10-601 Machine Learning, Fall 2011: Homework 2

Machine Learning Department
Carnegie Mellon University

Due: October 6th, 2011, 5pm

Instructions There are 2 questions on this assignment. The second question involves coding, so start early. Please submit your completed homework to Sharon Cavlovich (GHC 8215) by 5 PM on Thursday, October 6th, 2011. Submit your homework as 2 **separate** sets of pages, one for each question. Please staple or paperclip all pages from a single question together, but **DO NOT** staple pages from different questions together. This will make it much easier for the TA's to split up your homework for grading. Include your name and email address on each set.

1 MLE and MAP [William Bishop, 20 points]

In this problem we will find the maximum likelihood estimator (MLE) and maximum a posteriori (MAP) estimator for the mean of a univariate normal distribution. Specifically, we assume we have N samples, $x_1, \ldots, x_N$ independently drawn from a normal distribution with *known* variance σ^2 and *unknown* mean μ.

1. [5 Points] Please derive the MLE estimator for the mean μ. Make sure to show all of your work.
2. [12 Points] Now derive the MAP estimator for the mean μ. Assume that the prior distribution for the mean is itself a normal distribution with mean ν and variance β^2. Please show all of your work. HINT: You may want to make use of the fact that:

$$\beta^2\left(\sum_{i=1}^{N}(x_i-\mu)^2\right)+\sigma^2(\mu-\nu)^2 = \left[\mu\sqrt{N\beta^2+\sigma^2}-\frac{\sigma^2\nu+\beta^2\sum_{i=1}^{N}x_i}{\sqrt{N\beta^2+\sigma^2}}\right]^2-\frac{[\sigma^2\nu+\beta^2\sum_{i=1}^{N}x_i]^2}{N\beta^2+\sigma^2}+\beta^2\left(\sum_{i=1}^{N}x_i^2\right)+\sigma^2\nu^2$$

3. [3 Points] Please comment on what happens to the MLE and MAP estimators as the number of samples N goes to infinity.

1.1 Answer to Part A

First, we derive the likelihood term:

$$\begin{aligned}P(x_1,\ldots,x_N|\mu) &= \prod_{i=1}^{N}P(x_i|\mu)\\ &= \prod_{i=1}^{N}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\end{aligned}$$

Next, we note that log is a monotonically increase function, so we can maximize the log-likelihood:

$$\log\left(P(x_1,\ldots,x_N|\mu)\right) = \sum_{i=1}^{N} \log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

We take derivatives of this with respect to μ and find:

$$\frac{d\log\left(P(x_1,\ldots,x_N|\mu)\right)}{d\mu} = \sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2}$$

Setting the left hand side equal to zero, we find:

$$\begin{aligned}
0 &= \sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2} \\
0 &= \sum_{i=1}^{N} (x_i-\mu) \\
\sum_{i=1}^{N} \mu &= \sum_{i=1}^{N} x_i \\
N\mu &= \sum_{i=1}^{N} x_i \\
\hat{\mu} &= \frac{\sum_{i=1}^{N} x_i}{N}
\end{aligned}$$

1.2 Answer to Part B

There are two ways to solve this problem. We first show an easier way, which is sufficient to find the MAP estimator.

We use Bayes' rule to write:

$$P(\mu|x_1,\ldots x_n) = \frac{P(x_1,\ldots,x_N|\mu)P(\mu)}{P(x_1,\ldots,x_N)}$$

From part A, we know that we can write:

$$P(x_1,\ldots,x_N|\mu) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

and we are given that:

$$P(\mu) = \frac{1}{\sqrt{2\pi\beta^2}} e^{-\frac{(\mu-\nu)^2}{2\beta^2}}$$

Thus, we desire to find the value of μ which maximizes:

$$P(\mu|x_1,\ldots x_n) = \frac{\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right) \frac{1}{\sqrt{2\pi\beta^2}} e^{-\frac{(\mu-\nu)^2}{2\beta^2}}}{C}$$

where $C = P(x_1, \ldots, x_N)$.

