CS 340 Fall 2007: Homework 4

1 Bayes classifier for Gaussian data

[Note: you can solve this exercise by hand or using a computer (matlab, R, whatever). In either case, show your work.] Consider the following training set of heights x (in inches) and gender y (male/female) of some US college students.

x	y
67	m
79	m
71	m
68	f
67	f
60	f

1. Fit a Bayes classifier to this data, using maximum likelihood estimation, i.e., estimate the parameters of the class conditional likelihoods
$$p(x|y=c) = \mathcal{N}(x; \mu_c, \sigma_c) \tag{1}$$
and the class prior
$$p(y=c) = \pi_c \tag{2}$$
What are your values of μ_c, σ_c, π_c for $c = m, f$? Show your work (so you can get partial credit if you make an arithmetic error).

2. Compute $p(y=m|x, \hat{\theta})$, where $x = 72$, and $\hat{\theta}$ are the MLE parameters. (This is called a **plug-in** prediction.)

3. What would be a simple way to extend this technique if you had multiple attributes per person, such as height and weight? Write down your proposed model as an equation.

2 Irrelevant features with naive Bayes

Let $x_{iw} = 1$ if word w occurs in document i and $x_{iw} = 0$. Let θ_{cw} be the estimated probability that word w occurs in documents of class c. Then the log-likelihood that document $\mathbf{x}$ belongs to class c is

$$\begin{aligned}
\log p(\mathbf{x}_i|c, \theta) &= \log \prod_{w=1}^{W} \theta_{cw}^{x_{iw}} (1-\theta_{cw})^{1-x_{iw}} && (3)\\
&= \sum_{w=1}^{W} x_{iw} \log \theta_{cw} + (1-x_{iw}) \log(1-\theta_{cw}) && (4)\\
&= \sum_{w=1}^{W} x_{iw} \log \frac{\theta_{cw}}{1-\theta_{cw}} + \sum_{w} \log(1-\theta_{cw}) && (5)\\
&= [\mathbf{x}_i, 1]^T [\log \frac{\theta_c}{1-\theta_c}, \alpha_c] && (6)\\
&= \mathbf{y}_i^T \boldsymbol{\phi}_c && (7)
\end{aligned}$$

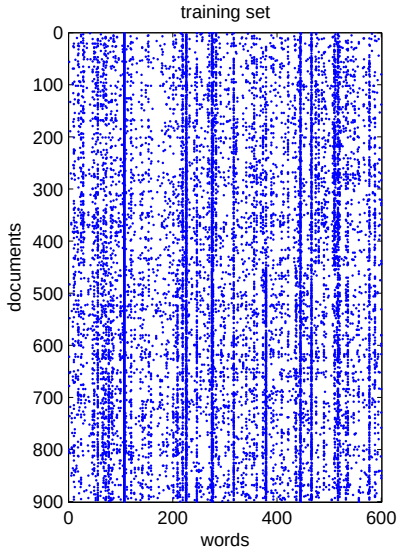

Figure 1: Document word co-occurrence matrix. A black dot in row i column j means document i contains word j at least once. The first 450 lines correspond to the documents about X windows, the second 450 lines to documents about MS windows. Can you see a difference in the patterns between the first 450 rows and the second 450 rows?

where W is the number of words in the vocabulary, $\alpha_c = \sum_w \log(1 - \theta_{cw})$ is a constant independent of $\mathbf{x}$ and $\boldsymbol{\phi}_c$ is the vector of parameters derived from the $\boldsymbol{\theta}_c$ and α_c. (So we see that this is a linear classifier.)

1. Assuming $p(C = 1) = p(C = 2) = 0.5$, write down an expression for the log posterior odds, $\log_2 \frac{p(c=1|\mathbf{x})}{p(c=2|\mathbf{x})}$, in terms of y_i and the parameters ϕ. Hint: since the priors are equal, the log posterior ratio is just the log likelihood ratio.

2. Intuitively, words that occur in both classes are not very "discriminative", and therefore should not affect our beliefs about the class label. Consider a particular word w. State the conditions on $\theta_{1,w}$ and $\theta_{2,w}$ (or equivalently the conditions on $\phi_{1,w}, \phi_{2,w}$) under which the presence or absence of w in a test document will have no effect on the class posterior (such a word will be ignored by the classifier). Hint: using your previous result, figure out when the posterior odds ratio is 0.5/0.5.

