

A tree is interactively constructed as follows. The user visualizes the multidimensional data in the Data Interaction window and selects a splitting attribute and one or more split-points. The current decision tree in the Knowledge Interaction window is expanded. The user selects a node of the decision tree. The user may either assign a class label to the node (which makes the node a leaf) or request the visualization of the training data corresponding to the node. This leads to a new visualization of every attribute except the ones used for splitting criteria on the same path from the root. The interactive process continues until a class has been assigned to each leaf of the decision tree.

The trees constructed with PBC were compared with trees generated by the CART, C4.5, and SPRINT algorithms from various data sets. The trees created with PBC were of comparable accuracy with the tree from the algorithmic approaches, yet were significantly smaller and, thus, easier to understand. Users can use their domain knowledge in building a decision tree, but also gain a deeper understanding of their data during the construction process.

8.3 Bayes Classification Methods

“*What are Bayesian classifiers?*” Bayesian classifiers are statistical classifiers. They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

Bayesian classification is based on Bayes’ theorem, described next. Studies comparing classification algorithms have found a simple Bayesian classifier known as the *naïve Bayesian classifier* to be comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases.

Naïve Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called *class-conditional independence*. It is made to simplify the computations involved and, in this sense, is considered “naïve.”

Section 8.3.1 reviews basic probability notation and Bayes’ theorem. In Section 8.3.2 you will learn how to do naïve Bayesian classification.

8.3.1 Bayes’ Theorem

Bayes’ theorem is named after Thomas Bayes, a nonconformist English clergyman who did early work in probability and decision theory during the 18th century. Let \mathbf{X} be a data tuple. In Bayesian terms, \mathbf{X} is considered “evidence.” As usual, it is described by measurements made on a set of n attributes. Let H be some hypothesis such as that the data tuple \mathbf{X} belongs to a specified class C . For classification problems, we want to determine $P(H|\mathbf{X})$, the probability that the hypothesis H holds given the “evidence” or observed data tuple \mathbf{X} . In other words, we are looking for the probability that tuple \mathbf{X} belongs to class C , given that we know the attribute description of \mathbf{X} .

$P(H|X)$ is the **posterior probability**, or a *posteriori probability*, of H conditioned on X . For example, suppose our world of data tuples is confined to customers described by the attributes *age* and *income*, respectively, and that X is a 35-year-old customer with an income of \$40,000. Suppose that H is the hypothesis that our customer will buy a computer. Then $P(H|X)$ reflects the probability that customer X will buy a computer given that we know the customer's age and income.

In contrast, $P(H)$ is the **prior probability**, or a *priori probability*, of H . For our example, this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter. The posterior probability, $P(H|X)$, is based on more information (e.g., customer information) than the prior probability, $P(H)$, which is independent of X .

Similarly, $P(X|H)$ is the posterior probability of X conditioned on H . That is, it is the probability that a customer, X , is 35 years old and earns \$40,000, given that we know the customer will buy a computer.

$P(X)$ is the prior probability of X . Using our example, it is the probability that a person from our set of customers is 35 years old and earns \$40,000.

"How are these probabilities estimated?" $P(H)$, $P(X|H)$, and $P(X)$ may be estimated from the given data, as we shall see next. **Bayes' theorem** is useful in that it provides a way of calculating the posterior probability, $P(H|X)$, from $P(H)$, $P(X|H)$, and $P(X)$. Bayes' theorem is

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}. \quad (8.10)$$

Now that we have that out of the way, in the next section, we will look at how Bayes' theorem is used in the naïve Bayesian classifier.

8.3.2 Naïve Bayesian Classification

The **naïve Bayesian** classifier, or **simple Bayesian** classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n -dimensional attribute vector, $X = (x_1, x_2, \dots, x_n)$, depicting n measurements made on the tuple from n attributes, respectively, A_1, A_2, \dots, A_n .
2. Suppose that there are m classes, C_1, C_2, \dots, C_m . Given a tuple, X , the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X . That is, the naïve Bayesian classifier predicts that tuple X belongs to the class C_i if and only if

$$P(C_i|X) > P(C_j|X) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

Thus, we maximize $P(C_i|X)$. The class C_i for which $P(C_i|X)$ is maximized is called the *maximum posteriori hypothesis*. By Bayes' theorem (Eq. 8.10),

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}. \quad (8.11)$$

3. As $P(\mathbf{X})$ is constant for all classes, only $P(\mathbf{X}|C_i)P(C_i)$ needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = \dots = P(C_m)$, and we would therefore maximize $P(\mathbf{X}|C_i)$. Otherwise, we maximize $P(\mathbf{X}|C_i)P(C_i)$. Note that the class prior probabilities may be estimated by $P(C_i) = |C_{i,D}|/|D|$, where $|C_{i,D}|$ is the number of training tuples of class C_i in D .
4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(\mathbf{X}|C_i)$. To reduce computation in evaluating $P(\mathbf{X}|C_i)$, the naïve assumption of **class-conditional independence** is made. This presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$\begin{aligned}
 P(\mathbf{X}|C_i) &= \prod_{k=1}^n P(x_k|C_i) \\
 &= P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i).
 \end{aligned} \tag{8.12}$$

