CHAPTER 4

Pruning Decision Trees

The recursive partitioning method of constructing decision trees described in Chapter 2 will continue to subdivide the set of training cases until each subset in the partition contains cases of a single class, or until no test offers any improvement. The result is often a very complex tree that "overfits the data" by inferring more structure than is justified by the training cases.

This effect is readily seen in the extreme example of random data in which the class of each case is quite unrelated to its attribute values. I constructed an artificial dataset of this kind with ten attributes, each of which took the value 0 or 1 with equal probability. The class was also binary, yes with probability 0.25 and no with probability 0.75. One thousand randomly generated cases were split into a training set of 500 and a test set of 500. From this data, C4.5's initial tree-building routine produces a nonsensical tree of 119 nodes that has an error rate of more than 35% on the test cases.

This small example illustrates the twin perils that can come from too gullible acceptance of the initial tree: It is often extremely complex, and can actually have a higher error rate than a simpler tree. For the random data above, a tree consisting of just the leaf no would have an expected error rate of 25% on unseen cases, yet the elaborate tree is noticeably less accurate. While the complexity comes as no surprise, the increased error attributable to overfitting is not intuitively obvious. To explain this, suppose we have a two-class task in which a case's class is inherently indeterminate, with proportion $p \geq 0.5$ of the cases belonging to the majority class (here no). If a classifier assigns all such cases to this majority class, its expected error rate is clearly $1 - p$. If, on the other hand, the classifier assigns a case to the majority class with probability p and to the other class with probability $1 - p$, its expected error rate is the sum of

- the probability that a case belonging to the majority class is assigned to the other class, $p \times (1 - p)$, and
- the probability that a case belonging to the other class is assigned to the majority class, $(1 - p) \times p$

which comes to $2 \times p \times (1 - p)$. Since p is at least 0.5, this is generally greater than $1 - p$, so the second classifier will have a higher error rate. Now, the complex decision tree bears a close resemblance to this second type of classifier. The tests are unrelated to class so, like a symbolic pachinko machine, the tree sends each case randomly to one of the leaves. We would expect the leaves themselves to be distributed in proportion to the class frequencies in the training set. Consequently, the tree's expected error rate for the random data above is $2 \times 0.25 \times 0.75$ or 37.5%, quite close to the observed value.

It may seem that this discussion of random classifiers and indeterminate classes is a far cry from real-world induction tasks. This is not so: Tasks are often at least partly indeterminate because the attributes do not capture all information relevant to classification. Again, when the training set has been split many times so that tests are selected from examination of a small subset of cases, several tests may appear equally promising and choosing a particular one of them has elements of randomness. In my experience, almost all decision trees can benefit from simplification.

A decision tree is not usually simplified by deleting the whole tree in favor of a leaf, as was done in the random data example. Instead, the idea is to remove parts of the tree that do not contribute to classification accuracy on unseen cases, producing something less complex and thus more comprehensible.

4.1 When to simplify?

There are basically two ways in which the recursive partitioning method can be modified to produce simpler trees: deciding not to divide a set of training cases any further, or removing retrospectively some of the structure built up by recursive partitioning.

The former approach, sometimes called *stopping* or *prepruning*, has the attraction that time is not wasted assembling structure that is not used in the final simplified tree. The typical approach is to look at the best way of splitting a subset and to assess the split from the point of view of statistical significance, information gain, error reduction, or whatever. If this assessment falls below some threshold, the division is rejected and the tree for the subset is just the most appropriate leaf. However, as Breiman *et al.* point out, such stopping rules are not easy to get right—too high a threshold can terminate division before the benefits

of subsequent splits become evident, while too low a value results in little simplification.

At one stage, I used a stopping criterion based on the χ^2 test of statistical significance [Quinlan, 1986a]. The results were quite satisfactory in some domains but were uneven, so I abandoned this approach; C4.5, like CART, now follows the second path. The divide-and-conquer process is given free rein and the overfitted tree that it produces is then pruned. The additional computation invested in building parts of the tree that are subsequently discarded can be substantial, but this cost is offset against benefits due to a more thorough exploration of possible partitions. Growing and pruning trees is slower but more reliable.

Pruning a decision tree will almost invariably cause it to misclassify more of the training cases. Consequently, the leaves of the pruned tree will not necessarily contain training cases from a single class, a phenomenon we have encountered in the previous chapter. Instead of a class associated with a leaf, there will again be a *class distribution* specifying, for each class, the probability that a training case at the leaf belongs to that class. This modification can slightly alter the determination of the most probable class for an unseen case. The fact that there is a distribution rather than a single class becomes important when we discuss the certainty of prediction (Chapter 8).

4.2 Error-based pruning

Decision trees are usually simplified by discarding one or more subtrees and replacing them with leaves; as when building trees, the class associated with a leaf is found by examining the training cases covered by the leaf and choosing the most frequent class. In addition, C4.5 allows replacement of a subtree by one of its branches. Both operations are illustrated in Figure 4–1 that shows a decision tree derived from congressional voting data[5] before and after pruning. (For the unpruned tree, recall that the (N) or (N/E) appearing after a leaf indicates that the leaf covers N training cases, E erroneously; similar numbers for the pruned tree are explained below.) The subtree

5. This dataset, collected by Jeff Schlimmer, records the votes of all United States congressmen on 16 key issues selected by the *Congressional Quarterly Almanac* for the second session of 1984. It is available in the University of California, Irvine data library.

