10-601 Machine Learning: Homework 2 Solutions

Due 5:30 p.m. Thursday, February 4, 2016

Problem 1: Independent events and Bayes Theorem

(a) **[5 Points]** For events A, B prove:

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

Solution: By the definition of conditional probabilities, we have that

$$P(A,B) = P(A|B)P(B) \qquad \text{and} \qquad P(A,B) = P(B|A)P(A).$$

Therefore, $P(A|B)P(B) = P(B|A)P(A)$ and dividing both sides of the equality by $P(B)$ gives the desired result.

(b) **[5 Points]** Let $A_1, ..., A_n$ be a partition of the sample space Ω, prove that for any event B,

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Definition: A partition of the sample space Ω is a sequence of disjoint events $A_1, A_2, ..., A_n$ such that $\bigcup_{i=1}^{n} A_i = \Omega$.

Solution: For each A_i define $C_i = B \cap A_i$. Since the A_i sets partition Ω, we know that the sets $C_1, \ldots, C_n$ are disjoint and $B = \bigcup_{i=1}^{n} C_i$ (that is, $C_1, \ldots, C_n$ partition B). With this, we have

$$P(B) = P\left(\bigcup_{i=1}^{n} C_i\right) = \sum_{i=1}^{n} P(C_i) = \sum_{i=1}^{n} P(B \cap A_i) = \sum_{i=1}^{n} P(B|A_i)P(A_i),$$

where the last equality follows from part (a).

(c) **[5 Points]** Let $A_1, ..., A_n$ be a partition of the sample space Ω such that $P(A_i) > 0$ for all $i = 1, ...n$. Prove that if $P(B) > 0$, then for each $i = 1, ..., n$,

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_k P(B|A_k)P(A_k)}.$$

Solution: Combining parts (a) and (b), we have

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(A_i \cap B)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)}.$$

(d) **[5 Points]** Let A, B, and C be any events. Which of the following statements are true? Justify your responses (e.g. *True, this is a direct implication of Bayes Theorem* or *False, it is only true if P(A)=P(B).* Alternatively, you can provide a counterexample to justify that something is false).

(1) $P(A, B, C) = P(A|B, C)P(B|C)P(C)$

Solution: True. Using the definition of conditional probability twice, we have

$$P(A,B,C) = P(A|B,C)P(B,C) = P(A|B,C)P(B|C)P(C).$$

(2) $P(A, B) = P(A|B)P(B|A)$
Solution: False. Given $P(A, B) = P(A|B)P(B)$, if the statement was true it would imply $P(B) = P(B|A)$, which is only true if A and B are independent.

(3) $P(A, B, C) = P(B|A, C)P(C, A)$
Solution: True. Using the definition of conditional probability, we have $P(A, B, C) = P(B|A, C)P(A, C)$.

(4) $P(A, B, C) = P(B|A, C)P(C, A)P(C)$
Solution: False. Any case in which $P(C) \neq 1$, $P(A, B, C) = P(B|A, C)P(A, C) \neq P(B|A, C)P(A, C)P(C)$.

(5) $P(A, B) = P(A)P(B)$
Solution: False. Only true if A and B are independent.

(e) **[5 Points]** Let A be any event, and let X be a random variable defined by

$$X = \begin{cases} -1 & \text{if event } A \text{ occurs} \\ 0 & \text{otherwise.} \end{cases}$$

Show that $P(A) + \mathbb{E}[X] = 0$, where $\mathbb{E}[X]$ denotes the *expected value* of X. Does this result still hold if the value of X is 1 when event A occurs, instead of -1?

Solution: Using the definition of expectations, we have that

$$\mathbb{E}[X] = -1 \cdot P(X = -1) + 0 \cdot P(X = 0) = -P(A).$$

Therefore, $P(A) + \mathbb{E}[X] = P(A) - P(A) = 0$. If we had replaced -1 with 1 in the definition of X, then $\mathbb{E}[X] = P(A)$ and we would have $P(A) + \mathbb{E}[X] = 2P(A)$, so the claim no longer holds.