We note that because we are simply looking for the value of μ that maximizes this expression we can take the log of both side and write:

$$\log\left(P(\mu|x_1, \ldots x_n)\right) = \left(\sum_{i=1}^{N} -\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{(x_i-\mu)^2}{2\sigma^2}\right) - \log\left(\sqrt{2\pi\beta^2}\right) - \frac{(\mu-\nu)^2}{2\beta^2}$$

Taking the derivative with respect to μ, we have:

$$\frac{\partial \log\left(P(\mu|x_1, \ldots x_n)\right)}{\partial \mu} = \left(\sum_{i=1}^{N} \frac{x_i-\mu}{\sigma^2}\right) - \frac{\mu-\nu}{\beta^2}$$

Setting this equal to zero, we have:

$$\begin{aligned}
0 &= \left(\sum_{i=1}^{N} \frac{x_i-\mu}{\sigma^2}\right) - \frac{\mu-\nu}{\beta^2} \\
\frac{\mu-\nu}{\beta^2} &= \sum_{i=1}^{N} \frac{x_i-\mu}{\sigma^2} \\
\frac{\mu-\nu}{\beta^2} &= -\frac{\sum_{i=1}^{N} x_i}{\sigma^2} - \frac{N\mu}{\sigma^2} \\
\frac{\mu}{\beta^2} + \frac{N\mu}{\sigma^2} &= \frac{\sum_{i=1}^{N} x_i}{\sigma^2} + \frac{\nu}{\beta^2} \\
\frac{(\sigma^2 + N\beta^2)\mu}{\sigma^2\beta^2} &= \frac{\sigma^2\nu + \beta^2\sum_{i=1}^{N} x_i}{\sigma^2\beta^2} \\
\hat{\mu} &= \frac{\sigma^2\nu + \beta^2\sum_{i=1}^{N} x_i}{\sigma^2 + N\beta^2}
\end{aligned}$$

The second way involves a little more work. In this way, we first show that the posterior distribution is itself a Gaussian, and we then use the fact that the mean of a Gaussian is where it achieves its maximum value to find the MAP.

We start with the fact that we again want to find:

$$P(\mu|x_1, \ldots x_n) = \frac{\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{\sigma^2}}\right) \frac{1}{\sqrt{2\pi\beta^2}} e^{-\frac{(\mu-\nu)^2}{2\beta^2}}}{C}$$

where $C = P(x_1, \ldots, x_N)$. However, instead of simply maximizing this function, we first show that $P(\mu|x_1, \ldots x_n)$ is itself a Gaussian. Note, that we can write:

$$\begin{aligned}
P(\mu|x_1, \ldots x_n) &\propto \left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right) \frac{1}{\sqrt{2\pi\beta^2}} e^{-\frac{(\mu-\nu)^2}{2\beta^2}} \\
&\propto \left(\prod_{i=1}^{N} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right) e^{-\frac{(\mu-\nu)^2}{2\beta^2}} \\
&= e^{-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{N}(x_i-\mu)^2\right)} e^{-\frac{(\mu-\nu)^2}{2\beta^2}} \\
&= e^{-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{N}(x_i-\mu)^2\right) - \frac{(\mu-\nu)^2}{2\beta^2}} \\
&= e^{-\frac{\beta^2\left(\sum_{i=1}^{N}(x_i-\mu)^2\right) + \sigma^2(\mu-\nu)^2}{2\sigma^2\beta^2}}
\end{aligned}$$

