3.6 INDUCTIVE BIAS IN DECISION TREE LEARNING

What is the policy by which ID3 generalizes from observed training examples to classify unseen instances? In other words, what is its inductive bias? Recall from Chapter 2 that inductive bias is the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances.

Given a collection of training examples, there are typically many decision trees consistent with these examples. Describing the inductive bias of ID3 therefore consists of describing the basis by which it chooses one of these consistent hypotheses over the others. Which of these decision trees does ID3 choose? It chooses the first acceptable tree it encounters in its simple-to-complex, hill-climbing search through the space of possible trees. Roughly speaking, then, the ID3 search strategy (a) selects in favor of shorter trees over longer ones, and (b) selects trees that place the attributes with highest information gain closest to the root. Because of the subtle interaction between the attribute selection heuristic used by ID3 and the particular training examples it encounters, it is difficult to characterize precisely the inductive bias exhibited by ID3. However, we can approximately characterize its bias as a preference for short decision trees over complex trees.

> **Approximate inductive bias of ID3:** Shorter trees are preferred over larger trees.

In fact, one could imagine an algorithm similar to ID3 that exhibits precisely this inductive bias. Consider an algorithm that begins with the empty tree and searches *breadth first* through progressively more complex trees, first considering all trees of depth 1, then all trees of depth 2, etc. Once it finds a decision tree consistent with the training data, it returns the smallest consistent tree at that search depth (e.g., the tree with the fewest nodes). Let us call this breadth-first search algorithm BFS-ID3. BFS-ID3 finds a shortest decision tree and thus exhibits precisely the bias "shorter trees are preferred over longer trees." ID3 can be viewed as an efficient approximation to BFS-ID3, using a greedy heuristic search to attempt to find the shortest tree without conducting the entire breadth-first search through the hypothesis space.

Because ID3 uses the information gain heuristic and a hill climbing strategy, it exhibits a more complex bias than BFS-ID3. In particular, it does not always find the shortest consistent tree, and it is biased to favor trees that place attributes with high information gain closest to the root.

> **A closer approximation to the inductive bias of ID3:** Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

3.6.1 Restriction Biases and Preference Biases

There is an interesting difference between the types of inductive bias exhibited by ID3 and by the CANDIDATE-ELIMINATION algorithm discussed in Chapter 2.

Consider the difference between the hypothesis space search in these two approaches:

- ID3 searches a *complete* hypothesis space (i.e., one capable of expressing any finite discrete-valued function). It searches *incompletely* through this space, from simple to complex hypotheses, until its termination condition is met (e.g., until it finds a hypothesis consistent with the data). Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy. Its hypothesis space introduces no additional bias.
- The version space CANDIDATE-ELIMINATION algorithm searches an *incomplete* hypothesis space (i.e., one that can express only a subset of the potentially teachable concepts). It searches this space *completely*, finding every hypothesis consistent with the training data. Its inductive bias is solely a consequence of the expressive power of its hypothesis representation. Its search strategy introduces no additional bias.

In brief, the inductive bias of ID3 follows from its *search strategy*, whereas the inductive bias of the CANDIDATE-ELIMINATION algorithm follows from the definition of its *search space*.

The inductive bias of ID3 is thus a *preference* for certain hypotheses over others (e.g., for shorter hypotheses), with no hard restriction on the hypotheses that can be eventually enumerated. This form of bias is typically called a *preference bias* (or, alternatively, a *search bias*). In contrast, the bias of the CANDIDATE-ELIMINATION algorithm is in the form of a categorical *restriction* on the set of hypotheses considered. This form of bias is typically called a *restriction bias* (or, alternatively, a *language bias*).

Given that some form of inductive bias is required in order to generalize beyond the training data (see Chapter 2), which type of inductive bias shall we prefer; a preference bias or restriction bias?

Typically, a preference bias is more desirable than a restriction bias, because it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function. In contrast, a restriction bias that strictly limits the set of potential hypotheses is generally less desirable, because it introduces the possibility of excluding the unknown target function altogether.

Whereas ID3 exhibits a purely preference bias and CANDIDATE-ELIMINATION a purely restriction bias, some learning systems combine both. Consider, for example, the program described in Chapter 1 for learning a numerical evaluation function for game playing. In this case, the learned evaluation function is represented by a linear combination of a fixed set of board features, and the learning algorithm adjusts the parameters of this linear combination to best fit the available training data. In this case, the decision to use a linear function to represent the evaluation function introduces a restriction bias (nonlinear evaluation functions cannot be represented in this form). At the same time, the choice of a particular parameter tuning method (the LMS algorithm in this case) introduces a preference bias stemming from the ordered search through the space of all possible parameter values.

