CHAPTER

2

CONCEPT LEARNING AND THE GENERAL-TO-SPECIFIC ORDERING

The problem of inducing general functions from specific training examples is central to learning. This chapter considers concept learning: acquiring the definition of a general category given a sample of positive and negative training examples of the category. Concept learning can be formulated as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples. In many cases this search can be efficiently organized by taking advantage of a naturally occurring structure over the hypothesis space—a general-to-specific ordering of hypotheses. This chapter presents several learning algorithms and considers situations under which they converge to the correct hypothesis. We also examine the nature of inductive learning and the justification by which any program may successfully generalize beyond the observed training data.

2.1 INTRODUCTION

Much of learning involves acquiring general concepts from specific training examples. People, for example, continually learn general concepts or categories such as "bird," "car," "situations in which I should study more in order to pass the exam," etc. Each such concept can be viewed as describing some subset of objects or events defined over a larger set (e.g., the subset of animals that constitute birds). Alternatively, each concept can be thought of as a boolean-valued function defined over this larger set (e.g., a function defined over all animals, whose value is true for birds and false for other animals).

In this chapter we consider the problem of automatically inferring the general definition of some concept, given examples labeled as members or nonmembers of the concept. This task is commonly referred to as *concept learning*, or approximating a boolean-valued function from examples.

> **Concept learning.** Inferring a boolean-valued function from training examples of its input and output.

2.2 A CONCEPT LEARNING TASK

To ground our discussion of concept learning, consider the example task of learning the target concept "days on which my friend Aldo enjoys his favorite water sport." Table 2.1 describes a set of example days, each represented by a set of *attributes*. The attribute *EnjoySport* indicates whether or not Aldo enjoys his favorite water sport on this day. The task is to learn to predict the value of *EnjoySport* for an arbitrary day, based on the values of its other attributes.

What hypothesis representation shall we provide to the learner in this case? Let us begin by considering a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes. In particular, let each hypothesis be a vector of six constraints, specifying the values of the six attributes *Sky, AirTemp, Humidity, Wind, Water,* and *Forecast*. For each attribute, the hypothesis will either

- indicate by a "?" that any value is acceptable for this attribute,
- specify a single required value (e.g., *Warm*) for the attribute, or
- indicate by a "Ø" that no value is acceptable.

If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a positive example ($h(x) = 1$). To illustrate, the hypothesis that Aldo enjoys his favorite sport only on cold days with high humidity (independent of the values of the other attributes) is represented by the expression

$$\langle ?, Cold, High, ?, ?, ? \rangle$$

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

The most general hypothesis—that every day is a positive example—is represented by

$$\langle ?, ?, ?, ?, ?, ? \rangle$$

and the most specific possible hypothesis—that *no* day is a positive example—is represented by

$$\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$$

To summarize, the *EnjoySport* concept learning task requires learning the set of days for which $EnjoySport = yes$, describing this set by a conjunction of constraints over the instance attributes. In general, any concept learning task can be described by the set of instances over which the target function is defined, the target function, the set of candidate hypotheses considered by the learner, and the set of available training examples. The definition of the *EnjoySport* concept learning task in this general form is given in Table 2.2.

2.2.1 Notation

Throughout this book, we employ the following terminology when discussing concept learning problems. The set of items over which the concept is defined is called the set of *instances*, which we denote by X. In the current example, X is the set of all possible days, each represented by the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The concept or function to be learned is called the *target concept*, which we denote by c. In general, c can be any boolean-valued function defined over the instances X; that is, $c : X \rightarrow \{0, 1\}$. In the current example, the target concept corresponds to the value of the attribute *EnjoySport* (i.e., $c(x) = 1$ if $EnjoySport = Yes$, and $c(x) = 0$ if $EnjoySport = No$).

- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "$\emptyset$" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

TABLE 2.2
The *EnjoySport* concept learning task.

