HOMEWORK 2:
MAXIMUM LIKELIHOOD ESTIMATION AND NAÏVE BAYES

CMU 10601: MACHINE LEARNING (SPRING 2016)
OUT: Sep. 7, 2016
DUE: 5:30 pm, Sep. 14, 2016
TAs: Pradeep Dasigi, Hsiao-Yu Fish Tung, Varshaa Naganathan

Instructions

- **Homework Submission:** For problems 1 and 2, and items 1-2 and 8-10 in question 3, submit your solutions as **BOTH** a hard-copy to the hanging folders outside Sandra Winkler's office (GHC 8221) **AND** an electronic version to Gradescope. Both must be submitted on time. You will use Autolab to submit your code in question 3 (items 3-7 in question 3). More information on using Autolab is provided with question 3. For Gradescope, you will need to specify which pages go with which question. We provide a LaTeX template which you can use to type up your solutions. This template ensures the correct sections start on new pages. Please check Piazza for updates about the homework.
- **Collaboration policy**: Please read the policy on the course webpage: https://mgormley.github.io/10601b-f16/about.html

Problem 1: Bayes Theorem and Random Variables

1. **[4 points]** For events A and B, prove

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)}$$

where $\neg A$ indicate the non-occurrence of event A.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)} \tag{1}$$

2. **[5 points]** For any events A, B, C and D, prove that

$$P(A \cap B \cap C \cap D) = P(A)P(B|A)P(C|A \cap B)P(D|A \cap B \cap C)$$

What would the above expression evaluate to if A, B, C and D are independent?

$$\begin{aligned} RHS &= P(A)P(B|A)P(C|A,B)P(D|A,B,C) && (2)\\ &= P(A)\frac{P(A,B)}{P(A)}\frac{P(A,B,C)}{P(A,B)}\frac{P(A,B,C,D)}{P(A,B,C)} && (3)\\ &= P(A,B,C,D) = LHS && (4)\end{aligned}$$

If they are independent, then $P(A \cap B \cap C \cap D) = P(A)P(B)P(C)P(D)$.

3. **[4 points]** Given a fair six sided die, let a random variable X be defined as follows

$$X = \begin{cases} 1 & \text{if the dice rolls even} \\ 0 & \text{if the dice rolls odd} \end{cases}$$

What is $\mathbb{E}(X)$?

$$\mathbb{E}(X) = \frac{1}{2} * 1 + \frac{1}{2} * 0 = \frac{1}{2}.$$

4. Let X, Y, and Z be three random variables taking values in $0, 1$. Given below is an incomplete table of probabilities of values taken by the variables.

	$Z = 0$		$Z = 1$	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
$Y = 0$	$1/24$	$1/12$	$1/12$	$5/24$
$Y = 1$	$1/12$	p	q	$7/24$

(a) **[5 points]** Given that X and Y are independent, find the values p and q.

Combining all possible Z, the table becomes

	$X = 0$	$X = 1$
$Y = 0$	$3/24$	$7/24$
$Y = 1$	$1/12 + q$	$p + \frac{7}{24}$.

Given X and Y are independent, we know P(X,Y) = P(X)P(Y). Thus we list the following equations:

$$P(X = 0)P(Y = 0) = \frac{3}{24} \tag{5}$$

$$P(X = 1)P(Y = 0) = \frac{7}{24} \tag{6}$$

$$P(X = 0)P(Y = 1) = \frac{1}{12} + q \tag{7}$$

$$P(X = 1)P(Y = 1) = \frac{7}{24} + p \tag{8}$$

Also we known these values sum to one, so we have

$$p + q = \frac{5}{24}$$

By the fact that $P(X=0,Y=0) = P(X=0)P(Y=0)$, we have

$$\begin{aligned}
\frac{3}{24} = P(X=0,Y=0) &= P(X=0)P(Y=0) &(9)\\
&= (P(X=0,Y=0) + P(X=0,Y=1)) &(10)\\
&\quad \cdot (P(X=0,Y=0) + P(X=1,Y=0)) &(11)\\
&= (\frac{3}{24} + \frac{1}{12} + q) \cdot (\frac{3}{24} + \frac{7}{24}) &(12)\\
q &= \frac{11}{120} &(13)\\
p &= \frac{5}{24} - q = \frac{7}{60} &(14)
\end{aligned}$$

(b) **[5 points]** Are X and Y conditionally independent given Z? Why?
No. We can examine whether $P(X,Y|Z=0) = P(X|Z=0)P(Y|Z=0)$. By first normalizing the table on the left hand side to have total probability sum to one and summing over the row for $Y=0$ and the column of $X=0$, we get $P(Y=0|Z=0) = \frac{5}{13}$, $P(Y=1|Z=0)1 - P(Y=0|Z=0) = \frac{8}{13}$ and $P(X=0|Z=0) = \frac{5}{13}$. Since

$$P(X=0,Y=1|Z=0) = \frac{10}{39} \neq \frac{5}{13} \cdot \frac{8}{13} = P(X=0|Z=0)P(Y=1|Z=0), \qquad (15)$$

we know X and Y are not independent conditioning on Z.