Problem 2: Maximum Likelihood Estimation

This problem explores maximum likelihood estimation, which is a technique for estimating an unknown parameter of a probability distribution based on observed samples. Suppose we observe the values of n iid[1] random variables $X_1, \ldots, X_n$ drawn from a single Geometric distribution with parameter θ. In other words, for each X_i and natural number k, we know

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

Our goal is to estimate the value of θ from these observed values of X_1 through X_n.

For any hypothetical value $\hat{\theta}$, we can compute the probability of observing the outcome $X_1, \ldots, X_n$ if the true parameter value θ were equal to $\hat{\theta}$. This probability of the observed data is often called the *data likelihood*, and the function $L(\hat{\theta}) = P(X_1, \ldots, X_n|\hat{\theta})$ that maps each $\hat{\theta}$ to the corresponding likelihood is called the *likelihood function*. A natural way to estimate the unknown parameter θ is to choose the $\hat{\theta}$ that maximizes the likelihood function. Formally,

$$\hat{\theta}^{\text{MLE}} = \operatorname*{argmax}_{\hat{\theta}} L(\hat{\theta}).$$

Often it is more convenient to work with the log likelihood function $\ell(\hat{\theta}) = \log L(\hat{\theta})$. Since the log function is increasing, we also have

$$\hat{\theta}^{\text{MLE}} = \operatorname*{argmax}_{\hat{\theta}} \ell(\hat{\theta}).$$

(a) **[5 Points]** Write a formula for the log likelihood function, $\ell(\hat{\theta})$. Your function should depend on the random variables $X_1, \ldots, X_n$, the hypothetical parameter $\hat{\theta}$, and should be simplified as far as possible (i.e., don't just write the definition of the log likelihood function). Does the log likelihood function depend on the order of the random variables?

[1] iid means Independent, Identically Distributed.

Solution: We can can calculate the likelihood function as follows:

$$\begin{aligned}
\ell(\hat{\theta}) &= \log(P(X_1, \ldots, X_n \,|\, \hat{\theta})) \\
&= \log\left(\prod_{i=1}^{n} P(X_i \,|\, \hat{\theta})\right) && \text{(the } X_i \text{ are iid)} \\
&= \sum_{i=1}^{n} \log\big(P(X_i \,|\, \hat{\theta})\big) && \text{(properties of logs)} \\
&= \sum_{i=1}^{n} \log\big((1-\hat{\theta})^{X_i} \cdot \hat{\theta}\big) && \text{(definition)} \\
&= \sum_{i=1}^{n} \big(X_i \log(1-\hat{\theta}) + \log(\hat{\theta})\big) && \text{(properties of logs)} \\
&= \log(1-\hat{\theta}) \sum_{i=1}^{n} X_i + n \log(\hat{\theta}).
\end{aligned}$$

The log likelihood does not depend on the order of the examples.

(b) **[5 Points]** Consider the following sequence of 15 samples:

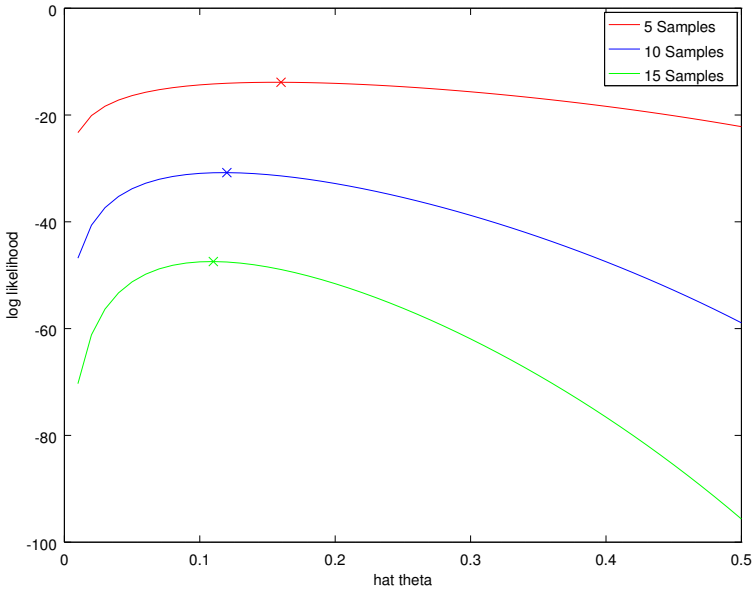
$$X = (1, 0, 3, 5, 18, 14, 5, 7, 13, 9, 0, 17, 4, 24, 3).$$

