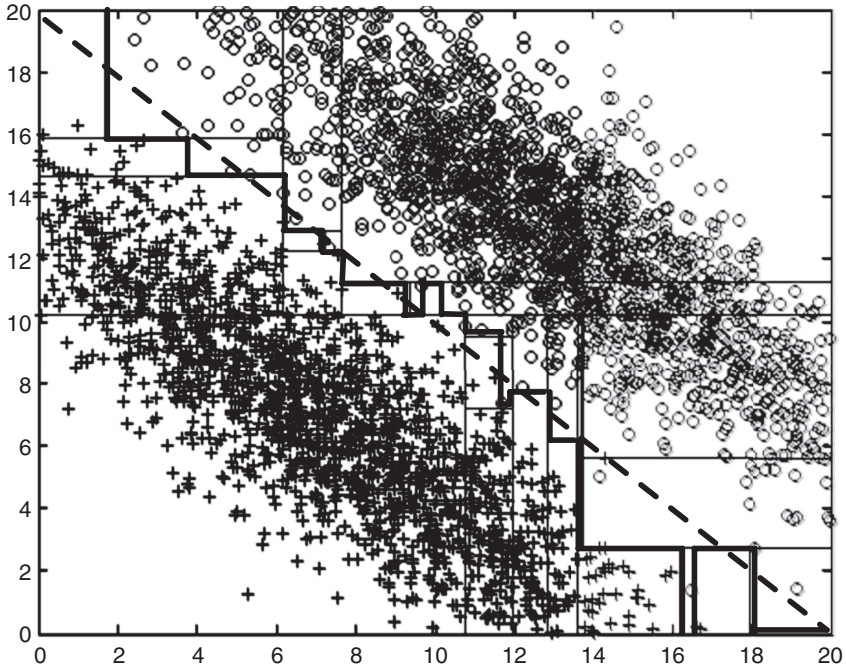


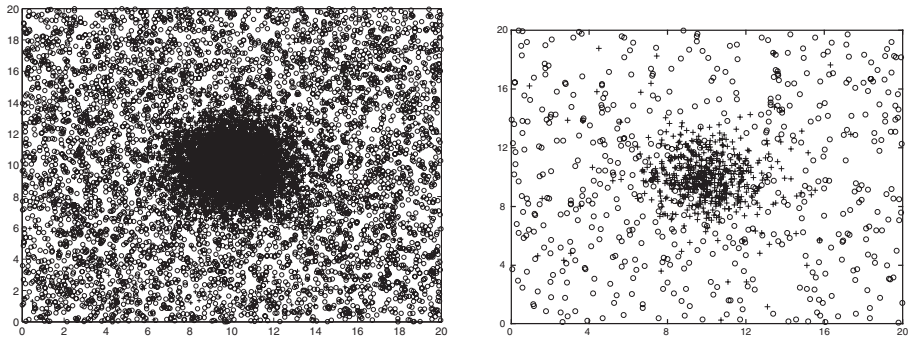
Figure 3.21. Example of data set that cannot be partitioned optimally using a decision tree with single attribute test conditions. The true decision boundary is shown by the dashed line.

Although an oblique decision tree is more expressive and can produce more compact trees, finding the optimal test condition is computationally more expensive.

9. **Choice of Impurity Measure:** It should be noted that the choice of impurity measure often has little effect on the performance of decision tree classifiers since many of the impurity measures are quite consistent with each other, as shown in Figure 3.11 on page 129. Instead, the strategy used to prune the tree has a greater impact on the final tree than the choice of impurity measure.

