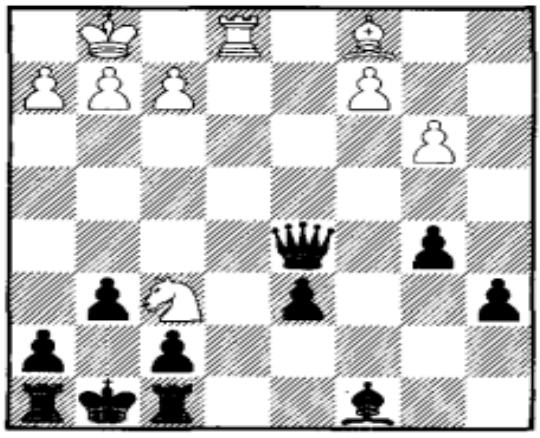
UNIT-5

Analytical Learning-1

Introduction

Previous chapters have considered a variety of inductive learning methods: that is, methods that generalize from observed training examples by identifying features that empirically distinguish positive from negative training examples. Decision tree learning, neural network learning, inductive logic programming, and genetic algorithms are all examples of inductive methods that operate in this fashion. The key practical limit on these inductive learners is that they perform poorly when insufficient data is available. In fact, as discussed in Chapter 7, theoretical analysis shows that there are fundamental bounds on the accuracy that can be achieved when learning inductively from a given number of training examples.

Can we develop learning methods that are not subject to these fundamental bounds on learning accuracy imposed by the amount of training data available? Yes, if we are willing to reconsider the formulation of the learning problem itself. One way is to develop learning algorithms that accept explicit prior knowledge as an input, in addition to the input training data. Explanation-based learning is one such approach. It uses prior knowledge to analyze, or explain, each training example in order to infer which example features are relevant to the target function and which are irrelevant. These explanations enable it to generalize more accurately than inductive systems that rely on the data alone. As we saw in the previous chapter, inductive logic programming systems such as CIGOL also use prior background knowledge to guide learning. However, they use their background knowledge to infer features that augment the input descriptions of instances, thereby increasing the complexity of the hypothesis space to be searched. In contrast, explanationbased learning uses prior knowledge to reduce the complexity of the hypothesis space to be searched, thereby reducing sample complexity and improving generalization accuracy of the learner. To capture the intuition underlying explanation-based learning, consider the task of learning to play chess. In particular, suppose we would like our chess program to learn to recognize important classes of game positions, such as the target concept "chessboard positions in which black will lose its queen within two moves." shows a positive training example of this target concept. Inductive learning methods could, of course, be employed to learn this target concept. However, because the chessboard is fairly complex (there are 32 pieces that may be on any of 64 squares), and because the particular patterns that capture this concept are fairly subtle (involving the relative positions of various pieces on the board), we would have to provide thousands of training examples similar to the one in to expect an inductively learned hypothesis to generalize correctly to new situations.



A positive example of the target concept "chess positions in which black will lose its queen within two moves." Note the white knight is simultaneously attacking both the black king and queen. Black must therefore move its king, enabling white to capture its queen.

Inductive and Analytical Learning Problems The essential difference between analytical and inductive learning methods is that they assume two different formulations of the learning problem: 0 In inductive learning, the learner is given a hypothesis space H from which it must select an output hypothesis, and a set of training examples ,

D = {(xl, f (x~)), . . . (x,, f (x,))} where f (xi) is the target value for the instance xi. The desired output of the learner is a hypothesis h from H that is consistent with these training examples. 0 In analytical learning, the input to the learner includes the same hypothesis space H and training examples D as for inductive learning. In addition, the learner is provided an additional input: A domain theory B consisting of background knowledge that can be used to explain observed training examples. The desired output of ,the learner is a hypothesis h from H that is consistent with both the training examples D and the domain theory B. To illustrate, in our chess example each instance xi would describe a particular chess position, and f (xi) would be True when xi is a position for which black will lose its queen within two moves, and False otherwise. We might define the hypothesis space H to consist of sets of Horn clauses (if-then rules) as in Chapter 10, where the predicates used by the rules refer to the positions or relative positions of specific pieces on the board. The domain theory B would consist of a formalization of the rules of chess, describing the legal moves, the fact

