CHAPTER 6

BAYESIAN LEARNING

Bayesian reasoning provides a probabilistic approach to inference. It is based on the assumption that the quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data. It is important to machine learning because it provides a quantitative approach to weighing the evidence supporting alternative hypotheses. Bayesian reasoning provides the basis for learning algorithms that directly manipulate probabilities, as well as a framework for analyzing the operation of other algorithms that do not explicitly manipulate probabilities.

6.1 INTRODUCTION

Bayesian learning methods are relevant to our study of machine learning for two different reasons. First, Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems. For example, Michie et al. (1994) provide a detailed study comparing the naive Bayes classifier to other learning algorithms, including decision tree and neural network algorithms. These researchers show that the naive Bayes classifier is competitive with these other learning algorithms in many cases and that in some cases it outperforms these other methods. In this chapter we describe the naive Bayes classifier and provide a detailed example of its use. In particular, we discuss its application to the problem of learning to classify text documents such as electronic news articles.

For such learning tasks, the naive Bayes classifier is among the most effective algorithms known.

The second reason that Bayesian methods are important to our study of machine learning is that they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities. For example, in this chapter we analyze algorithms such as the FIND-S and CANDIDATE-ELIMINATION algorithms of Chapter 2 to determine conditions under which they output the most probable hypothesis given the training data. We also use a Bayesian analysis to justify a key design choice in neural network learning algorithms: choosing to minimize the sum of squared errors when searching the space of possible neural networks. We also derive an alternative error function, cross entropy, that is more appropriate than sum of squared errors when learning target functions that predict probabilities. We use a Bayesian perspective to analyze the inductive bias of decision tree learning algorithms that favor short decision trees and examine the closely related Minimum Description Length principle. A basic familiarity with Bayesian methods is important to understanding and characterizing the operation of many algorithms in machine learning.

Features of Bayesian learning methods include:

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting (1) a prior probability for each candidate hypothesis, and (2) a probability distribution over observed data for each possible hypothesis.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
- Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions. A second practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses). In certain specialized situations, this computational cost can be significantly reduced.

The remainder of this chapter is organized as follows. Section 6.2 introduces Bayes theorem and defines maximum likelihood and maximum a posteriori probability hypotheses. The four subsequent sections then apply this probabilistic framework to analyze several issues and learning algorithms discussed in earlier chapters. For example, we show that several previously described algorithms output maximum likelihood hypotheses, under certain assumptions. The remaining sections then introduce a number of learning algorithms that explicitly manipulate probabilities. These include the Bayes optimal classifier, Gibbs algorithm, and naive Bayes classifier. Finally, we discuss Bayesian belief networks, a relatively recent approach to learning based on probabilistic reasoning, and the EM algorithm, a widely used algorithm for learning in the presence of unobserved variables.

6.2 BAYES THEOREM

In machine learning we are often interested in determining the best hypothesis from some space H, given the observed training data D. One way to specify what we mean by the *best* hypothesis is to say that we demand the *most probable* hypothesis, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H. Bayes theorem provides a direct method for calculating such probabilities. More precisely, Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

To define Bayes theorem precisely, let us first introduce a little notation. We shall write $P(h)$ to denote the initial probability that hypothesis h holds, before we have observed the training data. $P(h)$ is often called the *prior probability* of h and may reflect any background knowledge we have about the chance that h is a correct hypothesis. If we have no such prior knowledge, then we might simply assign the same prior probability to each candidate hypothesis. Similarly, we will write $P(D)$ to denote the prior probability that training data D will be observed (i.e., the probability of D given no knowledge about which hypothesis holds). Next, we will write $P(D|h)$ to denote the probability of observing data D given some world in which hypothesis h holds. More generally, we write $P(x|y)$ to denote the probability of x given y. In machine learning problems we are interested in the probability $P(h|D)$ that h holds given the observed training data D. $P(h|D)$ is called the *posterior probability* of h, because it reflects our confidence that h holds after we have seen the training data D. Notice the posterior probability $P(h|D)$ reflects the influence of the training data D, in contrast to the prior probability $P(h)$, which is independent of D.

Bayes theorem is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability $P(h|D)$, from the prior probability $P(h)$, together with $P(D)$ and $P(D|h)$.

Bayes theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{6.1}$$

As one might intuitively expect, $P(h|D)$ increases with $P(h)$ and with $P(D|h)$ according to Bayes theorem. It is also reasonable to see that $P(h|D)$ decreases as $P(D)$ increases, because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h.

In many learning scenarios, the learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis $h \in H$ given the observed data D (or at least one of the maximally probable if there are several). Any such maximally probable hypothesis is called a *maximum a posteriori* (MAP) hypothesis. We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis. More precisely, we will say that h_{MAP} is a MAP hypothesis provided

$$\begin{aligned} h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}}\, P(h|D) \\ &= \underset{h \in H}{\operatorname{argmax}}\, \frac{P(D|h)P(h)}{P(D)} \\ &= \underset{h \in H}{\operatorname{argmax}}\, P(D|h)P(h) \qquad (6.2) \end{aligned}$$

Notice in the final step above we dropped the term $P(D)$ because it is a constant independent of h.

In some cases, we will assume that every hypothesis in H is equally probable a priori ($P(h_i) = P(h_j)$ for all h_i and h_j in H). In this case we can further simplify Equation (6.2) and need only consider the term $P(D|h)$ to find the most probable hypothesis. $P(D|h)$ is often called the *likelihood* of the data D given h, and any hypothesis that maximizes $P(D|h)$ is called a *maximum likelihood* (ML) hypothesis, h_{ML}.