When learning the target concept, the learner is presented a set of *training examples*, each consisting of an instance x from X, along with its target concept value $c(x)$ (e.g., the training examples in Table 2.1). Instances for which $c(x) = 1$ are called *positive examples*, or members of the target concept. Instances for which $c(x) = 0$ are called *negative examples*, or nonmembers of the target concept. We will often write the ordered pair $\langle x, c(x) \rangle$ to describe the training example consisting of the instance x and its target concept value $c(x)$. We use the symbol D to denote the set of available training examples.

Given a set of training examples of the target concept c, the problem faced by the learner is to hypothesize, or estimate, c. We use the symbol H to denote the set of *all possible hypotheses* that the learner may consider regarding the identity of the target concept. Usually H is determined by the human designer's choice of hypothesis representation. In general, each hypothesis h in H represents a boolean-valued function defined over X; that is, $h : X \rightarrow \{0, 1\}$. The goal of the learner is to find a hypothesis h such that $h(x) = c(x)$ for all x in X.

2.2.2 The Inductive Learning Hypothesis

Notice that although the learning task is to determine a hypothesis h identical to the target concept c over the entire set of instances X, the only information available about c is its value over the training examples. Therefore, inductive learning algorithms can at best guarantee that the output hypothesis fits the target concept over the training data. Lacking any further information, our assumption is that the best hypothesis regarding unseen instances is the hypothesis that best fits the observed training data. This is the fundamental assumption of inductive learning, and we will have much more to say about it throughout this book. We state it here informally and will revisit and analyze this assumption more formally and more quantitatively in Chapters 5, 6, and 7.

> **The inductive learning hypothesis.** Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

2.3 CONCEPT LEARNING AS SEARCH

Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation. The goal of this search is to find the hypothesis that best fits the training examples. It is important to note that by selecting a hypothesis representation, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn. Consider, for example, the instances X and hypotheses H in the *EnjoySport* learning task. Given that the attribute *Sky* has three possible values, and that *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast* each have two possible values, the instance space X contains exactly

$3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 96$ distinct instances. A similar calculation shows that there are $5 \cdot 4 \cdot 4 \cdot 4 \cdot 4 \cdot 4 = 5120$ *syntactically distinct* hypotheses within H. Notice, however, that every hypothesis containing one or more "$\emptyset$" symbols represents the empty set of instances; that is, it classifies every instance as negative. Therefore, the number of *semantically distinct* hypotheses is only $1 + (4 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3) = 973$. Our *EnjoySport* example is a very simple learning task, with a relatively small, finite hypothesis space. Most practical learning tasks involve much larger, sometimes infinite, hypothesis spaces.

If we view learning as a search problem, then it is natural that our study of learning algorithms will examine different strategies for searching the hypothesis space. We will be particularly interested in algorithms capable of efficiently searching very large or infinite hypothesis spaces, to find the hypotheses that best fit the training data.

2.3.1 General-to-Specific Ordering of Hypotheses

Many algorithms for concept learning organize the search through the hypothesis space by relying on a very useful structure that exists for any concept learning problem: a general-to-specific ordering of hypotheses. By taking advantage of this naturally occurring structure over the hypothesis space, we can design learning algorithms that exhaustively search even infinite hypothesis spaces without explicitly enumerating every hypothesis. To illustrate the general-to-specific ordering, consider the two hypotheses

$$h_1 = \langle Sunny, ?, ?, Strong, ?, ? \rangle$$
$$h_2 = \langle Sunny, ?, ?, ?, ?, ? \rangle$$

Now consider the sets of instances that are classified positive by h_1 and by h_2. Because h_2 imposes fewer constraints on the instance, it classifies more instances as positive. In fact, any instance classified positive by h_1 will also be classified positive by h_2. Therefore, we say that h_2 is more general than h_1.

This intuitive "more general than" relationship between hypotheses can be defined more precisely as follows. First, for any instance x in X and hypothesis h in H, we say that x *satisfies* h if and only if $h(x) = 1$. We now define the *more_general_than_or_equal_to* relation in terms of the sets of instances that satisfy the two hypotheses: Given hypotheses h_j and h_k, h_j is *more_general_than_or_equal_to* h_k if and only if any instance that satisfies h_k also satisfies h_j.