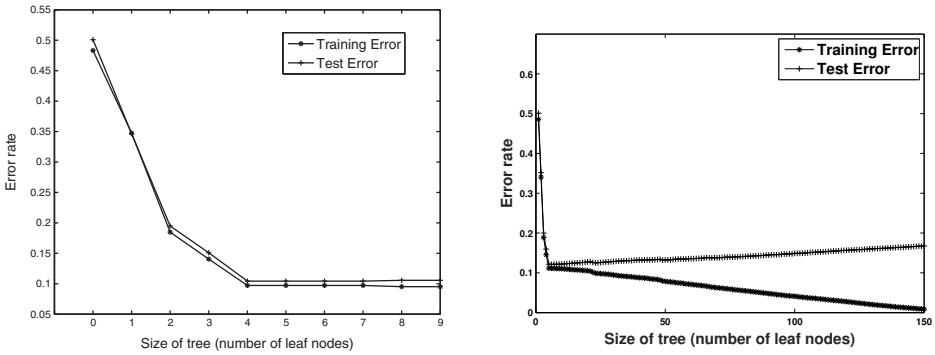
3.4 Model Overfitting

Methods presented so far try to learn classification models that show the lowest error on the training set. However, as we will show in the following example, even if a model fits well over the training data, it can still show poor generalization performance, a phenomenon known as model overfitting.



(a) Example of a 2-D data. (b) Training set using 10% data.

Figure 3.22. Examples of training and test sets of a two-dimensional classification problem.



(a) Varying tree size from 1 to 8. (b) Varying tree size from 1 to 150.

Figure 3.23. Effect of varying tree size (number of leaf nodes) on training and test errors.

Example 3.5. [Overfitting and Underfitting of Decision Trees] Consider the two-dimensional data set shown in Figure 3.22(a). The data set contains instances that belong to two separate classes, represented as + and o, respectively, where each class has 5400 instances. All instances belonging to the o class were generated from a uniform distribution. For the + class, 5000 instances were generated from a Gaussian distribution centered at (10,10) with unit variance, while the remaining 400 instances were sampled from the same uniform distribution as the o class. We can see from Figure 3.22(a) that

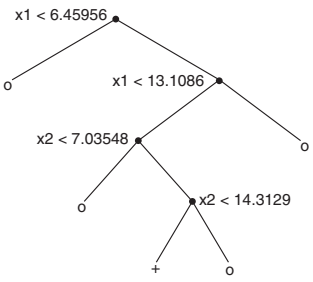
the + class can be largely distinguished from the o class by drawing a circle of appropriate size centered at (10,10). To learn a classifier using this two-dimensional data set, we randomly sampled 10% of the data for training and used the remaining 90% for testing. The training set, shown in Figure 3.22(b), looks quite representative of the overall data. We used Gini index as the impurity measure to construct decision trees of increasing sizes (number of leaf nodes), by recursively expanding a node into child nodes till every leaf node was pure, as described in Section 3.3.4.

Figure 3.23(a) shows changes in the training and test error rates as the size of the tree varies from 1 to 8. Both error rates are initially large when the tree has only one or two leaf nodes. This situation is known as **model underfitting**. Underfitting occurs when the learned decision tree is too simplistic, and thus, incapable of fully representing the true relationship between the attributes and the class labels. As we increase the tree size from 1 to 8, we can observe two effects. First, both the error rates decrease since larger trees are able to represent more complex decision boundaries. Second, the training and test error rates are quite close to each other, which indicates that the performance on the training set is fairly representative of the generalization performance. As we further increase the size of the tree from 8 to 150, the training error continues to steadily decrease till it eventually reaches zero, as shown in Figure 3.23(b). However, in a striking contrast, the test error rate ceases to decrease any further beyond a certain tree size, and then it begins to increase. The training error rate thus grossly under-estimates the test error rate once the tree becomes too large. Further, the gap between the training and test error rates keeps on widening as we increase the tree size. This behavior, which may seem counter-intuitive at first, can be attributed to the phenomena of **model overfitting**.

