CHAPTER 12

COMBINING INDUCTIVE AND ANALYTICAL LEARNING

Purely inductive learning methods formulate general hypotheses by finding empirical regularities over the training examples. Purely analytical methods use prior knowledge to derive general hypotheses deductively. This chapter considers methods that combine inductive and analytical mechanisms to obtain the benefits of both approaches: better generalization accuracy when prior knowledge is available and reliance on observed training data to overcome shortcomings in prior knowledge. The resulting combined methods outperform both purely inductive and purely analytical learning methods. This chapter considers inductive-analytical learning methods based on both symbolic and artificial neural network representations.

12.1 MOTIVATION

In previous chapters we have seen two paradigms for machine learning: inductive learning and analytical learning. Inductive methods, such as decision tree induction and neural network BACKPROPAGATION, seek general hypotheses that fit the observed training data. Analytical methods, such as PROLOG-EBG, seek general hypotheses that fit prior knowledge while covering the observed data. These two learning paradigms are based on fundamentally different justifications for learned hypotheses and offer complementary advantages and disadvantages. Combining them offers the possibility of more powerful learning methods.

Purely analytical learning methods offer the advantage of generalizing more accurately from less data by using prior knowledge to guide learning. However, they can be misled when given incorrect or insufficient prior knowledge. Purely inductive methods offer the advantage that they require no explicit prior knowledge and learn regularities based solely on the training data. However, they can fail when given insufficient training data, and can be misled by the implicit inductive bias they must adopt in order to generalize beyond the observed data. Table 12.1 summarizes these complementary advantages and pitfalls of inductive and analytical learning methods. This chapter considers the question of how to combine the two into a single algorithm that captures the best aspects of both.

The difference between inductive and analytical learning methods can be seen in the nature of the *justifications* that can be given for their learned hypotheses. Hypotheses output by purely analytical learning methods such as PROLOG-EBG carry a *logical* justification; the output hypothesis follows deductively from the domain theory and training examples. Hypotheses output by purely inductive learning methods such as BACKPROPAGATION carry a *statistical* justification; the output hypothesis follows from statistical arguments that the training sample is sufficiently large that it is probably representative of the underlying distribution of examples. This statistical justification for induction is clearly articulated in the PAC-learning results discussed in Chapter 7.

Given that analytical methods provide logically justified hypotheses and inductive methods provide statistically justified hypotheses, it is easy to see why combining them would be useful: Logical justifications are only as compelling as the assumptions, or prior knowledge, on which they are built. They are suspect or powerless if prior knowledge is incorrect or unavailable. Statistical justifications are only as compelling as the data and statistical assumptions on which they rest. They are suspect or powerless when assumptions about the underlying distributions cannot be trusted or when data is scarce. In short, the two approaches work well for different types of problems. By combining them we can hope to devise a more general learning approach that covers a more broad range of learning tasks.

Figure 12.1 summarizes a spectrum of learning problems that varies by the availability of prior knowledge and training data. At one extreme, a large volume

	Inductive learning	**Analytical learning**
Goal:	Hypothesis fits data	Hypothesis fits domain theory
Justification:	Statistical inference	Deductive inference
Advantages:	Requires little prior knowledge	Learns from scarce data
Pitfalls:	Scarce data, incorrect bias	Imperfect domain theory

TABLE 12.1
Comparison of purely analytical and purely inductive learning.

FIGURE 12.1
A spectrum of learning tasks. At the left extreme, no prior knowledge is available, and purely inductive learning methods with high sample complexity are therefore necessary. At the rightmost extreme, a perfect domain theory is available, enabling the use of purely analytical methods such as PROLOG-EBG. Most practical problems lie somewhere between these two extremes.

of training data is available, but no prior knowledge. At the other extreme, strong prior knowledge is available, but little training data. Most practical learning problems lie somewhere between these two extremes of the spectrum. For example, in analyzing a database of medical records to learn "symptoms for which treatment x is more effective than treatment y," one often begins with approximate prior knowledge (e.g., a qualitative model of the cause-effect mechanisms underlying the disease) that suggests the patient's temperature is more likely to be relevant than the patient's middle initial. Similarly, in analyzing a stock market database to learn the target concept "companies whose stock value will double over the next 10 months," one might have approximate knowledge of economic causes and effects, suggesting that the gross revenue of the company is more likely to be relevant than the color of the company logo. In both of these settings, our own prior knowledge is incomplete, but is clearly useful in helping discriminate relevant features from irrelevant.