3. The posterior mean estimate of θ, using a Beta(1,1) prior, is given by

$$\hat{\theta}_{cw} = \frac{1 + \sum_{i \in c} x_{iw}}{2 + n_c} \tag{8}$$

   where the sum is over the n_c documents in class c. Consider a particular word w, and suppose it always occurs in every document in both classes. Let there be n_1 documents of class 1 and n_2 be the number of documents in class 2, where $n_1 \neq n_2$ (since e.g., we get much more non-spam than spam; this is an example of class imbalance). If we use the above estimate for $\hat{\theta}_{cw}$ for this word, will it be ignored by our classifier?

4. What other ways can you think of which encourage "irrelevant" words to be ignored?

3 Naive Bayes classifier for document classification

Consider the problem of classifying email messages posted to online discussion boards into one of two classes, one for users of X Windows (class 1) and another for users of microsoft Windows (class 2). (This is analogous to **email spam**

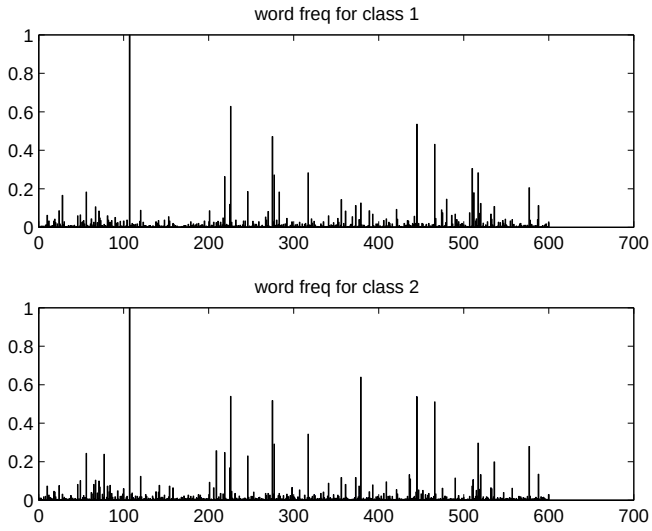

Figure 2: Class conditional densities $p(x_j = 1|c)$ for two document classes. The big spike at index 107 corresponds to the word "subject", which occurs in both classes with probability 1.

filtering, which we will consider below.) There are 900 documents from each class; we divided them into training and text sets of equal size. To save space (and time), we ran word detection on the documents, and the data available to you consist of binary feature vectors for each document. Upon loading `docdata.mat` the Matlab environment will contain

```
Name          Size              Bytes  Class     Attributes
  vocab       600x1               43914  cell
  xtest       900x600            147412  double    sparse
  xtrain      900x600            152884  double    sparse
  ytest       900x1                7200  double
  ytrain      900x1                7200  double
```

`xtrain(n,i)`=1 iff document n contains word i, otherise `xtrain(n,i)`=0. Note that `xtrain` and `xtest` are **sparse** matrices, since most entries are zero. You can visualize sparse matrices using the `spy` command: see Figure 1. The identity of the 600 words is stored in the cell array `vocab`. You can print out the first 10 words using

```
for t=1:10
   fprintf(2,'%2d %20s\n', t,  vocab{t});
end
```

which produces the list in Figure 3.

1. Implement the following function

```
function theta = NBtrain(X,Y)
% Posterior mean estimate of Naive Bayes parameters
% Input:
% X(i,j) = 1 if word j appears in document i, otherwise X(i,j)=0
% Y(i) = class label of doc i (assumed to be 1 or 2)
% Output:
% theta(j,c) = probability of word j appearing in class c
```

1	straight
2	magazines
3	issues
4	ray
5	enabled
6	head
7	improved
8	thread
9	libs
10	working

Figure 3: First 10 words in the vocabulary used for the NB exercise.

which computes the following posterior mean estimate

$$\hat{\theta}_{jc} = \frac{N_{jc} + 1}{N_c + 2} \tag{9}$$

where N_{jc} counts the number of times word j appears in class c, N_c is the total number of documents in class c, and we have assumed a Beta(1,1) prior. Turn in your code.

2. Implement a function to classify each document, assuming uniform class priors $p(Y = 1) = p(Y = 2) = 0.5$.

```
function y = NBapply(X,theta)
% X(i,j) = 1 if word j appears in document i, otherwise X(i,j)=0
% theta(j,c) = prob of word j in class c
% y(i) = most probable class for X(i,:)
```

Here $y(i) = \arg\max_y p(Y = y|X(i,:))$ is the most probable class label for document i. Since $p(Y = y|\mathbf{x}) \propto p(\mathbf{x}|Y = y)$ is a small number, you will need to use logs to avoid underflow. (You don't necessarily need the logsumexp trick, because it suffices to compute the log likelihood $p(\mathbf{x}|y)$ rather than the normalized posterior $p(y|\mathbf{x})$, but you will need to use logs somehow!) Turn in your code.