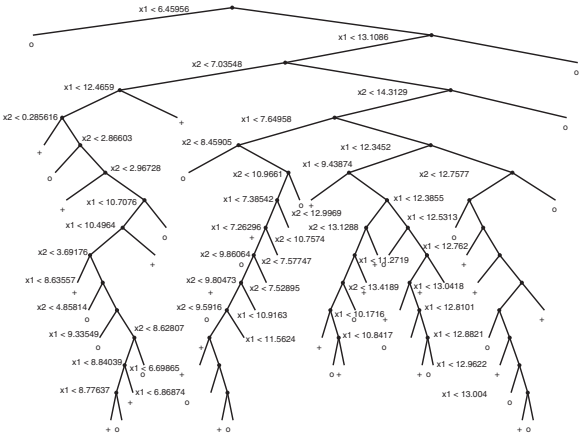
■

3.4.1 Reasons for Model Overfitting

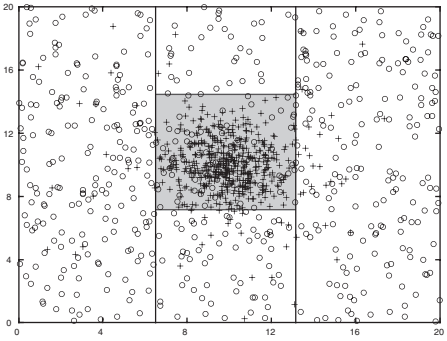
Model overfitting is the phenomena where, in the pursuit of minimizing the training error rate, an overly complex model is selected that captures specific patterns in the training data but fails to learn the *true* nature of relationships between attributes and class labels in the overall data. To illustrate this, Figure 3.24 shows decision trees and their corresponding decision boundaries (shaded rectangles represent regions assigned to the + class) for two trees of sizes 5 and 50. We can see that the decision tree of size 5 appears quite simple and its decision boundaries provide a reasonable approximation to the ideal decision boundary, which in this case corresponds to a circle centered around



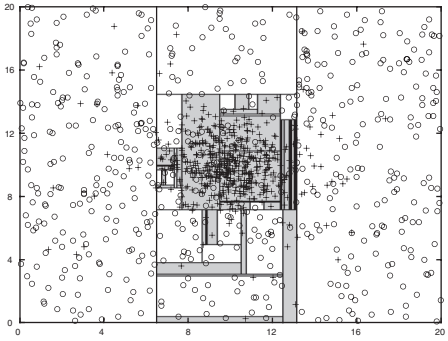
(a) Decision tree with 5 leaf nodes.



(b) Decision tree with 50 leaf nodes.



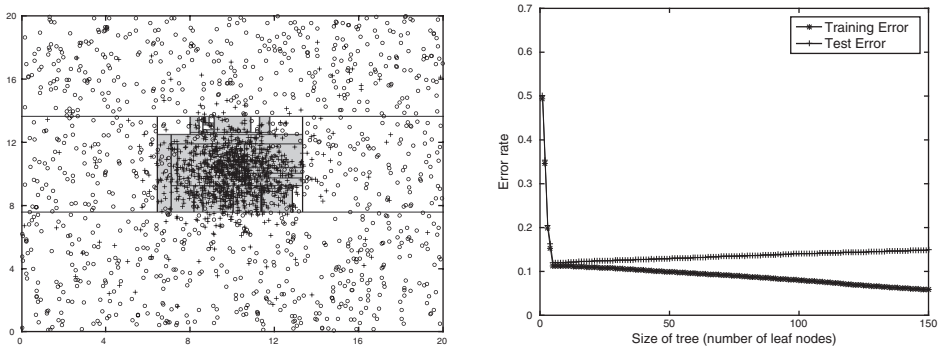
(c) Decision boundary for tree with 5 leaf nodes.



(d) Decision boundary for tree with 50 leaf nodes.

Figure 3.24. Decision trees with different model complexities.

the Gaussian distribution at (10, 10). Although its training and test error rates are non-zero, they are very close to each other, which indicates that the patterns learned in the training set should generalize well over the test set. On the other hand, the decision tree of size 50 appears much more complex than the tree of size 5, with complicated decision boundaries. For example, some of its shaded rectangles (assigned the + class) attempt to cover narrow regions in the input space that contain only one or two + training instances. Note that



(a) Decision boundary for tree with 50 leaf nodes using 20% data for training.

(b) Training and test error rates using 20% data for training.