The question considered in this chapter is "What kinds of learning algorithms can we devise that make use of approximate prior knowledge, together with available data, to form general hypotheses?" Notice that even when using a purely inductive learning algorithm, one has the opportunity to make design choices based on prior knowledge of the particular learning task. For example, when applying BACKPROPAGATION to a problem such as speech recognition, one must choose the encoding of input and output data, the error function to be minimized during gradient descent, the number of hidden units, the topology of the network, the learning rate and momentum, etc. In making these choices, human designers have the opportunity to embed task-specific knowledge into the learning algorithm. The result, however, is a purely inductive instantiation of BACKPROPAGATION, *specialized* by the designer's choices to the task of speech recognition. Our interest here lies in something different. We are interested in systems that take prior knowledge as an *explicit input* to the learner, in the same sense that the training data is an explicit input, so that they remain general purpose algorithms, even while taking advantage of domain-specific knowledge. In brief, our interest here lies in *domain-independent algorithms that employ explicitly input domain-dependent knowledge.*

What criteria should we use to compare alternative approaches to combining inductive and analytical learning? Given that the learner will generally not know the quality of the domain theory or the training data in advance, we are interested

in general methods that can operate robustly over the entire spectrum of problems of Figure 12.1. Some specific properties we would like from such a learning method include:

- Given no domain theory, it should learn at least as effectively as purely inductive methods.
- Given a perfect domain theory, it should learn at least as effectively as purely analytical methods.
- Given an imperfect domain theory and imperfect training data, it should combine the two to outperform either purely inductive or purely analytical methods.
- It should accommodate an unknown level of error in the training data.
- It should accommodate an unknown level of error in the domain theory.

Notice this list of desirable properties is quite ambitious. For example, accommodating errors in the training data is problematic even for statistically based induction without at least some prior knowledge or assumption regarding the distribution of errors. Combining inductive and analytical learning is an area of active current research. While the above list is a fair summary of what we would like our algorithms to accomplish, we do not yet have algorithms that satisfy all these constraints in a fully general fashion.

The next section provides a more detailed discussion of the combined inductive-analytical learning problem. Subsequent sections describe three different approaches to combining approximate prior knowledge with available training data to guide the learner's search for an appropriate hypothesis. Each of these three approaches has been demonstrated to outperform purely inductive methods in multiple task domains. For ease of comparison, we use a single example problem to illustrate all three approaches.

12.2 INDUCTIVE-ANALYTICAL APPROACHES TO LEARNING

12.2.1 The Learning Problem

To summarize, the learning problem considered in this chapter is

Given:

- A set of training examples D, possibly containing errors
- A domain theory B, possibly containing errors
- A space of candidate hypotheses H

Determine:

- A hypothesis that best fits the training examples and domain theory

What precisely shall we mean by "the hypothesis that best fits the training examples and domain theory?" In particular, shall we prefer hypotheses that fit

the data a little better at the expense of fitting the theory less well, or vice versa? We can be more precise by defining measures of hypothesis error with respect to the data and with respect to the domain theory, then phrasing the question in terms of these errors. Recall from Chapter 5 that $error_D(h)$ is defined to be the proportion of examples from D that are misclassified by h. Let us define the error $error_B(h)$ of h with respect to a domain theory B to be the probability that h will disagree with B on the classification of a randomly drawn instance. We can attempt to characterize the desired output hypothesis in terms of these errors. For example, we could require the hypothesis that minimizes some combined measure of these errors, such as

$$\underset{h \in H}{\text{argmin}}\ k_D error_D(h) + k_B error_B(h)$$

While this appears reasonable at first glance, it is not clear what values to assign to k_D and k_B to specify the relative importance of fitting the data versus fitting the theory. If we have a very poor theory and a great deal of reliable data, it will be best to weight $error_D(h)$ more heavily. Given a strong theory and a small sample of very noisy data, the best results would be obtained by weighting $error_B(h)$ more heavily. Of course if the learner does not know in advance the quality of the domain theory or training data, it will be unclear how it should weight these two error components.

