



VIVEKANANDA
COLLEGE OF ENGINEERING & TECHNOLOGY
NEHRUNAGAR POST, PUTTUR, D.K. 574203



Lecture Notes
on



Subject Code: 15CS73
(CBCS Scheme)

Prepared by

Mr. Harivinod N
Dept. of Computer Science and Engineering,
VCET Puttur

Module-4
Bayesian Learning

Course website:
www.techjourney.in



Module-3: Bayesian Learning

1.1 Introduction

Bayesian reasoning provides a probabilistic approach to inference. It assumes 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 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.
- 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.

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.



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 . 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.
- Similarly, we will write $P(D)$ to denote the **prior probability** that **training data D** will be observed
- Next, we will write $P(D|h)$ to denote the probability of observing data D given some world in which hypothesis h holds. In general, 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 provides a way to calculate the posterior probability $P(h|D)$, from the prior probability $P(h)$, together with $P(D)$ and $P(D|h)$.

$$\text{Bayes theorem: } P(h|D) = \frac{P(D|h)P(h)}{P(D)} \quad \dots(1)$$

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,

$$\begin{aligned} h_{MAP} &\equiv \operatorname{argmax}_{h \in H} P(h|D) \\ &= \operatorname{argmax}_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \operatorname{argmax}_{h \in H} P(D|h)P(h) \quad \dots(2) \end{aligned}$$



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 above equation 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}

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}} P(D|h) \quad \dots(3)$$

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.

Summary of basic probability formulas.

- **Product rule:** probability $P(A \wedge B)$ of a conjunction of two events A and B

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

- **Sum rule:** probability of a disjunction of two events A and B

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- **Bayes theorem:** the posterior probability $P(h|D)$ of h given D

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

- **Theorem of total probability:** if events A_1, \dots, A_n are mutually exclusive with $\sum_{i=1}^n P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$$

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.

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?

Solution: The above situation can be summarized by the following probabilities:

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



The maximum a posteriori hypothesis can be found using Equation (2):

$$P(\oplus|\text{cancer})P(\text{cancer}) = (.98).008 = .0078$$

$$P(\oplus|\neg\text{cancer})P(\neg\text{cancer}) = (.03).992 = .0298$$

Thus, $\mathbf{h}_{\text{map}} = \neg \text{cancer}$. (No Cancer)

Note: The exact posterior probabilities can also be determined by normalizing the above quantities so that they sum to 1.

$$P(\text{cancer}|\oplus) = \frac{.0078}{.0078+0.0298} = .21$$

$$P(\neg\text{cancer}|\oplus) = \frac{.0298}{.0078+0.0298} = .79$$

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 $P(\text{cancer}|\oplus)$ and $P(\neg\text{cancer}|\oplus)$ must sum to 1.

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.

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.

3.1 Brute-Force Bayes Concept Learning

Consider the concept learning problem first introduced in Module-1. Assume the learner considers some finite hypothesis space H defined over the instance space X , in which the task is to learn some target concept $c : X \rightarrow \{0,1\}$. As usual, we assume that the learner is given some sequence of training examples $((x_1, d_1) \dots (x_m, d_m))$ where x_i is some instance from X and where d_i is the target value of x_i (i.e., $d_i = c(x_i)$). To simplify the discussion in this section, we assume the sequence of instances $(x_1 \dots x_m)$ is held fixed, so that the training data D can be written simply as the sequence of target values $D = (d_1 \dots d_m)$

We can design a straightforward concept learning algorithm to output the maximum a posteriori hypothesis, based on Bayes theorem, as follows:

Brute-Force Map Learning Algorithm

1. For each hypothesis h in H , calculate the posterior probability

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

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D)$$

This algorithm may require significant computation, because it applies Bayes theorem to each hypothesis in H to calculate $P(h|D)$. While this may prove impractical for large hypothesis spaces, the algorithm is still of interest because it provides a standard against which we may judge the performance of other concept learning algorithms.

We assume the following.

1. The training data D is noise free (i.e., $d_i = c(x_i)$).
2. The target concept c is contained in the hypothesis space H
3. We have no a priori reason to believe that any hypothesis is more probable than any other.

Given no prior knowledge (i.e. $P(h)$ is not given) that one hypothesis is more likely than another, it is reasonable to assign the same prior probability to every hypothesis h in H .

$$P(h) = \frac{1}{|H|} \quad \text{for all } h \text{ in } H$$

Now, $P(D|h)$ is the probability of observing the target values $D = (d_1 \dots d_m)$ for the fixed set of instances $(x_1 \dots x_m)$, given a world in which hypothesis h holds (i.e., given a world in which h is the correct description of the target concept c). Since we assume noise-free training data, the probability of observing classification d_i given h is just 1 if $d_i = h(x_i)$ and 0 if $d_i \neq h(x_i)$. Therefore,

$$P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases} \quad \dots(4)$$

In other words, the probability of data D given hypothesis h is 1 if D is consistent with h , and 0 otherwise. Recalling Bayes theorem, we have,

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

First consider the case where h is inconsistent with the training data D . Here $P(D|h) = 0$ due to Equation (4). Thus, the posterior probability of hypothesis is

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0 \text{ if } h \text{ is inconsistent with } D$$

Now consider the case where h is consistent with D . Since Equation (4) defines $P(D|h) = 1$ when h is consistent with D , we have

$$\begin{aligned}
 P(h|D) &= \frac{1 \cdot \frac{1}{|H|}}{P(D)} = \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} \\
 &= \frac{1}{|VS_{H,D}|} \text{ if } h \text{ is consistent with } D
 \end{aligned}$$

where $VS_{H,D}$ is the Version Space (subset of hypotheses) from H that are consistent with D . The derivation for $P(D)$ is as follows

$$\begin{aligned}
 P(D) &= \sum_{h_i \in H} P(D|h_i)P(h_i) = \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|} \\
 &= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} = \frac{|VS_{H,D}|}{|H|}
 \end{aligned}$$

To summarize, Bayes theorem implies that the posterior probability $P(h|D)$ under our assumed $P(h)$ and $P(D|h)$ is

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Every consistent hypothesis is, therefore, a MAP hypothesis.

