision tree learning. Thus, best-first-based pre-pruning with the new 1SE rule is a better choice than the pre-pruning without the new 1SE rule.

4.6 Comparing best-first-based pre-pruning and post-pruning

This section compares best-first-based pre-pruning to best-first-based post-pruning in terms of classification accuracy, tree size and training time. We compare the two pruning methods with and without the 1SE rule separately. In other words, pre-pruning without the new 1SE rule is compared to post-pruning without the 1SE rule, and pre-pruning with the new 1SE rule is compared to post-pruning with the 1SE rule. Again, the experiments in this section used the Gini index and the error rate. The minimal number of instances at the terminal nodes was set to two.

4.6.1 Accuracy

Table 4.11 presents the accuracy of best-first-based pre-pruning and post-pruning both with and without the 1SE rule. Note that, as before, the third column (pre-pruning without the new 1SE rule) is compared to the second column (post-pruning without the 1SE rule), and the fifth column (pre-pruning with the new 1SE rule) is compared to the fourth column (post-pruning with the 1SE rule). The ◦ symbol in the third column indicates that pre-pruning without the 1SE rule produces significantly less accurate trees than post-pruning without the 1SE rule.

Table 4.11 shows that pre-pruning without the new 1SE rule produces significantly less accurate trees than post-pruning without the 1SE rule on 15 datasets. There is no significant difference between the two schemes on the other datasets. Furthermore,

Dataset	Without the 1SE rule			With the 1SE rule	
	post-pruning	pre-pruning		post-pruning	pre-pruning
anneal	98.32	96.19	○	98.32	98.26
arrhythmia	67.28	59.22	○	66.68	66.11
audiology	75.39	65.05	○	74.07	73.90
autos	75.37	60.47	○	74.16	73.78
balance-scale	78.87	66.70	○	78.47	77.82
breast-cancer	69.24	69.46		68.58	68.51
breast-w	94.34	94.15		94.28	94.22
horse-colic	84.04	85.45		84.72	85.56
credit-rating	84.09	85.13		84.61	85.29
german_credit	72.40	71.23		72.54	72.45
pima_diabetes	73.20	74.25		73.60	74.23
ecoli	82.80	81.52		82.21	81.89
glass	70.39	66.34		69.45	68.84
heart-c	76.20	74.21		75.96	75.59
heart-h	77.66	79.75		79.49	79.41
heart-statlog	77.00	73.19		76.04	74.59
hepatitis	77.87	78.40		78.29	78.20
hypothyroid	99.50	99.44		99.50	99.50
ionosphere	88.84	89.32		88.75	89.00
iris	94.20	94.53		94.27	94.27
kr-vs-kp	99.42	94.90	○	99.37	97.51
labor	84.63	79.70		83.43	82.13
letter	87.08	11.17	○	86.94	86.83
lymphography	78.01	78.02		77.60	77.00
mushroom	99.96	99.96		99.96	99.96
optdigits	90.35	50.10	○	90.20	90.14
pendigits	96.14	87.82		96.08	96.05
primary-tumor	39.27	25.55	○	38.32	36.79
segment	95.78	93.34	○	95.63	95.54
sick	98.87	98.17	○	98.79	98.74
sonar	71.64	72.11		71.88	71.45
soybean	91.23	87.93		90.56	90.82
newSplice	94.30	88.87	○	93.75	93.92
vehicle	70.06	62.91	○	69.52	69.01
vote	94.82	95.31		95.01	95.38
vowel	79.74	36.59	○	79.08	78.19
waveform	76.21	70.01	○	76.01	75.89
zoo	40.31	40.61		40.61	40.61
All	(●/ /○)	(0/23/15)			(0/38/0)

Table 4.11: Comparing the accuracy of best-first-based pre-pruning and post-pruning (both with and without the 1SE rule).

there is no significant difference in accuracy between pre-pruning with the new 1SE rule and post-pruning with the 1SE rule on any dataset. Thus, we can say that pre-pruning using the new 1SE rule is as good as post-pruning using the 1SE rule in terms of accuracy.

4.6.2 Tree size

Table 4.12 lists the tree size of best-first-based pre-pruning and post-pruning both with and without the 1SE rule. As before, the third column is compared to the second column and the fifth column is compared to the fourth column. The ∘ symbol in the third column indicates that pre-pruning without the new 1SE rule produces significantly smaller trees than post-pruning without the 1SE rule. The ∘ symbol in the fifth column indicates that pre-pruning with the new 1SE rule produces significantly smaller trees than post-pruning with the 1SE rule.

Table 4.12 shows that pre-pruning without the new 1SE rule produces significantly smaller trees than post-pruning without the 1SE rule on 35 datasets. There is no significant difference between the two schemes on the other datasets. Moreover, pre-pruning with the new 1SE rule generates significantly smaller trees than post-pruning with the 1SE rule on six datasets. On the other datasets there is no significant difference between the two schemes.

4.6.3 Training time

Table 4.13 presents the training time of best-first-based pre-pruning and post-pruning both with and without the 1SE rule. Note that the training time in the table is measured in seconds. As before, the third column is compared to the second column and the fifth column is compared to the fourth column. The ∘ symbol in the third column indicates that the training time of pre-pruning without the new 1SE rule is significantly smaller than the training time of post-pruning without the 1SE rule. The ∘ symbol in the fifth column indicates that the training time of pre-pruning with the new 1SE rule is significantly smaller than the training time of post-pruning with the 1SE rule. If there is no ∘ symbol it means that there is no significant difference between the two schemes.

Dataset	Without the 1SE rule			With the 1SE rule		
	post-pruning	pre-pruning		post-pruning	pre-pruning	
anneal	25.60	12.98	∘	25.14	24.56	
arrhythmia	66.84	5.04	∘	49.94	47.74	
audiology	47.12	22.16	∘	43.40	42.74	
autos	51.16	18.88	∘	47.50	46.10	
balance-scale	160.94	13.64	∘	139.76	125.78	
breast-cancer	28.46	2.28	∘	14.84	3.60	
breast-w	37.80	10.92	∘	28.28	21.72	
horse-colic	152.80	7.14	∘	93.06	9.72	∘
credit-rating	43.64	5.52	∘	19.12	4.02	
german_credit	94.96	7.06	∘	43.76	16.82	
pima_diabetes	69.90	6.66	∘	35.38	7.20	
ecoli	32.78	14.08	∘	25.42	23.18	
glass	41.98	13.56	∘	32.70	27.94	
heart-c	36.94	5.66	∘	24.12	11.96	
heart-h	31.22	4.76	∘	18.60	8.20	
heart-statlog	37.88	5.60	∘	28.40	16.04	∘
hepatitis	50.24	2.86	∘	32.92	9.40	∘
hypothyroid	46.52	22.86	∘	40.90	38.96	
ionosphere	19.08	6.16	∘	14.50	8.20	∘
iris	9.84	8.12		8.36	8.36	
kr-vs-kp	85.62	15.90	∘	82.36	56.70	∘
labor	21.18	7.92	∘	18.28	16.06	
letter	2341.30	42.68	∘	2277.28	2241.92	
lymphography	21.50	5.66	∘	17.14	12.18	
mushroom	13.38	13.40		13.38	13.36	
optdigits	386.04	76.02	∘	361.08	351.06	
pendigits	393.02	169.30	∘	378.94	370.66	
primary-tumor	123.50	4.26	∘	93.60	75.24	
segment	85.84	31.30	∘	80.58	77.72	
sick	85.06	12.40	∘	73.74	65.78	
sonar	19.12	3.92	∘	13.62	7.86	
soybean	284.36	80.96	∘	202.10	174.90	
newSplice	81.58	13.76	∘	70.88	53.40	
vehicle	139.52	15.40	∘	119.98	103.02	
vote	66.40	7.48	∘	50.60	6.54	∘
vowel	184.16	41.20	∘	176.22	169.82	
waveform	360.40	21.12	∘	240.48	172.08	
zoo	1.12	1.00		1.00	1.00	
All	(●/ /∘)	(0/3/35)			(0/32/6)	

Table 4.12: Comparing the tree size of best-first-based pre-pruning and post-pruning (both with and without the 1SE rule).

The table shows that the training time of pre-pruning without the new 1SE rule is significantly smaller than the training time of post-pruning without the 1SE rule on 36 datasets. On the other datasets there is no significant difference between the two schemes. The table also shows that the training time of pre-pruning with the new 1SE rule is significantly smaller than the training time of post-pruning with the 1SE rule on 19 datasets. There is no significant difference between the two schemes on the other datasets.

4.6.4 Discussion

The accuracy, the tree size and the training time of the two new pruning methods have been presented above. Considering the accuracy, best-fist-based pre-pruning without the new 1SE rule is not a good choice because the accuracy produced by the scheme is significantly lower than the accuracy of best-first-based post-pruning without the 1SE rule on 15 datasets. In other words, best-first-based pre-pruning without the new 1SE rule stops too early on about half of the datasets. However, this scheme has two advantages. The tree size generated by this scheme is significantly smaller on 35 datasets. Moreover, the training time of the scheme is significantly smaller on 36 datasets. Thus, for very large datasets, best-first-based pre-pruning without the new 1SE rule may be worthy of consideration.