An alternative perspective on the question of how to weight prior knowledge and data is the Bayesian perspective. Recall from Chapter 6 that Bayes theorem describes how to compute the posterior probability $P(h|D)$ of hypothesis h given observed training data D. In particular, Bayes theorem computes this posterior probability based on the observed data D, together with prior knowledge in the form of $P(h)$, $P(D)$, and $P(D|h)$. Thus we can think of $P(h)$, $P(D)$, and $P(D|h)$ as a form of background knowledge or domain theory, and we can think of Bayes theorem as a method for weighting this domain theory, together with the observed data D, to assign a posterior probability $P(h|D)$ to h. The Bayesian view is that one should simply choose the hypothesis whose posterior probability is greatest, and that Bayes theorem provides the proper method for weighting the contribution of this prior knowledge and observed data. Unfortunately, Bayes theorem implicitly assumes *perfect* knowledge about the probability distributions $P(h)$, $P(D)$, and $P(D|h)$. When these quantities are only imperfectly known, Bayes theorem alone does not prescribe how to combine them with the observed data. (One possible approach in such cases is to assume prior probability distributions over $P(h)$, $P(D)$, and $P(D|h)$ themselves, then calculate the expected value of the posterior $P(h|D)$. However, this requires additional knowledge about the priors over $P(h)$, $P(D)$, and $P(D|h)$, so it does not really solve the general problem.)

We will revisit the question of what we mean by "best" fit to the hypothesis and data as we examine specific algorithms. For now, we will simply say that the learning problem is to minimize some combined measure of the error of the hypothesis over the data and the domain theory.

12.2.2 Hypothesis Space Search

How can the domain theory and training data best be combined to constrain the search for an acceptable hypothesis? This remains an open question in machine learning. This chapter surveys a variety of approaches that have been proposed, many of which consist of extensions to inductive methods we have already studied (e.g., BACKPROPAGATION, FOIL).

One way to understand the range of possible approaches is to return to our view of learning as a task of searching through the space of alternative hypotheses. We can characterize most learning methods as search algorithms by describing the hypothesis space H they search, the initial hypothesis h_0 at which they begin their search, the set of search operators O that define individual search steps, and the goal criterion G that specifies the search objective. In this chapter we explore three different methods for using prior knowledge to alter the search performed by purely inductive methods.

- *Use prior knowledge to derive an initial hypothesis from which to begin the search.* In this approach the domain theory B is used to construct an initial hypothesis h_0 that is consistent with B. A standard inductive method is then applied, starting with the initial hypothesis h_0. For example, the KBANN system described below learns artificial neural networks in this way. It uses prior knowledge to design the interconnections and weights for an initial network, so that this initial network is perfectly consistent with the given domain theory. This initial network hypothesis is then refined inductively using the BACKPROPAGATION algorithm and available data. Beginning the search at a hypothesis consistent with the domain theory makes it more likely that the final output hypothesis will better fit this theory.
- *Use prior knowledge to alter the objective of the hypothesis space search.* In this approach, the goal criterion G is modified to require that the output hypothesis fits the domain theory as well as the training examples. For example, the EBNN system described below learns neural networks in this way. Whereas inductive learning of neural networks performs gradient descent search to minimize the squared error of the network over the training data, EBNN performs gradient descent to optimize a different criterion. This modified criterion includes an additional term that measures the error of the learned network relative to the domain theory.
- *Use prior knowledge to alter the available search steps.* In this approach, the set of search operators O is altered by the domain theory. For example, the FOCL system described below learns sets of Horn clauses in this way. It is based on the inductive system FOIL, which conducts a greedy search through the space of possible Horn clauses, at each step revising its current hypothesis by adding a single new literal. FOCL uses the domain theory to expand the set of alternatives available when revising the hypothesis, allowing the

addition of multiple literals in a single search step when warranted by the domain theory. In this way, FOCL allows single-step moves through the hypothesis space that would correspond to many steps using the original inductive algorithm. These "macro-moves" can dramatically alter the course of the search, so that the final hypothesis found consistent with the data is different from the one that would be found using only the inductive search steps.

The following sections describe each of these approaches in turn.

