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.

"Machine Learning" by Tom Mitchell, 1997

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 $\langle a_1, a_2 \ldots a_n \rangle$. 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 $\langle a_1, a_2 \ldots a_n \rangle$ that describe the instance.

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

We can use Bayes theorem to rewrite this expression as

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

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 $P(a_1, a_2 \ldots a_n | v_j)$ 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 $a_1, a_2 \ldots a_n$ is just the product of the probabilities for the individual attributes: $P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$. Substituting this into Equation (6.19), we have the approach used by the naive Bayes classifier.

Naive Bayes classifier:

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

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

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

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:**

$$\frac{n_c + mp}{n + m} \qquad (6.22)$$

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