3.6.2 Why Prefer Short Hypotheses?

Is ID3's inductive bias favoring shorter decision trees a sound basis for generalizing beyond the training data? Philosophers and others have debated this question for centuries, and the debate remains unresolved to this day. William of Occam was one of the first to discuss† the question, around the year 1320, so this bias often goes by the name of Occam's razor.

Occam's razor: Prefer the simplest hypothesis that fits the data.

Of course giving an inductive bias a name does not justify it. Why should one prefer simpler hypotheses? Notice that scientists sometimes appear to follow this inductive bias. Physicists, for example, prefer simple explanations for the motions of the planets, over more complex explanations. Why? One argument is that because there are fewer short hypotheses than long ones (based on straightforward combinatorial arguments), it is less likely that one will find a short hypothesis that coincidentally fits the training data. In contrast there are often many very complex hypotheses that fit the current training data but fail to generalize correctly to subsequent data. Consider decision tree hypotheses, for example. There are many more 500-node decision trees than 5-node decision trees. Given a small set of 20 training examples, we might expect to be able to find many 500-node decision trees consistent with these, whereas we would be more surprised if a 5-node decision tree could perfectly fit this data. We might therefore believe the 5-node tree is less likely to be a statistical coincidence and prefer this hypothesis over the 500-node hypothesis.

Upon closer examination, it turns out there is a major difficulty with the above argument. By the same reasoning we could have argued that one should prefer decision trees containing exactly 17 leaf nodes with 11 nonleaf nodes, that use the decision attribute A_1 at the root, and test attributes A_2 through A_{11}, in numerical order. There are relatively few such trees, and we might argue (by the same reasoning as above) that our a priori chance of finding one consistent with an arbitrary set of data is therefore small. The difficulty here is that there are very many small sets of hypotheses that one can define—most of them rather arcane. Why should we believe that the small set of hypotheses consisting of decision trees with *short descriptions* should be any more relevant than the multitude of other small sets of hypotheses that we might define?

A second problem with the above argument for Occam's razor is that the size of a hypothesis is determined by the particular representation used *internally* by the learner. Two learners using different internal representations could therefore arrive at different hypotheses, both justifying their contradictory conclusions by Occam's razor! For example, the function represented by the learned decision tree in Figure 3.1 could be represented as a tree with just one decision node, by a learner that uses the boolean attribute XYZ, where we define the attribute XYZ to

† Apparently while shaving.

be true for instances that are classified positive by the decision tree in Figure 3.1 and false otherwise. Thus, two learners, both applying Occam's razor, would generalize in different ways if one used the XYZ attribute to describe its examples and the other used only the attributes *Outlook, Temperature, Humidity*, and *Wind*.

This last argument shows that Occam's razor will produce two different hypotheses from the same training examples when it is applied by two learners that perceive these examples in terms of different internal representations. On this basis we might be tempted to reject Occam's razor altogether. However, consider the following scenario that examines the question of which internal representations might arise from a process of evolution and natural selection. Imagine a population of artificial learning agents created by a simulated evolutionary process involving reproduction, mutation, and natural selection of these agents. Let us assume that this evolutionary process can alter the perceptual systems of these agents from generation to generation, thereby changing the internal attributes by which they perceive their world. For the sake of argument, let us also assume that the learning agents employ a fixed learning algorithm (say ID3) that cannot be altered by evolution. It is reasonable to assume that over time evolution will produce internal representation that make these agents increasingly successful within their environment. Assuming that the success of an agent depends highly on its ability to generalize accurately, we would therefore expect evolution to develop internal representations that work well with whatever learning algorithm and inductive bias is present. If the species of agents employs a learning algorithm whose inductive bias is Occam's razor, then we expect evolution to produce internal representations for which Occam's razor is a successful strategy. The essence of the argument here is that evolution will create internal representations that make the learning algorithm's inductive bias a self-fulfilling prophecy, simply because it can alter the representation easier than it can alter the learning algorithm.

For now, we leave the debate regarding Occam's razor. We will revisit it in Chapter 6, where we discuss the Minimum Description Length principle, a version of Occam's razor that can be interpreted within a Bayesian framework.