12.3 USING PRIOR KNOWLEDGE TO INITIALIZE THE HYPOTHESIS

One approach to using prior knowledge is to initialize the hypothesis to perfectly fit the domain theory, then inductively refine this initial hypothesis as needed to fit the training data. This approach is used by the KBANN (Knowledge-Based Artificial Neural Network) algorithm to learn artificial neural networks. In KBANN an initial network is first constructed so that for every possible instance, the classification assigned by the network is identical to that assigned by the domain theory. The BACKPROPAGATION algorithm is then employed to adjust the weights of this initial network as needed to fit the training examples.

It is easy to see the motivation for this technique: if the domain theory is correct, the initial hypothesis will correctly classify all the training examples and there will be no need to revise it. However, if the initial hypothesis is found to imperfectly classify the training examples, then it will be refined inductively to improve its fit to the training examples. Recall that in the purely inductive BACKPROPAGATION algorithm, weights are typically initialized to small random values. The intuition behind KBANN is that even if the domain theory is only approximately correct, initializing the network to fit this domain theory will give a better starting approximation to the target function than initializing the network to random initial weights. This should lead, in turn, to better generalization accuracy for the final hypothesis.

This *initialize-the-hypothesis* approach to using the domain theory has been explored by several researchers, including Shavlik and Towell (1989), Towell and Shavlik (1994), Fu (1989, 1993), and Pratt (1993a, 1993b). We will use the KBANN algorithm described in Shavlik and Towell (1989) to illustrate this approach.

12.3.1 The KBANN Algorithm

The KBANN algorithm exemplifies the initialize-the-hypothesis approach to using domain theories. It assumes a domain theory represented by a set of propositional, nonrecursive Horn clauses. A Horn clause is propositional if it contains no variables. The input and output of KBANN are as follows:

KBANN(*Domain_Theory*, *Training_Examples*)
Domain_Theory: Set of propositional, nonrecursive Horn clauses.
Training_Examples: Set of ⟨*input output*⟩ *pairs of the target function.*

Analytical step: Create an initial network equivalent to the domain theory.

1. For each instance attribute create a network input.
2. For each Horn clause in the *Domain_Theory*, create a network unit as follows:
   - Connect the inputs of this unit to the attributes tested by the clause antecedents.
   - For each non-negated antecedent of the clause, assign a weight of W to the corresponding sigmoid unit input.
   - For each negated antecedent of the clause, assign a weight of $-W$ to the corresponding sigmoid unit input.
   - Set the threshold weight w_0 for this unit to $-(n - .5)W$, where n is the number of non-negated antecedents of the clause.
3. Add additional connections among the network units, connecting each network unit at depth i from the input layer to all network units at depth $i + 1$. Assign random near-zero weights to these additional connections.

Inductive step: Refine the initial network.

4. Apply the BACKPROPAGATION algorithm to adjust the initial network weights to fit the *Training_Examples*.

TABLE 12.2
The KBANN algorithm. The domain theory is translated into an equivalent neural network (steps 1–3), which is inductively refined using the BACKPROPAGATION algorithm (step 4). A typical value for the constant W is 4.0.

Given:

- A set of training examples
- A domain theory consisting of nonrecursive, propositional Horn clauses

Determine:

- An artificial neural network that fits the training examples, biased by the domain theory

The two stages of the KBANN algorithm are first to create an artificial neural network that perfectly fits the domain theory and second to use the BACKPROPAGATION algorithm to refine this initial network to fit the training examples. The details of this algorithm, including the algorithm for creating the initial network, are given in Table 12.2 and illustrated in Section 12.3.2.

12.3.2 An Illustrative Example

To illustrate the operation of KBANN, consider the simple learning problem summarized in Table 12.3, adapted from Towell and Shavlik (1989). Here each instance describes a physical object in terms of the material from which it is made, whether it is light, etc. The task is to learn the target concept *Cup* defined over such physical objects. Table 12.3 describes a set of training examples and a domain theory for the *Cup* target concept. Notice the domain theory defines a *Cup*

Domain theory:

$$Cup \leftarrow Stable, Liftable, OpenVessel$$
$$Stable \leftarrow BottomIsFlat$$
$$Liftable \leftarrow Graspable, Light$$
$$Graspable \leftarrow HasHandle$$
$$OpenVessel \leftarrow HasConcavity, ConcavityPointsUp$$

Training examples:

