Predicting a class label using naïve Bayes classification:

We wish to predict the class label of a tuple using naïve Bayesian classification, given the training data and frequency of occurrence to calculate conditional probabilities. The data tuples are described the attributes age, income, student, and credit_rating. The class label attribute, buys_computer, has two distinct values, namely {yes, no}. Let C1 correspond to the class buys_computer = yes and C2 correspond to buys_computer = no. The tuple we wish to classify is:

X = (age = youth, income = medium, student = yes, credit_rating = fair)

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

P(buys_computer = yes) = 9/14 = 0.643
P(buys_computer = no) = 5/14 = 0.357

To compute (PX|Ci), for i=1,2, we compute the following conditional probabilities (which come from simply counting the occurrences of the conditions in the dataset, thereby estimating probability through freqency):

P(age = youth | buys_computer = yes)	= 2/9	= 0.222
P(age = youth | buys_computer = no)	= 3/5	= 0.600
P(income = medium | buys_computer = yes)	= 4/9	= 0.444
P(income = medium | buys_computer = no)	= 2/5	= 0.400
P(student = yes | buys_computer = yes)	= 6/9	= 0.667
P(student = yes | buys_computer = no)	= 1/5	= 0.200
P(credit_rating = fair | buys_computer = yes)	= 6/9	= 0.667
P(credit_rating = fair | buys_computer = ne)	= 2/5	= 0.400

Using these probabilities, we obtain:

P(X | buys_computer = yes) = P(age = youth | buys_computer = yes)
* P(income = medium | buys_computer = yes)
* P(student = yes | buys_computer = yes)
* P(credit_rating = fair | buys_computer = yes)
= 0.222 * 0.444 * 0.667 * 0.667 = 0.044

Similarly,

P(X | buys_computer = no) = 0.600 * 0.400 * 0.200 * 0.400 = 0.019

To find the class, Ci, that maximizes P(X|Ci)P(Ci), we compute

P(X | buys_computer = yes)* P(buys_computer = yes) = 0.444 * 0.643 = 0.028
P(X | buys_computer = no) * P(buys_computer = no) = 0.019 * 0.357 = 0.007

These numbers can now be normalized to represent probability, but since order will be maintained, we can simply choose the maximum, or, predict the buys_computer = yes for tuple X.

What if I encounter probability values of 0?

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 0. This technique for probability estimation is known as the Laplacian correction or Laplace estimator.

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 one more tuple for each income-value pair. In this way, we instead obtain the following probabilities:

1/1003 = 0.001, 991/1003 = 0.988, and 11/1003 = 0.011

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