The evolution of probabilities associated with hypotheses is depicted schematically in Figure given below. Initially (Figure 6.1a) all hypotheses have the same probability. As training data accumulates (Figures 6.1b and 6.1c), the posterior probability for inconsistent hypotheses becomes zero while the total probability summing to one is shared equally among the remaining consistent hypotheses.

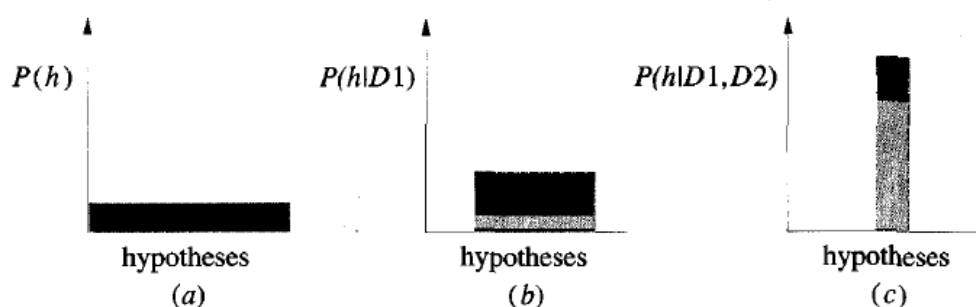


FIGURE 6.1

Evolution of posterior probabilities $P(h|D)$ with increasing training data. (a) Uniform priors assign equal probability to each hypothesis. As training data increases first to D_1 (b), then to $D_1 \wedge D_2$ (c), the posterior probability of inconsistent hypotheses becomes zero, while posterior probabilities increase for hypotheses remaining in the version space.

3.2 MAP Hypotheses and Consistent Learners

The above analysis shows that in the given setting, every hypothesis consistent with D is a MAP hypothesis. We will say that a learning algorithm is a **consistent learner** provided it outputs a hypothesis that commits zero errors over the training examples. Given the above analysis, we can conclude that every consistent learner outputs a MAP hypothesis, if we assume



a uniform prior probability distribution over H (i.e., $P(h_i) = P(h_j)$ for all i, j), and if we assume deterministic, noise free training data.

The Bayesian framework allows one way to characterize the behavior of learning algorithms (e.g., FIND-S), even when the learning algorithm does not explicitly manipulate probabilities. By identifying probability distributions $P(h)$ and $P(D|h)$ under which the algorithm outputs optimal (i.e., MAP) hypotheses, we can characterize the implicit assumptions, under which this algorithm behaves optimally. Thus, Bayesian analysis can be used to show that a particular learning algorithm outputs MAP hypothesis even though it may not explicitly use Bayes rule or calculate probabilities in any form.

So far we discussed a special case of Bayesian reasoning, where $P(D|h)$ takes on values of only 0 and 1, reflecting the deterministic predictions of hypotheses and the assumption of noise-free training data. In the next section, we model learning from noisy training data, by allowing $P(D|h)$ to take on values other than 0 and 1, and by introducing into $P(D|h)$ additional assumptions about the probability distributions that govern the noise.

4. Maximum Likelihood and Least-Squared Error Hypotheses

In this section we consider the problem of learning a *continuous-valued target function*. This is a problem faced by many learning approaches such as neural network learning, linear regression, and polynomial curve fitting. A straightforward Bayesian analysis will show that under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a maximum likelihood hypothesis.

Consider the following problem. Learner L considers an instance space X and a hypothesis space H consisting of some class of real-valued functions defined over X (i.e., each h in H is a function of the form $h : X \rightarrow \mathbb{R}$, where \mathbb{R} represents the set of real numbers). The problem faced by L is to learn an unknown target function $f : X \rightarrow \mathbb{R}$ drawn from H . A set of m training examples is provided, where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution. More precisely, each training example is a pair of the form (x_i, d_i) where $d_i = f(x_i) + e_i$. Here $f(x_i)$ is the noise-free value of the target function and e_i is a random variable representing the noise. It is assumed that the values of the e_i are drawn independently and that they are distributed according to a Normal distribution with zero mean. The task of the learner is to output a maximum likelihood hypothesis, or, equivalently, a MAP hypothesis assuming all hypotheses are equally probable a priori.

Example: A simple example of such a problem is learning a linear function, though our analysis applies to learning arbitrary real-valued functions. Figure 6.2 illustrates the whole scenario. Here notice that the maximum likelihood hypothesis is not necessarily identical to the correct hypothesis, f , because it is inferred from only a limited sample of noisy training data.

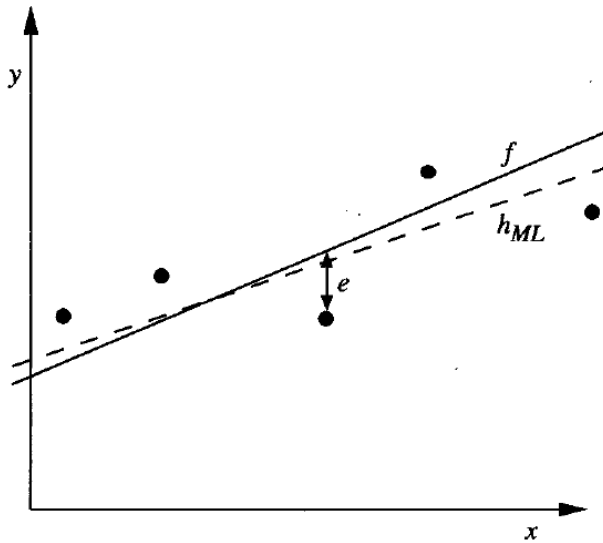


FIGURE 6.2
 Learning a real-valued function. The target function f corresponds to the solid line. The training examples $\langle x_i, d_i \rangle$ are assumed to have Normally distributed noise e_i with zero mean added to the true target value $f(x_i)$. The dashed line corresponds to the linear function that minimizes the sum of squared errors. Therefore, it is the maximum likelihood hypothesis h_{ML} , given these five training examples.