At this point, we make use of the hint provided and write:

$$\begin{aligned}
P(\mu|x_1,\dots x_n) &\propto e^{-\frac{1}{2\sigma^2\beta^2}\left(\left[\mu\sqrt{N\beta^2+\sigma^2}-\frac{\sigma^2\nu+\beta^2\sum_{i=1}^N x_i}{\sqrt{N\beta^2+\sigma^2}}\right]^2-\frac{[\sigma^2\nu+\beta^2\sum_{i=1}^N x_i]^2}{N\beta^2+\sigma^2}+\beta^2\left(\sum_{i=1}^N x_i^2\right)+\sigma^2\nu^2\right)} \\
&= e^{-\frac{1}{2\sigma^2\beta^2}\left(\left[\mu\sqrt{N\beta^2+\sigma^2}-\frac{\sigma^2\nu+\beta^2\sum_{i=1}^N x_i}{\sqrt{N\beta^2+\sigma^2}}\right]^2\right)} e^{\frac{1}{2\sigma^2\beta^2}\left(\frac{[\sigma^2\nu+\beta^2\sum_{i=1}^N x_i]^2}{N\beta^2+\sigma^2}+\beta^2\left(\sum_{i=1}^N x_i^2\right)+\sigma^2\nu^2\right)} \\
&\propto e^{-\frac{1}{2\sigma^2\beta^2}\left(\left[\mu\sqrt{N\beta^2+\sigma^2}-\frac{\sigma^2\nu+\beta^2\sum_{i=1}^N x_i}{\sqrt{N\beta^2+\sigma^2}}\right]^2\right)} \\
&= e^{-\frac{N\beta^2+\sigma^2}{2\sigma^2\beta^2}\left(\left[\mu-\frac{\sigma^2\nu+\beta^2\sum_{i=1}^N x_i}{N\beta^2+\sigma^2}\right]^2\right)}
\end{aligned}$$

Note this is just the kernel of a Normal distribution with mean $\frac{\sigma^2\nu+\beta^2\sum_{i=1}^N x_i}{N\beta^2+\sigma^2}$ and variance $\frac{\sigma^2\beta^2}{N\beta^2+\sigma^2}$ (so the posterior is a normal distribution). Therefore, since a normal achieves at its maximum its mean, the MAP estimator must be:

$$\hat{\mu}_{\text{MAP}} = \frac{\sigma^2\nu+\beta^2\sum_{i=1}^N x_i}{N\beta^2+\sigma^2}$$

1.3 Answer to Part C

The MAP estimator is:

$$\begin{aligned}
\hat{\mu}_{\text{MAP}} &= \frac{\sigma^2\nu+\beta^2\sum_{i=1}^N x_i}{N\beta^2+\sigma^2} \\
&= \frac{\sigma^2\nu}{N\beta^2+\sigma^2}+\frac{\beta^2\sum_{i=1}^N x_i}{N\beta^2+\sigma^2} \\
&= \frac{\sigma^2\nu}{N\beta^2+\sigma^2}+\frac{\sum_{i=1}^N x_i}{N+\frac{\sigma^2}{\beta^2}} \\
&= \frac{\sigma^2\nu}{N\beta^2+\sigma^2}+\frac{\frac{1}{N}\sum_{i=1}^N x_i}{1+\frac{\sigma^2}{N\beta^2}}
\end{aligned}$$

and the MLE estimator is:

$$\hat{\mu}_{\text{MLE}} = \frac{\sum_{i=1}^N x_i}{N}$$

Notice that as $N\to\infty$, $\frac{\sigma^2}{N\beta^2}\to 0$, $\frac{\sigma^2\nu}{N\beta^2+\sigma^2}\to 0$ and therefore $\hat{\mu}_{\text{MAP}}\to\frac{\sum_{i=1}^N x_i}{N}=\hat{\mu}_{\text{MLE}}$. Thus, as the number of samples increases, the two estimators become identical.

2 Bag of Words [Shing-hon Lau, 40 points]

In this problem, you will code up a Naive Bayes classifier to perform "Bag of Words"-style classification of text documents. Our goal is to train a classifier that can accurately predict whether newsgroup posts come from a newsgroup about baseball or a newsgroup about (ice) hockey.[1] Recall that in "Bag of Words"-style

[1]For those of your unfamiliar with these sports, you can read about them at http://en.wikipedia.org/wiki/Baseball and http://en.wikipedia.org/wiki/Ice_hockey.

classification, we make the assumption that the probability a word appears in document is conditionally independent of the word position given the class of the document.