When the new 1SE rule is considered, the accuracy of best-first-based pre-pruning is statistically indistinguishable from the accuracy of best-first-based post-pruning. Moreover, the tree size generated by the scheme is significantly smaller on six datasets. The training time of the scheme is also significantly smaller on half of the datasets. Thus best-first-based pre-pruning with the new 1SE rule is a better choice than best-first-based post-pruning with the 1SE rule on the UCI datasets used in the experiments. (Note that the three tables also show the different performance between pre-pruning with the new 1SE rule and post-pruning without the 1SE rule. For example, pre-pruning with the new 1SE rule yields 1.91 significant losses in accuracy when compared to post-pruning without the 1SE rule on the *kr-vs-kp* dataset.)

Dataset	Without the 1SE rule		With the rule	
	post-pruning	pre-pruning	post-pruning	pre-pruning
anneal	2.56	2.08 ○	2.54	2.52
arrhythmia	12.54	2.82 ○	12.29	11.51
audiology	2.00	1.23 ○	2.00	1.96
autos	0.86	0.52 ○	0.85	0.84
balance-scale	0.31	0.06 ○	0.29	0.28
breast-cancer	0.51	0.16 ○	0.49	0.26 ○
breast-w	0.24	0.12 ○	0.22	0.20
horse-colic	2.14	0.44 ○	2.10	0.88 ○
credit-rating	1.33	0.59 ○	1.27	0.68 ○
german_credit	4.02	1.25 ○	3.91	2.41 ○
pima_diabetes	0.57	0.14 ○	0.55	0.30 ○
ecoli	0.15	0.09 ○	0.14	0.13
glass	0.20	0.11 ○	0.20	0.19
heart-c	0.45	0.20 ○	0.43	0.32 ○
heart-h	0.37	0.18 ○	0.35	0.25 ○
heart-statlog	0.18	0.07 ○	0.17	0.12
hepatitis	0.36	0.08 ○	0.34	0.18 ○
hypothyroid	8.31	5.22 ○	8.19	7.76
ionosphere	0.69	0.42 ○	0.66	0.50 ○
iris	0.02	0.02	0.02	0.02
kr-vs-kp	8.83	6.14 ○	8.82	7.85 ○
labor	0.06	0.04 ○	0.06	0.06
letter	265.57	21.78 ○	312.70	351.75
lymphography	0.32	0.18 ○	0.31	0.27
mushroom	22.63	22.55	22.62	22.42 ○
optdigits	25.30	9.87 ○	24.96	24.53
pendigits	18.78	10.22 ○	18.69	18.20 ○
primary-tumor	1.33	0.24 ○	1.33	1.12 ○
segment	3.29	2.48 ○	3.28	3.26
sick	10.45	3.68 ○	10.31	9.12 ○
sonar	0.88	0.44 ○	0.85	0.65 ○
soybean	8.43	4.81 ○	8.40	8.27
newSplice	62.93	29.01 ○	62.89	44.75 ○
vehicle	1.27	0.40 ○	1.21	1.13
vote	0.68	0.23 ○	0.66	0.30 ○
vowel	4.34	1.85 ○	4.31	4.26
waveform	25.61	6.28 ○	25.35	20.25 ○
zoo	0.45	0.33 ○	0.45	0.33 ○
All	(●/ /○)	(0/2/36)		(0/19/19)

Table 4.13: Comparing the training time of best-first-based pre-pruning and post-pruning (both with and without the 1SE rule).

4.7 Comparing best-first-based pre-pruning and post-pruning to minimal cost-complexity pruning

This section compares best-first-based pre-pruning and post-pruning to minimal cost-complexity pruning in terms of classification accuracy, tree size and training time. The comparison is performed both with and without the 1SE rule. Best-first-based pre-pruning and post-pruning without the 1SE rule were compared to minimal cost-complexity pruning without the 1SE rule, and best-first-based pre-pruning and post-pruning with the 1SE rule were compared to minimal cost-complexity pruning with the 1SE rule. As before, the Gini index was selected as the splitting criterion. The error rate was selected as the error estimate in the internal cross-validation. The minimal number of instances at the terminal nodes was set to two. Note that in Table 4.14, Table 4.15 and Table 4.16, the third column and the fourth column are compared to the second column, and the sixth column and the seven column are compared to the fifth column.

4.7.1 Accuracy

Table 4.14 presents the accuracy of the three pruning methods both with and without the 1SE rule. The ∘ symbol in the third and the fourth column indicates that best-first-based post-pruning and pre-pruning without the 1SE rule produces significantly less accurate trees than minimal cost-complexity pruning without the 1SE rule respectively. The ∘ symbol in the seventh column indicates that best-first-based pre-pruning with the new 1SE rule produces significantly less accurate trees than minimal cost-complexity pruning with the 1SE rule. The • symbol in the sixth and the seventh column indicates that best-first-based post-pruning and pre-pruning with the 1SE rule produces significantly more accurate trees than minimal cost-complexity pruning with the 1SE rule.

We can see that if the 1SE rule is not used, best-first-based post-pruning produces significantly less accurate trees than minimal cost-complexity pruning on one dataset: *arrhythmia*. On the other datasets there is no significant difference between the two schemes. Best-first-based pre-pruning produces significantly less accurate trees than minimal cost-complexity pruning on about half of the datasets. On the other datasets there is no significant difference between the two schemes.

If the 1SE rule is used, best-first-based post-pruning produces significantly more accurate trees than minimal cost-complexity pruning on one dataset: *pendigits*. On the other datasets there is no significant difference between the two schemes. Best-first-based pre-pruning also produces significantly more accurate trees than minimal cost-complexity pruning on the *pendigits* dataset. However, the latter scheme produces significantly more accurate trees than the former scheme on the *arrhythmia* dataset. On the other datasets there is no significant difference between the two schemes.

4.7.2 Tree size

Table 4.15 presents the tree size of the three pruning methods both with and without the 1SE rule. The $\bullet$ symbol in the third column indicates that best-first-based post-pruning without the 1SE rule generates significantly larger trees than minimal cost-complexity pruning without the 1SE rule. The $\circ$ symbol in the fourth column indicates that best-first-based pre-pruning without the new 1SE rule generates significantly smaller trees than minimal cost-complexity pruning without the 1SE rule. The $\bullet$ symbol in the sixth and the seventh column indicates that best-first-based post-pruning and pre-pruning with the 1SE rule generates significantly larger trees than minimal cost-complexity pruning with the 1SE rule respectively.

We can see that if the 1SE rule is not used, best-first-based post-pruning generates significantly larger trees than minimal cost-complexity pruning on 32 datasets. On the other datasets there is no significant difference between the two schemes. Best-first-based pre-pruning produces significantly smaller trees than minimal cost-complexity pruning on 27 datasets. On the other datasets there is no significant difference between the two schemes. If the 1SE rule is used, best-first-based post-pruning produces significantly larger trees than minimal cost-complexity pruning on 30 datasets. On the other datasets there is no significant difference between the two schemes. Best-first-based pre-pruning produces significantly larger trees than minimal cost-complexity pruning on 22 datasets. On the other datasets there is no significant difference between the two schemes.

Dataset	Without the 1SE rule			With the 1SE rule		
	CART pruning	post-pruning	pre-pruning	CART pruning	post-pruning	pre-pruning
anneal	98.33	98.32	96.19 ○	98.11	98.32	98.26
arrhythmia	71.49	67.28 ○	59.22 ○	69.79	66.68	66.11 ○
audiology	74.56	75.39	65.05 ○	71.69	74.07	73.90
autos	75.08	75.37	60.47 ○	70.71	74.16	73.78
balance-scale	78.76	78.87	66.70 ○	77.76	78.47	77.82
breast-cancer	70.64	69.24	69.46	70.36	68.58	68.51
breast-w	94.71	94.34	94.15	94.29	94.28	94.22
horse-colic	84.86	84.04	85.45	85.32	84.72	85.56
credit-rating	84.96	84.09	85.13	85.29	84.61	85.29
german_credit	73.69	72.40	71.23 ○	73.61	72.54	72.45
pima_diabetes	74.57	73.20	74.25	75.05	73.60	74.23
ecoli	82.54	82.80	81.52	81.95	82.21	81.89
glass	70.93	70.39	66.34	68.93	69.45	68.84
heart-c	78.54	76.20	74.21	77.89	75.96	75.59
heart-h	78.55	77.66	79.75	79.68	79.49	79.41
heart-statlog	78.41	77.00	73.19 ○	77.63	76.04	74.59
hepatitis	77.73	77.87	78.40	77.82	78.29	78.20
hypothyroid	99.48	99.50	99.44	99.52	99.50	99.50
ionosphere	88.87	88.84	89.32	89.15	88.75	89.00
iris	94.47	94.20	94.53	93.67	94.27	94.27
kr-vs-kp	99.33	99.42	94.90 ○	99.21	99.37	97.51
labor	80.33	84.63	79.70	80.00	83.43	82.13
letter	87.23	87.08	11.17 ○	86.89	86.94	86.83
lymphography	77.47	78.01	78.02	75.99	77.60	77.00
mushroom	99.95	99.96	99.96	99.94	99.96	99.96
optdigits	90.43	90.35	50.10 ○	89.91	90.20	90.14
pendigits	96.21	96.14	87.82	95.65	96.08 ●	96.05 ●
primary-tumor	41.16	39.27	25.55 ○	39.62	38.32	36.79
segment	95.90	95.78	93.34 ○	95.24	95.63	95.54
sick	98.87	98.87	98.17 ○	98.75	98.79	98.74
sonar	71.35	71.64	72.11	71.09	71.88	71.45
soybean	91.45	91.23	87.93	90.01	90.56	90.82
newSplice	94.57	94.30	88.87 ○	94.39	93.75	93.92
vehicle	69.73	70.06	62.91 ○	68.82	69.52	69.01
vote	95.93	94.82	95.31	95.49	95.01	95.38
vowel	80.18	79.74	36.59 ○	78.10	79.08	78.19
waveform	76.71	76.21	70.01 ○	76.27	76.01	75.89
zoo	40.61	40.31	40.61	40.61	40.61	40.61
All	(●/ /○)	(0/37/1)	(0/21/17)		(1/37/0)	(1/36/1)