Before showing why a hypothesis that minimizes the sum of squared errors in this setting is also a maximum likelihood hypothesis, let us quickly review two basic concepts from probability theory: probability densities and Normal distributions.

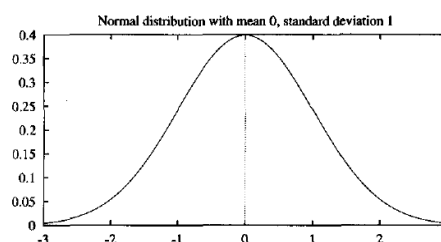
Probability densities:

First, in order to discuss probabilities over continuous variables such as e , we must introduce probability densities. The reason, roughly, is that we wish for the total probability over all possible values of the random variable to sum to one. In the case of continuous variables we cannot achieve this by assigning a finite probability to each of the infinite set of possible values for the random variable. Instead, we speak of a probability density for continuous variables such as e and require that the integral of this probability density over all possible values be one. In general, we will use lower case p to refer to the probability density function, to distinguish it from a finite probability P (which we will sometimes refer to as a probability mass). The probability density $p(x_0)$ is the limit as ϵ goes to zero, of times the probability that x will take on a value in the interval $[x_0, x_0 + \epsilon)$.

Probability density function:
$$p(x_0) \equiv \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} P(x_0 \leq x < x_0 + \epsilon)$$

Normal Distribution: Random noise variable e is generated by a Normal probability distribution. A Normal distribution (also called a Gaussian distribution) is a smooth, bell-shaped distribution that can be completely characterized by its mean μ and its standard deviation σ . It can be defined by the probability density function.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



A Normal distribution is fully determined by two parameters in the above formula: μ and σ . If the random variable X follows a normal distribution, then:

- The probability that X will fall into the interval (a, b) is given by $\int_a^b p(x)dx$
- The expected, or mean value of X , $E[X]$, is $E[X] = \mu$
- The variance of X , $\text{Var}(X)$, is $\text{Var}(X) = \sigma^2$
- The standard deviation of X , σ_x , is $\sigma_x = \sigma$

The Central Limit Theorem states that the sum of a large number of independent, identically distributed random variables follows a distribution that is approximately Normal.

Prove: Maximum likelihood hypothesis h_{ML} minimizes the sum of the squared errors between the observed training values d_i and the hypothesis predictions $h(x_i)$

Proof: From equation (3) we have

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} p(D|h)$$

Let set of training instances be (x_1, \dots, x_m) and therefore consider the data D to be the corresponding sequence of target values $D = (d_1, \dots, d_m)$. Here $d_i = f(x_i) + e_i$. Assuming the training examples are mutually independent given h , we can write $P(D|h)$ as the product of the various $p(d_i|h)$

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^m p(d_i|h)$$

Given that the noise e_i obeys a Normal distribution with zero mean and unknown variance σ^2 , each d_i must also obey a Normal distribution with variance σ^2 centered around the true target value $f(x_i)$ rather than 0. Therefore $p(d_i|h)$ can be written as a Normal distribution with variance σ^2 and mean $\mu = f(x_i)$. Let us write the formula for this Normal distribution to describe $p(d_i|h)$, using general formula for a Normal distribution and substituting the appropriate μ and σ^2 . Because we are writing the expression for the probability of d_i given that h is the correct description of the target function f , we will also substitute $\mu = f(x_i) = h(x_i)$, yielding

$$\begin{aligned} h_{ML} &= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - \mu)^2} \\ &= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - h(x_i))^2} \end{aligned}$$

We now apply a transformation that is common in maximum likelihood calculations: Rather than maximizing the above complicated expression we shall choose to maximize its (less complicated) logarithm. This is justified because $\ln p$ is a monotonic function of p . Therefore, maximizing $\ln p$ also maximizes p .

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^m \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

The first term in this expression is a constant independent of h , and can therefore be discarded, yielding,

$$h_{ML} = \operatorname{argmax}_{h \in H} \sum_{i=1}^m -\frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

Maximizing this negative quantity is equivalent to minimizing the corresponding positive quantity.

$$h_{ML} = \operatorname{argmin}_{h \in H} \sum_{i=1}^m \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

Finally, we can again discard constants that are independent of h .

$$h_{ML} = \operatorname{argmin}_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2$$

Above equation shows that the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of the squared errors between the observed training values d_i and the hypothesis predictions $h(x_i)$.

Limitations: The above analysis considers noise only in the target value of the training example and does not consider noise in the attributes describing the instances themselves.

5. Maximum Likelihood Hypotheses for Predicting Probabilities

In the problem setting of the previous section we determined that the maximum likelihood hypothesis is the one that minimizes the sum of squared errors over the training examples. In this section we derive an analogous criterion for a second setting that is common in neural network learning: learning to predict probabilities.

Consider the setting in which we wish to learn a nondeterministic (probabilistic) function $f : X \rightarrow \{0, 1\}$, which has two discrete output values. For example, the instance space X might represent medical patients in terms of their symptoms, and the target function $f(x)$ might be 1 if the patient survives the disease and 0 if not. Alternatively, X might represent loan applicants in terms of their past credit history, and $f(x)$ might be 1 if the applicant successfully repays their next loan and 0 if not. In both of these cases we might well expect f to be probabilistic. For example, among a collection of patients exhibiting the same set of observable symptoms, we might find that 92% survive, and 8% do not. This unpredictability could arise from our inability to observe all the important distinguishing features of the patients, or from some genuinely probabilistic mechanism in the evolution of the disease. Whatever the source of the problem, the effect is that we have a target function $f(x)$ whose output is a probabilistic function of the input.