If we have a particular document D_i with w_i words in it, we can compute the probability that D_i comes from a class y as:

$$\mathbb{P}(D_i|Y=y) = \prod_{j=1}^{w_i} \mathbb{P}(X_j = x_j|Y=y).$$

Here, X_j represents the j^{th} position in document D_i and x_j represents the actual word that appears in the j^{th} position in document D_i. As a concrete example, we might have the first document (D_1) which contains 300 words ($w_1 = 300$). The document might belong to the hockey newsgroup ($y =$ 'hockey') and the 15^{th} word in the document might be 'player' ($x_{15} =$ 'player').

Of course, for the purposes of classification, we are more interested in the probability distribution over the document classes given a particular document. We can use Bayes Rule to write:

$$\mathbb{P}(Y=y|D_i) = \frac{\mathbb{P}(Y=y)\prod_{j=1}^{w_i}\mathbb{P}(X_j=x_j|Y=y)}{\mathbb{P}(X_1=x_1,\ldots,X_{w_i}=x_{w_i})}.$$

Note that, for the purposes of classification, we can actually ignore the denominator here and write:

$$\mathbb{P}(Y=y|D_i) \propto \mathbb{P}(Y=y)\prod_{j=1}^{w_i}\mathbb{P}(X_j=x_j|Y=y).$$

1. [2 Points] Why is it that we can ignore the denominator?

★ **SOLUTION:** The denominator is a constant with respect to y, so it has no effect on which value of y achieves a maximum. Alternatively, one can note that the denominator is the same for every class y.

Our prediction, $\hat{y}_i$, for the class is then,

$$\hat{y}_i = \arg\max_y \mathbb{P}(Y=y|D_i) = \arg\max_y \mathbb{P}(Y=y)\prod_{j=1}^{w_i}\mathbb{P}(X_j=x_j|Y=y).$$

When working with products of many probabilities, it is often advisable to take a logarithm. This provides us with two advantages. It turns products into sums, which is often more convenient. It also avoids underflow and numerical accuracy issues. Many computing systems will round 10^{-1000} to 0, which is undesirable when we may be working with thousand-word documents. Taking the logarithm gives us:

$$\hat{y}_i = \arg\max_y log(\mathbb{P}(Y=y)) + \sum_{j=1}^{w_i} log(\mathbb{P}(X_j=x_j|Y=y)).$$

2. [2 Points] Why does $\arg\max_y \mathbb{P}(Y=y|D_i) = \arg\max_y log(\mathbb{P}(Y=y|D_i))$?

★ **SOLUTION:** The logarithm is a strictly monotonic function on [0,1] and all of the inputs are probabilities that must lie in [0,1].

The ZIP file associated with this assignment contains a total of five items:

- vocabulary.txt
- baseball_train_set.csv
- baseball_test_set.csv
- hockey_train_set.csv
- hockey_test_set.csv

The last four files are CSV (comma-separated value) files. Each of these files consists of 50 lines, with each line containing 5822 different values. Each line corresponds to a single post on a newsgroup, and each of the values indicates the number of times a particular word appears in that post. The words are contained in `vocabulary.txt`; there are 5822 lines in total in the file, corresponding to the 5822 distinct words that appeared in either the training or test set.[2] For example, the 5236^{th} line of `vocabulary.txt` is 'the'. Therefore, if we see a '3' in the 5236^{th} position of a line in the CSV file, we know that the word 'the' appeared three times in that message. You need not use `vocabulary.txt` to code up a successful solution, but it is provided if you wish to sanity check your knowledge of the data format.

[2]Typically, one would not know the words that might appear in the test set and you would need to do slightly more bookkeeping to keep track of what words you saw in the training set. Then, you would check if a word in the test set appeared in the training set. However, we're cheating here a little bit to make the assignment easier to code up.