	Cups				*Non-Cups*					
BottomIsFlat	√	√	√	√	√	√	√			√
ConcavityPointsUp	√	√	√	√	√		√	√		
Expensive	√		√				√		√	
Fragile	√	√			√	√		√		√
HandleOnTop					√		√			
HandleOnSide	√			√					√	
HasConcavity	√	√	√	√	√		√	√	√	√
HasHandle	√			√	√		√		√	
Light	√	√	√	√	√	√	√		√	
MadeOfCeramic	√				√		√	√		
MadeOfPaper				√					√	
MadeOfStyrofoam		√	√			√				√

TABLE 12.3
The *Cup* learning task. An approximate domain theory and a set of training examples for the target concept *Cup*.

as an object that is *Stable*, *Liftable*, and an *OpenVessel*. The domain theory also defines each of these three attributes in terms of more primitive attributes, terminating in the primitive, operational attributes that describe the instances. Note the domain theory is not perfectly consistent with the training examples. For example, the domain theory fails to classify the second and third training examples as positive examples. Nevertheless, the domain theory forms a useful approximation to the target concept. KBANN uses the domain theory and training examples together to learn the target concept more accurately than it could from either alone.

In the first stage of the KBANN algorithm (steps 1–3 in the algorithm), an initial network is constructed that is consistent with the domain theory. For example, the network constructed from the *Cup* domain theory is shown in Figure 12.2. In general the network is constructed by creating a sigmoid threshold unit for each Horn clause in the domain theory. KBANN follows the convention that a sigmoid output value greater than 0.5 is interpreted as *True* and a value below 0.5 as *False*. Each unit is therefore constructed so that its output will be greater than 0.5 just in those cases where the corresponding Horn clause applies. For each antecedent to the Horn clause, an input is created to the corresponding sigmoid unit. The weights of the sigmoid unit are then set so that it computes the logical AND of its inputs. In particular, for each input corresponding to a non-negated antecedent,

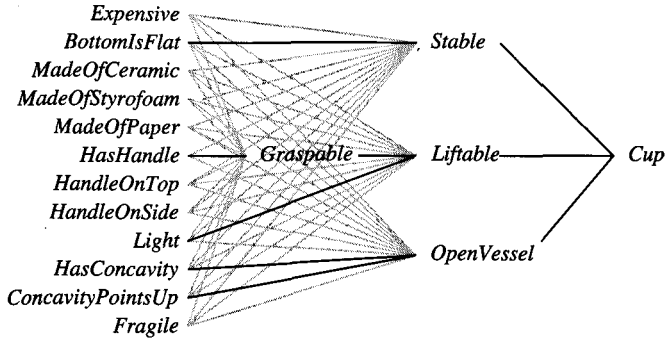

FIGURE 12.2
A neural network equivalent to the domain theory. This network, created in the first stage of the KBANN algorithm, produces output classifications identical to those of the given domain theory clauses. Dark lines indicate connections with weight W and correspond to antecedents of clauses from the domain theory. Light lines indicate connections with weights of approximately zero.

the weight is set to some positive constant W. For each input corresponding to a negated antecedent, the weight is set to $-W$. The threshold weight of the unit, w_0 is then set to $-(n-.5)W$, where n is the number of non-negated antecedents. When unit input values are 1 or 0, this assures that their weighted sum plus w_0 will be positive (and the sigmoid output will therefore be greater than 0.5) if and only if all clause antecedents are satisfied. Note for sigmoid units at the second and subsequent layers, unit inputs will not necessarily be 1 and 0 and the above argument may not apply. However, if a sufficiently large value is chosen for W, this KBANN algorithm can correctly encode the domain theory for arbitrarily deep networks. Towell and Shavlik (1994) report using $W = 4.0$ in many of their experiments.

Each sigmoid unit input is connected to the appropriate network input or to the output of another sigmoid unit, to mirror the graph of dependencies among the corresponding attributes in the domain theory. As a final step many additional inputs are added to each threshold unit, with their weights set approximately to zero. The role of these additional connections is to enable the network to inductively learn additional dependencies beyond those suggested by the given domain theory. The solid lines in the network of Figure 12.2 indicate unit inputs with weights of W, whereas the lightly shaded lines indicate connections with initial weights near zero. It is easy to verify that for sufficiently large values of W this network will output values identical to the predictions of the domain theory.