Given this problem setting, we might wish to learn a neural network (or other real-valued function approximator) whose output is the probability that $f(x) = 1$. In other words, we seek to learn the target function, $f' : X \rightarrow \{0, 1\}$, such that $f'(x) = P(f(x) = 1)$. In the above medical



patient example, if x is one of those indistinguishable patients of which 92% survive, then $f'(x) = 0.92$ whereas the probabilistic function $f(x)$ will be equal to 1 in 92% of cases and equal to 0 in the remaining 8%.

How can we learn f' using, say, a neural network? One obvious, brute-force way would be to first collect the observed frequencies of 1's and 0's for each possible value of x and to then train the neural network to output the target frequency for each x . As we shall see below, we can instead train a neural network directly from the observed training examples of f , yet still derive a maximum likelihood hypothesis for f' .

What criterion should we optimize in order to find a maximum likelihood hypothesis for f' in this setting? To answer this question, we must first obtain an expression for $P(D|h)$. Let us assume the training data D is of the form $D = \{(x_1, d_1) \dots (x_m, d_m)\}$, where d_i is the observed 0 or 1 value for $f(x_i)$. Recall that in the maximum likelihood, least-squared error analysis of the previous section, we made the simplifying assumption that the instances $(x_1 \dots x_m)$ were fixed. This enabled us to characterize the data by considering only the target values d_i . Although we could make a similar simplifying assumption in this case, let us avoid it here in order to demonstrate that it has no impact on the final outcome. Thus, treating both x_i and d_i as random variables, and assuming that each training example is drawn independently, we can write $P(D|h)$ as

$$P(D|h) = \prod_{i=1}^m P(x_i, d_i|h)$$

It is reasonable to assume, furthermore, that the probability of encountering any particular instance x_i is independent of the hypothesis h . For example, the probability that our training set contains a particular patient x_i is independent of our hypothesis about survival rates (though of course the survival d_i of the patient does depend strongly on h). When x is independent of h we can rewrite the above expression as

$$P(D|h) = \prod_{i=1}^m P(x_i, d_i|h) = \prod_{i=1}^m P(d_i|h, x_i) P(x_i) \quad \dots(8)$$

Now what is the probability $P(d_i | h, x_i)$ of observing $d_i = 1$ for a single instance x_i , given a world in which hypothesis h holds? Recall that h is our hypothesis regarding the target function, which computes this very probability.

Therefore, $P(d_i = 1 | h, x_i) = h(x_i)$, and in general

$$P(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ (1 - h(x_i)) & \text{if } d_i = 0 \end{cases} \quad \dots(9)$$

In order to substitute for $P(D|h)$ in (8), let us first "re-express it in a more mathematically manipulable form, as

$$P(d_i|h, x_i) = h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \quad \dots(10)$$

It is easy to verify that the expressions in Equations (6.9) and (6.10) are equivalent. Notice that when $d_i = 1$, the second term from Equation (6.10), $(1 - h(x_i))^{1-d_i}$, becomes equal to 1. Hence $P(d_i = 1|h, x_i) = h(x_i)$, which is equivalent to the first case in Equation (6.9). A similar analysis shows that the two equations are also equivalent when $d_i = 0$.

We can use Equation (6.10) to substitute for $P(d_i|h, x_i)$ in Equation (6.8) to obtain

$$P(D|h) = \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i) \quad (6.11)$$

Now we write an expression for the maximum likelihood hypothesis

$$h_{ML} = \operatorname{argmax}_{h \in H} \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i)$$

The last term is a constant independent of h , so it can be dropped

$$h_{ML} = \operatorname{argmax}_{h \in H} \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \quad (6.12)$$

The expression on the right side of Equation (12) can be seen as a generalization of the **Binomial distribution**. The expression in Equation (12) describes the probability that flipping each of m distinct coins will produce the outcome (d_1, \dots, d_m) , assuming that each coin x_i has probability $h(x_i)$ of producing a heads. Note the Binomial distribution is similar, but makes the additional assumption that the coins have identical probabilities of turning up heads (i.e., that $h(x_i) = h(x_j)$, for every i, j). In both cases we assume the outcomes of the coin flips are mutually independent—an assumption that fits our current setting.

As in earlier cases, we will find it easier to work with the log of the likelihood, yielding

$$h_{ML} = \operatorname{argmax}_{h \in H} \sum_{i=1}^m d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \quad \dots(13)$$

Equation (13) describes the quantity that must be maximized in order to obtain the maximum likelihood hypothesis in our current problem setting. This result is analogous to our earlier result showing that minimizing the sum of squared errors produces the maximum likelihood hypothesis in the earlier problem setting. Note the similarity between Equation (13) and the general form of the entropy function, $-x_i \log x_i$, discussed in Chapter 3. Because of this similarity, the negation of the above quantity is sometimes called the cross entropy.

6. Minimum Description Length Principle

Recall from Module-3 the discussion of Occam's razor, a popular inductive bias that can be summarized as “choose the shortest explanation for the observed data”. There we discussed several arguments in the long-standing debate regarding Occam's razor. Here we consider a



Bayesian perspective on this issue and a closely related principle called the Minimum Description Length (MDL) principle.

The Minimum Description Length principle is motivated by interpreting the definition of h_{MAP} in light of basic concepts from information theory. Consider again the now familiar definition of MAP

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(D|h)P(h)$$

$$h_{MAP} = \operatorname{argmax}_{h \in H} \log_2 P(D|h) + \log_2 P(h)$$

$$h_{MAP} = \operatorname{argmin}_{h \in H} -\log_2 P(D|h) - \log_2 P(h)$$

Above equation can be interpreted as a statement that short hypotheses are preferred, assuming a particular representation scheme for encoding hypotheses and data.