Figure 3.25. Performance of decision trees using 20% data for training (twice the original training size).

the prevalence of + instances in such regions is highly specific to the training set, as these regions are mostly dominated by − instances in the overall data. Hence, in an attempt to perfectly fit the training data, the decision tree of size 50 starts fine tuning itself to specific patterns in the training data, leading to poor performance on an independently chosen test set.

There are a number of factors that influence model overfitting. In the following, we provide brief descriptions of two of the major factors: limited training size and high model complexity. Though they are not exhaustive, the interplay between them can help explain most of the common model overfitting phenomena in real-world applications.

Limited Training Size

Note that a training set consisting of a finite number of instances can only provide a limited representation of the overall data. Hence, it is possible that the patterns learned from a training set do not fully represent the true patterns in the overall data, leading to model overfitting. In general, as we increase the size of a training set (number of training instances), the patterns learned from the training set start resembling the true patterns in the overall data. Hence, the effect of overfitting can be reduced by increasing the training size, as illustrated in the following example.

Example 3.6. [Effect of Training Size] Suppose that we use twice the number of training instances than what we had used in the experiments conducted in Example 3.5. Specifically, we use 20% data for training and use the remainder for testing. Figure 3.25(b) shows the training and test error rates as the size of the tree is varied from 1 to 150. There are two major differences in the trends shown in this figure and those shown in Figure 3.23(b) (using only 10% of the data for training). First, even though the training error rate decreases with increasing tree size in both figures, its rate of decrease is much smaller when we use twice the training size. Second, for a given tree size, the gap between the training and test error rates is much smaller when we use twice the training size. These differences suggest that the patterns learned using 20% of data for training are more generalizable than those learned using 10% of data for training.

Figure 3.25(a) shows the decision boundaries for the tree of size 50, learned using 20% of data for training. In contrast to the tree of the same size learned using 10% data for training (see Figure 3.24(d)), we can see that the decision tree is not capturing specific patterns of noisy + instances in the training set. Instead, the high model complexity of 50 leaf nodes is being effectively used to learn the boundaries of the + instances centered at (10, 10).

■

High Model Complexity

Generally, a more complex model has a better ability to represent complex patterns in the data. For example, decision trees with larger number of leaf nodes can represent more complex decision boundaries than decision trees with fewer leaf nodes. However, an overly complex model also has a tendency to learn specific patterns in the training set that do not generalize well over unseen instances. Models with high complexity should thus be judiciously used to avoid overfitting.

One measure of model complexity is the number of "parameters" that need to be inferred from the training set. For example, in the case of decision tree induction, the attribute test conditions at internal nodes correspond to the parameters of the model that need to be inferred from the training set. A decision tree with larger number of attribute test conditions (and consequently more leaf nodes) thus involves more "parameters" and hence is more complex.

Given a class of models with a certain number of parameters, a learning algorithm attempts to select the best combination of parameter values that maximizes an evaluation metric (e.g., accuracy) over the training set. If the number of parameter value combinations (and hence the complexity) is large,

the learning algorithm has to select the best combination from a large number of possibilities, using a limited training set. In such cases, there is a high chance for the learning algorithm to pick a *spurious* combination of parameters that maximizes the evaluation metric just by random chance. This is similar to the **multiple comparisons problem** (also referred as multiple testing problem) in statistics.

As an illustration of the multiple comparisons problem, consider the task of predicting whether the stock market will rise or fall in the next ten trading days. If a stock analyst simply makes random guesses, the probability that her prediction is correct on any trading day is 0.5. However, the probability that she will predict correctly at least nine out of ten times is

$$\frac{\binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0107,$$

which is extremely low.

Suppose we are interested in choosing an investment advisor from a pool of 200 stock analysts. Our strategy is to select the analyst who makes the most number of correct predictions in the next ten trading days. The flaw in this strategy is that even if all the analysts make their predictions in a random fashion, the probability that at least one of them makes at least nine correct predictions is

$$1 - (1 - 0.0107)^{200} = 0.8847,$$

which is very high. Although each analyst has a low probability of predicting at least nine times correctly, considered together, we have a high probability of finding at least one analyst who can do so. However, there is no guarantee in the future that such an analyst will continue to make accurate predictions by random guessing.