3. [20 Points] Code up a "Bag of Words"-style Naive Bayes classifier to handle the provided data. Your classifier should take as input a training set consisting of baseball and hockey posts and a test set also consisting of baseball and hockey posts. Your classifier should output predictions for the classes of the test set (either 'baseball' or 'hockey'). Use the MLE estimate for all parameters in your classifier. If it arises in your code, define $0 \times log(0) = 0$. You may assume that the two classes occur with equal prior probability, (i.e., $\mathbb{P}(Y = \text{'baseball'}) = \mathbb{P}(Y = \text{'hockey'})$). In case of ties, you should predict 'baseball'.

   You may use the programming language of your choice. If you decide to use Matlab, you can import the CSV files by using File → Import Data. Please turn in a printed version of all your code (including the extension below) and also submit your code electronically via Blackboard. (Hint: If you have more than 100 lines of code, you are almost certainly doing something wrong.)

   ★ **SOLUTION:** The code solution for part 3 and part 5 can be found at the end of the solutions.

4. [3 Points] Train your classifier using all of the training set (50 posts each from baseball and hockey) and have it classify all of the test set (50 posts each from baseball and hockey). Report, in writing, the answers to the following questions: What is your test set accuracy? What did your classifier end up predicting? Why is using the MLE estimate is a bad idea in this situation?

   ★ **SOLUTION:** The classification accuracy is 50%. All of the posts were classified as baseball. The reason that this occurs is because words that appear in the test data without appearing in the training data have log-likelihood -Inf. For each post, both baseball and hockey have a log-likelihood of -Inf, so all posts are predicted as baseball by the tie-breaker rule. Obviously, using the MLE estimate is bad in this situation, since it is very likely that we will see words in our test data that are not present in the training data and we should do better than just guessing baseball for all posts.

5. [5 Points] Extend your classifier so that it can compute a MAP estimate using a fair Dirichlet prior. The prior is fair in the sense that it "hallucinates" that each word appears β times in the train set. Formally, all of the parameters of the Dirichlet prior are equal to β. The easiest way to do this is to modify your classifier to take β as an additional parameter. (Hint: This extension should take no more than 3 lines of code.)

   ★ **SOLUTION:** The code solution for part 3 and part 5 can be found at the end of the solutions.

6. [5 Points] Train your classifier using all of the training set and have it classify all of the test set, using the following values of β: 0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 5, and 10. Submit a plot with β on the x-axis and classification accuracy on the y-axis. Report, in writing, the answers to the following questions: How does classification accuracy change with respect to β? Is using a large value of β good or bad? Why?

   ★ **SOLUTION:** The plot is attached at the end of the solutions. Classification accuracy is high for small values of β, but it drops dramatically as β gets larger. Using a large value of β is bad because then our prior "drowns out" the data; in other words, we end up largely ignoring the data.