3.7 ISSUES IN DECISION TREE LEARNING

Practical issues in learning decision trees include determining how deeply to grow the decision tree, handling continuous attributes, choosing an appropriate attribute selection measure, handling training data with missing attribute values, handling attributes with differing costs, and improving computational efficiency. Below we discuss each of these issues and extensions to the basic ID3 algorithm that address them. ID3 has itself been extended to address most of these issues, with the resulting system renamed C4.5 (Quinlan 1993).

3.7.1 Avoiding Overfitting the Data

The algorithm described in Table 3.1 grows each branch of the tree just deeply enough to perfectly classify the training examples. While this is sometimes a

reasonable strategy, in fact it can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that *overfit* the training examples.

We will say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set).

> ***Definition***: Given a hypothesis space H, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

Figure 3.6 illustrates the impact of overfitting in a typical application of decision tree learning. In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes. The horizontal axis of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed. The vertical axis indicates the accuracy of predictions made by the tree. The solid line shows the accuracy of the decision tree over the training examples, whereas the broken line shows accuracy measured over an independent set of test examples (not included in the training set). Predictably, the accuracy of the tree over the training examples increases monotonically as the tree is grown. However, the accuracy measured over the independent test examples first increases, then decreases. As can be seen, once the tree size exceeds approximately 25 nodes,

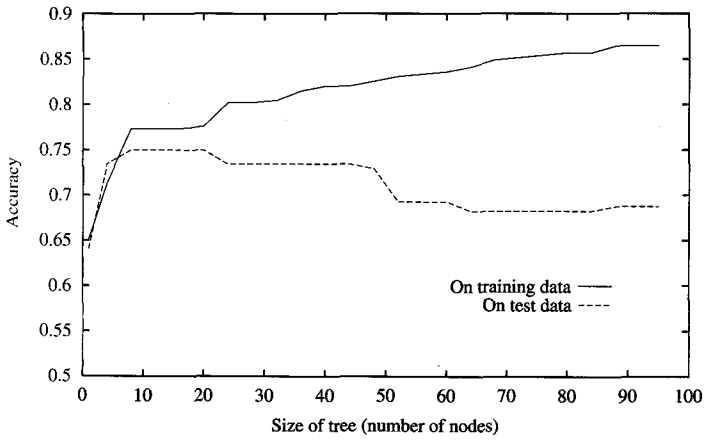

FIGURE 3.6
Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases. Software and data for experimenting with variations on this plot are available on the World Wide Web at http://www.cs.cmu.edu/~tom/mlbook.html.

further elaboration of the tree decreases its accuracy over the test examples despite increasing its accuracy on the training examples.

How can it be possible for tree h to fit the training examples better than h', but for it to perform more poorly over subsequent examples? One way this can occur is when the training examples contain random errors or noise. To illustrate, consider the effect of adding the following positive training example, incorrectly labeled as negative, to the (otherwise correct) examples in Table 3.2.

$$\langle Outlook = Sunny, Temperature = Hot, Humidity = Normal,$$
$$Wind = Strong, PlayTennis = No \rangle$$

Given the original error-free data, ID3 produces the decision tree shown in Figure 3.1. However, the addition of this incorrect example will now cause ID3 to construct a more complex tree. In particular, the new example will be sorted into the second leaf node from the left in the learned tree of Figure 3.1, along with the previous positive examples D9 and D11. Because the new example is labeled as a negative example, ID3 will search for further refinements to the tree below this node. Of course as long as the new erroneous example differs in some arbitrary way from the other examples affiliated with this node, ID3 will succeed in finding a new decision attribute to separate out this new example from the two previous positive examples at this tree node. The result is that ID3 will output a decision tree (h) that is more complex than the original tree from Figure 3.1 (h'). Of course h will fit the collection of training examples perfectly, whereas the simpler h' will not. However, given that the new decision node is simply a consequence of fitting the noisy training example, we expect h to outperform h' over subsequent data drawn from the same instance distribution.

The above example illustrates how random noise in the training examples can lead to overfitting. In fact, overfitting is possible even when the training data are noise-free, especially when small numbers of examples are associated with leaf nodes. In this case, it is quite possible for coincidental regularities to occur, in which some attribute happens to partition the examples very well, despite being unrelated to the actual target function. Whenever such coincidental regularities exist, there is a risk of overfitting.

Overfitting is a significant practical difficulty for decision tree learning and many other learning methods. For example, in one experimental study of ID3 involving five different learning tasks with noisy, nondeterministic data (Mingers 1989b), overfitting was found to decrease the accuracy of learned decision trees by 10–25% on most problems.

There are several approaches to avoiding overfitting in decision tree learning. These can be grouped into two classes:

- approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
- approaches that allow the tree to overfit the data, and then post-prune the tree.