How does the multiple comparisons problem relate to model overfitting? In the context of learning a classification model, each combination of parameter values corresponds to an analyst, while the number of training instances corresponds to the number of days. Analogous to the task of selecting the best analyst who makes the most accurate predictions on consecutive days, the task of a learning algorithm is to select the best combination of parameters that results in the highest accuracy on the training set. If the number of parameter combinations is large but the training size is small, it is highly likely for the learning algorithm to choose a spurious parameter combination that provides high training accuracy just by random chance. In the following example, we illustrate the phenomena of overfitting due to multiple comparisons in the context of decision tree induction.

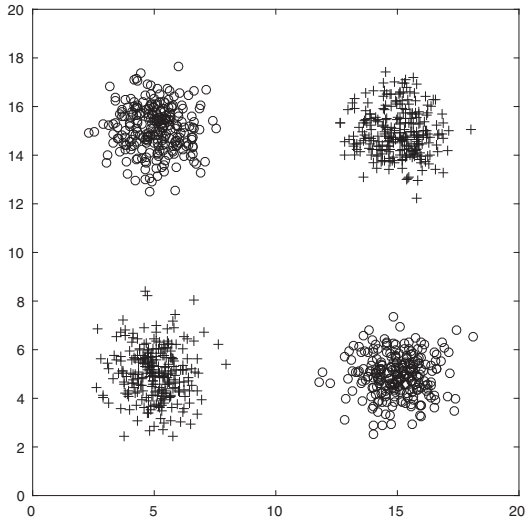
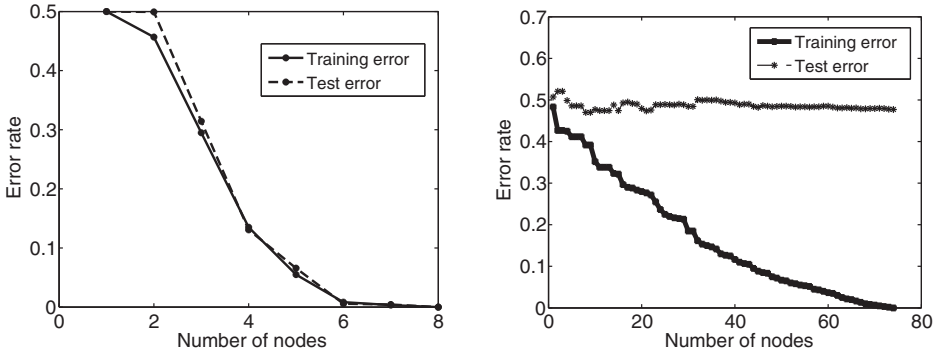


Figure 3.26. Example of a two-dimensional (X-Y) data set.

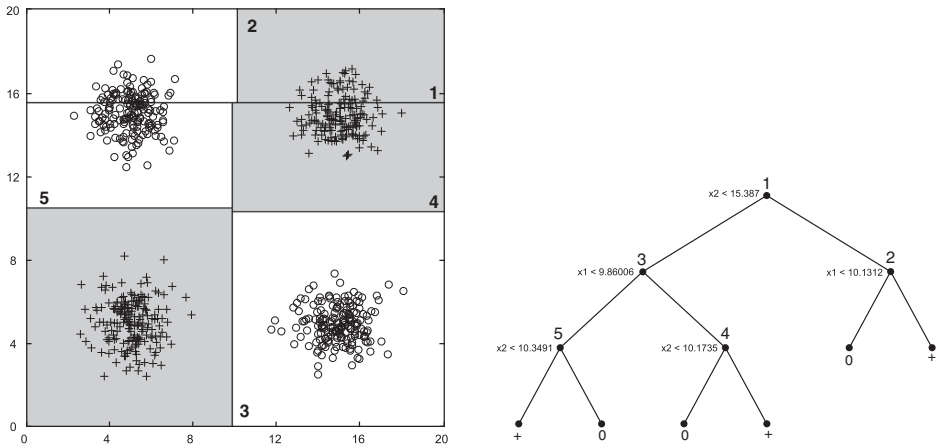


(a) Using X and Y attributes only.