The second stage of KBANN (step 4 in the algorithm of Table 12.2) uses the training examples and the BACKPROPAGATION algorithm to refine the initial network weights. Of course if the domain theory and training examples contain no errors, the initial network will already fit the training data. In the *Cup* example, however, the domain theory and training data are inconsistent, and this step therefore alters the initial network weights. The resulting trained network is summarized in Figure 12.3, with dark solid lines indicating the largest positive weights, dashed lines indicating the largest negative weights, and light lines

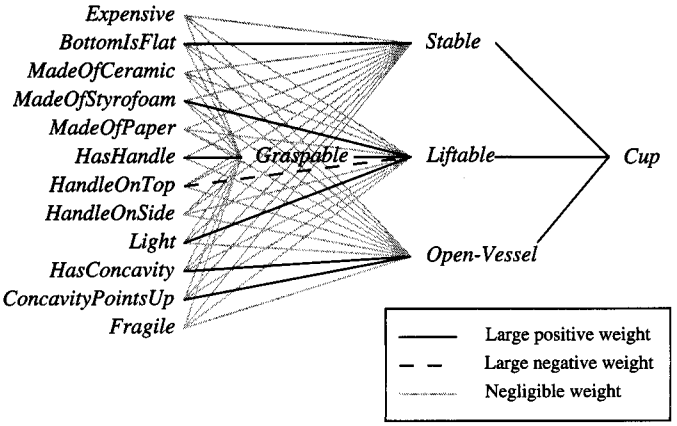

FIGURE 12.3
Result of inductively refining the initial network. KBANN uses the training examples to modify the network weights derived from the domain theory. Notice the new dependency of *Liftable* on *MadeOfStyrofoam* and *HandleOnTop*.

indicating negligible weights. Although the initial network misclassifies several training examples from Table 12.3, the refined network of Figure 12.3 perfectly classifies all of these training examples.

It is interesting to compare the final, inductively refined network weights to the initial weights derived from the domain theory. As can be seen in Figure 12.3, significant new dependencies were discovered during the inductive step, including the dependency of the *Liftable* unit on the feature *MadeOfStyrofoam*. It is important to keep in mind that while the unit labeled *Liftable* was initially defined by the given Horn clause for *Liftable*, the subsequent weight changes performed by BACKPROPAGATION may have dramatically changed the meaning of this hidden unit. After training of the network, this unit may take on a very different meaning unrelated to the initial notion of *Liftable*.

12.3.3 Remarks

To summarize, KBANN analytically creates a network equivalent to the given domain theory, then inductively refines this initial hypothesis to better fit the training data. In doing so, it modifies the network weights as needed to overcome inconsistencies between the domain theory and observed data.

The chief benefit of KBANN over purely inductive BACKPROPAGATION (beginning with random initial weights) is that it typically generalizes more accurately than BACKPROPAGATION when given an approximately correct domain theory, especially when training data is scarce. KBANN and other initialize-the-hypothesis approaches have been demonstrated to outperform purely inductive systems in several practical problems. For example, Towell et al. (1990) describe the application of KBANN to a molecular genetics problem. Here the task was to learn to

recognize DNA segments called promoter regions, which influence gene activity. In this experiment KBANN was given an initial domain theory obtained from a molecular geneticist, and a set of 53 positive and 53 negative training examples of promoter regions. Performance was evaluated using a leave-one-out strategy in which the system was run 106 different times. On each iteration KBANN was trained using 105 of the 106 examples and tested on the remaining example. The results of these 106 experiments were accumulated to provide an estimate of the true error rate. KBANN obtained an error rate of 4/106, compared to an error rate of 8/106 using standard BACKPROPAGATION. A variant of the KBANN approach was applied by Fu (1993), who reports an error rate of 2/106 on the same data. Thus, the impact of prior knowledge in these experiments was to reduce significantly the error rate. The training data for this experiment is available at World Wide Web site http://www.ics.uci.edu/~mlearn/MLRepository.html.

Both Fu (1993) and Towell et al. (1990) report that Horn clauses extracted from the final trained network provided a refined domain theory that better fit the observed data. Although it is sometimes possible to map from the learned network weights back to a refined set of Horn clauses, in the general case this is problematic because some weight settings have no direct Horn clause analog. Craven and Shavlik (1994) and Craven (1996) describe alternative methods for extracting symbolic rules from learned networks.