that players must take turns, and the fact that the game is won when one player captures her opponent's king. Note in analytical learning, the learner must output a hypothesis that is consistent with both the training data and the domain theory. We say that hypothesis h is consistent with domain theory B provided B does not entail the negation of h (i.e., B -h). This additional constraint that the output hypothesis must be consistent with B reduces the ambiguity faced by the learner when the data alone cannot resolve among all hypotheses in H. The net effect, provided the domain theory is correct, is to increase the accuracy of the output hypothesis. Let us introduce in detail a second example of an analytical learning problem--one that we will use for illustration throughout this chapter. Consider an instance space X in which each instance is a pair of physical objects. Each of the two physical objects in the instance is described by the predicates Color, Volume, Owner, Material, Type, and Density, and

the relationship between the two objects is described by the predicate On. Given this instance space, the task is to learn the target concept "pairs of physical objects, such that one can be stacked safely on the other," denoted by the predicate SafeToStack(x,y). Learning this target concept might be useful, for example, to a robot system that has the task of storing various physical objects within a limited workspace.

LEARNING WITH PERFECT DOMAIN THEORIES: PROLOG-EBG

As stated earlier, in this chapter we consider explanation-based learning from domain theories that are perfect, that is, domain theories that are correct and complete. A domain theory is said to be correct if each of its assertions is a truthful statement about the world. A domain theory is said to be complete with respect to a given target concept and instance space, if the domain theory covers every positive example in the instance space. Put another way, it is complete if every instance that satisfies the target concept can be proven by the domain theory to satisfy it. Notice our definition of completeness does not require that the domain theory be able to prove that negative examples do not satisfy the target concept. However, if we follow the usual PROLOG convention that unprovable assertions are assumed to be false, then this definition of completeness includes full coverage of both positive and negative examples by the domain theory. The reader may well ask at this point whether it is reasonable to assume that such perfect domain theories are available to the learner. After all, if the learner had a perfect domain theory, why would it need to learn? There are two responses to this question.

First, there are cases in which it is feasible to provide a perfect domain theory. Our earlier chess problem provides one such case, in which the legal moves of chess form a perfect domain theory from which the optimal chess playing

strategy can (in principle) be inferred. Furthermore, although it is quite easy to write down the legal moves of chess that constitute this domain theory, it is extremely difficult to write down the optimal chess-playing strategy. In such cases, we prefer to provide the domain theory to the learner and rely on the learner to formulate a useful description of the target concept (e.g., "board states in which I am about to lose my queen") by examining and generalizing from specific training examples. Section 11.4 describes the successful application of explanation-based learning with perfect domain theories to automatically improve performance at several search-intensive planning and optimization problems.

Second, in many other cases it is unreasonable to assume that a perfect domain theory is available. It is difficult to write a perfectly correct and complete theory even for our relatively simple SafeToStack problem. A more realistic assumption is that plausible explanations based on imperfect domain theories must be used, rather than exact proofs based on perfect knowledge. Nevertheless, we can begin to understand the role of explanations in learning by considering the ideal case of perfect domain theories. In Chapter 12 we will consider learning from imperfect domain theories.

REMARKS ON EXPLANATION-BASED LEARNING

As we saw in the above example, PROLOG-EBG conducts a detailed analysis of individual training examples to determine how best to generalize from the specific example to a general Horn clause hypothesis. The following are the key properties of this algorithm. 0 Unlike inductive methods,

PROLOG-EBG produces justified general hypotheses by using prior knowledge to analyze individual examples. 0 The explanation of how the example satisfies the target concept determines which example attributes are relevant: those mentioned by the explanation. 0 The further analysis of the explanation, regressing the target concept to determine its weakest preimage with respect to the explanation, allows deriving more general constraints on the values of the relevant features. 0 Each learned Horn clause corresponds to a sufficient condition for satisfying the target concept. The set of learned Horn clauses covers the positive training examples encountered by the learner, as well as other instances that share the same explanations. 0 The generality of the learned Horn clauses will depend on the formulation of the domain theory and on the sequence in which training examples are considered. 0 PROLOG-EBG implicitly assumes that the domain theory is correct and complete. If the domain theory is incorrect or incomplete, the resulting learned concept may also be incorrect.