> ***Definition***: Let h_j and h_k be boolean-valued functions defined over X. Then h_j is **more_general_than_or_equal_to** h_k (written $h_j \geq_g h_k$) if and only if
>
> $$(\forall x \in X)[(h_k(x) = 1) \rightarrow (h_j(x) = 1)]$$

We will also find it useful to consider cases where one hypothesis is strictly more general than the other. Therefore, we will say that h_j is (strictly) *more_general_than*

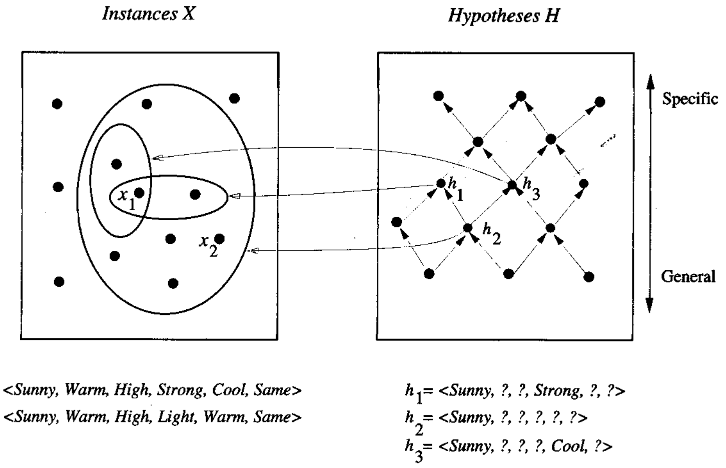

FIGURE 2.1
Instances, hypotheses, and the *more_general_than* relation. The box on the left represents the set X of all instances, the box on the right the set H of all hypotheses. Each hypothesis corresponds to some subset of X—the subset of instances that it classifies positive. The arrows connecting hypotheses represent the *more_general_than* relation, with the arrow pointing toward the less general hypothesis. Note the subset of instances characterized by h_2 subsumes the subset characterized by h_1, hence h_2 is *more_general_than* h_1.

h_k (written $h_j >_g h_k$) if and only if $(h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$. Finally, we will sometimes find the inverse useful and will say that h_j is *more_specific_than* h_k when h_k is *more_general_than* h_j.

To illustrate these definitions, consider the three hypotheses h_1, h_2, and h_3 from our *EnjoySport* example, shown in Figure 2.1. How are these three hypotheses related by the $\geq_g$ relation? As noted earlier, hypothesis h_2 is more general than h_1 because every instance that satisfies h_1 also satisfies h_2. Similarly, h_2 is more general than h_3. Note that neither h_1 nor h_3 is more general than the other; although the instances satisfied by these two hypotheses intersect, neither set subsumes the other. Notice also that the $\geq_g$ and $>_g$ relations are defined independent of the target concept. They depend only on which instances satisfy the two hypotheses and not on the classification of those instances according to the target concept. Formally, the $\geq_g$ relation defines a partial order over the hypothesis space H (the relation is reflexive, antisymmetric, and transitive). Informally, when we say the structure is a partial (as opposed to total) order, we mean there may be pairs of hypotheses such as h_1 and h_3, such that $h_1 \not\geq_g h_3$ and $h_3 \not\geq_g h_1$.

The $\geq_g$ relation is important because it provides a useful structure over the hypothesis space H for *any* concept learning problem. The following sections present concept learning algorithms that take advantage of this partial order to efficiently organize the search for hypotheses that fit the training data.

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   - For each attribute constraint a_i in h
     If the constraint a_i is satisfied by x
     Then do nothing
     Else replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

TABLE 2.3
FIND-S Algorithm.

2.4 FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

How can we use the *more_general_than* partial ordering to organize the search for a hypothesis consistent with the observed training examples? One way is to begin with the most specific possible hypothesis in H, then generalize this hypothesis each time it fails to cover an observed positive training example. (We say that a hypothesis "covers" a positive example if it correctly classifies the example as positive.) To be more precise about how the partial ordering is used, consider the FIND-S algorithm defined in Table 2.3.