Problem 2: Maximum Likelihood Estimation and Maximum a Posteriori Estimation

In this problem, we will look at two different ways of estimating parameters in a probability distribution. Suppose we observe n iid random variables X_1, . . ., X_n, drawn from a Geometric distribution with parameter θ. That is, for each X_i and a natural number k,

$$P(X_i = k) = (1-\theta)^k\theta$$

Given some observed values of X_1 to X_n, we want to estimate the value of θ.

Maximum Likelihood Estimation

The first kind of estimator for θ we will consider is the Maximum Likelihood Estimator (MLE). The probability of observing given data is called the likelihood of the data, and the function that gives the likelihood for a given parameter $\hat{\theta}$ (which may or may not be equal to the true parameter θ) is called the likelihood function, written as $L(\hat{\theta})$. When we use MLE, we estimate θ by choosing the $\hat{\theta}$ that maximizes the likelihood.

$$\hat{\theta}^{\text{MLE}} = \arg\max_{\hat{\theta}} L(\hat{\theta})$$

It is often convenient to deal with the log-likelihood ($\ell(\hat{\theta}) = \log L(\hat{\theta})$) instead, and since log is an increasing function, we also have

$$\hat{\theta}^{\text{MLE}} = \arg\max_{\hat{\theta}} \ell(\hat{\theta})$$

1. **[4 points]** Given a dataset $\mathcal{D}$, containing observations $\{X_1 = k_1, X_2 = k_2, \ldots, X_n = k_n\}$, write an expression for $\ell(\hat{\theta})$ as a function of $\mathcal{D}$ and $\hat{\theta}$. How does the order of the variables affect the function?
   Since $X_1, \cdots, X_n$ are drawn independently, their joint distribution can be written as the product of individual probabilities.

   $$\ell(\hat{\theta}) = \log \Pi_{i=1}^n P(X_i) = \log \Pi_{i=1}^n (1-\theta)^{k_i}\theta$$

   The order of variables does not affect the function

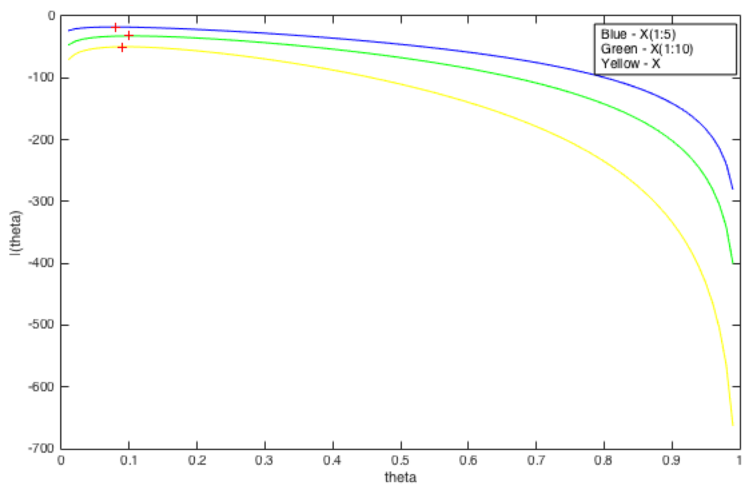
2. **[4 points]** Write a function in Octave `plotMLE(X, theta)` that takes as input a sequence of samples, and values for $\hat{\theta}$ and produces a plot of the log likelihood function ($\ell(\hat{\theta})$ on the Y-axis and $\hat{\theta}$ on the X-axis). This program is not autograded on Autolab. So, please include your code in your pdf submission.

```
function [] = plotMLE(X, theta)
    % Input:
    % X is a 1 x n array
    % theta is a 1 x m array
    n = numel(X);
    l = sum(X)*log(1-theta) + n*log(theta);
    plot(theta,l)
    xlabel('theta')
    ylabel('l(theta)')
end
```

3. **[3 points]** Consider the following sequence of 15 samples:

$$X = (0, 21, 23, 8, 9, 2, 9, 0, 7, 8, 20, 9, 7, 4, 17)$$

Use your program to produce three plots, first one with the first five samples $(0, 21, 23, 8, 9)$, second one with the first ten, and the third one with all fifteen. For all three plots, use values of $\hat{\theta}$ from $0.01, 0.02, \ldots, 1.0$. In each plot, mark the location of the maximum likelihood estimator. Does the estimate change across the three plots?