Write a short computer program that plots three log likelihood functions, one based on the first 5 samples in X (i.e., $(1, 0, 3, 5, 18)$), one based on the first 10 samples, and one based on all 15 samples. Each plot should show the log likelihood of each value of $\hat{\theta}$ in $\{0.01, 0.02, \ldots, 0.5\}$. The x-axis should be $\hat{\theta}$ and the y-axis should be $\ell(\hat{\theta})$. For each plot, mark the location of the maximum likelihood estimator. Please include both the plots and the code used to generate them (but, since this code is not autograded, please just include it in your written submission).

Solution: The following code generates the plot:

```
X = [1, 0, 3, 5, 18, 14, 5, 7, 13, 9, 0, 17, 4, 24, 3];
thetas = 0.01:0.01:0.5;

# Computes the log likelihood of the data in X given parameter theta
function l = log_likelihood(X, theta)
    l = log(1-theta)*sum(X) + numel(X)*log(theta);
endfunction

function handle = make_plot(X, thetas, style)
    ls = arrayfun(@(theta) log_likelihood(X, theta), thetas);
    handle = plot(thetas, ls, style);
    % Plot the MLE
    [max_l, ind] = max(ls);
    plot([thetas(ind)], [max_l], [style 'x'])
endfunction

hold on
h5 = make_plot(X(1:5), thetas, "r-")
h10 = make_plot(X(1:10), thetas, "b-")
h15 = make_plot(X, thetas, "g-")

legend([h5, h10, h15], {'5 Samples', '10 Samples', '15 Samples'})
xlabel("hat theta");
```

```
ylabel("log likelihood")

print -color -deps "problem2b.eps"
```



(c) **[5 Points]** Derive a closed form expression for the maximum likelihood estimate (hint: recall that if x^* maximizes $f(x)$, then $f'(x^*) = 0$). Does your closed form expression agree with the plots?

Solution: Since the MLE $\hat{\theta}^{\text{MLE}}$ maximizes the log likelihood function, we know that the derivative of the log likelihood function is zero at $\hat{\theta}^{\text{MLE}}$. The derivative of ℓ is given by

$$\ell'(\hat{\theta}) = -\frac{\sum_{i=1}^{n} X_i}{1-\hat{\theta}} + \frac{n}{\theta}.$$

Solving $\ell'(\hat{\theta}^{\text{MLE}}) = 0$ gives $\hat{\theta}^{\text{MLE}} = (\bar{X}_n + 1)^{-1}$, where $\bar{X}_n$ is the average of the samples. The formula for the MLE agrees with all three plots.

(d) **[5 Points]** Briefly explain (in 1-2 sentences) why the log likelihood functions become more negative as the number of samples increases.

Solution: Since the probability of observing any given example is a number strictly smaller than 1, the likelihood of observing a data set goes to zero as we increase the size (since we are multiplying together many numbers smaller than one). As a result, the log likelihood tends towards $-\infty$ as the size of the data set increases.

Problem 3: Implementing Naive Bayes

In this question you will implement a Naive Bayes classifier for a text classification problem. You will be given a collection of text articles, each coming from either the serious European magazine *The Economist*, or from the not-so-serious American magazine *The Onion*. The goal is to learn a classifier that can distinguish between articles from each magazine.

We have pre-processed the articles so that they are easier to use in your experiments. We extracted the set of all words that occur in any of the articles. This set is called the *vocabulary* and we let V be the number of words in the vocabulary. For each article, we produced a feature vector $X = \langle X_1, \ldots, X_V \rangle$, where X_i is

equal to 1 if the i^{th} word appears in the article and 0 otherwise. Each article is also accompanied by a class label of either 1 for The Economist or 2 for The Onion. Later in the question we give instructions for loading this data into Octave.