Table 4.14: Comparing the accuracy of the two new pruning methods and minimal cost-complexity pruning (both with and without the 1SE rule).

Dataset	Without the 1SE rule			With the 1SE rule		
	CART pruning	post-pruning	pre-pruning	CART pruning	post-pruning	pre-pruning
anneal	20.46	25.60 ●	12.98 ○	18.38	25.14 ●	24.56 ●
arrhythmia	21.74	66.84 ●	5.04 ○	16.66	49.94 ●	47.74 ●
audiology	38.32	47.12 ●	22.16 ○	24.06	43.40 ●	42.74 ●
autos	45.38	51.16 ●	18.88 ○	33.18	47.50 ●	46.10 ●
balance-scale	66.38	160.94 ●	13.64 ○	25.80	139.76 ●	125.78 ●
breast-c	7.14	28.46 ●	2.28	2.02	14.84	3.60
breast-w	17.12	37.80 ●	10.92 ○	9.56	28.28 ●	21.72 ●
horse-colic	12.12	152.80 ●	7.14	5.90	93.06 ●	9.72
credit-rating	6.56	43.64 ●	5.52	3.34	19.12	4.02
german_credit	24.14	94.96 ●	7.06 ○	10.46	43.76	16.82
pima_diabetes	24.48	69.90 ●	6.66 ○	7.10	35.38	7.20
ecoli	25.24	32.78	14.08 ○	14.18	25.42 ●	23.18 ●
glass	25.84	41.98 ●	13.56 ○	13.40	32.70 ●	27.94 ●
heart-c	12.72	36.94 ●	5.66 ○	8.66	24.12 ●	11.96
heart-h	14.62	31.22 ●	4.76 ○	5.72	18.60	8.20
heart-statlog	15.68	37.88 ●	5.60 ○	8.76	28.40 ●	16.04
hepatitis	11.58	50.24 ●	2.86	2.30	32.92 ●	9.40
hypothyroid	20.42	46.52 ●	22.86	14.60	40.90 ●	38.96 ●
ionosphere	10.78	19.08 ●	6.16	5.78	14.50 ●	8.20
iris	7.82	9.84	8.12	6.24	8.36	8.36
kr-vs-kp	69.06	85.62 ●	15.90 ○	55.90	82.36 ●	56.70
labor	13.00	21.18 ●	7.92	5.80	18.28 ●	16.06 ●
letter	2256.90	2341.30 ●	42.68 ○	1855.16	2277.28 ●	2241.92 ●
lymphography	13.74	21.50 ●	4.26 ○	5.82	17.14 ●	12.18 ●
mushroom	13.26	13.38	13.40	13.10	13.38	13.36
optdigits	323.14	386.04 ●	76.02 ○	216.60	361.08 ●	351.06 ●
pendigits	358.84	393.02 ●	169.30 ○	273.62	378.94 ●	370.66 ●
primary-t	36.20	123.50 ●	4.26 ○	15.86	93.60 ●	75.24 ●
segment	82.38	85.84	31.30 ○	64.28	80.58 ●	77.72 ●
sick	51.10	85.06 ●	12.40 ○	34.20	73.74 ●	65.78 ●
sonar	12.84	19.12 ●	3.92 ○	4.94	13.62 ●	7.86
soybean	106.00	284.36 ●	80.96 ○	68.36	202.10 ●	174.90 ●
newSplice	41.60	81.58 ●	13.76 ○	26.08	70.88 ●	53.40 ●
vehicle	98.22	139.52 ●	15.40 ○	42.34	119.98 ●	103.02 ●
vote	9.66	66.40 ●	7.48	6.14	50.60 ●	6.54
vowel	184.28	184.16	41.20 ○	144.10	176.22 ●	169.82 ●
waveform	125.98	360.40 ●	21.12 ○	61.10	240.48 ●	172.08 ●
zoo	1.00	1.12	1.00	1.00	1.00	1.00
All	(●/ /○)	(32/6/0)	(0/11/27)		(30/8/0)	(22/16/0)

Table 4.15: Comparing the tree size of the two new pruning methods and minimal cost-complexity pruning (both with and without the 1SE rule).

4.7.3 Training time

Table 4.16 lists the training time of the three pruning methods both with and without the 1SE rule in seconds. The • symbol in the third column indicates that the training time of best-first-based post-pruning without the 1SE rule is significantly larger than that of minimal cost-complexity pruning without the 1SE rule, and the ◦ symbol in the third column indicates that the training time of the former scheme is significantly smaller. The ◦ symbol in the fourth column indicates that the training time of best-first-based pre-pruning without the new 1SE rule is significantly smaller than that of minimal cost-complexity pruning without the 1SE rule. The • symbol in the sixth column indicates that the training time of best-first-based post-pruning with the 1SE rule is significantly larger than that of minimal cost-complexity pruning with the 1SE rule, and the ◦ symbol in the sixth column indicates that the training time of the former scheme is significantly smaller. The • symbol in the seventh column indicates that the training time of best-first-based pre-pruning with the new 1SE rule is significantly larger than that of minimal cost-complexity pruning with the 1SE rule, and the ◦ symbol in the seventh column indicates that the training time of the former scheme is significantly smaller.

We can see that compared to the training time of minimal cost-complexity pruning, if the 1SE rule is not used, the training time of best-first-based post-pruning is significantly smaller on 16 datasets and significantly larger on 6 datasets. On the other 16 datasets there is no significant difference between the two schemes. The training time of best-first-based pre-pruning is significantly smaller on all datasets used in the experiments. If the 1SE rule is used, the training time of best-first-based post-pruning is significantly smaller on 15 datasets and significantly larger on 7 datasets. On the datasets, there is no significant difference between the two schemes. The training time of best-first-based pre-pruning is significantly larger on 4 datasets and significantly smaller on 28 datasets. On the datasets, there is no significant difference between the two schemes.

4.7.4 Discussion

Recall that the goal of decision tree learning is to find accurate and small models. If the 1SE rule is not used, minimal cost-complexity is better than best-first-based post-pruning because its accuracy is not significantly different from post-pruning