To explain this, let us introduce a basic result from information theory: Consider the problem of designing a code to transmit messages drawn at random, where the probability of encountering message i is p_i . We are interested here in the most compact code; that is, we are interested in the code that minimizes the expected number of bits we must transmit in order to encode a message drawn at random. Clearly, to minimize the expected code length we should assign shorter codes to messages that are more probable. Shannon and Weaver (1949) showed that the optimal code (i.e., the code that minimizes the message length) assigns $-\log_2 p_i$ bits to encode message i . We will refer to the number of bits required to encode message i using code C as the description length of message i with respect to C , which we denote by $L_C(i)$.

Let us interpret above equation in light of the above result from coding theory.

- $-\log_2 P(h)$ is the description length of h under the optimal encoding for the hypothesis space H . In other words, this is the size of the description of hypothesis h using this optimal representation. In our notation, $L_{C_H}(h) = -\log_2 P(h)$, where C_H is the optimal code for hypothesis space H .
- $-\log_2 P(D|h)$ is the description length of the training data D given hypothesis h , under its optimal encoding. In our notation, $L_{C_{D|h}}(D|h) = -\log_2 P(D|h)$, where $C_{D|h}$ is the optimal code for describing data D assuming that both the sender and receiver know the hypothesis h .
- Therefore we can rewrite Equation (6.16) to show that h_{MAP} is the hypothesis h that minimizes the sum given by the description length of the hypothesis plus the description length of the data given the hypothesis.

$$h_{MAP} = \operatorname{argmin}_h L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

where C_H and $C_{D|h}$ are the optimal encodings for H and for D given h , respectively.

The Minimum Description Length (MDL) principle recommends choosing the hypothesis that minimizes the sum of these two description lengths. Of course, to apply this principle in practice we must choose specific encodings or representations appropriate for the given



learning task. Assuming we use the codes C_1 and C_2 to represent the hypothesis and the data given the hypothesis, we can state the MDL principle as

Minimum Description Length principle: Choose h_{MDL} where

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} L_{C_1}(h) + L_{C_2}(D|h)$$

The above analysis shows that if we choose C_1 to be the optimal encoding of hypotheses C_H , and if we choose C_2 to be the optimal encoding $C_{D|h}$ then $h_{MDL} = h_{MAP}$.

Intuitively, we can think of the MDL principle as recommending the shortest method for re-encoding the training data, where we count both the size of the hypothesis and any additional cost of encoding the data given this hypothesis.

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.

7. 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.

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 (a_1, a_2, \dots, a_n) . 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 (a_1, a_2, \dots, a_n) that describe the instance.

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

We can use Bayes theorem to rewrite this expression as

$$\begin{aligned} v_{MAP} &= \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)} \\ &= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2, \dots, a_n | v_j) P(v_j) \end{aligned} \quad \dots(19)$$

Now we could attempt to estimate the two terms in Equation (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 no. of these terms = no. of possible instances * no. of possible target values.)



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.

$$\text{Naive Bayes classifier: } v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j) \quad \dots(20)$$

where v_{NB} denotes the target value output by the naive Bayes classifier. (Here total terms are only n)

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 (20).

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. Instead, the hypothesis is formed without searching, simply by counting the frequency of various data combinations within the training examples.

Illustration: Consider the following data.

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Let us use the naive Bayes classifier and the training data from this table to classify the following novel instance:

(*Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong*)

Our task is to predict the target value (yes or no) of the target concept *PlayTennis* for this new instance. Instantiating Equation (20) to fit the current task, the target value v_{NB} is given by

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



The probabilities of the different target values can easily be estimated based on their frequencies over the 14 training examples

$$P(\text{PlayTennis} = \text{yes}) = 9/14 = .64$$

$$P(\text{PlayTennis} = \text{no}) = 5/14 = .36$$

$$P(\text{Wind} = \text{strong} | \text{PlayTennis} = \text{yes}) = 3/9 = .33$$

$$P(\text{Wind} = \text{strong} | \text{PlayTennis} = \text{no}) = 3/5 = .60 \quad \dots \text{and so on (remaining 10)}$$

$$\text{We have } P(\text{yes}) P(\text{sunny}|\text{yes}) P(\text{cool}|\text{yes}) P(\text{high}|\text{yes}) P(\text{strong}|\text{yes}) = .0053$$

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

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 $0.0206 / (0.0206 + 0.0053) = 0.795$

Estimating Probabilities: In the above computations, conditional fraction

$$P(\text{Wind} = \text{strong} | \text{PlayTennis} = \text{no}) = 3/5 = n_c/n$$

from the training samples provides a good estimate of the probability in many cases, but estimate is poor when n is very small or n_c is 0. There are two difficulties. 1) First, n_c/n produces a biased underestimate of the probability. 2) Second, when this probability estimate is zero, this probability term will dominate the Bayes classifier if the future query contains $\text{Wind} = \text{strong}$. The reason is that the quantity calculated in Equation (20) requires multiplying all the other probability terms by these zero values.

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

$$\text{m-estimate of probability: } \frac{n_c + mp}{n + m} \quad \dots(22)$$

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 $p = 1/k$. For example, in estimating $P(\text{Wind} = \text{strong} | \text{PlayTennis} = \text{no})$ 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 n_c/n . If both n and m are nonzero, then the observed fraction 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 (22) can be interpreted as augmenting the n actual observations by an additional m virtual samples distributed according to p .



8. Bayesian Belief Networks

The naive Bayes classifier makes significant use of the assumption that the values of the attributes a_1, \dots, 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 (or Bayesian 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. Bayesian networks allow stating conditional independence assumptions that apply to subsets of the variables. They are an active focus of current research, and a variety of algorithms have been proposed for learning them and for using them for inference.

Bayesian networks describes the probability distribution over a set of variables. The probability distribution over these joint variables are called the joint probability distribution. The joint probability distribution specifies the probability for each of the possible variable bindings for the tuple (Y_1, \dots, Y_2) . A Bayesian belief network describes the joint probability distribution for a set of variables.