EXPLANATION-BASED LEARNING OF SEARCH CONTROL KNOWLEDGE

As noted above, the practical applicability of the PROLOG-EBG algorithm is restricted by its requirement that the domain theory be correct and complete. One important class of learning problems where this requirement is easily satisfied is learning to speed up complex search programs. In fact, the largest scale attempts to apply explanation-based learning have addressed the problem of learning to control search, or what is sometimes called "speedup" learning. For example, playing games such as chess involves searching through a vast space of possible moves and board positions to

find the best move. Many practical scheduling and optimization problems are easily formulated as large search problems, in which the task is to find some move toward the goal state. In such problems the definitions of the legal search operators, together with the definition of the search objective, provide a complete and correct domain theory for learning search control knowledge Exactly how should we formulate the problem of learning search control so that we can apply explanation-based learning? Consider a general search problem where S is the set of possible search states, 0 is a set of legal search operators that transform one search state into another, and G is a predicate defined over S that indicates which states are goal states. The problem in general is to find a sequence of operators that will transform an arbitrary initial state si to some final state sf that satisfies the goal predicate G. One way to formulate the learning problem is to have our system learn a separate target concept for each of the operators in 0. In particular, for each operator o in 0 it might attempt to learn the target concept "the set of states for which o leads toward a goal state." Of course the exact choice of which target concepts to learn depends on the internal structure of problem solver that must

use this learned knowledge. For example, if the problem solver is a means-ends planning system that works by establishing and solving subgoals, then we might instead wish to learn target concepts such as "the set of planning states in which subgoals of type A should be solved before subgoals of type B." One system that employs explanation-based learning to improve its search is PRODIGY (Carbonell et al. 1990). PRODIGY is a domain-independent planning system that accepts the definition of a problem domain in terms of the state space S and operators 0. It then solves problems of the form "find a sequence of operators that leads from initial state si to a state that satisfies goal predicate G." PRODIGY uses a means-ends planner that decomposes problems into subgoals, solves them, then combines their solutions into a solution for the full problem. Thus, during its search for problem

solutions PRODIGY repeatedly faces questions such as "Which subgoal should be solved next?'and "Which operator should be considered for solving this subgoal?' Minton (1988) describes the integration of explanation-based learning into PRODIGY by defining a set of target concepts appropriate for these kinds of control decisions that it repeatedly confronts. For example, one target concept is "the set of states in which subgoal A should be solved before subgoal B." An example of a rule learned by PRODIGY for this target concept in a simple block-stacking problem domain is IF One subgoal to be solved is On@, y), and One subgoal to be solved is On(y, z) THEN Solve the subgoal On(y, z) before On(x, y) To understand this rule, consider again the simple block stacking problem illustrated in Figure 9.3. In the problem illustrated by that figure, the goal is to stack the blocks so that they spell the word "universal." PRODIGY would decompose this problem into several subgoals to be achieved, including On(U, N), On(N, I), etc. Notice the above rule matches the subgoals On(U, N) and On(N, I), and recommends solving the subproblem On(N, I) before solving On(U, N). The justification for this rule (and the explanation used by PRODIGY to learn the rule) is that if we solve the subgoals in the reverse sequence, we will encounter a conflict in which we must undo the solution to the On(U, N) subgoal in order to achieve the other subgoal On(N, I). PRODIGY learns by first encountering such a conflict, then explaining to itself the reason for this conflict and creating a rule such as the one above. The net effect is that PRODIGY uses domain-independent knowledge about possible subgoal conflicts, together with domain-specific knowledge of specific operators (e.g., the fact that the robot can pick up only one block at a time), to learn useful domain-specific planning rules such as the one illustrated above.

Combining Inductive and Analytical Learning

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 justiJications that can be given for their learned hypotheses. Hypotheses output by purely analytical learning methods such as PROLOGEBG 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.

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.

INDUCTIVE-ANALYTICAL APPROACHES TO LEARNING

The Learning Problem To summarize, the learning problem considered in this chapter is Given: 0 A set of training examples D, possibly containing errors 0 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 errorD(h) is defined to be the proportion of examples from D that are misclassified by h. Let us define the error error~(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 argmin kDerrorD (h) + kBerrorB (h) h€H While this appears reasonable at first glance, it is not clear what values to assign to k~ and kg 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~(h) more heavily. Given a strong theory and a small sample of very noisy data, the best results would be obtained by weighting errorB(h) more heavily. Of course if the learner does not know in advance the quality of the domain theor 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(h1D) 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(Dlh). Thus we can think of P(h), P(D), and P(Dlh) 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(hlD) 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 pe$ect knowledge about the probability distributions P(h), P(D), and P(Dlh). 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(D1h) themselves, then calculate the expected value of the posterior P (h 1 D) . However, this requires additional knowledge about the priors over P(h), P(D), and P(Dlh), 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. theory.

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 Towel1 (1989), Towel1 and Shavlik (1994), Fu (1989, 1993), and Pratt (1993a, 1993b). We will use the KBANN algorithm described in Shavlik and Towel1 (1989) to illustrate this approach