Dataset	Without 1SE			With 1SE		
	CART pruning	post-pruning	pre-pruning	CART pruning	post-pruning	pre-pruning
anneal	3.18	2.56 ○	2.08 ○	2.73	2.54 ○	2.52 ○
arrhythmia	12.95	12.54 ○	2.82 ○	12.71	12.29 ○	11.51 ○
audiology	1.92	2.00 ●	1.23 ○	1.86	2.00 ●	1.96 ●
autos	0.86	0.86	0.52 ○	0.84	0.85	0.84
balance-scale	0.33	0.31	0.06 ○	0.31	0.29	0.28
breast-cancer	0.54	0.51	0.16 ○	0.52	0.49 ○	0.26 ○
breast-w	0.32	0.24 ○	0.12 ○	0.29	0.22 ○	0.20 ○
horse-colic	2.20	2.14	0.44 ○	2.13	2.10	0.88 ○
credit-rating	1.38	1.33 ○	0.59 ○	1.33	1.27 ○	0.68 ○
german_credit	4.09	4.02 ○	1.25 ○	3.88	3.91	2.41 ○
pima_diabetes	0.66	0.57 ○	0.14 ○	0.63	0.55 ○	0.30 ○
ecoli	0.19	0.15 ○	0.09 ○	0.18	0.14 ○	0.13 ○
glass	0.24	0.20	0.11 ○	0.22	0.20	0.19 ○
heart-c	0.47	0.45	0.20 ○	0.46	0.43	0.32 ○
heart-h	0.39	0.37	0.18 ○	0.38	0.35	0.25 ○
heart-statlog	0.19	0.18	0.07 ○	0.18	0.17	0.12 ○
hepatitis	0.37	0.36	0.08 ○	0.35	0.34	0.18 ○
hypothyroid	9.69	8.31 ○	5.22 ○	9.59	8.19 ○	7.76 ○
ionosphere	0.73	0.69	0.42 ○	0.71	0.66	0.50 ○
iris	0.03	0.02 ○	0.02 ○	0.03	0.02 ○	0.02 ○
kr-vs-kp	9.82	8.83 ○	6.14 ○	9.64	8.82 ○	7.85 ○
labor	0.07	0.06	0.04 ○	0.07	0.06	0.06
letter	104.48	265.57 ●	21.78 ○	150.11	312.70 ●	351.75 ●
lymphography	0.32	0.32	0.18 ○	0.31	0.31	0.27 ○
mushroom	25.27	22.63 ○	22.55 ○	24.86	22.62 ○	22.42 ○
optdigits	23.66	25.30	9.87 ○	23.34	24.96 ●	24.53 ●
pendigits	17.70	18.78 ●	10.22 ○	17.54	18.69 ●	18.20 ●
primary-tumor	1.24	1.33 ●	0.24 ○	1.21	1.33 ●	1.12
segment	3.98	3.29 ○	2.48 ○	3.73	3.28 ○	3.26 ○
sick	12.14	10.45 ○	3.68 ○	11.89	10.31 ○	9.12 ○
sonar	0.88	0.88	0.44 ○	0.85	0.85	0.65 ○
soybean	8.22	8.43 ●	4.81 ○	8.11	8.40 ●	8.27
newSplice	65.91	62.93 ○	29.01 ○	65.49	62.89 ○	44.75 ○
vehicle	1.27	1.27	0.40 ○	1.26	1.21	1.13 ○
vote	0.71	0.68	0.23 ○	0.68	0.66	0.30 ○
vowel	4.46	4.34 ○	1.85 ○	4.38	4.31 ○	4.26
waveform	24.84	25.61 ●	6.28 ○	24.66	25.35 ●	20.25 ○
zoo	0.48	0.45 ○	0.33 ○	0.48	0.45	0.33 ○
All	(●/ /○)	(6/16/16)	(0/0/38)		(7/16/15)	(4/6/28)

Table 4.16: Comparing the training time of the two new pruning methods and minimal cost-complexity pruning (both with and without the 1SE rule).

on 37 datasets and its accuracy is significantly better on one dataset. Moreover, minimal cost-complexity pruning generates significantly smaller trees than post-pruning on most datasets. Compared to minimal cost-complexity pruning, best-first-based pre-pruning performs significantly worse in terms of accuracy on about half of the datasets, but it generates significantly smaller trees on most datasets. Thus, if the 1SE rule is not used, minimal cost-complexity pruning is preferable to the other two pruning methods in terms of accuracy. If the 1SE rule is used, compared to the two new pruning methods, minimal cost-complexity pruning generates significantly less accurate trees on only one dataset. However, the tree size generated by minimal cost-complexity is significantly smaller on most datasets. To sum up, minimal cost-complexity pruning is preferable to both best-first-based pre-pruning and post-pruning because it generates smaller trees with similar accuracy.

4.8 The effects of training set size on tree complexity for best-first-based pre-pruning and post-pruning

As mentioned in Chapter 2, Oates and Jensen (1997) have found that tree size generated by four pruning methods (error based pruning, reduced-error pruning, MDL pruning and minimal cost-complexity pruning) is strongly dependent on training set size. Normally, an increase of the training set size results in an increase of tree size. But Oates and Jensen also found that minimal cost-complexity pruning can appropriately limit tree growth much more frequently than the other three pruning methods. This section discusses the effects of training set size on tree complexity for best-first-based pre-pruning and post-pruning (both with and without the 1SE rule). We first describe the effect of training set size on an artificial dataset and then on 18 real-world datasets. For each dataset, 10 levels of training set size (i.e. 10%, 20% ... 100%) were selected. The minimal number of instances was set to two. The results of accuracy and tree size are still based on ten-times ten-fold cross validation.

The artificial dataset called *arti1000* in the experiments is generated by the data generator RDG1 in WEKA (Witten & Frank, 2005). It consists of 20 binary attributes and a binary class attribute. It has 1000 instances. Table 4.17 presents the accuracy and tree size of best-first-based pre-pruning on the *arti1000* dataset.

Set size	without 1SE rule		with 1SE rule	
	accuracy	tree size	accuracy	tree size
10%	74.48	4.02	74.91	3.80
20%	77.70	5.96	78.52	7.42
30%	77.70	5.72	78.59	7.80
40%	78.40	6.06	80.67	14.94
50%	77.91	6.04	81.19	17.94
60%	77.88	6.14	83.90	29.34
70%	79.26	7.72	85.14	38.38
80%	79.49	8.38	86.07	48.70
90%	**80.81**	9.26	88.81	60.96
100%	81.50	9.22	**89.50**	69.08

Table 4.17: The accuracy and tree size of best-first-based pre-pruning on different training set sizes (both with and without the new 1SE rule).

Set size	without 1SE rule		with 1SE rule	
	accuracy	tree size	accuracy	tree size
10%	83.97	20.98	83.49	19.08
20%	88.02	36.42	86.91	31.88
30%	89.34	51.26	88.12	43.58
40%	91.02	66.34	89.79	58.86
50%	**92.18**	77.98	91.30	69.32
60%	92.24	88.36	**91.59**	78.42
70%	92.45	96.98	92.00	88.00
80%	92.76	107.80	92.13	95.02
90%	92.42	113.22	91.91	97.36
100%	92.35	122.20	92.23	108.80

Table 4.18: The accuracy and tree size of best-first-based post-pruning for different training set sizes (both with and without the new 1SE rule).

Table 4.18 presents the accuracy and tree size of best-first-based post-pruning on the *arti1000* dataset. The percentage at which accuracy ceases to grow is marked in bold. The description of how this point is computed is presented below. Considering best-first-based pre-pruning, if the new 1SE rule is not used, the accuracy ceases to grow when 90% of the full training set is used. If the 1SE rule is used, the accuracy ceases to grow when the full training set is used. Considering post-pruning, when 50% (without the 1SE rule) or 60% (with the 1SE rule) of the full dataset is used, the accuracy ceases to grow while the tree size still increases significantly.

The rest of this section discusses the experiments on 18 real-world datasets. The experimental method used is similar to the one used by Oates and Jensen (1997). For each dataset, the training set size at which accuracy ceases to grow was found.

Dataset	without 1SE rule			with 1SE rule		
	cease %	$\triangle accuracy$	$\triangle size$	*cease* %	$\triangle accuracy$	$\triangle size$
anneal	100	0.0	0.0	90	0.77	11.32
arrhythmia	40	0.81	-9.13	80	0.73	16.13
audiology	100	0.0	0.0	100	0.0	0.0
autos	100	0.0	0.0	100	0.0	0.0
balance-scale	90	0.96	5.43	90	1.04	23.82
breast-cancer	10	-1.05	4.39	10	-1.9	37.78
breast-w	10	1.13	52.75	20	0.93	68.05
horse-colic	50	0.79	15.97	60	1.12	32.1
credit-rating	10	-0.26	26.09	10	-0.04	-2.99
german_credit	10	0.97	72.52	70	1.18	38.41
pima_diabetes	50	0.45	25.23	50	0.87	-1.11
ecoli	80	1.28	12.22	50	0.79	35.38
glass	100	0.0	0.0	80	-0.06	8.16
heart-c	30	0.1	5.65	40	1.2	30.94
heart-h	30	0.65	4.62	30	0.96	26.34
heart-statlog	50	0.0	-5.71	50	1.55	54.61
hepatitis	10	-0.51	35.66	10	-0.65	80.85
hypothyroid	10	0.64	53.46	10	0.81	74.49
	14	0.33	16.62	16	0.52	29.68

Table 4.19: The effect of random data reduction on tree complexity for best-first-based pre-pruning (both with and without the new 1SE rule).

This size was obtained by scanning the accuracy curves from left to right, scanning stopped when the mean of three adjacent accuracy estimates was no more than 1% from the accuracy of the tree built using the full training set. Using a threshold of 1% means that any reduction in tree size costs very little in terms of accuracy. Table 4.19 presents the results of best-first-based pre-pruning and Table 4.20 presents the results of best-first-based post-pruning. The two tables list the percentage of the training instances at which accuracy ceased to grow (% *cease*), the decrease in accuracy ($\triangle accuracy$) between the tree built from the full training set T_f and the tree built from reduced training set T_r, and the decrease in tree size ($\triangle size = 100 * (size(T_f) - size(T_r))/size(T_f)$). The last line in the tables shows the number of datasets for which accuracy ceases to grow before the full training set is used, the mean of $\triangle accuracy$ and the mean of $\triangle size$. The full results of accuracy and tree size on the 18 datasets are listed in Appendix B.