3. Use `NBtrain` on the data in `xtrain,ytrain`. Compute the misclassification rates (i.e., the number of documents that you mis-classified) on the training set (by using `NBapply` on `xtrain,ytrain`) and on the test set (by using `NBapply` on `xtest,ytest`). Sanity check: You should get test error of **0.1867**.

4. The provided function `NBcv` computes the K-fold cross-validation error. (This calls your functions `NBtrain` and `NBapply`.) $K = 1$ means no cross-validation, that is error is simply computed on the whole training set. Use this to compute the 10-fold error rate on the training set. How does this compare to the (non cross validated) training and test error?

5. Plot (as histograms) the class-conditional densities $p(x_j = 1|y = c, \theta_c)$ for classes $c = 1, 2$ and words $j = 1 : 600$. You should get the same result as Figure 2.

6. What are the 5 most likely words in each class?

7. It is clear that the most probable words are not very discriminative. One way to measure how much information a word (feature) $X_j \in \{0, 1\}$ conveys about the class label $Y \in \{1, 2\}$ is by computing the **mutual information** between X_j and Y, denoted $I(X_j, Y)$, and defined as

$$mi(j) = I(X_j, Y) \quad = \quad \sum_{x=0}^{1} \sum_{c=1}^{2} p(X_j = x, Y = c) \log \frac{p(X_j = x, Y = c)}{p(X_j = x)p(Y = c)} \tag{10}$$

If we assume equal class priors, $p(Y = 1) = p(Y = 2) = 0.5$, then

$$p(X_j = 1, Y = c) = p(X_j|Y = y)p(Y = c) = \frac{\theta_{jc}}{2} \tag{11}$$

Use the provided function `MI` to compute the 5 words with the highest mutual information with the class label. (Use the θ's estimated on `xtrain,ytrain`.) List the words along with the corresponding values of `MI`. (As a sanity check, the first word should be "windows" with an MI of 0.2150.)

4 Naive Bayes classifier for email spam filtering

You will now apply the code you wrote in the previous question to a much larger data set (so your computer will need much more time and memory). Consider the problem of classifying email messages into spam and non-spam (ham). There are 1000 spam emails and 2000 ham emails in our collection; we divided them into training and text sets of equal size and class ratio. We consider 10,000 words. Upon loading `emaildata.mat`, the Matlab environment will contain

```
Name          Size              Bytes  Class     Attributes
 vocab          1x10000            745408  cell
  xtest       1500x10000           4043516  double    sparse
  xtrain      1500x10000           4043516  double    sparse
  ytest          1x1500              12000  double
  ytrain         1x1500              12000  double
```

Now `xtrain(n,k)` contains the *number of times* word k occurs in document n; thus it is not a binary matrix. `vocab` is a cell array, as before.

1. Convert the X data (training and test) to binary (word present/ absent) form and fit a naive Bayes classifier using your code from above. What is the training and test error? Hint: if you run out of memory, use the 'clear' command to remove variables once they are no longer needed. Also, consider using single precision.

2. What are the 50 most likely words in each class? List them with their probability.

3. Use the provided function `MI.m` to compute the 50 words with the highest mutual information with the class label. (Use the θ's estimated on `xtrain,ytrain`.) List the words along with the corresponding values of `MI`. (As a sanity check, the first word should be "our" with an MI of 0.23376.)

4. Plot the error rate on the test set as a function of the number of features used, in order of decreasing mutual information. You should get a plot like Figure 4. Turn in your code and plot. How many features should we use, and why?

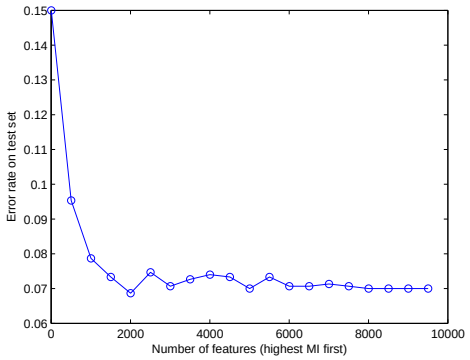

Figure 4: Test error rate vs number of features, in order of decreasing MI.