(b) After adding 100 irrelevant attributes.

Figure 3.27. Training and test error rates illustrating the effect of multiple comparisons problem on model overfitting.

Example 3.7. [Multiple Comparisons and Overfitting] Consider the two-dimensional data set shown in Figure 3.26 containing 500 + and 500 o instances, which is similar to the data shown in Figure 3.19. In this data set, the distributions of both classes are well-separated in the two-dimensional (X-Y) attribute space, but none of the two attributes (X or Y) are sufficiently informative to be used alone for separating the two classes. Hence, splitting



(a) Decision boundary for tree with 6 leaf nodes.

(b) Decision tree with 6 leaf nodes.

Figure 3.28. Decision tree with 6 leaf nodes using X and Y as attributes. Splits have been numbered from 1 to 5 in order of other occurrence in the tree.

the data set based on any value of an X or Y attribute will provide close to zero reduction in an impurity measure. However, if X and Y attributes are used together in the splitting criterion (e.g., splitting X at 10 *and* Y at 10), the two classes can be effectively separated.

Figure 3.27(a) shows the training and test error rates for learning decision trees of varying sizes, when 30% of the data is used for training and the remainder of the data for testing. We can see that the two classes can be separated using a small number of leaf nodes. Figure 3.28 shows the decision boundaries for the tree with six leaf nodes, where the splits have been numbered according to their order of appearance in the tree. Note that the even though splits 1 and 3 provide trivial gains, their consequent splits (2, 4, and 5) provide large gains, resulting in effective discrimination of the two classes.

Assume we add 100 irrelevant attributes to the two-dimensional X-Y data. Learning a decision tree from this resultant data will be challenging because the number of candidate attributes to choose for splitting at every internal node will increase from two to 102. With such a large number of candidate attribute test conditions to choose from, it is quite likely that spurious attribute test conditions will be selected at internal nodes because of the multiple comparisons problem. Figure 3.27(b) shows the training and test error rates after adding 100 irrelevant attributes to the training set. We can see that the

test error rate remains close to 0.5 even after using 50 leaf nodes, while the training error rate keeps on declining and eventually becomes 0.

■

3.5 Model Selection

There are many possible classification models with varying levels of model complexity that can be used to capture patterns in the training data. Among these possibilities, we want to select the model that shows lowest generalization error rate. The process of selecting a model with the right level of complexity, which is expected to generalize well over unseen test instances, is known as **model selection**. As described in the previous section, the training error rate cannot be reliably used as the sole criterion for model selection. In the following, we present three generic approaches to estimate the generalization performance of a model that can be used for model selection. We conclude this section by presenting specific strategies for using these approaches in the context of decision tree induction.

3.5.1 Using a Validation Set

Note that we can always estimate the generalization error rate of a model by using "out-of-sample" estimates, i.e. by evaluating the model on a separate **validation set** that is not used for training the model. The error rate on the validation set, termed as the validation error rate, is a better indicator of generalization performance than the training error rate, since the validation set has not been used for training the model. The validation error rate can be used for model selection as follows.

Given a training set $D.train$, we can partition $D.train$ into two smaller subsets, $D.tr$ and $D.val$, such that $D.tr$ is used for training while $D.val$ is used as the validation set. For example, two-thirds of $D.train$ can be reserved as $D.tr$ for training, while the remaining one-third is used as $D.val$ for computing validation error rate. For any choice of classification model m that is trained on $D.tr$, we can estimate its validation error rate on $D.val$, $err_{val}(m)$. The model that shows the lowest value of $err_{val}(m)$ can then be selected as the preferred choice of model.

The use of validation set provides a generic approach for model selection. However, one limitation of this approach is that it is sensitive to the sizes of $D.tr$ and $D.val$, obtained by partitioning $D.train$. If the size of $D.tr$ is too small, it may result in the learning of a poor classification model with sub-standard performance, since a smaller training set will be less representative