Algorithm 3.2: Predicting with a naive bayes classifier for binary features

```
for i = 1 : N do
    for c = 1 : C do
        L_ic = log π̂_c;
        for j = 1 : D do
            if x_ij = 1 then L_ic := L_ic + log θ̂_jc else L_ic := L_ic + log(1 − θ̂_jc)
    p_ic = exp(L_ic − logsumexp(L_i,:));
    ŷ_i = argmax_c p_ic;
```

take the top K, where K is chosen based on some tradeoff between accuracy and complexity. This approach is known as variable **ranking**, **filtering**, or **screening**.

One way to measure relevance is to use mutual information (Section 2.8.3) between feature X_j and the class label Y:

$$I(X,Y) = \sum_{x_j} \sum_{y} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)} \tag{3.75}$$

The mutual information can be thought of as the reduction in entropy on the label distribution once we observe the value of feature j. If the features are binary, it is easy to show (Exercise 3.21) that the MI can be computed as follows

$$I_j = \sum_{c} \left[\theta_{jc} \pi_c \log \frac{\theta_{jc}}{\theta_j} + (1 - \theta_{jc}) \pi_c \log \frac{1 - \theta_{jc}}{1 - \theta_j} \right] \tag{3.76}$$

where $\pi_c = p(y = c)$, $\theta_{jc} = p(x_j = 1 | y = c)$, and $\theta_j = p(x_j = 1) = \sum_c \pi_c \theta_{jc}$. (All of these quantities can be computed as a by-product of fitting a naive Bayes classifier.)

Figure 3.1 illustrates what happens if we apply this to the binary bag of words dataset used in Figure 3.8. We see that the words with highest mutual information are much more discriminative than the words which are most probable. For example, the most probable word in both classes is "subject", which always occurs because this is newsgroup data, which always has a subject line. But obviously this is not very discriminative. The words with highest MI with the class label are (in decreasing order) "windows", "microsoft", "DOS" and "motif", which makes sense, since the classes correspond to Microsoft Windows and X Windows.

3.5.5 Classifying documents using bag of words

Document classification is the problem of classifying text documents into different categories. One simple approach is to represent each document as a binary vector, which records whether each word is present or not, so $x_{ij} = 1$ iff word j occurs in document i, otherwise $x_{ij} = 0$. We can then use the following class conditional density:

$$p(\mathbf{x}_i | y_i = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} \text{Ber}(x_{ij} | \theta_{jc}) = \prod_{j=1}^{D} \theta_{jc}^{\mathbb{I}(x_{ij})} (1 - \theta_{jc})^{\mathbb{I}(1 - x_{ij})} \tag{3.77}$$