The estimate changes very little across the 3 cases.

Maximum a Posteriori Estimation

Now we assume that we have some prior knowledge about the true parameter θ. We express it by treating θ itself as a random variable and definining a prior probability distribution over it. Precisely, we suppose that the data X_1, . . ., X_n are drawn as follows:

- θ is drawn from the prior probability distribution
- Then X_1, . . ., X_n are drawn independently from a Geometric distribution with θ as the parameter.

Now both X_i and θ are random variables, and they have a joint probability distribution. We now estimate θ as follows

$$\hat{\theta}^{\text{MAP}} = \arg\max_{\hat{\theta}} P(\theta = \hat{\theta} | X_1, \ldots, X_n)$$

This is called Maximum a Posteriori (MAP) estimation. Using Bayes rule, we can rewrite the posterior probability as follows.

$$P(\theta = \hat{\theta}) = \frac{P(X_i, \dots, X_n | \theta = \hat{\theta}) P(\theta = \hat{\theta})}{P(X_1, \dots, X_n)}$$

Applying this to the MAP estimate, we get the following expression. Notice that we can ignore the denominator since it is not a function of $\hat{\theta}$.

$$\begin{aligned} \hat{\theta}^{\text{MAP}} &= \arg\max_{\hat{\theta}} P(X_1, \dots, X_n | \theta = \hat{\theta}) P(\theta = \hat{\theta}) \\ &= \arg\max_{\hat{\theta}} L(\hat{\theta}) P(\theta = \hat{\theta}) \\ &= \arg\max_{\hat{\theta}} \left(\ell(\hat{\theta}) + \log P(\theta = \hat{\theta}) \right) \end{aligned}$$

Thus, the MAP estimator maximizes the sum of the log-likelihood and the log-probability of the prior distribution on θ. When the prior is a continuous distribution with density function p, we have

$$\hat{\theta}^{\text{MAP}} = \arg\max_{\hat{\theta}} \left(\ell(\hat{\theta}) + \log p(\hat{\theta}) \right)$$

For this problem, we will use the Beta distribution (a popular choice when the data distribution is Geometric or Bernoulli) as the prior, and the density function is given by

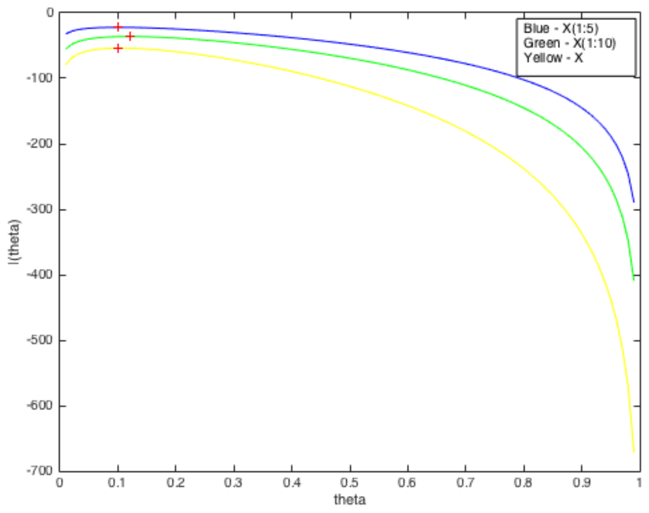
$$p(\hat{\theta}) = \frac{\hat{\theta}^{\alpha-1}(1-\hat{\theta})^{\beta-1}}{B(\alpha, \beta)}$$

where $B(\alpha, \beta)$ is the beta function.

4. **[4 points]** Modify the program you wrote for plotting log-likelihood values to plot $\hat{\theta} \mapsto \ell(\hat{\theta}) + \log p(\hat{\theta})$ instead. Please submit this program in your pdf as well.

```
function [] = plotMAP(X, theta)
    % Input:
    % X is a 1 x n array
    % theta is a 1 x m array
    n = numel(X);
    B = 0.5;
    p = (1-theta)/B;
    l = sum(X)*log(1-theta) + n*log(theta) + log(p);
    plot(theta,l)
    xlabel('theta')
    ylabel('l(theta)')
end
```

5. **[5 points]** Redo the three plots you made above, but with the log-posterior function instead, and mark the MAP estimators. Set $\alpha = 1$, $\beta = 2$. Note that $B(1, 2) = 0.5$.