8.1 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 $X_1 \dots X_i$ is conditionally independent of the set of variables $Y_1 \dots Y_m$ given the set of variables $Z_1 \dots Z_n$, if

$$P(X_1 \dots X_i | Y_1 \dots Y_m, Z_1 \dots Z_n) = P(X_1 \dots X_i | Z_1 \dots Z_n)$$

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 (20) as follows

$$P(A_1, A_2|V) = P(A_1|A_2, V)P(A_2|V) \quad (6.23)$$

$$= P(A_1|V)P(A_2|V) \quad (6.24)$$

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 $P(A_1 | A_2, V) = P(A_1 | V)$.

8.2 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.

1. First, the network arcs represent the assertion that the variable is conditionally independent of its non-descendants 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*.
2. 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 (y_1, \dots, y_n) to the tuple of network variables (Y_1, \dots, Y_n) can be computed by the formula

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

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 .

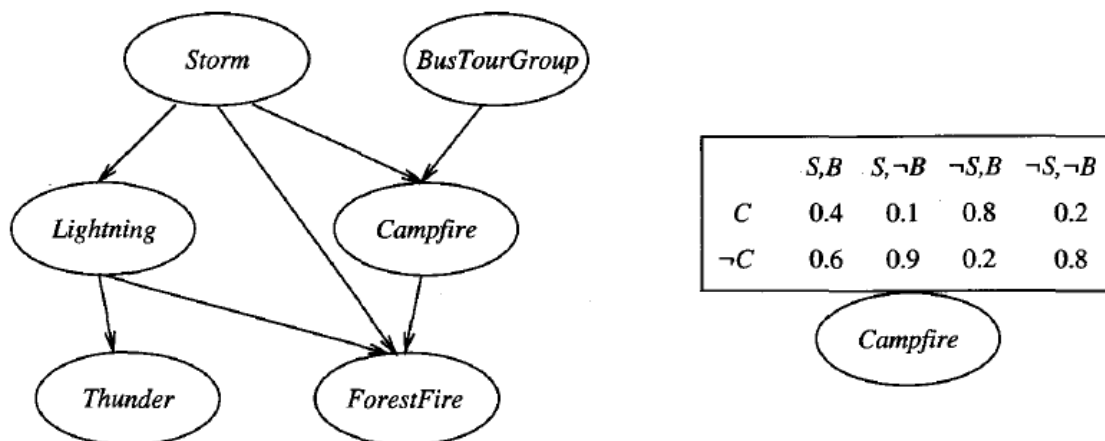


FIGURE 6.3

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*.

To illustrate, the Bayesian network in Figure 6.3 represents the joint probability distribution over the boolean variables *Storm*, *Lightning*, *Thunder*, *ForestFire*, *Campfire*, and *BusTourGroup*. Consider the node *Campfire*. The network nodes and arcs represent the



assertion that *Campfire* is conditionally independent of its non-descendants *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

$$P(\text{Campfire} = \text{True} \mid \text{Storm} = \text{True}, \text{BusTourGroup} = \text{True}) = 0.4$$

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*.

8.3 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.

8.4 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.

Note: Refer lecture slides for more examples/illustrations

9. The EM Algorithm

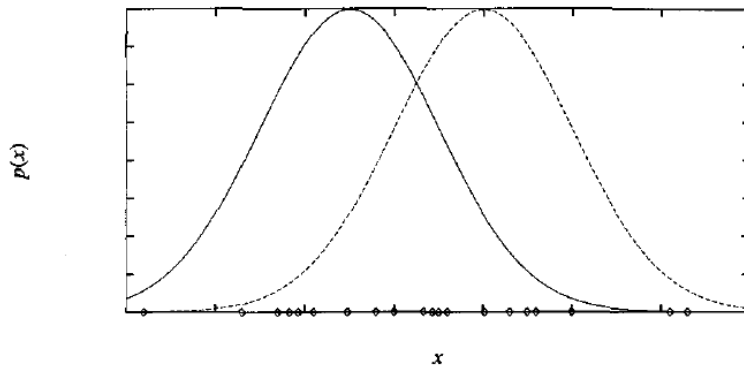
In many practical learning settings, only a subset of the relevant instance features might be observable. For example, in training or using the Bayesian belief network, we might have data where only a subset of the network variables *Storm*, *Lightning*, *Thunder*, *ForestFire*, *Campfire*, and *BusTourGroup* have been observed. Many approaches have been proposed to handle the problem of learning in the presence of unobserved variables. 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.

Application: The EM algorithm has been used to train Bayesian belief networks (Heckerman 1995) as well as radial basis function neural networks. 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).

9.1 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 , known value. The learning task is to output a hypothesis $h = (\mu_1, \dots, \mu_k)$

**FIGURE 6.4**

Instances generated by a mixture of two Normal distributions with identical variance σ . The instances are shown by the points along the x axis. If the means of the Normal distributions are unknown, the EM algorithm can be used to search for their maximum likelihood estimates.

that describes the means of each of the k distributions. We would like to find a maximum likelihood hypothesis for these means; that is, a hypothesis h that maximizes $p(D|h)$.

Note it is easy to calculate the maximum likelihood hypothesis for the mean of a single Normal distribution given the observed data instances x_1, x_2, \dots, x_m drawn from this single distribution. Earlier where we showed that the maximum likelihood hypothesis is the one that minimizes the sum of squared errors over the m training instances. Now the problem of finding the mean of a single distribution is just a special case of the problem discussed. Restating using our current notation, we have

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^m (x_i - \mu)^2 \quad \dots(6.27)$$

In this case, the sum of squared errors is minimized by the sample mean

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^m x_i \quad \dots (6.28)$$

Our problem here, however, involves a mixture of k different Normal distributions, and we cannot observe which instances were generated by which distribution. Thus, we have a prototypical example of a problem involving hidden variables. In the example of Figure 6.4, we can think of the full description of each instance as the triple (x_i, z_{i1}, z_{i2}) , where x_i is the observed value of the i^{th} instance and where z_{i1} and z_{i2} indicate which of the two Normal distributions was used to generate the value x_i . In particular, z_{ij} has the value 1 if x_i was created by the j^{th} Normal distribution and 0 otherwise. Here x_i is the observed variable in the description of the instance, and z_{i1} and z_{i2} are hidden variables. If the values of z_{i1} and z_{i2} were observed, we could use Equation (6.27) to solve for the means μ_1 and μ_2 . Because they are not, we will instead use the EM algorithm.