Original decision tree:

```
physician fee freeze = n:
|   adoption of the budget resolution = y: democrat (151)
|   adoption of the budget resolution = u: democrat (1)
|   adoption of the budget resolution = n:
|   |   education spending = n: democrat (6)
|   |   education spending = y: democrat (9)
|   |   education spending = u: republican (1)
physician fee freeze = y:
|   synfuels corporation cutback = n: republican (97/3)
|   synfuels corporation cutback = u: republican (4)
|   synfuels corporation cutback = y:
|   |   duty free exports = y: democrat (2)
|   |   duty free exports = u: republican (1)
|   |   duty free exports = n:
|   |   |   education spending = n: democrat (5/2)
|   |   |   education spending = y: republican (13/2)
|   |   |   education spending = u: democrat (1)
physician fee freeze = u:
|   water project cost sharing = n: democrat (0)
|   water project cost sharing = y: democrat (4)
|   water project cost sharing = u:
|   |   mx missile = n: republican (0)
|   |   mx missile = y: democrat (3/1)
|   |   mx missile = u: republican (2)
```

pruning:

```
/sician fee freeze = n: democrat (168/2.6)
physician fee freeze = y: republican (123/13.9)
physician fee freeze = u:
|   mx missile = n: democrat (3/1.1)
|   mx missile = y: democrat (4/2.2)
|   mx missile = u: republican (2/1)
```

Figure 4–1. Decision tree before and after pruning


```
adoption of the budget resolution = y: democrat
adoption of the budget resolution = u: democrat
adoption of the budget resolution = n:
|   education spending = n: democrat
|   education spending = y: democrat
|   education spending = u: republican
```

has been replaced by the leaf democrat, the subtree

```
synfuels corporation cutback = n: republican
synfuels corporation cutback = u: republican
synfuels corporation cutback = y:
|   duty free exports = y: democrat
|   duty free exports = u: republican
|   duty free exports = n:
|   |   education spending = n: democrat
|   |   education spending = y: republican
|   |   education spending = u: democrat
```

has become the leaf republican, and the subtree

```
water project cost sharing = n: democrat
water project cost sharing = y: democrat
water project cost sharing = u:
|   mx missile = n: republican
|   mx missile = y: democrat
|   mx missile = u: republican
```

has been replaced by the subtree at its third branch.

Suppose that it was possible to predict the error rate of a tree and of its subtrees (including leaves). This would immediately suggest the following simple pruning rationale: Start from the bottom of the tree and examine each nonleaf subtree. If replacement of this subtree with a leaf, or with its most frequently used branch, would lead to a lower predicted error rate, then prune the tree accordingly, remembering that the predicted error rate for all trees that include this one will be affected. Since the error rate for the whole tree decreases as the error rate of any of its subtrees is reduced, this process will lead to a tree whose predicted error rate is minimal with respect to the allowable forms of pruning.

How can we predict these error rates? It is clear that error rate on the training set from which the tree was built (*resubstitution error*, in the terminology of Breiman *et al.*) does not provide a suitable estimate; as far as the training set is concerned, pruning always increases error. In the tree of Figure 4–1, the first replaced subtree separates 1 Republican

from 167 Democrats (with no errors on the training set), so pruning this subtree to the leaf democrat causes one of the training cases to be misclassified. The second subtree misclassifies 7 training cases when sorting out 11 Democrats and 112 Republicans, but this increases to 11 errors when the tree is replaced by the leaf republican.

This search for a way of predicting error rates leads once again to two families of techniques. The first family predicts the error rate of the tree and its subtrees using a new set of cases that is distinct from the training set. Since these cases were not examined at the time the tree was constructed, the estimates obtained from them are clearly unbiased and, if there are enough of them, reliable. Examples of such techniques are:

- *Cost-complexity* pruning [Breiman *et al.*, 1984], in which the predicted error rate of a tree is modeled as the weighted sum of its complexity and its error on the training cases, with the separate cases used primarily to determine an appropriate weighting.

- *Reduced-error* pruning [Quinlan, 1987e], which assesses the error rates of the tree and its components directly on the set of separate cases.

The drawback associated with this family of techniques is simply that some of the available data must be reserved for the separate set, so the original tree must be constructed from a smaller training set. This may not be much of a disadvantage when data is abundant, but can lead to inferior trees when data is scarce. One way around this problem is to use a *cross-validation* approach. In essence, the available cases are divided into C equal-sized blocks and, for each block, a tree is constructed from cases in all the other blocks and tested on cases in the "holdout" block. For moderate values of C, the assumption is made that the tree constructed from all but one block will not differ much from the tree constructed from all data. Of course, C trees must be grown instead of just one. See Breiman *et al.* [1984] and Chapter 9 for more on cross-validation.