6. **[3 points]** Do you see any significant differences between MLE and MAP estimates?

   There are no significant differences observed.

7. **[5 points]** Derive a close form expression for the maximum a posteriori estimate. (hint: If x^* maximizes f, $f'(x^*) = 0$). Does this expression agree with your plots?

$$\nabla_\theta \log \Pi_{i=1}^n (1-\theta)^{k_i}\theta + \log \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} = \nabla_\theta \sum_{i=1}^n [k_i \log(1-\theta) + \log(\theta)] + \log \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)}$$
$$= \sum_{i=1}^n [-k_i \frac{1}{1-\theta}] + \frac{n}{\theta} + \frac{\alpha-1}{\theta} - \frac{\beta-1}{1-\theta} = 0$$

   By solving the equation, we have

$$\frac{n+\alpha-1}{\theta} = \frac{\sum_{i=1}^n k_i + \beta - 1}{1-\theta} \tag{16}$$
$$\theta = \frac{n+\alpha-1}{n+\alpha+\beta-2+\sum_{i=1}^n k_i} \tag{17}$$

   Compute θ from the above expression and cross check with the values from graphs to check if they agree. The values are very close.

Problem 3: Implementing Naïve Bayes

For this question, you will implement a Naïve Bayes (NB) classifier. You are given a dataset containing text articles coming from two sources: *The Economist*, a serious news source, and *The Onion*, a sarcastic news source. You will train a classifier to distinguish between the articles from the two sources.
The features used in the classifier are the words themselves. The set of all words from all the articles in our data is called the *vocabulary*, and let's say its size is V. We will represent each article as a feature vector $X = \langle X_1, \ldots, X_V \rangle$, such that

$$X_w = \begin{cases} 1 & \text{if } w \text{ is present in the document} \\ 0 & \text{otherwise} \end{cases}$$

We also associate each article with a label Y such that

$$Y = \begin{cases} 0 & \text{if the article is from } \textit{The Economist} \\ 1 & \text{if the article is from } \textit{The Onion} \end{cases}$$

Two key assumption we make when we apply the Naïve Bayes classifier are that our data are drawn iid from a joint probability distribution over feature vectors X and labels Y, and more importantly for each pair of features, X_i

and X_j with $i \neq j$ X_i is conditionally independent of X_j given the label Y (hence the "naïve" in Naïve Bayes). To predict the label of an article, we choose the most probable class label given x.

$$\hat{Y} = \arg\max_y P(Y = y|X)$$

Using the Bayes rule and the NB assumption, we can rewrite the above expression as follows

$$\begin{aligned}
\hat{Y} &= \arg\max_y \frac{P(X|Y = y)P(Y = y)}{P(x)} && \text{(Bayes rule)} \\
&= \arg\max_y P(X|Y = y)P(Y = y) && \text{(denominator does not depend on } y) \\
&= \arg\max_y P(X_1, \ldots, X_n|Y = y)P(Y = y) \\
&= \arg\max_y \prod_{w=1}^{V} P(X_w|Y = y)P(Y = y) && \text{(Naïve Bayes assumption)}
\end{aligned}$$

As we can see, by making the NB assumption, we can factor the probability distribution $P(X|Y = y)$ as a product of all $P(X_w|Y = y)$. Factoring it this way usually lets us define significantly fewer parameters.

1. **[1 point]** How many parameters will the model need under the NB assumption, assuming that $P(X_w|Y = y)$ and $P(Y = y)$ are both Bernoulli distributions? Give your answer as a function of the vocabulary size, V.

   We would need one parameter for each $P(X_w|Y = y)$, for each of the two classes. This would be $2V$ parameters. We also need one parameter for estimating $P(Y = y)$ (Since it is a binary case, estimating probability of one class is sufficient). Hence, total number of parameter needed is $2V + 1$.

2. **[2 points]** How many parameters (also as a function of V) will the model need if we do not make the NB assumption, assuming $P(Y = y)$ is Bernoulli again and $P(X|Y = y)$ is a categorical distribution?