The two tables show that all four schemes (i.e. best-first-based post-pruning with and without the 1SE rule and best-first-based pre-pruning with and without the new 1SE rule) have the same overfitting problems, which is similar to the pruning methods

Dataset	without 1SE rule			with 1SE rule		
	cease %	△*accuracy*	△*size*	*cease* %	△*accuracy*	△*size*
anneal	60	0.84	19.84	60	1.01	20.76
arrhythmia	60	0.56	33.15	80	0.53	7.89
audiology	90	0.96	7.94	100	0.0	0.0
autos	90	0.73	6.84	100	0.0	0.0
balance-scale	60	0.76	43.21	80	0.7	26.29
breast-cancer	10	-1.41	82.01	10	-1.89	68.06
breast-w	10	0.82	83.92	10	0.9	79.14
horse-colic	10	-0.52	93.48	10	0.48	89.81
credit-rating	10	-0.91	82.49	10	-0.39	62.34
german_credit	10	0.55	85.95	10	0.95	72.99
pima_diabetes	10	-0.16	83.78	10	0.74	71.4
ecoli	40	0.32	39.23	40	0.06	31.16
glass	60	0.16	28.16	60	0.41	24.22
heart-c	20	0.39	71.47	20	0.12	58.96
heart-h	10	0.11	83.92	10	2.15	73.98
heart-statlog	40	1.11	57.76	40	0.6	52.32
hepatitis	10	-0.8	95.06	10	-0.31	92.53
hypothyroid	10	0.4	72.01	10	0.44	69.88
	18	0.22	59.46	16	0.36	50.1

Table 4.20: The effect of random data reduction on tree complexity for best-first-based post-pruning (both with and without the new 1SE rule).

used by Oates and Jensen (1997). For all four schemes , the accuracy ceases to grow on most datasets when 60% of full training data is used. On some datasets, the accuracy ceases to grow when only 10% of the full training set is used. Overfitting occurs because we can see that tree size continues to grow until the full training set is used. If we only use reduced training set (e.g. 50% on *ecoli* for best-first-based post-pruning without the 1SE rule) to build trees on each dataset, and compare the trees to those that are generated on the full training set, post-pruning without the 1SE rule can appropriately limit tree growth more frequently: 59.46, and loses the lowest accuracy: 0.22%.

4.9 Summary

This chapter evaluated best-first decision tree learning on real-world datasets. We first presented the datasets and methodology used in the experiments. Then we evaluated post-pruning for best-first decision trees using different splitting criteria: the Gini index and the information. The experimental results showed that the Gini index appeared to be a better choice than the information in terms of both accuracy and tree size. After this comparison, an experiment was run using the error rate and the root mean squared error respectively as the error estimate in the internal cross-validation to determine the number of expansions. We found that the error rate was overall better in terms of accuracy and that the root mean squared error is better in terms of tree size. Considering heuristic search and exhaustive search for finding binary splits on nominal attributes in post-pruning in multi-class problems, the experimental results indicated that there was no significant difference between them in terms of accuracy on all datasets investigated.

We then investigated the effects of the 1SE rule on the two new pruning methods. Considering best-first-based post-pruning, using the 1SE rule could significantly reduce tree size on most datasets without significantly losing accuracy on any dataset. Considering best-first-based pre-pruning, although using the new 1SE rule produced significantly larger trees on about half of the datasets, it also produced significantly more accurate trees on about half of the datasets. Thus, the 1SE rule is a good choice for both pruning methods.

Then best-first-based pre-pruning was compared to best-first-based post-pruning in terms of classification accuracy, tree size and training time. If the 1SE rule was not used, although best-first-based pre-pruning could significantly reduce tree size and training time on almost all datasets, the accuracy was significantly lower on about half of the datasets. If the 1SE rule was used, the two new pruning methods did not exhibit any significant difference in accuracy on any of the datasets. The tree size generated by pre-pruning was significantly smaller on several datasets. The training time of pre-pruning was significantly smaller on half of the datasets. Thus, best-first-based pre-pruning using the new 1SE rule is a better choice than best-first-based post-pruning using the 1SE rule.

Following that, the two best-first-based pruning methods were compared to minimal cost-complexity pruning in term of classification accuracy, tree size and training time. If the 1SE rule was not used, minimal cost-complexity pruning was better than the other two pruning methods because its accuracy was as good as that of post-pruning, which in turn has accuracy much better than pre-pruning. Moreover, the tree size of minimal cost-complexity pruning was significantly smaller on most datasets. If the 1SE rule was used, post-pruning, pre-pruning and minimal cost-complexity pruning had nearly equivalent performance in terms of accuracy. Furthermore, cost-complexity pruning can still generated significantly smaller trees on most datasets, but its training time is significantly larger on more datasets. Thus, considering accuracy and tree size, minimal cost-complexity pruning is the best pruning method considered.

Lastly, we investigated the effects of training set size on tree complexity for the two best-first-based pruning methods on an artificial dataset and 18 UCI datasets. Regarding the artificial dataset, the accuracy of pre-pruning increased up to 90% (without the new 1SE rule) or the full set (with the new 1SE rule) of the dataset, as well as the size of the trees. For post-pruning, accuracy ceased to grow when 50% (without the 1SE rule) or 60% (with the 1SE rule) of the full training data was used but tree size still increases significantly. Considering the 18 UCI datasets, we found that the tree size generated by the two pruning methods was also influenced by training set size regardless of whether the 1SE rule was used. For most datasets, if 60% of the full training set was used, the accuracy of the two pruning methods (both with and without using the 1SE rule) ceased to grow while the tree size still increased significantly. This indicates that both methods suffer from overfitting.

Chapter 5
Conclusions

Decision tree learning is a very good choice for classification learning because it produces accurate and small models that offer insight into how predictions are made. Best-first decision tree learning expands nodes in best-first order. This is different from standard top-down decision tree learning which expands nodes in a fixed order. This thesis investigated the applicability of best-first learning to the induction of binary decision trees. Two new pruning methods for best-first decision trees based on cross-validation were presented and compared to another cross-validation-based pruning method: minimal cost-complexity pruning (Breiman et al., 1984). This chapter summarises the material that was presented, draws some conclusions and describes some possibilities for future work.

5.1 Summary and conclusions

This thesis first presented the background for best-first decision trees and cross-validation-based pruning methods (Chapter 2). We discussed minimal cost-complexity pruning and the effect of the 1SE rule when choosing the final pruned subtree from a sequence of pruned subtrees. Three other related pruning methods that are based on cross-validation or a hold-out set, namely critical value pruning (Mingers, 1987), reduced-error pruning (Quinlan, 1987) and the wrapper approach (Kohavi, 1995a), were briefly discussed. Then, pre-pruning was compared to post-pruning for standard decision trees to see their respective advantages and disadvantages. Friedman's work (2000) that applies best-first decision trees to boosting was discussed and this has been the only work that investigated best-first decision trees so far.

The following chapter discussed best-first decision tree learning in detail (Chapter 3). In order to measure node impurity, two splitting criteria were used in best-first decision trees: the Gini index and the information. An example was given to calculate node impurity on the *weather* dataset. Splitting rules for best-first decision tree learning were discussed and an example was also given. The idea is that, for numeric attributes the method used is the same as the one used in the CART system (Breiman et al., 1984) and C4.5 (Quinlan, 1993). For nominal attributes, we discussed both exhaustive search and heuristic search. Then the best-first decision tree learning algorithm, the best-first-based pre-pruning and post-pruning algorithms were presented in detail. We mentioned that both the error rate and the root mean squared error could be used to determine the number of expansions in the internal cross-validation. We also discussed the time complexity of both the basic best-first decision tree algorithm and the pruning algorithms in big O notation.

Chapter 4 explored the performance of best-first decision tree learning on standard benchmark datasets from the UCI repository. We first considered an experiment comparing best-first decision tree learning using the Gini index and the information as the splitting criterion respectively in best-first-based post-pruning and found that the Gini index was overall somehow better in terms of both accuracy and tree size. An experiment was then run using the error rate and the root mean squared error as the error estimate respectively and the error rate was found to be the better choice in terms of accuracy. Following that, we compared the performance of exhaustive search and heuristic search for finding binary splits on nominal attributes in multi-class problems, and found that there was no significant difference between these two search methods in terms of accuracy on all datasets used. Moreover, we also found that the 1SE rule was a good choice for best-first-based post-pruning because it could significantly reduce tree size on most datasets without significantly degrading accuracy on any dataset. The new 1SE rule for best-first-based pre-pruning was also a good choice because it could prevent stopping too early on about half of the datasets.

Following that, we compared best-first-based post-pruning to best-first-based pre-pruning and found that if the 1SE rule is not used, although pre-pruning can significantly reduce tree size and training time on almost all datasets, the accuracy is significantly lower on about half of the datasets. That is to say, pre-pruning (without the new 1SE rule) stops too early on about half of the datasets. If the 1SE rule was used, the two pruning methods performed similarly on all datasets in terms of accuracy. However, the trees generated by pre-pruning were significantly smaller on several datasets. The training time of pre-pruning was also significantly smaller on half of the datasets.