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}}\, P(D|h) \qquad (6.3)$$

In order to make clear the connection to machine learning problems, we introduced Bayes theorem above by referring to the data D as training examples of some target function and referring to H as the space of candidate target functions. In fact, Bayes theorem is much more general than suggested by this discussion. It can be applied equally well to any set H of mutually exclusive propositions whose probabilities sum to one (e.g., "the sky is blue," and "the sky is not blue"). In this chapter, we will at times consider cases where H is a hypothesis space containing possible target functions and the data D are training examples. At other times we will consider cases where H is some other set of mutually exclusive propositions, and D is some other kind of data.

6.2.1 An Example

To illustrate Bayes rule, consider a medical diagnosis problem in which there are two alternative hypotheses: (1) that the patient has a particular form of cancer, and (2) that the patient does not. The available data is from a particular laboratory test with two possible outcomes: $\oplus$ (positive) and $\ominus$ (negative). We have prior knowledge that over the entire population of people only .008 have this disease. Furthermore, the lab test is only an imperfect indicator of the disease. The test returns a correct positive result in only 98% of the cases in which the disease is actually present and a correct negative result in only 97% of the cases in which the disease is not present. In other cases, the test returns the opposite result. The above situation can be summarized by the following probabilities:

$$\begin{aligned} P(cancer) &= .008, & P(\neg cancer) &= .992 \\ P(\oplus|cancer) &= .98, & P(\ominus|cancer) &= .02 \\ P(\oplus|\neg cancer) &= .03, & P(\ominus|\neg cancer) &= .97 \end{aligned}$$

Suppose we now observe a new patient for whom the lab test returns a positive result. Should we diagnose the patient as having cancer or not? The maximum a posteriori hypothesis can be found using Equation (6.2):

$$\begin{aligned} P(\oplus|cancer)P(cancer) &= (.98).008 = .0078 \\ P(\oplus|\neg cancer)P(\neg cancer) &= (.03).992 = .0298 \end{aligned}$$

Thus, $h_{MAP} = \neg cancer$. The exact posterior probabilities can also be determined by normalizing the above quantities so that they sum to 1 (e.g., $P(cancer|\oplus) = \frac{.0078}{.0078+.0298} = .21$). This step is warranted because Bayes theorem states that the posterior probabilities are just the above quantities divided by the probability of the data, $P(\oplus)$. Although $P(\oplus)$ was not provided directly as part of the problem statement, we can calculate it in this fashion because we know that $P(cancer|\oplus)$ and $P(\neg cancer|\oplus)$ must sum to 1 (i.e., either the patient has cancer or they do not). Notice that while the posterior probability of *cancer* is significantly higher than its prior probability, the most probable hypothesis is still that the patient does not have cancer.

As this example illustrates, the result of Bayesian inference depends strongly on the prior probabilities, which must be available in order to apply the method directly. Note also that in this example the hypotheses are not completely accepted or rejected, but rather become more or less probable as more data is observed.

Basic formulas for calculating probabilities are summarized in Table 6.1.

6.3 BAYES THEOREM AND CONCEPT LEARNING

What is the relationship between Bayes theorem and the problem of concept learning? Since Bayes theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data, we can use it as the basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable. This section considers such a brute-force Bayesian concept learning algorithm, then compares it to concept learning algorithms we considered in Chapter 2. As we shall see, one interesting result of this comparison is that under certain conditions several algorithms discussed in earlier chapters output the same hypotheses as this brute-force Bayesian

~~as its correct classification (which can be done using at most $\log_2 k$ bits, where k is the number of possible classifications). The hypothesis h_{MDL} under the encodings C_1 and C_2 is just the one that minimizes the sum of these description lengths.~~

~~Thus the MDL principle provides a way of trading off hypothesis complexity for the number of errors committed by the hypothesis. It might select a shorter hypothesis that makes a few errors over a longer hypothesis that perfectly classifies the training data. Viewed in this light, it provides one method for dealing with the issue of *overfitting* the data.~~

~~Quinlan and Rivest (1989) describe experiments applying the MDL principle to choose the best size for a decision tree. They report that the MDL-based method produced learned trees whose accuracy was comparable to that of the standard tree-pruning methods discussed in Chapter 3. Mehta et al. (1995) describe an alternative MDL-based approach to decision tree pruning, and describe experiments in which an MDL-based approach produced results comparable to standard tree-pruning methods.~~

~~What shall we conclude from this analysis of the Minimum Description Length principle? Does this prove once and for all that short hypotheses are best? No. What we have shown is only that *if* a representation of hypotheses is chosen so that the size of hypothesis h is $-\log_2 P(h)$, and *if* a representation for exceptions is chosen so that the encoding length of D given h is equal to $-\log_2 P(D|h)$, *then* the MDL principle produces MAP hypotheses. However, to show that we have such a representation we must know all the prior probabilities $P(h)$, as well as the $P(D|h)$. There is no reason to believe that the MDL hypothesis relative to *arbitrary* encodings C_1 and C_2 should be preferred. As a practical matter it might sometimes be easier for a human designer to specify a representation that captures knowledge about the relative probabilities of hypotheses than it is to fully specify the probability of each hypothesis. Descriptions in the literature on the application of MDL to practical learning problems often include arguments providing some form of justification for the encodings chosen for C_1 and C_2.~~

6.7 BAYES OPTIMAL CLASSIFIER

So far we have considered the question "what is the most probable *hypothesis* given the training data?" In fact, the question that is often of most significance is the closely related question "what is the most probable *classification* of the new instance given the training data?" Although it may seem that this second question can be answered by simply applying the MAP hypothesis to the new instance, in fact it is possible to do better.