   We need to estimate $P(X_1, \ldots, X_n|Y = y)$ for every possible combination of values of X_i. Since each X_i can be 0 or 1, there are 2^V possible combinations. Since the sum of probabilities for each such combination for a particular class must be 1, estimating $P(X_1, \ldots, X_n|Y = y)$ for $2^V - 1$ combinations is sufficient. Again, this needs to be done for each of the two classes and we need to estimate $P(Y = y)$ (Since it is a binary case, estimating probability of one class is sufficient). Hence, total number of parameters needed is $2(2^V - 1) + 1$

Since we do not know the true joint distribution over X and Y, we need to estimate $P(X|Y = y)$ and $P(Y = y)$ from the training data. For each word index $w \in 1, \ldots, V$ and $y \in 0, 1$, suppose that the distribution of X_w given Y is a Bernoulli distribution with the parameter θ_{yw}, such that

$$P(X = 1|Y = y) = \theta_{yw} \quad \text{and} \quad P(X = 0|Y = y) = 1 - \theta_{yw}$$

A common problem in language related ML problems is dealing with words not seen in training data. Without any prior information, the probability of unseen words is zero. But we know that it is not a good estimate, and we would want to assign a small probability ro any word in the vocabulary occurring in either *The Economist* or *The Onion*. We can achieve that by imposing a Beta(α, β) prior on θ_{yw}, and perform a MAP estimate from the training data. You will experiment with two combinations of α and β values in this problem.
Similarly, suppose the distribution of Y is a Bernoulli distribution (taking values 1 or 2), as given below.

$$P(Y = 1) = \rho \quad \text{and} \quad P(Y = 0) = 1 - \rho$$

Since we have enough articles in both classes, we need not worry about zero probabilities, and will not impose a prior on ρ.

Programming Instructions

You will implement some functions for training and testing a Naïve Bayes classifier for this question. You will submit your code online through the CMU autolab system, which will execute it remotely against a suite of tests. Your grade will be automatically determined from the testing results. Since you get immediate feedback after submitting your code and you are allowed to submit as many different versions as you like (without any penalty), it is easy for you to check your code as you go.

Our autograder requires that you write your code in Octave. Octave is a free scientific programming language with syntax identical to that of MATLAB. Installation instructions can be found on the Octave website (http://www.gnu.org/software/octave/), and we have posted links to several Octave and MATLAB tutorials on Piazza.
To get started, you can log into the autolab website (https://autolab.cs.cmu.edu). From there you should see 10-601B in your list of courses. Download the template for Homework 2 and extract the contents (i.e., by executing `tar xvf hw2.tar` at the command line). In the archive you will find one `.m` file for each of the functions that you are asked to implement and a file that contains the data for this problem, `HW2Data.mat`. To finish each programming part of this problem, open the corresponding `.m` file and complete the function defined in that file. When you are ready to submit your solutions, you will create a new tar archive of the top-level directory (i.e., by executing `tar cvf hw2.tar hw2`) and upload that through the Autolab website.
The file `HW2Data.mat` contains the data that you will use in this problem. We have preprocessed the data so that you can directly run experiments. You can load it from Octave by executing `load("HW2Data.mat")` in the Octave interpreter. After loading the data, you will see that there are 7 variables: `Vocabulary`, `XTrain`, `yTrain`, `XTest`, `yTest`, `XTrainSmall`, and `yTrainSmall`.

- `Vocabulary` is a $V \times 1$ dimensional cell array that that contains every word appearing in the documents. When we refer to the j^{th} word, we mean `Vocabulary(j,1)`.
- `XTrain` is a $n \times V$ dimensional matrix describing the n documents used for training your Naive Bayes classifier. The entry `XTrain(i,j)` is 1 if word j appears in the i^{th} training document and 0 otherwise.
- `yTrain` is a $n\times 1$ dimensional matrix containing the class labels for the training documents. `yTrain(i,1)` is 1 if the i^{th} document belongs to The Economist and 2 if it belongs to The Onion.
- `XTest` and `yTest` are the same as `XTrain` and `yTrain`, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.
- Finally, `XTrainSmall` and `yTrainSmall` are subsets of `XTrain` and `yTrain` which are used in the final question.

Code

Logspace Arithmetic

When working with very large or very small numbers (such as probabilities), it is useful to work in *logspace* to avoid numerical precision issues. In logspace, we keep track of the logs of numbers, instead of the numbers themselves. For example, if $p(x)$ and $p(y)$ are probability values, instead of storing $p(x)$ and $p(y)$ and computing $p(x) * p(y)$, we work in log space by storing $\log(p(x))$, $\log(p(y))$, and we can compute the log of the product, $\log(p(x) * p(y))$ by taking the sum: $\log(p(x) * p(y)) = log(p(x)) + log(p(y))$.