To understand the significance of KBANN it is useful to consider how its hypothesis search differs from that of the purely inductive BACKPROPAGATION algorithm. The hypothesis space search conducted by both algorithms is depicted schematically in Figure 12.4. As shown there, the key difference is the initial hypothesis from which weight tuning is performed. In the case that multiple hypotheses (weight vectors) can be found that fit the data—a condition that will be especially likely when training data is scarce—KBANN is likely to converge to a hypothesis that generalizes beyond the data in a way that is more similar to the domain theory predictions. On the other hand, the particular hypothesis to which BACKPROPAGATION converges will more likely be a hypothesis with small weights, corresponding roughly to a generalization bias of smoothly interpolating between training examples. In brief, KBANN uses a domain-specific theory to bias generalization, whereas BACKPROPAGATION uses a domain-independent syntactic bias toward small weight values. Note in this summary we have ignored the effect of local minima on the search.

Limitations of KBANN include the fact that it can accommodate only propositional domain theories; that is, collections of variable-free Horn clauses. It is also possible for KBANN to be misled when given highly inaccurate domain theories, so that its generalization accuracy can deteriorate below the level of BACKPROPAGATION. Nevertheless, it and related algorithms have been shown to be useful for several practical problems.

KBANN illustrates the initialize-the-hypothesis approach to combining analytical and inductive learning. Other examples of this approach include Fu (1993); Gallant (1988); Bradshaw et al. (1989); Yang and Bhargava (1990); Lacher et al. (1991). These approaches vary in the exact technique for constructing the initial

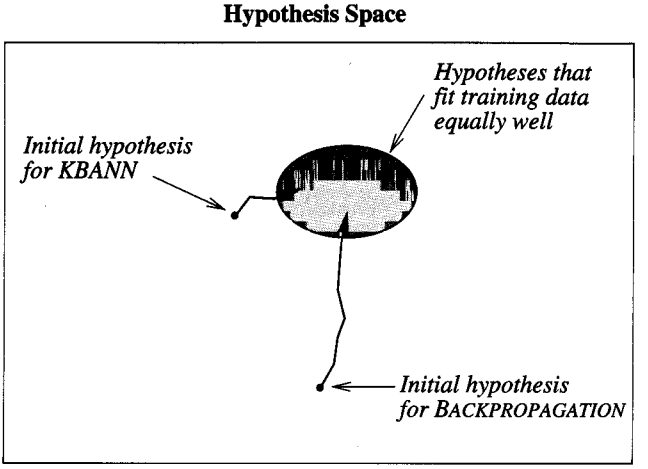

FIGURE 12.4
Hypothesis space search in KBANN. KBANN initializes the network to fit the domain theory, whereas BACKPROPAGATION initializes the network to small random weights. Both then refine the weights iteratively using the same gradient descent rule. When multiple hypotheses can be found that fit the training data (shaded region), KBANN and BACKPROPAGATION are likely to find different hypotheses due to their different starting points.

network, the application of BACKPROPAGATION to weight tuning, and in methods for extracting symbolic descriptions from the refined network. Pratt (1993a, 1993b) describes an initialize-the-hypothesis approach in which the prior knowledge is provided by a previously learned neural network for a related task, rather than a manually provided symbolic domain theory. Methods for training the values of Bayesian belief networks, as discussed in Section 6.11, can also be viewed as using prior knowledge to initialize the hypothesis. Here the prior knowledge corresponds to a set of conditional independence assumptions that determine the graph structure of the Bayes net, whose conditional probability tables are then induced from the training data.

12.4 USING PRIOR KNOWLEDGE TO ALTER THE SEARCH OBJECTIVE

The above approach begins the gradient descent search with a hypothesis that perfectly fits the domain theory, then perturbs this hypothesis as needed to maximize the fit to the training data. An alternative way of using prior knowledge is to incorporate it into the error criterion minimized by gradient descent, so that the network must fit a combined function of the training data and domain theory. In this section, we consider using prior knowledge in this fashion. In particular, we consider prior knowledge in the form of known derivatives of the target function. Certain types of prior knowledge can be expressed quite naturally in this form. For example, in training a neural network to recognize handwritten characters we