To illustrate this algorithm, assume the learner is given the sequence of training examples from Table 2.1 for the *EnjoySport* task. The first step of FIND-S is to initialize h to the most specific hypothesis in H

$$h \leftarrow \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$$

Upon observing the first training example from Table 2.1, which happens to be a positive example, it becomes clear that our hypothesis is too specific. In particular, none of the "$\emptyset$" constraints in h are satisfied by this example, so each is replaced by the next more general constraint that fits the example; namely, the attribute values for this training example.

$$h \leftarrow \langle Sunny, Warm, Normal, Strong, Warm, Same \rangle$$

This h is still very specific; it asserts that all instances are negative except for the single positive training example we have observed. Next, the second training example (also positive in this case) forces the algorithm to further generalize h, this time substituting a "?" in place of any attribute value in h that is not satisfied by the new example. The refined hypothesis in this case is

$$h \leftarrow \langle Sunny, Warm, ?, Strong, Warm, Same \rangle$$

Upon encountering the third training example—in this case a negative example—the algorithm makes no change to h. In fact, the FIND-S algorithm simply *ignores every negative example*! While this may at first seem strange, notice that in the current case our hypothesis h is already consistent with the new negative example (i.e., h correctly classifies this example as negative), and hence no revision is needed. In the general case, as long as we assume that the hypothesis space H contains a hypothesis that describes the true target concept c and that the training data contains no errors, then the current hypothesis h can never require a revision in response to a negative example. To see why, recall that the current hypothesis h is the most specific hypothesis in H consistent with the observed positive examples. Because the target concept c is also assumed to be in H and to be consistent with the positive training examples, c must be *more_general_than_or_equal_to* h. But the target concept c will never cover a negative example, thus neither will h (by the definition of *more_general_than*). Therefore, no revision to h will be required in response to any negative example.

To complete our trace of FIND-S, the fourth (positive) example leads to a further generalization of h

$$h \leftarrow \langle Sunny, Warm, ?, Strong, ?, ? \rangle$$

The FIND-S algorithm illustrates one way in which the *more_general_than* partial ordering can be used to organize the search for an acceptable hypothesis. The search moves from hypothesis to hypothesis, searching from the most specific to progressively more general hypotheses along one chain of the partial ordering. Figure 2.2 illustrates this search in terms of the instance and hypothesis spaces. At each step, the hypothesis is generalized only as far as necessary to cover the new positive example. Therefore, at each stage the hypothesis is the most specific hypothesis consistent with the training examples observed up to this point (hence the name FIND-S). The literature on concept learning is

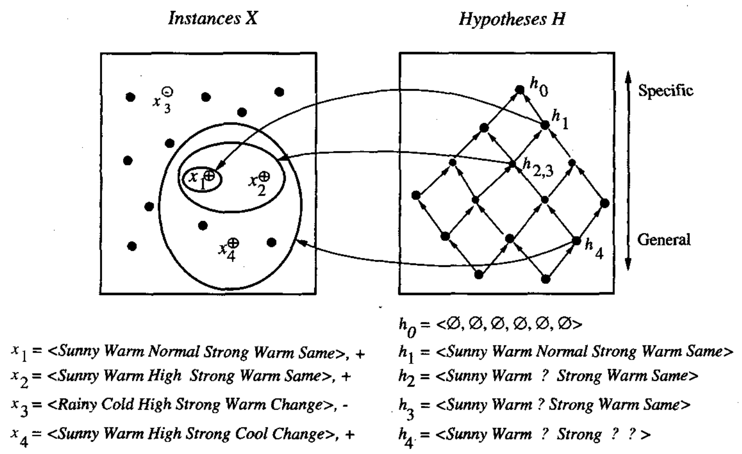

FIGURE 2.2
The hypothesis space search performed by FIND-S. The search begins (h_0) with the most specific hypothesis in H, then considers increasingly general hypotheses (h_1 through h_4) as mandated by the training examples. In the instance space diagram, positive training examples are denoted by "+," negative by "−," and instances that have not been presented as training examples are denoted by a solid circle.