We can easily estimate the probabilities $P(x_1|C_i), P(x_2|C_i), \dots, P(x_n|C_i)$ from the training tuples. Recall that here x_k refers to the value of attribute A_k for tuple \mathbf{X} . For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute $P(\mathbf{X}|C_i)$, we consider the following:

- (a) If A_k is categorical, then $P(x_k|C_i)$ is the number of tuples of class C_i in D having the value x_k for A_k , divided by $|C_{i,D}|$, the number of tuples of class C_i in D .
- (b) If A_k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ , defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{8.13}$$

so that

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}). \tag{8.14}$$

These equations may appear daunting, but hold on! We need to compute μ_{C_i} and σ_{C_i} , which are the mean (i.e., average) and standard deviation, respectively, of the values of attribute A_k for training tuples of class C_i . We then plug these two quantities into Eq. (8.13), together with x_k , to estimate $P(x_k|C_i)$.

For example, let $\mathbf{X} = (35, \$40,000)$, where A_1 and A_2 are the attributes *age* and *income*, respectively. Let the class label attribute be *buys_computer*. The associated class label for \mathbf{X} is *yes* (i.e., *buys_computer* = *yes*). Let's suppose that *age* has not been discretized and therefore exists as a continuous-valued attribute. Suppose that from the training set, we find that customers in D who buy a computer are

38 ± 12 years of age. In other words, for attribute *age* and this class, we have $\mu = 38$ years and $\sigma = 12$. We can plug these quantities, along with $x_1 = 35$ for our tuple \mathbf{X} , into Eq. (8.13) to estimate $P(\text{age} = 35 | \text{buys_computer} = \text{yes})$. For a quick review of mean and standard deviation calculations, please see Section 2.2.

5. To predict the class label of \mathbf{X} , $P(\mathbf{X}|C_i)P(C_i)$ is evaluated for each class C_i . The classifier predicts that the class label of tuple \mathbf{X} is the class C_i if and only if

$$P(\mathbf{X}|C_i)P(C_i) > P(\mathbf{X}|C_j)P(C_j) \quad \text{for } 1 \leq j \leq m, j \neq i. \quad (8.15)$$

In other words, the predicted class label is the class C_i for which $P(\mathbf{X}|C_i)P(C_i)$ is the maximum.

“How effective are Bayesian classifiers?” Various empirical studies of this classifier in comparison to decision tree and neural network classifiers have found it to be comparable in some domains. In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case, owing to inaccuracies in the assumptions made for its use, such as class-conditional independence, and the lack of available probability data.

Bayesian classifiers are also useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes’ theorem. For example, under certain assumptions, it can be shown that many neural network and curve-fitting algorithms output the *maximum posteriori* hypothesis, as does the naïve Bayesian classifier.

Example 8.4 Predicting a class label using naïve Bayesian classification. We wish to predict the class label of a tuple using naïve Bayesian classification, given the same training data as in Example 8.3 for decision tree induction. The training data were shown earlier in Table 8.1. The data tuples are described by the attributes *age*, *income*, *student*, and *credit_rating*. The class label attribute, *buys_computer*, has two distinct values (namely, {yes, no}). Let C_1 correspond to the class *buys_computer* = yes and C_2 correspond to *buys_computer* = no. The tuple we wish to classify is

$$\mathbf{X} = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$$

We need to maximize $P(\mathbf{X}|C_i)P(C_i)$, for $i = 1, 2$. $P(C_i)$, the prior probability of each class, can be computed based on the training tuples:

$$P(\text{buys_computer} = \text{yes}) = 9/14 = 0.643$$

$$P(\text{buys_computer} = \text{no}) = 5/14 = 0.357$$

To compute $P(\mathbf{X}|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$$P(\text{age} = \text{youth} | \text{buys_computer} = \text{yes}) = 2/9 = 0.222$$

$$P(\text{age} = \text{youth} | \text{buys_computer} = \text{no}) = 3/5 = 0.600$$

$$P(\text{income} = \text{medium} | \text{buys_computer} = \text{yes}) = 4/9 = 0.444$$

$$P(\text{income} = \text{medium} | \text{buys_computer} = \text{no}) = 2/5 = 0.400$$

$$P(\text{student} = \text{yes} | \text{buys_computer} = \text{yes}) = 6/9 = 0.667$$

$$P(\text{student} = \text{yes} \mid \text{buys_computer} = \text{no}) = 1/5 = 0.200$$

$$P(\text{credit_rating} = \text{fair} \mid \text{buys_computer} = \text{yes}) = 6/9 = 0.667$$

$$P(\text{credit_rating} = \text{fair} \mid \text{buys_computer} = \text{no}) = 2/5 = 0.400$$

Using these probabilities, we obtain

$$\begin{aligned} P(\mathbf{X} \mid \text{buys_computer} = \text{yes}) &= P(\text{age} = \text{youth} \mid \text{buys_computer} = \text{yes}) \\ &\quad \times P(\text{income} = \text{medium} \mid \text{buys_computer} = \text{yes}) \\ &\quad \times P(\text{student} = \text{yes} \mid \text{buys_computer} = \text{yes}) \\ &\quad \times P(\text{credit_rating} = \text{fair} \mid \text{buys_computer} = \text{yes}) \\ &= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044. \end{aligned}$$