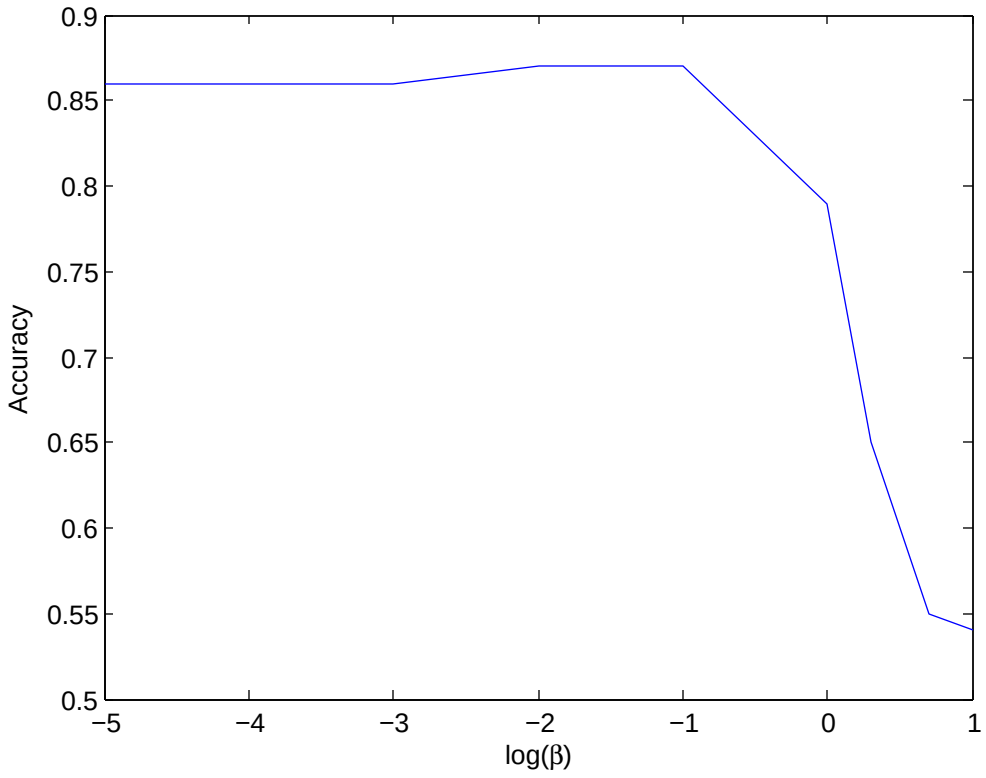
7. [3 Points] Fix $\beta = 0.1$. Train your classifier using the first n posts from the baseball training set and the first n posts from the hockey training set (for a total of $2n$ training examples) for the following values of n: 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50. Test your classifier on the full test set (50 baseball and 50 hockey posts). Submit a plot with n on the x-axis and classification accuracy on the y-axis.

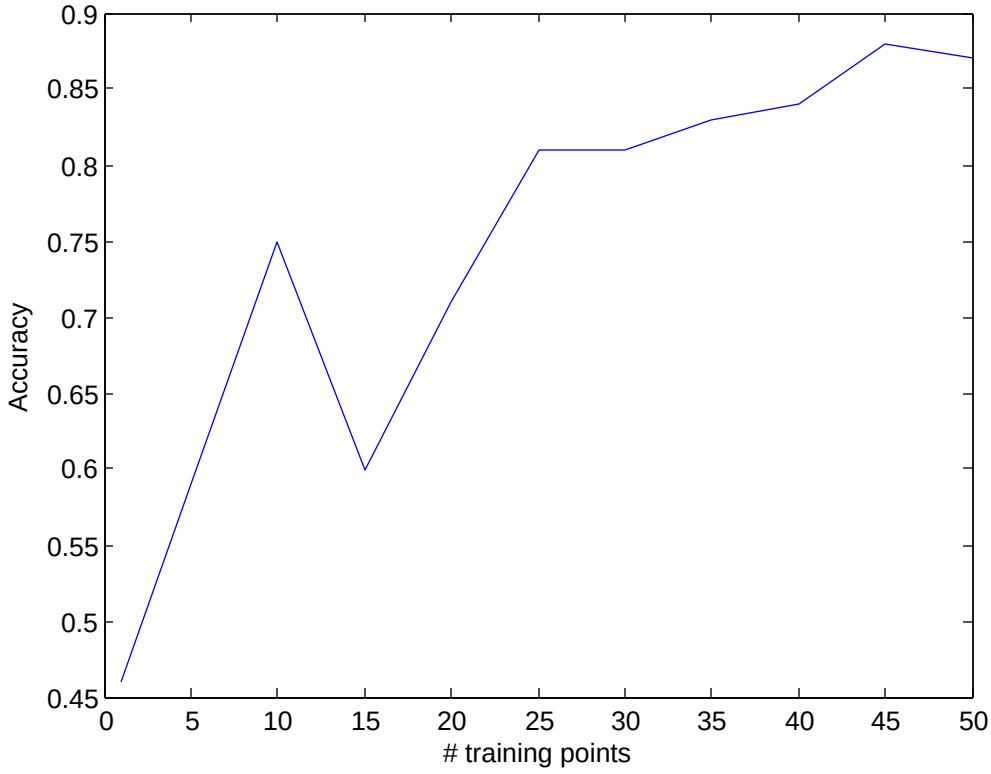
   ★ **SOLUTION:** The plot is attached at the end of the solutions.