3. **[1 Point]** Complete the function `[log_product] = logProd(x)` which takes as input a vector of numbers in logspace (i.e., $x_i = \log p_i$) and returns the product of those numbers in logspace—i.e., $\texttt{logProd}(\texttt{x}) = \log\left(\prod_i p_i\right)$.

```
function[p] = logProd(x)
    p = sum(x);
end
```

Training Naïve Bayes

4. **[4 Points]** Complete the function `[D] = NB_XGivenY(XTrain, yTrain, alpha, beta)`. The output `D` is a $2 \times V$ matrix, where for any word index $w \in \{1, \ldots, V\}$ and class index $y \in \{1, 2\}$, the entry `D(y,w)` is the MAP estimate of $\theta_{yw} = P(X_w = 1|Y = y)$ with a Beta(α, β) prior distribution. Note that the parameters of the Beta distribution are also given as input to the function.

```
function [D] = NB_XGivenY(XTrain, yTrain, alpha, beta)
    classE = (yTrain == 0); % Economist
    classO = (yTrain == 1); % Onion

    D = [(sum(XTrain(classE,:),1) + (alpha - 1)) / (sum(classE) + alpha + beta - 2);
         (sum(XTrain(classO,:),1) + (alpha - 1)) / (sum(classO) + alpha + beta - 2)]
end
```

5. **[4 Points]** Complete the function `[p] = NB_YPrior(yTrain)`. The output `p` is the MLE for $\rho = P(Y = 1)$.

```
function [p] = NB_YPrior(yTrain)
    p = sum(yTrain == 0)/numel(yTrain);
end
```

6. **[8 Points]** Complete the function `[yHat] = NB_Classify(D, p, XTest)`. The input `X` is an $m \times V$ matrix containing m feature vectors (stored as its rows). The output `yHat` is a $m \times 1$ vector of predicted class labels, where `yHat(i)` is the predicted label for the i^{th} row of `X`. [Hint: In this function, you will want to use the `logProd` function to avoid numerical problems.]

```
function [yHat] = NB_Classify(D, p, X)
    m = size(X,1);
    yHat = zeros(m,1);

    for i = 1:m
        classE = D(1,:) .* X(i,:) + (1 - D(1,:)) .* (1 - X(i,:));
        classO = D(2,:) .* X(i,:) + (1 - D(2,:)) .* (1 - X(i,:));

        probE = logProd([log(classE), log(p)]);
        probO = logProd([log(classO), log(1-p)]);

        if probE > probO
            yHat(i) = 0;
        else
            yHat(i) = 1;
        end
    end
end
```

7. **[1 Point]** Complete the function `[classification_error] = ClassificationError(yHat, yTruth)`, which takes two vectors of equal length and returns the proportion of entries that they disagree on.

```
function [error] = ClassificationError(yHat, yTruth)
    error = sum(yHat ~= yTruth)/numel(yTruth);
end
```

Experiments

8. **[4 Points]** Train your classifier on the data contained in `XTrain` and `yTrain`, with $\alpha = 2$ and $\beta = 1$ by running

   ```
   D = NB_XGivenY(XTrain, yTrain, 2, 1);
   p = NB_YPrior(yTrain);
   ```

   Use the learned classifier to predict the labels for the article feature vectors in `XTrain` and `XTest` by running

   ```
   yHatTrain = NB_Classify(D, p, XTrain);
   yHatTest = NB_Classify(D, p, XTest);
   ```

   Use the function `ClassificationError` to measure and report the training and testing error by running

   ```
   trainError = ClassificationError(yHatTrain, yTrain);
   testError = ClassificationError(yHatTest, yTest);
   ```

   How do the train and test errors compare? Explain any significant differences.

   Train Error = 0
   Test Error = 0.0207

9. **[4 points]** Repeat the above experiment with $\alpha = 2$ and $\beta = 5$. How do the errors in this experiment compare with those from the previous experiment? Do the differences tell us anything about the prior?

   Train Error = 0
   Test Error = 0.0138

10. **[4 points]** Repeat the experiment with $\alpha = 2$ and $\beta = 1$ as in the first experiment, but with the smaller training set `XTrainSmall` and `yTrainSmall`. How do the results compare to those from the first experiment? Does the effect of the prior vary with the training data size?

    Train Error = 0.23
    Test Error = 0.2897