When we apply the Naive Bayes classification algorithm, we make two assumptions about the data: first, we assume that our data is drawn iid from a joint probability distribution over the possible feature vectors X and the corresponding class labels Y; second, we assume for each pair of features X_i and X_j with $i \neq j$ that X_i is conditionally independent of X_j given the class label Y (this is the Naive Bayes assumption). Under these assumptions, a natural classification rule is as follows: Given a new input X, predict the most probable class label $\hat{Y}$ given X. Formally,

$$\hat{Y} = \operatorname*{argmax}_y P(Y = y|X).$$

(a) **[5 points]** Prove the classification rule can be rewritten as

$$\hat{Y} = \operatorname*{argmax}_y \left(\prod_{w=1}^{V} P(X_w|Y = y) \right) P(Y = y).$$

Solution: Using Bayes Rule and the Naive Bayes assumption, we can rewrite this classification rule as follows:

$$\begin{aligned}
\hat{Y} &= \operatorname*{argmax}_y \frac{P(X|Y = y)P(Y = y)}{P(X)} && \text{(Bayes Rule)} \\
&= \operatorname*{argmax}_y P(X|Y = y)P(Y = y) && \text{(Denominator does not depend on } y\text{)} \\
&= \operatorname*{argmax}_y P(X_1, \ldots, X_V|Y = y)P(Y = y) \\
&= \operatorname*{argmax}_y \left(\prod_{w=1}^{V} P(X_w|Y = y) \right) P(Y = y) && \text{(Conditional independence).}
\end{aligned}$$

(b) **[5 points]** How many parameters are needed to represent the distribution $P(X|Y = y)$ when using the Naive Bayes assumption? How many are needed if we do not use the Naive Bayes assumption? Based on this difference, in which cases is there a big gain from making this assumption?

Solution: Using the Naive Bayes assumption, since all the random variables are binary, we only need one parameter to represent the distribution of X_w given Y for each $w \in \{1, \ldots, V\}$ and $y \in \{1, 2\}$. This gives a total of $2V$ parameters.

Without the Naive Bayes assumption, it is not possible to factor the probability as above, and therefore we need one parameter for all but one of the 2^V possible feature vectors X and each class label $y \in \{1, 2\}$. This gives a total of $2(2^V - 1)$ parameters.

The use of Naive Bayes assumption allows us to drastically reduce the number of parameters, especially when the dimension of the feature space is very high.

Of course, since we don't know the true joint distribution over feature vectors X and class labels Y, we need to estimate the probabilities $P(X|Y = y)$ and $P(Y = y)$ from the training data. For each word index $w \in \{1, \ldots, V\}$ and class label $y \in \{1, 2\}$, the distribution of X_w given $Y = y$ is a Bernoulli distribution with parameter θ_{yw}. In other words, there is some unknown number θ_{yw} such that

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

For both The Economist and The Onion, we believe that each word w has a non-zero chance of appearing, but it is more likely that w will not occur in any particular document. We incorporate this belief by computing a MAP estimate using a Beta$(1.001, 1.9)$ prior on θ_{wy}. This has the added benefit of ensuring that none of our estimates of θ_{wy} are equal to 0 or 1 (which can cause problems for Naive Bayes).

Note: The formula for the MAP estimate of θ_{yw} is given by

$$\hat{\theta}_{yw}^{\text{MAP}} = \frac{\text{Number of times word } w \text{ appears in document from class } y + 1.001 - 1}{\text{Number of documents from class } y + 1.001 + 1.9 - 2}.$$

This is formula (8) on page 10 from the chapter 2 draft of Tom's book.

Similarly, the distribution of Y (when we consider it alone) is a Bernoulli distribution (except taking values 1 and 2 instead of 0 and 1) with parameter ρ. In other words, there is some unknown number ρ such that

$$P(Y = 1) = \rho \quad \text{and} \quad P(Y = 2) = 1 - \rho.$$

In this case, since we have many examples of articles from both The Economist and The Onion, there is no risk of having zero-probability estimates, so we will instead use the MLE.

Note: the formula for the MLE estimate of ρ is given by

$$\hat{\rho}^{\text{MLE}} = \frac{\text{Number of documents from class 1 (The Economist)}}{\text{Total number of documents}}.$$

Programming Instructions

Parts (c) through (e) of this question each ask you to implement one function related to the Naive Bayes classifier. 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 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.andrew.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 you 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. 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 5 variables: `Vocabulary`, `XTrain`, `yTrain`, `XTest`, and `yTest`.