Then best-first-based pre-pruning and post-pruning were compared to minimal cost-complexity pruning. We found that if the 1SE rule was used, minimal cost-complexity pruning had nearly equivalent accuracy to the two new pruning methods and generated significantly smaller trees on most datasets. If the 1SE rule was not used, minimal cost-complexity pruning and best-first-based post-pruning performed similarly in terms of accuracy, and much better than best-first-based pre-pruning. We also found that the tree size generated by the two new pruning methods was strongly dependent on training data size. For most datasets, even if the accuracy did not increase further, the tree size still significantly increased when training set size increased. This indicates that these methods are susceptible to overfitting.

5.2 Future work

This thesis only evaluated best-first learning for binary decision trees. For a binary tree, it is time-consuming to find the best binary split if the number of values of a nominal attribute in multi-class problems is large, even if heuristic search is used. The original *splice* dataset is an example of where this is problematic. Thus, it may be useful to consider constructing best-first decision trees with multi-way splits, where we extend a branch for each value of a nominal attribute, in the same way as in ID3 (Quinlan, 1986). Then we can evaluate the two new pruning methods for best-first decision trees in the same way as what was done in this thesis.

Appendix A
Possible binary splits on the attribute *temperature*

The possible binary splits for the attribute *temperature* are shown in Figure A.1. Their corresponding information gains are (the largest gain value is marked with an asterisk):

infoGain(temperature<64.5) = 0.048 *bits*

infoGain(temperature<66.5) = 0.010 *bits*

infoGain(temperature<68.5) = 0.0005 *bits*

infoGain(temperature<69.5) = 0.015 *bits*

infoGain(temperature<70.5) = 0.045 *bits*

infoGain(temperature<71.5) = 0.001 *bits*

infoGain(temperature<73.5) = 0.001 *bits*

infoGain(temperature<77.5) = 0.025 *bits*

infoGain(temperature<80.5) = 0.0005 *bits*

infoGain(temperature<82) = 0.010 *bits*

infoGain(temperature<84) = 0.113 *bits* *

Their corresponding Gini gains are (the largest gain value is marked with an asterisk):

giniGain(temperature<64.5) = 0.020

giniGain(temperature<66.5) = 0.007

giniGain(temperature<68.5) = 0.0003

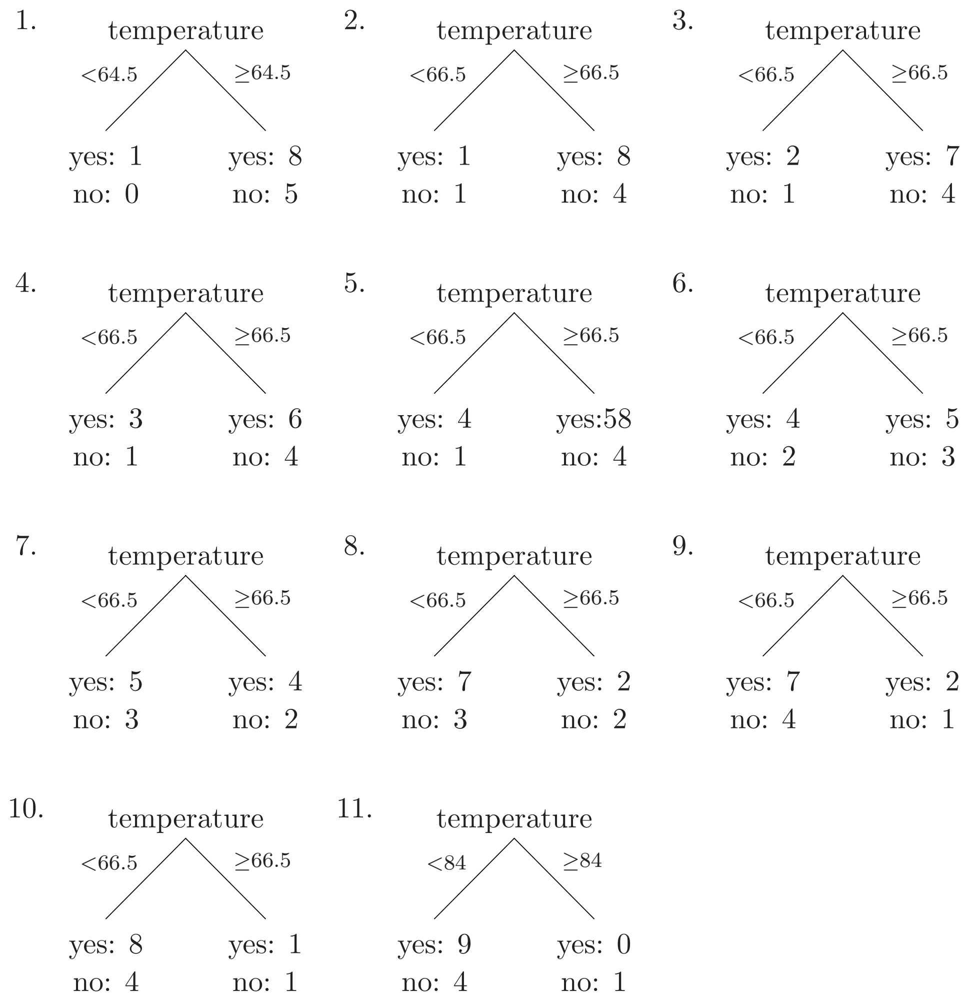
1. temperature <64.5 ≥64.5 yes: 1 no: 0 yes: 8 no: 5
2. temperature <66.5 ≥66.5 yes: 1 no: 1 yes: 8 no: 4
3. temperature <66.5 ≥66.5 yes: 2 no: 1 yes: 7 no: 4
4. temperature <66.5 ≥66.5 yes: 3 no: 1 yes: 6 no: 4
5. temperature <66.5 ≥66.5 yes: 4 no: 1 yes:58 no: 4
6. temperature <66.5 ≥66.5 yes: 4 no: 2 yes: 5 no: 3
7. temperature <66.5 ≥66.5 yes: 5 no: 3 yes: 4 no: 2
8. temperature <66.5 ≥66.5 yes: 7 no: 3 yes: 2 no: 2
9. temperature <66.5 ≥66.5 yes: 7 no: 4 yes: 2 no: 1
10. temperature <66.5 ≥66.5 yes: 8 no: 4 yes: 1 no: 1
11. temperature <84 ≥84 yes: 9 no: 4 yes: 0 no: 1

Figure A.1: Possible binary splits on the attribute *temperature*.

.

giniGain(temperature<69.5) = 0.009

giniGain(temperature<70.5) = 0.027

giniGain(temperature<71.5) = 0.0008

giniGain(temperature<73.5) = 0.0008

giniGain(temperature<77.5) = 0.016

giniGain(temperature<80.5) = 0.003

giniGain(temperature<82) = 0.007

giniGain(temperature<84) = 0.064 *

Appendix B Accuracy and tree size of pre-pruning and post-pruning for different training set sizes

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	90.45	91.46	92.97	92.75	93.59	93.50	94.04	94.49	94.66	96.19
arrhythmia	56.48	57.32	58.50	58.41	59.12	58.80	59.07	59.11	59.20	59.22
audiology	53.28	56.16	57.33	58.73	58.93	60.55	60.14	63.86	62.97	65.05
autos	41.96	43.10	47.12	51.98	53.75	56.08	57.39	56.64	57.82	60.47
balance-scale	60.46	62.62	63.23	63.69	64.99	64.89	65.89	64.99	65.74	66.70
breast-cancer	70.51	69.65	69.95	70.67	70.16	70.08	70.12	70.06	70.12	69.46
breast-w	93.02	93.41	93.42	93.83	93.61	93.82	93.89	93.86	93.95	94.15
horse-colic	80.37	83.31	83.82	84.50	84.66	85.15	85.51	85.53	85.56	85.45
credit-rating	85.39	85.36	85.41	85.36	85.32	85.39	85.51	85.25	85.28	85.13
german_credit	70.26	70.36	70.79	70.12	70.91	70.71	70.66	70.77	70.93	71.23
pima_diabetes	70.86	71.68	72.02	73.11	73.80	73.72	74.21	74.06	74.35	74.25
ecoli	71.27	76.72	79.10	79.80	80.12	80.39	80.27	80.24	81.28	81.52
glass	45.81	51.78	54.26	57.23	60.50	60.35	62.93	62.26	63.42	66.34
heart-c	71.30	72.76	74.11	73.89	74.28	74.32	74.55	74.58	74.34	74.21
heart-h	76.05	78.73	79.10	79.22	79.48	79.58	79.31	79.34	79.93	79.75
heart-statlog	68.81	70.15	71.59	71.74	73.19	72.56	73.63	73.22	73.41	73.19
hepatitis	78.91	78.53	78.99	79.24	78.35	78.85	78.46	78.46	78.53	78.40
hypothyroid	98.80	99.13	99.26	99.28	99.34	99.33	99.31	99.40	99.42	99.44

Table B.1: The accuracy of best-first-based pre-pruning without the new 1SE rule for different training set sizes.