To develop some intuitions consider a hypothesis space containing three hypotheses, h_1, h_2, and h_3. Suppose that the posterior probabilities of these hypotheses given the training data are .4, .3, and .3 respectively. Thus, h_1 is the MAP hypothesis. Suppose a new instance x is encountered, which is classified positive by h_1, but negative by h_2 and h_3. Taking all hypotheses into account, the probability that x is positive is .4 (the probability associated with h_1), and the probability that it is negative is therefore .6. The most probable classification (negative) in this case is different from the classification generated by the MAP hypothesis.

In general, the most probable classification of the new instance is obtained by combining the predictions of all hypotheses, weighted by their posterior probabilities. If the possible classification of the new example can take on any value v_j from some set V, then the probability $P(v_j|D)$ that the correct classification for the new instance is v_j, is just

$$P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

The optimal classification of the new instance is the value v_j, for which $P(v_j|D)$ is maximum.

Bayes optimal classification:

$$\underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) \tag{6.18}$$

To illustrate in terms of the above example, the set of possible classifications of the new instance is $V = \{\oplus, \ominus\}$, and

$$P(h_1|D) = .4, \; P(\ominus|h_1) = 0, \; P(\oplus|h_1) = 1$$
$$P(h_2|D) = .3, \; P(\ominus|h_2) = 1, \; P(\oplus|h_2) = 0$$
$$P(h_3|D) = .3, \; P(\ominus|h_3) = 1, \; P(\oplus|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(\oplus|h_i)P(h_i|D) = .4$$
$$\sum_{h_i \in H} P(\ominus|h_i)P(h_i|D) = .6$$

and

$$\underset{v_j \in \{\oplus,\ominus\}}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = \ominus$$

Any system that classifies new instances according to Equation (6.18) is called a *Bayes optimal classifier*, or Bayes optimal learner. No other classification method using the same hypothesis space and same prior knowledge can outperform this method on average. This method maximizes the probability that the new instance is classified correctly, given the available data, hypothesis space, and prior probabilities over the hypotheses.

For example, in learning boolean concepts using version spaces as in the earlier section, the Bayes optimal classification of a new instance is obtained by taking a weighted vote among all members of the version space, with each candidate hypothesis weighted by its posterior probability.

Note one curious property of the Bayes optimal classifier is that the predictions it makes can correspond to a hypothesis not contained in H! Imagine using Equation (6.18) to classify every instance in X. The labeling of instances defined in this way need not correspond to the instance labeling of any single hypothesis h from H. One way to view this situation is to think of the Bayes optimal classifier as effectively considering a hypothesis space H' different from the space of hypotheses H to which Bayes theorem is being applied. In particular, H' effectively includes hypotheses that perform comparisons between linear combinations of predictions from multiple hypotheses in H.

~~6.8 GIBBS ALGORITHM~~

~~Although the Bayes optimal classifier obtains the best performance that can be achieved from the given training data, it can be quite costly to apply. The expense is due to the fact that it computes the posterior probability for every hypothesis in H and then combines the predictions of each hypothesis to classify each new instance.~~

~~An alternative, less optimal method is the Gibbs algorithm (see Opper and Haussler 1991), defined as follows:~~

1. ~~Choose a hypothesis h from H at random, according to the posterior probability distribution over H.~~
2. ~~Use h to predict the classification of the next instance x.~~

~~Given a new instance to classify, the Gibbs algorithm simply applies a hypothesis drawn at random according to the current posterior probability distribution. Surprisingly, it can be shown that under certain conditions the expected misclassification error for the Gibbs algorithm is at most twice the expected error of the Bayes optimal classifier (Haussler et al. 1994). More precisely, the expected value is taken over target concepts drawn at random according to the prior probability distribution assumed by the learner. Under this condition, the expected value of the error of the Gibbs algorithm is at worst twice the expected value of the error of the Bayes optimal classifier.~~

~~This result has an interesting implication for the concept learning problem described earlier. In particular, it implies that if the learner assumes a uniform prior over H, and if target concepts are in fact drawn from such a distribution when presented to the learner, *then classifying the next instance according to a hypothesis drawn at random from the current version space (according to a uniform distribution), will have expected error at most twice that of the Bayes optimal classifier.* Again, we have an example where a Bayesian analysis of a non-Bayesian algorithm yields insight into the performance of that algorithm.~~

6.9 NAIVE BAYES CLASSIFIER

One highly practical Bayesian learning method is the naive Bayes learner, often called the *naive Bayes classifier.* In some domains its performance has been shown to be comparable to that of neural network and decision tree learning. This section introduces the naive Bayes classifier; the next section applies it to the practical problem of learning to classify natural language text documents.

The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $\langle a_1, a_2 \dots a_n \rangle$. The learner is asked to predict the target value, or classification, for this new instance.

The Bayesian approach to classifying the new instance is to assign the most probable target value, v_{MAP}, given the attribute values $\langle a_1, a_2 \dots a_n \rangle$ that describe the instance.

$$v_{MAP} = \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \dots a_n)$$

We can use Bayes theorem to rewrite this expression as

$$\begin{aligned} v_{MAP} &= \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} \\ &= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j) \qquad (6.19) \end{aligned}$$

Now we could attempt to estimate the two terms in Equation (6.19) based on the training data. It is easy to estimate each of the $P(v_j)$ simply by counting the frequency with which each target value v_j occurs in the training data. However, estimating the different $P(a_1, a_2 \dots a_n | v_j)$ terms in this fashion is not feasible unless we have a very, very large set of training data. The problem is that the number of these terms is equal to the number of possible instances times the number of possible target values. Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates.