The core code for performing Naive Bayes follows. This is the code to produce a MAP estimate. The MLE estimate can be produced by setting beta to 0.

```
function [ accuracy baseball_predictions hockey_predictions] =
naive_bayes(baseball_train_set, hockey_train_set, baseball_test_set, hockey_test_set, beta, num_train)
%NAIVE_BAYES_SOLUTION
%   INPUT:
%   baseball_train_set - 50 by 5822 matrix
%   hockey_train_set - 50 by 5822 matrix
%   baseball_test_set - 50 by 5822 matrix
%   hockey_test_set - 50 by 5822 matrix
%   alpha - single number specifying the number of times each word was
%           "hallucinated" in the prior
%   num_train - number of training points to use

nrows = 50;
ncols = 5822;

% make sure that everything is the correct size
assert(size(baseball_train_set, 1) == nrows);
assert(size(baseball_train_set, 2) == ncols);
assert(size(baseball_test_set, 1) == nrows);
assert(size(baseball_test_set, 2) == ncols);
assert(size(hockey_train_set, 1) == nrows);
assert(size(hockey_train_set, 2) == ncols);
assert(size(hockey_test_set, 1) == nrows);
assert(size(hockey_test_set, 2) == ncols);

% initialize the counts to whatever we have hallucinated
baseball_counts = beta * ones(1, ncols);
hockey_counts = beta * ones(1, ncols);

baseball_counts = baseball_counts + sum(baseball_train_set(1:num_train,:), 1);
hockey_counts = hockey_counts + sum(hockey_train_set(1:num_train,:), 1);

% we take the log
log_baseball_probs = log(baseball_counts / sum(baseball_counts));
log_hockey_probs = log(hockey_counts / sum(hockey_counts));

% predictions for the posts that are ACTUALLY from the baseball or hockey
% newsgroups
baseball_predictions = -1 * ones(1, nrows);
hockey_predictions = -1 * ones(1, nrows);

% predict the baseball posts
for i=1:nrows,
    baseball_prob = 0;
    hockey_prob = 0;
    for j=1:ncols,
        b_term = baseball_test_set(i,j) * log_baseball_probs(j);
        h_term = baseball_test_set(i,j) * log_hockey_probs(j);
        % 0 * -Inf = NaN; we ignore the term since we define 0 * log(0) = 0
        if(~isnan(b_term))
            baseball_prob = baseball_prob + b_term;
```

```
        end
        if(~isnan(h_term))
            hockey_prob = hockey_prob + h_term;
        end
    end
    % predict baseball in the event of a tie
    if(baseball_prob >= hockey_prob),
        baseball_predictions(i) = 1; % correct prediction
    else
        baseball_predictions(i) = 0; % incorrect prediction
    end
end

assert(isempty(find(baseball_predictions == -1)));

% predict the hockey posts
for i=1:nrows,
    baseball_prob = 0;
    hockey_prob = 0;
    for j=1:ncols,
        b_term = hockey_test_set(i,j) * log_baseball_probs(j);
        h_term = hockey_test_set(i,j) * log_hockey_probs(j);
        if(~isnan(b_term))
            baseball_prob = baseball_prob + b_term;
        end
        if(~isnan(h_term))
            hockey_prob = hockey_prob + h_term;
        end
    end
    % predict baseball in the event of a tie
    if(hockey_prob > baseball_prob),
        hockey_predictions(i) = 1; % correct prediction
    else
        hockey_predictions(i) = 0; % incorrect prediction
    end
end

assert(isempty(find(hockey_predictions == -1)));

accuracy = sum(baseball_predictions) + sum(hockey_predictions);
accuracy = accuracy / (2 * nrows);

end
```


0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
Accuracy
−5
−4
−3
−2
−1
0
1
log(β)


0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
Accuracy
0
5
10
15
20
25
30
35
40
45
50
training points