The approach taken in C4.5 belongs to the second family of techniques that use only the training set from which the tree was built. The raw resubstitution estimate of error rate is adjusted to reflect this estimate's bias. In earlier work, I developed a method called *pessimistic* pruning, inspired by a statistical correction, that effectively increased the number of errors observed at each leaf by 0.5 [Quinlan, 1987e]. C4.5 now employs a far more pessimistic estimate as follows.

When N training cases are covered by a leaf, E of them incorrectly, the resubstitution error rate for this leaf is E/N. However, we can regard

this somewhat naively as observing E "events" in N trials. If this set of N training cases could be regarded as a sample (which, of course, it is not), we could ask what this result tells us about the probability of an event (error) over the entire population of cases covered by this leaf. The probability of error cannot be determined exactly, but has itself a (posterior) probability distribution that is usually summarized by a pair of confidence limits. For a given confidence level CF, the upper limit on this probability can be found from the confidence limits for the binomial distribution; this upper limit is here written $U_{CF}(E, N)$[6]. Then, C4.5 simply equates the predicted error rate at a leaf with this upper limit, on the argument that the tree has been constructed to minimize the observed error rate. Now, this description does violence to statistical notions of sampling and confidence limits, so the reasoning should be taken with a large grain of salt. Like many heuristics with questionable underpinnings, however, the estimates that it produces seem frequently to yield acceptable results.

To simplify the accounting, error estimates for leaves and subtrees are computed assuming that they were used to classify a set of unseen cases of the same size as the training set. So, a leaf covering N training cases with a predicted error rate of $U_{CF}(E, N)$ would give rise to a predicted $N \times U_{CF}(E, N)$ errors. Similarly, the number of predicted errors associated with a (sub)tree is just the sum of the predicted errors of its branches.

4.3 Example: Democrats and Republicans

To illustrate what is happening, we will return to the example in Figure 4–1. The subtree

```
education spending = n: democrat (6)
education spending = y: democrat (9)
education spending = u: republican (1)
```

has no associated errors on the training set. For the first leaf, $N = 6$, $E = 0$, and (using C4.5's default confidence level of 25%), $U_{25\%}(0, 6) = 0.206$, so the predicted number of errors if this leaf were used to classify 6 unseen cases is 6×0.206. For the remaining leaves, $U_{25\%}(0, 9) = 0.143$ and $U_{25\%}(0, 1) = 0.750$, so the number of predicted errors for this subtree is given by

$$6 \times 0.206 + 9 \times 0.143 + 1 \times 0.750 = 3.273.$$

6. The upper and lower limits are symmetrical, so that the probability that the real error rate exceeds $U_{CF}(E, N)$ is $CF/2$.

If the subtree were replaced by the leaf democrat, it would cover the same 16 cases with one error, so the corresponding predicted errors come to

$$16 \times U_{25\%}(1, 16) = 16 \times 0.157 = 2.512.$$

Since the existing subtree has a higher number of predicted errors, it is pruned to a leaf.

The subtree immediately above this now looks like

```
adoption of the budget resolution = y: democrat (151)
adoption of the budget resolution = u: democrat (1)
adoption of the budget resolution = n: democrat (16/1)
```

The number of predicted errors for this subtree is

$$151 \times U_{25\%}(0, 151) + 1 \times U_{25\%}(0, 1) + 2.512 \text{ (from above)}$$

which comes to 4.642. If this subtree were replaced by the leaf democrat, the predicted errors would be $168 \times U_{25\%}(1, 168) = 2.610$. The predicted error rate for the leaf again is lower than that for the subtree, so this subtree is also pruned to a leaf.

4.4 Estimating error rates for trees

The numbers (N/E) at the leaves of the pruned tree in Figure 4–1 can now be explained. As before, N is the number of training cases covered by the leaf. E is just the number of predicted errors if a set of N unseen cases were classified by the tree.

The sum of the predicted errors at the leaves, divided by the number of cases in the training set, provides an immediate estimate of the error rate of the pruned tree on unseen cases. For this tree, the sum of the predicted errors at the leaves is 20.8 for a training set of size 300. By this estimate, then, the pruned tree will misclassify 6.9% of unseen cases.

The summary of results on the training cases and a set of test cases appears in Figure 4–2. For this particular dataset, the error rate of the pruned tree is higher than that of the unpruned tree for the training data, but, as hoped, the pruned tree has a lower error rate than the original tree on the unseen cases. The estimate here of 6.9% turns out to be somewhat high as the observed error rate on unseen cases is 3%. On a ten-way cross-validation, however, the average actual and predicted error rates on unseen cases, 5.3% and 5.6% respectively, are much closer.

Evaluation on training data (300 items):

Before Pruning		After Pruning		
Size	Errors	Size	Errors	Estimate
25	8(2.7%)	7	13(4.3%)	(6.9%)

Evaluation on test data (135 items):

Before Pruning		After Pruning		
Size	Errors	Size	Errors	Estimate
25	7(5.2%)	7	4(3.0%)	(6.9%)

Figure 4–2. Results with congressional voting data