The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction $a_1, a_2 \dots a_n$ is just the product of the probabilities for the individual attributes: $P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$. Substituting this into Equation (6.19), we have the approach used by the naive Bayes classifier.

Naive Bayes classifier:

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) \qquad (6.20)$$

where v_{NB} denotes the target value output by the naive Bayes classifier. Notice that in a naive Bayes classifier the number of distinct $P(a_i | v_j)$ terms that must be estimated from the training data is just the number of distinct attribute values times the number of distinct target values—a much smaller number than if we were to estimate the $P(a_1, a_2 \ldots a_n | v_j)$ terms as first contemplated.

To summarize, the naive Bayes learning method involves a learning step in which the various $P(v_j)$ and $P(a_i | v_j)$ terms are estimated, based on their frequencies over the training data. The set of these estimates corresponds to the learned hypothesis. This hypothesis is then used to classify each new instance by applying the rule in Equation (6.20). Whenever the naive Bayes assumption of conditional independence is satisfied, this naive Bayes classification v_{NB} is identical to the MAP classification.

One interesting difference between the naive Bayes learning method and other learning methods we have considered is that there is no explicit search through the space of possible hypotheses (in this case, the space of possible hypotheses is the space of possible values that can be assigned to the various $P(v_j)$ and $P(a_i | v_j)$ terms). Instead, the hypothesis is formed without searching, simply by counting the frequency of various data combinations within the training examples.

6.9.1 An Illustrative Example

Let us apply the naive Bayes classifier to a concept learning problem we considered during our discussion of decision tree learning: classifying days according to whether someone will play tennis. Table 3.2 from Chapter 3 provides a set of 14 training examples of the target concept *PlayTennis*, where each day is described by the attributes *Outlook, Temperature, Humidity,* and *Wind.* Here we use the naive Bayes classifier and the training data from this table to classify the following novel instance:

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

Our task is to predict the target value (*yes* or *no*) of the target concept *PlayTennis* for this new instance. Instantiating Equation (6.20) to fit the current task, the target value v_{NB} is given by

$$\begin{aligned} v_{NB} &= \operatorname*{argmax}_{v_j \in \{yes, no\}} P(v_j) \prod_i P(a_i | v_j) \\ &= \operatorname*{argmax}_{v_j \in \{yes, no\}} P(v_j) \quad P(Outlook = sunny | v_j) P(Temperature = cool | v_j) \\ &\qquad\qquad \cdot P(Humidity = high | v_j) P(Wind = strong | v_j) \end{aligned} \tag{6.21}$$

Notice in the final expression that a_i has been instantiated using the particular attribute values of the new instance. To calculate v_{NB} we now require 10 probabilities that can be estimated from the training data. First, the probabilities of the different target values can easily be estimated based on their frequencies over the 14 training examples

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

Similarly, we can estimate the conditional probabilities. For example, those for $Wind = strong$ are

$$P(Wind = strong|PlayTennis = yes) = 3/9 = .33$$

$$P(Wind = strong|PlayTennis = no) = 3/5 = .60$$

Using these probability estimates and similar estimates for the remaining attribute values, we calculate v_{NB} according to Equation (6.21) as follows (now omitting attribute names for brevity)

$$P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$

$$P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

Thus, the naive Bayes classifier assigns the target value $PlayTennis = no$ to this new instance, based on the probability estimates learned from the training data. Furthermore, by normalizing the above quantities to sum to one we can calculate the conditional probability that the target value is no, given the observed attribute values. For the current example, this probability is $\frac{.0206}{.0206+.0053} = .795$.

6.9.1.1 ESTIMATING PROBABILITIES

Up to this point we have estimated probabilities by the fraction of times the event is observed to occur over the total number of opportunities. For example, in the above case we estimated $P(Wind = strong|PlayTennis = no)$ by the fraction $\frac{n_c}{n}$ where $n = 5$ is the total number of training examples for which $PlayTennis = no$, and $n_c = 3$ is the number of these for which $Wind = strong$.

While this observed fraction provides a good estimate of the probability in many cases, it provides poor estimates when n_c is very small. To see the difficulty, imagine that, in fact, the value of $P(Wind = strong|PlayTennis = no)$ is .08 and that we have a sample containing only 5 examples for which $PlayTennis = no$. Then the most probable value for n_c is 0. This raises two difficulties. First, $\frac{n_c}{n}$ produces a biased underestimate of the probability. Second, when this probability estimate is zero, this probability term will dominate the Bayes classifier if the future query contains $Wind = strong$. The reason is that the quantity calculated in Equation (6.20) requires multiplying all the other probability terms by this zero value.

To avoid this difficulty we can adopt a Bayesian approach to estimating the probability, using the m-estimate defined as follows.

m-estimate of probability:

$$\frac{n_c + mp}{n + m} \tag{6.22}$$

Here, n_c and n are defined as before, p is our prior estimate of the probability we wish to determine, and m is a constant called the *equivalent sample size*, which determines how heavily to weight p relative to the observed data. A typical method for choosing p in the absence of other information is to assume uniform priors; that is, if an attribute has k possible values we set $p = \frac{1}{k}$. For example, in estimating $P(Wind = strong|PlayTennis = no)$ we note the attribute $Wind$ has two possible values, so uniform priors would correspond to choosing $p = .5$. Note that if m is zero, the m-estimate is equivalent to the simple fraction $\frac{n_c}{n}$. If both n and m are nonzero, then the observed fraction $\frac{n_c}{n}$ and prior p will be combined according to the weight m. The reason m is called the equivalent sample size is that Equation (6.22) can be interpreted as augmenting the n actual observations by an additional m virtual samples distributed according to p.