Applied to our k -means problem the EM algorithm searches for a maximum likelihood hypothesis by repeatedly re-estimating the expected values of the hidden variables z_{ij} given its



current hypothesis $(\mu_1 \dots \mu_k)$, then recalculating the maximum likelihood hypothesis using these expected values for the hidden variables.

We will first describe this instance of the EM algorithm, and later state the EM algorithm in its general form.

Applied to the problem of estimating the two means for Figure 6.4, the EM algorithm first initializes the hypothesis to $h = (\mu_1, \mu_2)$, where μ_1 and μ_2 are arbitrary initial values. It then iteratively re-estimates h by repeating the following two steps until the procedure converges to a stationary value for h .

Step 1: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij} , assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.

Step 2: Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated in Step 1. Then replace the hypothesis $h = \langle \mu_1, \mu_2 \rangle$ by the new hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$ and iterate.

Let us examine how both of these steps can be implemented in practice. Step 1 must calculate the expected value of each z_i . This $E[z_{ij}]$ is just the probability that instance x_i was generated by the j^{th} Normal distribution.

$$\begin{aligned} E[z_{ij}] &= \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^2 p(x = x_i | \mu = \mu_n)} \\ &= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^2 e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \end{aligned}$$

Thus, the first step is implemented by substituting the current values (μ_1, μ_2) and the observed x_i into the above expression.

In the second step we use the $E[z_{ij}]$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = (\mu'_1, \mu'_2)$. As we will discuss later, the maximum likelihood hypothesis in this case is given by

$$\mu_j \leftarrow \frac{\sum_{i=1}^m E[z_{ij}] x_i}{\sum_{i=1}^m E[z_{ij}]}$$

Note this expression is similar to the sample mean from Equation (6.28) that is used to estimate μ for a single Normal distribution. Our new expression is just the weighted sample mean for μ_j , with each instance weighted by the expectation $E[z_{ij}]$ that it was generated by the j^{th} Normal distribution.

The above algorithm for estimating the means of a mixture of k Normal distributions illustrates the essence of the EM approach: **The current hypothesis is used to estimate the unobserved variables, and the expected values of these variables are then used to calculate an improved hypothesis.** It can be proved that on each iteration through this loop, the EM



algorithm increases the likelihood $P(D|h)$ unless it is at a local maximum. The algorithm thus converges to a local maximum likelihood hypothesis for (μ_1, μ_2) .

9.2 General Statement of EM Algorithm

Above we described an EM algorithm for the problem of estimating means of a mixture of Normal distributions. More generally, the EM algorithm can be applied in many settings where we wish to estimate some set of parameters θ that describe an underlying probability distribution, given only the observed portion of the full data produced by this distribution. In the above two-means example the parameters of interest were $\theta = (\mu_1, \mu_2)$, and the full data were the triples (x_i, z_{i1}, z_{i2}) of which only the x_i were observed. In general let $X = \{x_1, \dots, x_m\}$ denote the observed data in a set of m independently drawn instances, let $Z = \{z_1, \dots, z_m\}$ denote the unobserved data in these same instances, and let $Y = X \cup Z$ denote the full data. Note the unobserved Z can be treated as a random variable whose probability distribution depends on the unknown parameters θ and on the observed data X . Similarly, Y is a random variable because it is defined in terms of the random variable Z . In the remainder of this section we describe the general form of the EM algorithm. We use h to denote the current hypothesized values of the parameters θ , and h' to denote the revised hypothesis that is estimated on each iteration of the EM algorithm.

The EM algorithm searches for the maximum likelihood hypothesis h' by seeking the h' that maximizes $E[\ln P(Y|h')]$. This expected value is taken over the probability distribution governing Y , which is determined by the unknown parameters θ . Let us consider exactly what this expression signifies. First, $P(Y|h')$ is the likelihood of the full data Y given hypothesis h' . It is reasonable that we wish to find a h' that maximizes some function of this quantity. Second, maximizing the logarithm of this quantity $\ln(P(Y|h'))$ also maximizes $P(Y|h')$, as we have discussed on several occasions already. Third, we introduce the expected value $E[\ln P(Y|h')]$ because the full data Y is itself a random variable. Given that the full data Y is a combination of the observed data X and unobserved data Z , we must average over the possible values of the unobserved Z , weighting each according to its probability. In other words we take the expected value $E[\ln P(Y|h')]$ over the probability distribution governing the random variable Y . The distribution governing Y is determined by the completely known values for X , plus the distribution governing Z .

What is the probability distribution governing Y ? In general, we will not know this distribution because it is determined by the parameters θ that we are trying to estimate. Therefore, the EM algorithm uses its current hypothesis h in place of the actual parameters θ to estimate the distribution governing Y . Let us define a function $Q(h'|h)$ that gives $E[\ln P(Y|h')]$ as a function of h' , under the assumption that $\theta = h$ and given the observed portion X of the full data Y .

$$Q(h'|h) = E[\ln p(Y|h')|h, X]$$

We write this function Q in the form $Q(h'|h)$ to indicate that it is defined in part by the assumption that the current hypothesis h is equal to θ . In its general form, the EM algorithm repeats the following two steps until convergence:

Step 1: Estimation (E) step: Calculate $Q(h'|h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y .