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	7.12	8.70	11.06	11.46	12.46	12.24	13.10	13.14	13.10	12.98
arrhythmia	2.50	4.04	5.38	5.50	5.62	5.62	5.50	6.00	5.84	5.04
audiology	6.46	8.34	9.36	11.34	11.66	14.34	14.02	18.92	18.82	22.16
autos	3.46	4.42	6.96	10.44	12.04	14.62	15.22	15.90	16.88	18.88
balance-scale	5.64	8.92	8.42	9.08	11.34	12.28	13.40	11.00	12.90	13.64
breast-cancer	2.18	2.10	2.10	2.22	2.42	2.14	2.56	2.48	2.54	2.28
breast-w	5.16	6.46	6.96	8.14	7.86	8.50	7.86	8.44	9.02	10.92
horse-colic	4.36	4.92	5.34	6.12	6.00	5.82	6.50	6.42	6.98	7.14
credit-rating	4.08	4.02	4.06	4.64	4.62	4.86	5.34	5.40	5.26	5.52
german_credit	1.94	3.10	3.84	3.94	4.14	4.74	4.68	5.32	6.30	7.06
pima_diabetes	3.36	3.76	4.04	4.92	4.98	5.24	5.50	5.76	6.40	6.66
ecoli	5.96	9.12	9.90	10.74	12.22	11.84	12.48	12.36	13.44	14.08
glass	3.42	5.52	7.02	8.10	9.32	10.42	10.80	11.06	11.92	13.56
heart-c	4.42	4.56	5.34	5.46	5.38	6.10	5.88	5.56	5.12	5.66
heart-h	3.84	4.06	4.54	4.70	5.12	4.88	5.18	5.40	5.42	4.76
heart-statlog	3.56	4.72	4.68	5.04	5.92	5.54	5.56	5.42	5.68	5.60
hepatitis	1.84	2.72	3.08	2.92	2.72	3.34	3.08	3.10	2.34	2.86
hypothyroid	10.64	14.88	16.86	18.50	19.36	19.00	20.32	22.30	23.22	22.86

Table B.2: The tree size of best-first-based pre-pruning without the new 1SE rule for different training set sizes.

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	91.37	92.99	94.86	95.37	95.71	95.99	96.52	96.74	97.49	98.26
arrhythmia	56.39	57.15	59.05	60.33	62.01	62.79	64.20	65.38	65.86	66.11
audiology	51.03	56.21	60.38	62.83	65.62	68.92	70.10	71.46	72.44	73.90
autos	41.24	45.72	52.93	59.50	62.72	66.57	67.59	71.62	71.82	73.78
balance-scale	61.22	64.59	66.87	70.00	71.49	74.10	75.17	76.50	76.78	77.82
breast-cancer	70.41	69.60	69.91	70.50	70.48	69.87	69.85	69.42	68.55	68.51
breast-w	93.11	93.29	93.73	94.08	94.14	93.84	94.09	94.06	94.17	94.22
horse-colic	80.76	82.71	83.93	84.01	84.72	84.44	85.16	85.40	85.48	85.56
credit-rating	85.33	85.57	85.42	85.46	85.46	85.42	85.48	85.57	85.41	85.29
german_credit	70.01	70.52	70.78	70.28	70.91	71.24	71.27	71.93	71.84	72.45
pima_diabetes	70.91	71.78	72.36	73.14	73.36	73.53	74.08	74.01	74.24	74.23
ecoli	71.56	77.45	79.76	80.96	81.10	80.71	81.08	81.52	81.88	81.89
glass	45.05	52.55	57.24	61.20	65.16	66.24	67.61	68.90	68.75	68.84
heart-c	71.10	73.46	75.00	74.39	75.07	75.05	75.74	75.59	75.60	75.59
heart-h	76.22	78.62	78.45	79.32	79.24	79.41	79.21	78.52	79.06	79.41
heart-statlog	68.93	70.19	73.44	73.41	73.04	74.44	74.11	74.78	74.67	74.59
hepatitis	78.85	78.73	78.93	78.72	78.86	78.92	78.46	78.10	77.92	78.20
hypothyroid	98.69	99.22	99.32	99.41	99.41	99.45	99.46	99.49	99.48	99.50

Table B.3: The accuracy of best-first-based pre-pruning with the new 1SE rule for different training set sizes.

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	7.58	9.88	13.44	14.78	16.44	18.02	19.68	20.70	21.78	24.56
arrhythmia	2.38	4.48	10.10	12.22	20.60	24.36	31.24	40.04	41.44	47.74
audiology	5.64	8.82	13.24	17.64	22.58	29.06	32.76	35.44	37.66	42.74
autos	3.22	6.28	11.50	16.92	21.92	27.30	31.54	39.02	41.76	46.10
balance-scale	7.32	14.10	20.70	33.68	44.54	58.06	70.86	86.74	95.82	125.78
breast-cancer	2.24	1.72	2.34	2.24	2.48	2.12	2.70	4.12	2.98	3.60
breast-w	5.30	6.94	9.00	10.16	11.30	12.02	14.70	14.38	17.30	21.72
horse-colic	4.42	4.94	5.18	6.72	6.12	6.60	6.08	13.94	13.24	9.72
credit-rating	4.14	3.66	3.92	3.72	3.52	3.44	4.44	4.06	4.72	4.02
german_credit	2.14	3.24	3.92	4.18	5.32	11.60	10.36	15.72	21.28	16.82
pima_diabetes	3.54	4.42	5.18	6.72	7.28	7.34	9.42	7.70	9.54	7.20
ecoli	6.00	9.34	11.70	13.10	14.98	16.00	16.96	19.40	20.46	23.18
glass	3.16	6.14	8.70	11.68	15.90	19.16	22.64	25.66	26.80	27.94
heart-c	4.38	5.64	6.84	8.26	8.10	11.14	10.84	10.94	11.62	11.96
heart-h	3.88	4.58	6.04	6.22	6.44	6.94	6.34	8.66	6.92	8.20
heart-statlog	3.46	5.44	6.84	6.74	7.28	10.78	9.94	11.88	11.82	16.04
hepatitis	1.80	2.60	3.48	3.68	3.72	5.32	4.74	5.74	8.50	9.40
hypothyroid	9.94	15.56	19.18	22.90	25.80	27.78	31.98	36.46	37.84	38.96

Table B.4: The tree size of best-first-based pre-pruning with the new 1SE rule for different training set sizes.

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	95.36	96.35	96.78	97.04	97.14	97.48	97.66	97.88	98.06	98.32
arrhythmia	57.43	60.87	62.92	64.87	66.13	66.72	67.14	67.74	67.72	67.28
audiology	55.80	60.75	63.78	66.21	68.44	71.22	72.48	73.94	74.43	75.39
autos	47.00	57.17	62.26	65.89	68.03	69.73	71.79	73.68	74.64	75.37
balance-scale	66.85	70.65	73.45	75.72	77.26	78.11	78.60	78.80	78.84	78.87
breast-cancer	70.65	70.44	70.65	70.90	70.02	70.78	69.56	69.52	69.23	69.24
breast-w	93.52	94.04	94.28	94.81	94.74	94.67	94.65	94.67	94.59	94.34
horse-colic	84.56	85.16	84.97	84.37	84.47	84.67	84.83	84.67	84.26	84.04
credit-rating	85.00	85.41	85.16	85.45	85.09	84.96	84.93	84.59	84.68	84.09
german_credit	71.85	72.59	72.98	72.68	72.95	72.81	73.15	72.67	72.34	72.40
pima_diabetes	73.36	74.03	74.47	74.22	74.03	74.01	74.09	73.37	73.49	73.20
ecoli	77.60	80.84	81.40	82.48	82.03	82.60	82.63	82.75	83.07	82.80
glass	54.66	64.14	66.81	67.73	70.03	70.23	70.49	70.63	70.91	70.39
heart-c	73.82	75.81	76.37	77.82	77.45	76.46	77.13	76.70	76.27	76.20
heart-h	77.55	79.74	78.89	79.58	79.10	78.96	78.69	78.04	77.76	77.66
heart-statlog	71.89	73.96	75.59	75.89	76.63	76.93	77.04	77.48	77.41	77.00
hepatitis	78.67	78.47	77.65	78.16	77.64	77.35	78.50	77.88	78.50	77.87
hypothyroid	99.10	99.30	99.41	99.45	99.50	99.50	99.53	99.51	99.50	99.50

Table B.5: The accuracy of best-first-based post-pruning without the 1SE rule on for different training set sizes.