~~6.10 AN EXAMPLE: LEARNING TO CLASSIFY TEXT~~

~~To illustrate the practical importance of Bayesian learning methods, consider learning problems in which the instances are text documents. For example, we might wish to learn the target concept "electronic news articles that I find interesting," or "pages on the World Wide Web that discuss machine learning topics." In both cases, if a computer could learn the target concept accurately, it could automatically filter the large volume of online text documents to present only the most relevant documents to the user.~~

~~We present here a general algorithm for learning to classify text, based on the naive Bayes classifier. Interestingly, probabilistic approaches such as the one described here are among the most effective algorithms currently known for learning to classify text documents. Examples of such systems are described by Lewis (1991), Lang (1995), and Joachims (1996).~~

~~The naive Bayes algorithm that we shall present applies in the following general setting. Consider an instance space X consisting of all possible *text documents* (i.e., all possible strings of words and punctuation of all possible lengths). We are given training examples of some unknown target function $f(x)$, which can take on any value from some finite set V. The task is to learn from these training examples to predict the target value for subsequent text documents. For illustration, we will consider the target function classifying documents as interesting or uninteresting to a particular person, using the target values *like* and *dislike* to indicate these two classes.~~

~~The two main design issues involved in applying the naive Bayes classifier to such text classification problems are first to decide how to represent an arbitrary text document in terms of attribute values, and second to decide how to estimate the probabilities required by the naive Bayes classifier.~~

~~Our approach to representing arbitrary text documents is disturbingly simple: Given a text document, such as this paragraph, we define an attribute for each word position in the document and define the value of that attribute to be the English word found in that position. Thus, the current paragraph would be described by 111 attribute values, corresponding to the 111 word positions. The value of the first attribute is the word "our," the value of the second attribute is the word "approach," and so on. Notice that long text documents will require a larger number of attributes than short documents. As we shall see, this will not cause us any trouble.~~

comp.graphics	misc.forsale	soc.religion.christian	sci.space
comp.os.ms-windows.misc	rec.autos	talk.politics.guns	sci.crypt
comp.sys.ibm.pc.hardware	rec.motorcycles	talk.politics.mideast	sci.electronics
comp.sys.mac.hardware	rec.sport.baseball	talk.politics.misc	sci.med
comp.windows.x	rec.sport.hockey	talk.religion.misc	
		alt.atheism	

~~**TABLE 6.3**~~
~~Twenty usenet newsgroups used in the text classification experiment. After training on 667 articles from each newsgroup, a naive Bayes classifier achieved an accuracy of 89% predicting to which newsgroup subsequent articles belonged. Random guessing would produce an accuracy of only 5%.~~

~~training examples to learn to predict which subsequent articles will be of interest to the user, so that it can bring these to the user's attention. Lang (1995) reports experiments in which NEWSWEEDER used its learned profile of user interests to suggest the most highly rated new articles each day. By presenting the user with the top 10% of its automatically rated new articles each day, it created a pool of articles containing three to four times as many interesting articles as the general pool of articles read by the user. For example, for one user the fraction of articles rated "interesting" was 16% overall, but was 59% among the articles recommended by NEWSWEEDER.~~

~~Several other, non-Bayesian, statistical text learning algorithms are common, many based on similarity metrics initially developed for information retrieval (e.g., see Rocchio 1971; Salton 1991). Additional text learning algorithms are described in Hearst and Hirsh (1996).~~

6.11 BAYESIAN BELIEF NETWORKS

As discussed in the previous two sections, the naive Bayes classifier makes significant use of the assumption that the values of the attributes $a_1 \ldots a_n$ are conditionally independent given the target value v. This assumption dramatically reduces the complexity of learning the target function. When it is met, the naive Bayes classifier outputs the optimal Bayes classification. However, in many cases this conditional independence assumption is clearly overly restrictive.

A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities. In contrast to the naive Bayes classifier, which assumes that *all* the variables are conditionally independent given the value of the target variable, Bayesian belief networks allow stating conditional independence assumptions that apply to *subsets* of the variables. Thus, Bayesian belief networks provide an intermediate approach that is less constraining than the global assumption of conditional independence made by the naive Bayes classifier, but more tractable than avoiding conditional independence assumptions altogether. Bayesian belief networks are an active focus of current research, and a variety of algorithms have been proposed for learning them and for using them for inference.

In this section we introduce the key concepts and the representation of Bayesian belief networks. More detailed treatments are given by Pearl (1988), Russell and Norvig (1995), Heckerman et al. (1995), and Jensen (1996).

In general, a Bayesian belief network describes the probability distribution over a set of variables. Consider an arbitrary set of random variables $Y_1 \ldots Y_n$, where each variable Y_i can take on the set of possible values $V(Y_i)$. We define the *joint space* of the set of variables Y to be the cross product $V(Y_1) \times V(Y_2) \times \ldots V(Y_n)$. In other words, each item in the joint space corresponds to one of the possible assignments of values to the tuple of variables $\langle Y_1 \ldots Y_n \rangle$. The probability distribution over this joint space is called the *joint probability distribution.* The joint probability distribution specifies the probability for each of the possible variable bindings for the tuple $\langle Y_1 \ldots Y_n \rangle$. A Bayesian belief network describes the joint probability distribution for a set of variables.