Similarly,

$$P(\mathbf{X} \mid \text{buys_computer} = \text{no}) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

To find the class, C_i , that maximizes $P(\mathbf{X} \mid C_i)P(C_i)$, we compute

$$P(\mathbf{X} \mid \text{buys_computer} = \text{yes})P(\text{buys_computer} = \text{yes}) = 0.044 \times 0.643 = 0.028$$

$$P(\mathbf{X} \mid \text{buys_computer} = \text{no})P(\text{buys_computer} = \text{no}) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts *buys_computer = yes* for tuple \mathbf{X} . ■

“What if I encounter probability values of zero?” Recall that in Eq. (8.12), we estimate $P(\mathbf{X} \mid C_i)$ as the product of the probabilities $P(x_1 \mid C_i)$, $P(x_2 \mid C_i)$, ..., $P(x_n \mid C_i)$, based on the assumption of class-conditional independence. These probabilities can be estimated from the training tuples (step 4). We need to compute $P(\mathbf{X} \mid C_i)$ for *each* class ($i = 1, 2, \dots, m$) to find the class C_i for which $P(\mathbf{X} \mid C_i)P(C_i)$ is the maximum (step 5). Let’s consider this calculation. For each attribute–value pair (i.e., $A_k = x_k$, for $k = 1, 2, \dots, n$) in tuple \mathbf{X} , we need to count the number of tuples having that attribute–value pair, per class (i.e., per C_i , for $i = 1, \dots, m$). In Example 8.4, we have two classes ($m = 2$), namely *buys_computer = yes* and *buys_computer = no*. Therefore, for the attribute–value pair *student = yes* of \mathbf{X} , say, we need two counts—the number of customers who are students and for which *buys_computer = yes* (which contributes to $P(\mathbf{X} \mid \text{buys_computer} = \text{yes})$) and the number of customers who are students and for which *buys_computer = no* (which contributes to $P(\mathbf{X} \mid \text{buys_computer} = \text{no})$).

But what if, say, there are no training tuples representing students for the class *buys_computer = no*, resulting in $P(\text{student} = \text{yes} \mid \text{buys_computer} = \text{no}) = 0$? In other words, what happens if we should end up with a probability value of zero for some $P(x_k \mid C_i)$? Plugging this zero value into Eq. (8.12) would return a zero probability for $P(\mathbf{X} \mid C_i)$, even though, without the zero probability, we may have ended up with a high probability, suggesting that \mathbf{X} belonged to class C_i ! A zero probability cancels the effects of all the other (posteriori) probabilities (on C_i) involved in the product.

There is a simple trick to avoid this problem. We can assume that our training database, D , is so large that adding one to each count that we need would only make a negligible difference in the estimated probability value, yet would conveniently avoid the

case of probability values of zero. This technique for probability estimation is known as the **Laplacian correction** or **Laplace estimator**, named after Pierre Laplace, a French mathematician who lived from 1749 to 1827. If we have, say, q counts to which we each add one, then we must remember to add q to the corresponding denominator used in the probability calculation. We illustrate this technique in Example 8.5.

Example 8.5 Using the Laplacian correction to avoid computing probability values of zero. Suppose that for the class *buys_computer = yes* in some training database, D , containing 1000 tuples, we have 0 tuples with *income = low*, 990 tuples with *income = medium*, and 10 tuples with *income = high*. The probabilities of these events, without the Laplacian correction, are 0, 0.990 (from 990/1000), and 0.010 (from 10/1000), respectively. Using the Laplacian correction for the three quantities, we pretend that we have 1 more tuple for each income-value pair. In this way, we instead obtain the following probabilities (rounded up to three decimal places):

$$\frac{1}{1003} = 0.001, \frac{991}{1003} = 0.988, \text{ and } \frac{11}{1003} = 0.011,$$

respectively. The “corrected” probability estimates are close to their “uncorrected” counterparts, yet the zero probability value is avoided. ■

8.4 Rule-Based Classification

In this section, we look at rule-based classifiers, where the learned model is represented as a set of IF-THEN rules. We first examine how such rules are used for classification (Section 8.4.1). We then study ways in which they can be generated, either from a decision tree (Section 8.4.2) or directly from the training data using a *sequential covering algorithm* (Section 8.4.3).

8.4.1 Using IF-THEN Rules for Classification

Rules are a good way of representing information or bits of knowledge. A **rule-based classifier** uses a set of IF-THEN rules for classification. An **IF-THEN** rule is an expression of the form

IF *condition* THEN *conclusion*.

An example is rule R_1 ,

R_1 : IF *age = youth* AND *student = yes* THEN *buys_computer = yes*.

The “IF” part (or left side) of a rule is known as the **rule antecedent** or **precondition**. The “THEN” part (or right side) is the **rule consequent**. In the rule antecedent, the condition consists of one or more *attribute tests* (e.g., *age = youth* and *student = yes*)