- `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.
- Finally, `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.

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: $\log(p(x) * p(y)) = log(p(x)) + log(p(y))$.

We provide the function `logProd(x)` so you can use it in your implementation. This function 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., `logProd(x)` $= \log\left(\prod_i p_i\right)$.

Training Naive Bayes

(c) **[8 Points]** Complete the function `[D] = NB_XGivenY(XTrain, yTrain)`. 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(1.001,1.9) prior distribution.

Solution:

```
function [D] = NB_XGivenY(XTrain, yTrain)
  EconoRows = yTrain == 1;
  OnionRows = yTrain == 2;

  alpha = 1.001;
  beta = 1.9;

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

(d) **[8 Points]** Complete the function `[p] = NB_YPrior(yTrain)`. The output `p` is the MLE for $\rho = P(Y = 1)$.

Solution:

```
function [p] = NB_YPrior(yTrain)
  p = sum(yTrain == 1) / length(yTrain);
end
```

(e) **[8 Points]** Complete the function `[yHat] = NB_Classify(D, p, X)`. 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.]

Solution:

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

  for i = 1:m
    econo_probs = D(1,:) .* XTest(i,:) + (1 - D(1,:)) .* (1 - XTest(i,:));
    onion_probs = D(2,:) .* XTest(i,:) + (1 - D(2,:)) .* (1 - XTest(i,:));

    econo_score = logProd([log(econo_probs), log(p)]);
    onion_score = logProd([log(onion_probs), log(1-p)]);

    if econo_score > onion_score
```

```
      yHat(i) = 1;
    else
      yHat(i) = 2;
    end
  end
end
```

Evaluating Naive Bayes

To help you evaluate your results, we provide the function `[error] = ClassificationError(yHat, yTruth)`, which takes two vectors of equal length and returns the proportion of entries that they disagree on.

Questions

(f) **[5 Points]** Train your classifier on the data contained in `XTrain` and `yTrain` by running

```
D = NB_XGivenY(XTrain, yTrain);
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? Which is more representative of the error we would expect to have on a new collection of articles? Does Naive Bayes attempt to minimize the training error?

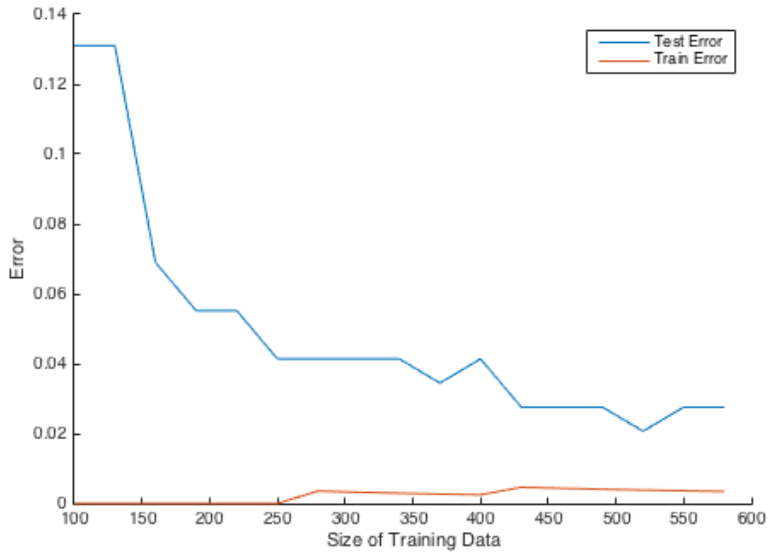
Solution: The training error is 0.0034 and the testing error is 0.0276. Since the data used for testing is not the same data that was used to train the model, we expect it to give a more realistic estimation of the error we would see on new articles. Naive Bayes does not directly minimize training error.

(g) **[8 Points]** In this question we explore how the size of the training data set affects the test and train error. For each value of m in $\{100, 130, 160, \ldots, 580\}$, train your Naive Bayes classifier on the first m training examples (that is, use the data given by `XTrain(1:m,:)` and `yTrain(1:m)`). Plot the training and testing error for each such value of m. The x-axis of your plot should be m, the y-axis should be error, and there should be one curve for training error and one curve for testing error. Explain the general trend of both the training and testing error curves.

As the size of the data set increases, the train error increases and the test error decreases, approaching each other. This behavior can be explained in great part by *overfitting*. Overfitting occurs when the classifier models noise present in the training data; when this happens, the classifier has a very good performance on the data that was used for training, but it lacks prediction accuracy on new data, which explains why the testing error is large. Overfitting decreases as the size of the data set increases.
This exercise should help you understand the importance of using a test set (or alternative techniques like cross-validation) to evaluate the performance of a classifier.

The following code is used to produce the plot:

```
load HW2Data.mat

train_sizes = 100:30:580;
num_runs = numel(train_sizes);

train_errors = zeros(num_runs, 1);
test_errors = zeros(num_runs, 1);

for i = 1:num_runs
    m = train_sizes(i)

    D = NB_XGivenY(XTrain(1:m,:), yTrain(1:m));
    p = NB_YPrior(yTrain(1:m));
    yHatTrain = NB_Classify(D, p, XTrain(1:m,:));
    yHatTest = NB_Classify(D, p, XTest);
    train_errors(i) = ClassificationError(yHatTrain, yTrain(1:m));
    test_errors(i) = ClassificationError(yHatTest, yTest);
end

hold on
plot(train_sizes, train_errors, 'r-')
plot(train_sizes, test_errors, 'b-')
xlabel 'Training Set Size'
ylabel 'Error'
legend('train', 'test')
print -color -deps 'error_plot.eps'
```



(h) [**8 Points**] Finally, we will try to interpret the learned parameters. Train your classifier on the data contained in `XTrain` and `yTrain`. For each class label $y \in \{1, 2\}$, create three lists according to the following criteria (Note that some of the words may look a little strange because we have run them through a stemming algorithm that tries to make words with common roots look the same. For example, "stemming" and "stemmed" would both become "stem"):

- Top five words that the model says are most likely to occur in a document from class y. That is,

the top five words according to this metric:

$$P(X_w = 1|Y = y)$$

- Top five words w according to this metric:

$$\frac{P(X_w = 1|Y = y)}{P(X_w = 1|Y \neq y)}.$$

- Top five words w according to this metric:

$$\frac{P(X_w = 1|Y = y)}{\max_v P(X_v = 1|Y = y)}.$$

Which list of words is more informative about the class y? Briefly explain your reasoning.

Solution:

```
load('HW3Data.mat')
D = NB_XGivenY(XTrain, yTrain);
[sorted, ixs] = sort(D(1,:), 'descend');
econo_most_likely = Vocabulary(ixs);
[sorted, ixs] = sort(D(2,:), 'descend');
onion_most_likely = Vocabulary(ixs);
[sorted, ixs] = sort(D(1,:) ./ D(2,:), 'descend');
econo_most_discriminating = Vocabulary(ixs);
[sorted, ixs] = sort(D(2,:) ./ D(1,:), 'descend');
onion_most_discriminating = Vocabulary(ixs);
[sorted, ixs] = sort(D(1,:) ./ max(D(2,:)), 'descend');
econo_alt_metric = Vocabulary(ixs);
[sorted, ixs] = sort(D(2,:) ./ max(D(1,:)), 'descend');
onion_alt_metric = Vocabulary(ixs);

% econo_most_likely(1:5) = ['the', 'to', 'of', 'in', 'a']
% onion_most_likely(1:5) = ['a', 'and', 'the', 'to', 'of']
% econo_most_discriminating(1:5) = ['organis', 'reckon', 'favour', 'centr',
% 'labour']
% onion_most_discriminating(1:5) = ['4enlarg', '5enlarg', 'percent',
% 'realiz', 'coach']
% econo_alt_metric(1:5) = ['the', 'to', 'of', 'in', 'a']
% onion_alt_metric(1:5)  = ['a', 'and', 'the', 'to', 'of']
```

The second list is the most informative one. It indicates the words that are probable for the class y and not for the other one (i.e., those that maximize the ratio). This appears to be more descriptive, since the words "the", "to", "of", "in", "a", and "and" appear in almost all written articles. It is interesting to notice that the words "favour" and "labour" (with their British spellings!) are two of the most discriminating words for articles from The Economist, since it is a British magazine.