6.11.1 Conditional Independence

Let us begin our discussion of Bayesian belief networks by defining precisely the notion of conditional independence. Let X, Y, and Z be three discrete-valued random variables. We say that X is *conditionally independent* of Y given Z if the probability distribution governing X is independent of the value of Y given a value for Z; that is, if

$$(\forall x_i, y_j, z_k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

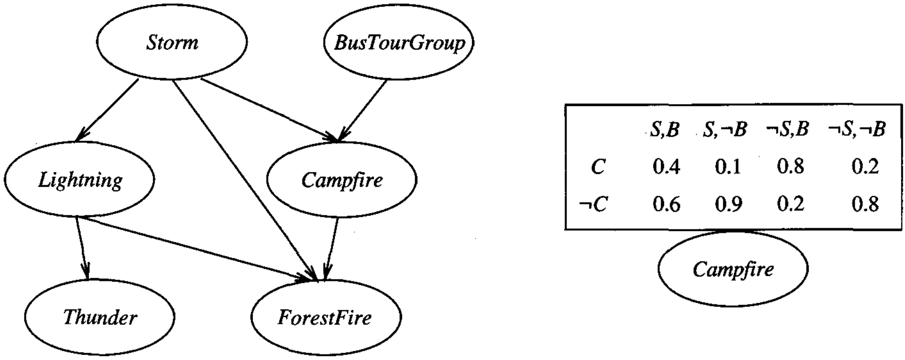
where $x_i \in V(X)$, $y_j \in V(Y)$, and $z_k \in V(Z)$. We commonly write the above expression in abbreviated form as $P(X|Y, Z) = P(X|Z)$. This definition of conditional independence can be extended to sets of variables as well. We say that the set of variables $X_1 \ldots X_l$ is conditionally independent of the set of variables $Y_1 \ldots Y_m$ given the set of variables $Z_1 \ldots Z_n$ if

$$P(X_1 \ldots X_l | Y_1 \ldots Y_m, Z_1 \ldots Z_n) = P(X_1 \ldots X_l | Z_1 \ldots Z_n)$$

Note the correspondence between this definition and our use of conditional independence in the definition of the naive Bayes classifier. The naive Bayes classifier assumes that the instance attribute A_1 is conditionally independent of instance attribute A_2 given the target value V. This allows the naive Bayes classifier to calculate $P(A_1, A_2|V)$ in Equation (6.20) as follows

$$\begin{aligned} P(A_1, A_2|V) &= P(A_1|A_2, V)P(A_2|V) && (6.23) \\ &= P(A_1|V)P(A_2|V) && (6.24) \end{aligned}$$

Equation (6.23) is just the general form of the product rule of probability from Table 6.1. Equation (6.24) follows because if A_1 is conditionally independent of A_2 given V, then by our definition of conditional independence $P(A_1|A_2, V) = P(A_1|V)$.



	S,B	S,¬B	¬S,B	¬S,¬B
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

FIGURE 6.3
A Bayesian belief network. The network on the left represents a set of conditional independence assumptions. In particular, each node is asserted to be conditionally independent of its nondescendants, given its immediate parents. Associated with each node is a conditional probability table, which specifies the conditional distribution for the variable given its immediate parents in the graph. The conditional probability table for the *Campfire* node is shown at the right, where *Campfire* is abbreviated to C, *Storm* abbreviated to S, and *BusTourGroup* abbreviated to B.

6.11.2 Representation

A *Bayesian belief network* (Bayesian network for short) represents the joint probability distribution for a set of variables. For example, the Bayesian network in Figure 6.3 represents the joint probability distribution over the boolean variables *Storm, Lightning, Thunder, ForestFire, Campfire*, and *BusTourGroup*. In general, a Bayesian network represents the joint probability distribution by specifying a set of conditional independence assumptions (represented by a directed acyclic graph), together with sets of local conditional probabilities. Each variable in the joint space is represented by a node in the Bayesian network. For each variable two types of information are specified. First, the network arcs represent the assertion that the variable is conditionally independent of its nondescendants in the network given its immediate predecessors in the network. We say X is a *descendant* of Y if there is a directed path from Y to X. Second, a conditional probability table is given for each variable, describing the probability distribution for that variable given the values of its immediate predecessors. The joint probability for any desired assignment of values $\langle y_1, \ldots, y_n \rangle$ to the tuple of network variables $\langle Y_1 \ldots Y_n \rangle$ can be computed by the formula

$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

where $Parents(Y_i)$ denotes the set of immediate predecessors of Y_i in the network. Note the values of $P(y_i | Parents(Y_i))$ are precisely the values stored in the conditional probability table associated with node Y_i.

To illustrate, the Bayesian network in Figure 6.3 represents the joint probability distribution over the boolean variables *Storm, Lightning, Thunder, Forest-*

Fire, *Campfire*, and *BusTourGroup*. Consider the node *Campfire*. The network nodes and arcs represent the assertion that *Campfire* is conditionally independent of its nondescendants *Lightning* and *Thunder*, given its immediate parents *Storm* and *BusTourGroup*. This means that once we know the value of the variables *Storm* and *BusTourGroup*, the variables *Lightning* and *Thunder* provide no additional information about *Campfire*. The right side of the figure shows the conditional probability table associated with the variable *Campfire*. The top left entry in this table, for example, expresses the assertion that

$$P(Campfire = True|Storm = True, BusTourGroup = True) = 0.4$$

Note this table provides only the conditional probabilities of *Campfire* given its parent variables *Storm* and *BusTourGroup*. The set of local conditional probability tables for all the variables, together with the set of conditional independence assumptions described by the network, describe the full joint probability distribution for the network.