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

Step 2: Maximization (M) step: Replace hypothesis h by the hypothesis h' that maximizes this Q function.

$$h \leftarrow \underset{h'}{\operatorname{argmax}} Q(h'|h)$$

When the function Q is continuous, the EM algorithm converges to a stationary point of the likelihood function $P(Y|h')$. When this likelihood function has a single maximum, EM will converge to this global maximum likelihood estimate for h' . Otherwise, it is guaranteed only to converge to a local maximum. In this respect, EM shares some of the same limitations as other optimization methods such as gradient descent, line search, and conjugate gradient discussed in Chapter 4.

9.3 Derivation of the k Means Algorithm

To illustrate the general EM algorithm, let us use it to derive the algorithm given in Section 6.12.1 for estimating the means of a mixture of k Normal distributions. As discussed above, the k -means problem is to estimate the parameters $\theta = \langle \mu_1 \dots \mu_k \rangle$ that define the means of the k Normal distributions. We are given the observed data $X = \{\langle x_i \rangle\}$. The hidden variables $Z = \{\langle z_{i1}, \dots, z_{ik} \rangle\}$ in this case indicate which of the k Normal distributions was used to generate x_i .

To apply EM we must derive an expression for $Q(h|h')$ that applies to our k -means problem. First, let us derive an expression for $\ln p(Y|h')$. Note the probability $p(y_i|h')$ of a single instance $y_i = \langle x_i, z_{i1}, \dots, z_{ik} \rangle$ of the full data can be written

$$p(y_i|h') = p(x_i, z_{i1}, \dots, z_{ik}|h') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2} \sum_{j=1}^k z_{ij}(x_i - \mu'_j)^2}$$

To verify this note that only one of the z_{ij} can have the value 1, and all others must be 0. Therefore, this expression gives the probability distribution for x_i generated by the selected Normal distribution. Given this probability for a single instance $p(y_i|h')$, the logarithm of the probability $\ln P(Y|h')$ for all m instances in the data is

$$\begin{aligned} \ln P(Y|h') &= \ln \prod_{i=1}^m p(y_i|h') = \sum_{i=1}^m \ln p(y_i|h') \\ &= \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k z_{ij}(x_i - \mu'_j)^2 \right) \end{aligned}$$

Finally we must take the expected value of this $\ln P(Y|h')$ over the probability distribution governing Y or, equivalently, over the distribution governing the unobserved components z_{ij} of Y . Note the above expression for $\ln P(Y|h')$ is a linear function of these z_{ij} . In general, for any function $f(z)$ that is a *linear* function of z , the following equality holds

$$E[f(z)] = f(E[z])$$

This general fact about linear functions allows us to write

$$\begin{aligned} E[\ln P(Y|h')] &= E \left[\sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k z_{ij} (x_i - \mu'_j)^2 \right) \right] \\ &= \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \right) \end{aligned}$$

To summarize, the function $Q(h'|h)$ for the k means problem is

$$Q(h'|h) = \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \right)$$

where $h' = \langle \mu'_1, \dots, \mu'_k \rangle$ and where $E[z_{ij}]$ is calculated based on the current hypothesis h and observed data X . As discussed earlier

$$E[z_{ij}] = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^k e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \quad (6.29)$$

Thus, the first (estimation) step of the EM algorithm defines the Q function based on the estimated $E[z_{ij}]$ terms. The second (maximization) step then finds the values μ'_1, \dots, μ'_k that maximize this Q function. In the current case

$$\begin{aligned} \operatorname{argmax}_{h'} Q(h'|h) &= \operatorname{argmax}_{h'} \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \right) \\ &= \operatorname{argmin}_{h'} \sum_{i=1}^m \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \end{aligned} \quad (6.30)$$

Thus, the maximum likelihood hypothesis here minimizes a weighted sum of squared errors, where the contribution of each instance x_i to the error that defines μ'_j is weighted by $E[z_{ij}]$. The quantity given by Equation (6.30) is minimized by setting each μ'_j to the weighted sample mean

$$\mu_j \leftarrow \frac{\sum_{i=1}^m E[z_{ij}] x_i}{\sum_{i=1}^m E[z_{ij}]} \quad (6.31)$$

Note that Equations (6.29) and (6.31) define the two steps in the k -means algorithm described in Section 6.12.1.



10. Summary

- Bayesian methods provide the basis for probabilistic learning methods that accommodate (and require) knowledge about the prior probabilities of alternative hypotheses and about the probability of observing various data given the hypothesis. Bayesian methods allow assigning a posterior probability to each candidate hypothesis, based on these assumed priors and the observed data.
- Bayesian methods can be used to determine the most probable hypothesis given the data-the maximum a posteriori (MAP) hypothesis. This is the optimal hypothesis in the sense that no other hypothesis is more likely.
- The naive Bayes classifier is a Bayesian learning method that has been found to be useful in many practical applications. It is called "naive" because it incorporates the simplifying assumption that attribute values are conditionally independent, given the classification of the instance. When this assumption is met, the naive Bayes classifier outputs the MAP classification. Even when this assumption is not met, as in the case of learning to classify text, the naive Bayes classifier is often quite effective. Bayesian belief networks provide a more expressive representation for sets of conditional independence assumptions among subsets of the attributes.
- The framework of Bayesian reasoning can provide a useful basis for analyzing certain learning methods that do not directly apply Bayes theorem. For example, under certain conditions it can be shown that minimizing the squared error when learning a real-valued target function corresponds to computing the maximum likelihood hypothesis.
- The Minimum Description Length principle recommends choosing the hypothesis that minimizes the description length of the hypothesis plus the description length of the data given the hypothesis. Bayes theorem and basic results from information theory can be used to provide a rationale for this principle.
- In many practical learning tasks, some of the relevant instance variables may be unobservable. The EM algorithm provides a quite general approach to learning in the presence of unobservable variables. This algorithm begins with an arbitrary initial hypothesis. It then repeatedly calculates the expected values of the hidden variables (assuming the current hypothesis is correct), and then recalculates the maximum likelihood hypothesis (assuming the hidden variables have the expected values calculated by the first step). This procedure converges to a local maximum likelihood hypothesis, along with estimated values for the hidden variables.