Although the first of these approaches might seem more direct, the second approach of post-pruning overfit trees has been found to be more successful in practice. This is due to the difficulty in the first approach of estimating precisely when to stop growing the tree.

Regardless of whether the correct tree size is found by stopping early or by post-pruning, a key question is what criterion is to be used to determine the correct final tree size. Approaches include:

- Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
- Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set. For example, Quinlan (1986) uses a chi-square test to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data.
- Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. This approach, based on a heuristic called the Minimum Description Length principle, is discussed further in Chapter 6, as well as in Quinlan and Rivest (1989) and Mehta et al. (1995).

The first of the above approaches is the most common and is often referred to as a *training and validation set* approach. We discuss the two main variants of this approach below. In this approach, the available data are separated into two sets of examples: a *training set*, which is used to form the learned hypothesis, and a separate *validation set*, which is used to evaluate the accuracy of this hypothesis over subsequent data and, in particular, to evaluate the impact of pruning this hypothesis. The motivation is this: Even though the learner may be misled by random errors and coincidental regularities within the training set, the validation set is unlikely to exhibit the same random fluctuations. Therefore, the validation set can be expected to provide a safety check against overfitting the spurious characteristics of the training set. Of course, it is important that the validation set be large enough to itself provide a statistically significant sample of the instances. One common heuristic is to withhold one-third of the available examples for the validation set, using the other two-thirds for training.

3.7.1.1 REDUCED ERROR PRUNING

How exactly might we use a validation set to prevent overfitting? One approach, called *reduced-error pruning* (Quinlan 1987), is to consider each of the decision nodes in the tree to be candidates for pruning. Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node. Nodes are removed only if the resulting pruned tree performs no worse than the

original over the validation set. This has the effect that any leaf node added due to coincidental regularities in the training set is likely to be pruned because these same coincidences are unlikely to occur in the validation set. Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set. Pruning of nodes continues until further pruning is harmful (i.e., decreases accuracy of the tree over the validation set).

The impact of reduced-error pruning on the accuracy of the decision tree is illustrated in Figure 3.7. As in Figure 3.6, the accuracy of the tree is shown measured over both training examples and test examples. The additional line in Figure 3.7 shows accuracy over the test examples as the tree is pruned. When pruning begins, the tree is at its maximum size and lowest accuracy over the test set. As pruning proceeds, the number of nodes is reduced and accuracy over the test set increases. Here, the available data has been split into three subsets: the training examples, the validation examples used for pruning the tree, and a set of test examples used to provide an unbiased estimate of accuracy over future unseen examples. The plot shows accuracy over the training and test sets. Accuracy over the validation set used for pruning is not shown.

Using a separate set of data to guide pruning is an effective approach provided a large amount of data is available. The major drawback of this approach is that when data is limited, withholding part of it for the validation set reduces even further the number of examples available for training. The following section presents an alternative approach to pruning that has been found useful in many practical situations where data is limited. Many additional techniques have been proposed as well, involving partitioning the available data several different times in

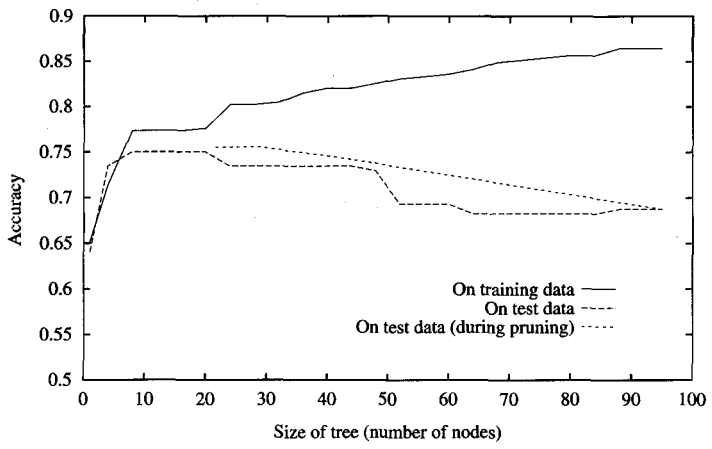

FIGURE 3.7
Effect of reduced-error pruning in decision tree learning. This plot shows the same curves of training and test set accuracy as in Figure 3.6. In addition, it shows the impact of reduced error pruning of the tree produced by ID3. Notice the increase in accuracy over the test set as nodes are pruned from the tree. Here, the validation set used for pruning is distinct from both the training and test sets.