One attractive feature of Bayesian belief networks is that they allow a convenient way to represent causal knowledge such as the fact that *Lightning* causes *Thunder*. In the terminology of conditional independence, we express this by stating that *Thunder* is conditionally independent of other variables in the network, given the value of *Lightning*. Note this conditional independence assumption is implied by the arcs in the Bayesian network of Figure 6.3.

6.11.3 Inference

We might wish to use a Bayesian network to infer the value of some target variable (e.g., *ForestFire*) given the observed values of the other variables. Of course, given that we are dealing with random variables it will not generally be correct to assign the target variable a single determined value. What we really wish to infer is the probability distribution for the target variable, which specifies the probability that it will take on each of its possible values given the observed values of the other variables. This inference step can be straightforward if values for all of the other variables in the network are known exactly. In the more general case we may wish to infer the probability distribution for some variable (e.g., *ForestFire*) given observed values for only a subset of the other variables (e.g., *Thunder* and *BusTourGroup* may be the only observed values available). In general, a Bayesian network can be used to compute the probability distribution for any subset of network variables given the values or distributions for any subset of the remaining variables.

Exact inference of probabilities in general for an arbitrary Bayesian network is known to be NP-hard (Cooper 1990). Numerous methods have been proposed for probabilistic inference in Bayesian networks, including exact inference methods and approximate inference methods that sacrifice precision to gain efficiency. For example, Monte Carlo methods provide approximate solutions by randomly sampling the distributions of the unobserved variables (Pradham and Dagum 1996). In theory, even approximate inference of probabilities in Bayesian networks can be NP-hard (Dagum and Luby 1993). Fortunately, in practice approximate methods have been shown to be useful in many cases. Discussions of inference methods for Bayesian networks are provided by Russell and Norvig (1995) and by Jensen (1996).

6.11.4 Learning Bayesian Belief Networks

Can we devise effective algorithms for learning Bayesian belief networks from training data? This question is a focus of much current research. Several different settings for this learning problem can be considered. First, the network structure might be given in advance, or it might have to be inferred from the training data. Second, all the network variables might be directly observable in each training example, or some might be unobservable.

In the case where the network structure is given in advance and the variables are fully observable in the training examples, learning the conditional probability tables is straightforward. We simply estimate the conditional probability table entries just as we would for a naive Bayes classifier.

In the case where the network structure is given but only some of the variable values are observable in the training data, the learning problem is more difficult. This problem is somewhat analogous to learning the weights for the hidden units in an artificial neural network, where the input and output node values are given but the hidden unit values are left unspecified by the training examples. In fact, Russell et al. (1995) propose a similar gradient ascent procedure that learns the entries in the conditional probability tables. This gradient ascent procedure searches through a space of hypotheses that corresponds to the set of all possible entries for the conditional probability tables. The objective function that is maximized during gradient ascent is the probability $P(D|h)$ of the observed training data D given the hypothesis h. By definition, this corresponds to searching for the maximum likelihood hypothesis for the table entries.

6.11.5 Gradient Ascent Training of Bayesian Networks

The gradient ascent rule given by Russell et al. (1995) maximizes $P(D|h)$ by following the gradient of $\ln P(D|h)$ with respect to the parameters that define the conditional probability tables of the Bayesian network. Let w_{ijk} denote a single entry in one of the conditional probability tables. In particular, let w_{ijk} denote the conditional probability that the network variable Y_i will take on the value y_{ij} given that its immediate parents U_i take on the values given by u_{ik}. For example, if w_{ijk} is the top right entry in the conditional probability table in Figure 6.3, then Y_i is the variable *Campfire*, U_i is the tuple of its parents $\langle Storm, BusTourGroup\rangle$, $y_{ij} = True$, and $u_{ik} = \langle False, False\rangle$. The gradient of $\ln P(D|h)$ is given by the derivatives $\frac{\partial \ln P(D|h)}{\partial w_{ijk}}$ for each of the w_{ijk}. As we show below, each of these derivatives can be calculated as

$$\frac{\partial \ln P(D|h)}{\partial w_{ij}} = \sum_{d \in D} \frac{P(Y_i = y_{ij}, U_i = u_{ik}|d)}{w_{ijk}} \tag{6.25}$$

For example, to calculate the derivative of $\ln P(D|h)$ with respect to the upper-rightmost entry in the table of Figure 6.3 we will have to calculate the quantity $P(Campfire = True, Storm = False, BusTourGroup = False|d)$ for each training example d in D. When these variables are unobservable for the training example d, this required probability can be calculated from the observed variables in d using standard Bayesian network inference. In fact, these required quantities are easily derived from the calculations performed during most Bayesian network inference, so learning can be performed at little additional cost whenever the Bayesian network is used for inference and new evidence is subsequently obtained.

Below we derive Equation (6.25) following Russell et al. (1995). The remainder of this section may be skipped on a first reading without loss of continuity. To simplify notation, in this derivation we will write the abbreviation $P_h(D)$ to represent $P(D|h)$. Thus, our problem is to derive the gradient defined by the set of derivatives $\frac{\partial P_h(D)}{\partial w_{ijk}}$ for all i, j, and k. Assuming the training examples d in the data set D are drawn independently, we write this derivative as

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \frac{\partial}{\partial w_{ijk}} \ln \prod_{d \in D} P_h(d)\\ &= \sum_{d \in D} \frac{\partial \ln P_h(d)}{\partial w_{ijk}}\\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial P_h(d)}{\partial w_{ijk}}\end{aligned}$$

This last step makes use of the general equality $\frac{\partial \ln f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$. We can now introduce the values of the variables Y_i and $U_i = Parents(Y_i)$, by summing over their possible values $y_{ij'}$ and $u_{ik'}$.