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	9.70	12.46	15.44	17.38	18.86	20.52	22.16	23.12	24.00	25.60
arrhythmia	5.80	15.28	24.90	31.68	38.88	44.68	52.30	58.58	61.44	66.84
audiology	7.68	13.62	19.00	23.52	28.66	34.02	37.34	40.92	43.38	47.12
autos	6.08	13.54	18.70	22.98	28.48	33.74	38.60	43.44	47.66	51.16
audiology	7.62	13.56	18.88	23.52	28.18	34.38	38.26	41.06	43.02	46.98
autos	6.00	13.74	19.24	24.72	29.06	34.10	37.74	42.44	45.58	48.22
balance-scale	17.12	32.34	47.38	63.38	78.34	91.40	106.86	123.48	137.88	160.94
breast-cancer	5.12	9.46	13.38	14.36	18.74	22.06	25.78	26.38	27.04	28.46
breast-w	6.08	9.66	13.66	18.34	21.26	25.12	27.30	33.64	36.40	37.80
horse-colic	9.96	20.98	32.68	47.30	63.90	89.18	90.50	103.26	137.72	152.80
credit-rating	7.64	14.02	17.42	22.80	24.12	31.50	36.86	37.56	42.02	43.64
german_credit	13.34	22.70	35.44	41.56	56.46	60.28	78.22	81.58	99.32	94.96
pima_diabetes	11.34	21.44	29.24	35.14	41.44	46.18	44.60	48.00	52.16	69.90
ecoli	8.56	13.56	16.08	19.92	23.22	25.44	28.84	29.76	31.54	32.78
glass	6.12	11.96	16.22	20.02	25.66	30.16	32.92	37.28	40.38	41.98
heart-c	5.88	10.54	13.98	17.68	20.76	26.98	30.10	29.24	33.62	36.94
heart-h	5.02	9.86	14.34	18.74	21.00	20.72	24.80	27.06	28.60	31.22
heart-statlog	5.00	9.72	12.86	16.00	17.58	23.00	25.00	29.04	32.86	37.88
hepatitis	2.48	5.60	9.66	13.16	18.54	22.84	30.86	33.74	40.24	50.24
hypothyroid	13.02	17.80	22.72	26.28	30.30	33.98	37.64	41.84	44.20	46.52

Table B.6: The tree size of best-first-based post-pruning without the 1SE rule for different training set sizes.

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	95.09	96.29	96.70	97.00	97.12	97.31	97.62	97.71	97.93	98.32
arrhythmia	57.38	60.29	61.90	63.30	64.20	64.75	65.59	66.15	66.55	66.68
audiology	54.98	59.30	62.46	65.41	66.90	69.98	71.20	71.72	73.14	74.07
autos	46.96	54.72	60.29	63.41	66.32	68.08	69.69	72.18	72.80	74.16
balance-scale	66.59	69.83	72.36	74.69	76.00	76.67	76.87	77.77	78.05	78.47
breast-cancer	70.47	70.58	70.40	70.67	69.91	70.56	69.21	69.00	67.96	68.58
breast-w	93.38	93.79	94.05	94.78	94.69	94.45	94.36	94.65	94.44	94.28
horse-colic	84.24	84.56	84.72	84.07	84.77	83.93	84.94	84.86	84.80	84.72
credit-rating	85.00	85.54	85.26	85.39	85.25	85.06	85.20	84.99	85.04	84.61
german_credit	71.59	72.08	72.39	71.86	72.40	72.30	72.74	72.96	72.56	72.54
pima_diabetes	72.86	73.97	73.75	73.75	73.75	73.50	73.88	74.01	74.08	73.60
ecoli	76.92	80.21	81.02	82.15	81.82	81.91	81.82	82.00	82.41	82.21
glass	53.73	62.58	65.31	66.75	68.39	69.04	69.23	69.32	69.83	69.45
heart-c	73.75	75.84	76.20	77.04	77.12	76.07	76.60	76.75	75.74	75.96
heart-h	77.34	79.71	78.93	79.82	78.76	79.34	78.49	78.42	78.24	79.49
heart-statlog	71.44	72.96	74.89	75.44	75.07	75.78	76.30	76.19	76.41	76.04
hepatitis	78.60	78.61	78.23	77.97	78.48	78.40	79.24	78.01	79.04	78.29
hypothyroid	99.06	99.27	99.36	99.41	99.47	99.46	99.49	99.50	99.49	99.50

Table B.7: The accuracy of best-first-based post-pruning with the 1SE rule for different training set sizes.

Dataset	Training set size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
anneal	9.56	12.16	15.14	16.98	18.36	19.92	21.82	22.62	23.30	25.14
arrhythmia	5.16	13.32	20.92	26.38	30.64	34.24	42.28	46.00	48.40	49.94
audiology	7.32	11.98	16.44	21.88	24.96	31.18	34.22	36.62	38.84	43.40
autos	5.98	12.02	17.32	20.96	25.58	30.86	34.62	40.40	43.64	47.50
balance-scale	16.06	29.72	41.74	56.84	67.60	78.60	88.32	103.02	115.64	139.76
breast-cancer	4.74	7.86	9.60	9.16	13.38	14.02	15.92	14.14	14.12	14.84
breast-w	5.90	8.64	12.42	16.78	18.56	19.92	20.92	26.96	27.88	28.28
horse-colic	9.48	17.72	26.88	38.24	54.84	67.54	63.12	66.26	87.58	93.06
credit-rating	7.20	12.32	13.42	18.94	17.28	24.46	26.08	24.56	21.56	19.12
german_credit	11.82	19.08	24.98	25.46	36.86	34.38	49.78	53.70	56.96	43.76
pima_diabetes	10.12	18.54	21.52	26.46	29.36	31.92	24.08	19.44	27.98	35.38
ecoli	8.24	12.34	14.66	17.50	19.06	20.54	22.80	23.18	25.48	25.42
glass	5.80	11.02	14.34	17.86	21.60	24.78	28.42	29.64	33.12	32.70
heart-c	5.74	9.90	11.86	15.08	16.76	22.18	23.64	20.42	23.22	24.12
heart-h	4.84	8.68	11.94	15.80	17.28	15.94	18.00	19.08	15.94	18.60
heart-statlog	4.66	8.48	10.54	13.54	14.14	17.86	21.44	21.40	23.14	28.40
hepatitis	2.46	5.08	7.80	10.64	14.12	16.96	21.26	25.78	27.56	32.92
hypothyroid	12.32	16.70	20.28	24.26	27.82	30.84	33.88	37.24	39.74	40.90

Table B.8: The tree size of best-first-based post-pruning with the 1SE rule for different training set sizes.

Bibliography

Blake, C., Keogh, E. & Merz, C. (1998). UCI repository of machine learning databases [http://www.ics.uci.edu/m̃learn/mlrepository.html]. Technical report, Department of Information and Computer Science, University of California, Irvine, CA.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth.

Coppersmith, D., Hong, S. J. & Kosking, J. (1999). Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, *3(2)*, 197–217.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10(7)*, 1895–1923.

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

Frank, E. (2000). *Pruning decision trees and lists*. PhD thesis, The University of Waikato.

Friedman, J., Hastie, J. & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, *28(2)*, 337–407.

Kass, G. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, *29(2)*.

Kearns, M. (1996). A bound on the error of cross-validation using approximation and estimation rates, with consequences for the training-test split. In Touretzky, MI, D. S., Mozer, M. G. & Hasselmo, M. E. (Eds.), *Advances in Neural Information Processing System* (pp. 183–189). MIT Press, Cambridge, MA.

Kohavi, R. (1995a). *Wrappers for performance enhancement and oblivious decision graphs*. PhD thesis, Stanford University.

Kohavi, R. (1995b). A study of cross-validation and bootstrap for accuracy estimate model selection. In *Proc of the 14th Int Joint Conf on Artificial Intelligence* (pp. 1137–1143). Montreal, Canada: Morgan Kaufmann, San Francisco, CA.

Kohavi, R. & John, G. (1995). Automatic parameter selection by minimising estimated error. In Prieditis, A. & Russell, S. (Eds.), *Machine Learning: Proceedings of the Twelfth International conference*. Morgan Kaufmann.

Mingers, J. (1987). Expert systems - rule induction with statistical data. *The Journal of the Operational Research Society, 38(1)*, 39–47.

Mitchell, T. (1997). Decision tree learning. *The McGraw-Hill Companies, Inc* (pp. 52–78).

Nadeau, C. & Bengio, Y. (2003). Inference for the generalization error. *Machine Learning, 52(3)*, 239–281.

Oates, T. & Jensen, D. (1997). The effects of training set size on decision tree complexity. In *Proc of the 14th Int Conf on Machine Learning* (pp. 254–262). Morgan Kaufmann.

Quinlan, J. & Rivest, R. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation, 80(3)*, 227–248.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1(1)*, 81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies, 27(3)*, 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco, CA: Morgan Kaufmann.

Witten, I. H. & Frank, E. (2005). *Data Mining: Practical machine Learning tools and techniques with Java implementations* (2 Ed.). San Francisco, CA: Morgan Kaufmann.