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}, u_{ik'})\\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}|u_{ik'}) P_h(u_{ik'})\end{aligned}$$

This last step follows from the product rule of probability, Table 6.1. Now consider the rightmost sum in the final expression above. Given that $w_{ijk} \equiv P_h(y_{ij}|u_{ik})$, the only term in this sum for which $\frac{\partial}{\partial w_{ijk}}$ is nonzero is the term for which $j' = j$ and $i' = i$. Therefore

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) P_h(y_{ij}|u_{ik}) P_h(u_{ik})\\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) w_{ijk} P_h(u_{ik})\\ &= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d|y_{ij}, u_{ik}) P_h(u_{ik})\end{aligned}$$

Applying Bayes theorem to rewrite $P_h(d|y_{ij}, u_{ik})$, we have

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d\in D} \frac{1}{P_h(d)} \frac{P_h(y_{ij}, u_{ik}|d) P_h(d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\ &= \sum_{d\in D} \frac{P_h(y_{ij}, u_{ik}|d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\ &= \sum_{d\in D} \frac{P_h(y_{ij}, u_{ik}|d)}{P_h(y_{ij}|u_{ik})} \\ &= \sum_{d\in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}} \qquad (6.26)\end{aligned}$$

Thus, we have derived the gradient given in Equation (6.25). There is one more item that must be considered before we can state the gradient ascent training procedure. In particular, we require that as the weights w_{ijk} are updated they must remain valid probabilities in the interval [0,1]. We also require that the sum $\sum_j w_{ijk}$ remains 1 for all i, k. These constraints can be satisfied by updating weights in a two-step process. First we update each w_{ijk} by gradient ascent

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d\in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}}$$

where η is a small constant called the learning rate. Second, we renormalize the weights w_{ijk} to assure that the above constraints are satisfied. As discussed by Russell et al., this process will converge to a locally maximum likelihood hypothesis for the conditional probabilities in the Bayesian network.

As in other gradient-based approaches, this algorithm is guaranteed only to find some local optimum solution. An alternative to gradient ascent is the EM algorithm discussed in Section 6.12, which also finds locally maximum likelihood solutions.

6.11.6 Learning the Structure of Bayesian Networks

Learning Bayesian networks when the network structure is not known in advance is also difficult. Cooper and Herskovits (1992) present a Bayesian scoring metric for choosing among alternative networks. They also present a heuristic search algorithm called K2 for learning network structure when the data is fully observable. Like most algorithms for learning the structure of Bayesian networks, K2 performs a greedy search that trades off network complexity for accuracy over the training data. In one experiment K2 was given a set of 3,000 training examples generated at random from a manually constructed Bayesian network containing 37 nodes and 46 arcs. This particular network described potential anesthesia problems in a hospital operating room. In addition to the data, the program was also given an initial ordering over the 37 variables that was consistent with the partial ordering of variable dependencies in the actual network. The program succeeded in reconstructing the correct Bayesian network structure almost exactly, with the exception of one incorrectly deleted arc and one incorrectly added arc.

Constraint-based approaches to learning Bayesian network structure have also been developed (e.g., Spirtes et al. 1993). These approaches infer independence and dependence relationships from the data, and then use these relationships to construct Bayesian networks. Surveys of current approaches to learning Bayesian networks are provided by Heckerman (1995) and Buntine (1994).

~~6.12 THE EM ALGORITHM~~

~~In many practical learning settings, only a subset of the relevant instance features might be observable. For example, in training or using the Bayesian belief network of Figure 6.3, we might have data where only a subset of the network variables *Storm*, *Lightning*, *Thunder*, *ForestFire*, *Campfire*, and *BusTourGroup* have been observed. Many approaches have been proposed to handle the problem of learning in the presence of unobserved variables. As we saw in Chapter 3, if some variable is sometimes observed and sometimes not, then we can use the cases for which it has been observed to learn to predict its values when it is not. In this section we describe the EM algorithm (Dempster et al. 1977), a widely used approach to learning in the presence of unobserved variables. The EM algorithm can be used even for variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known. The EM algorithm has been used to train Bayesian belief networks (see Heckerman 1995) as well as radial basis function networks discussed in Section 8.4. The EM algorithm is also the basis for many unsupervised clustering algorithms (e.g., Cheeseman et al. 1988), and it is the basis for the widely used Baum-Welch forward-backward algorithm for learning Partially Observable Markov Models (Rabiner 1989).~~

~~6.12.1 Estimating Means of k Gaussians~~

~~The easiest way to introduce the EM algorithm is via an example. Consider a problem in which the data D is a set of instances generated by a probability distribution that is a mixture of k distinct Normal distributions. This problem setting is illustrated in Figure 6.4 for the case where $k = 2$ and where the instances are the points shown along the x axis. Each instance is generated using a two-step process. First, one of the k Normal distributions is selected at random. Second, a single random instance x_i is generated according to this selected distribution. This process is repeated to generate a set of data points as shown in the figure. To simplify our discussion, we consider the special case where the selection of the single Normal distribution at each step is based on choosing each with uniform probability, where each of the k Normal distributions has the same variance σ^2, and where σ^2 is known. The learning task is to output a hypothesis $h = \langle\mu_1, \ldots \mu_k\rangle$ that describes the means of each of the k distributions. We would like to find~~