**Naïve Bayes Project Report**

**Introduction:**

The **Naive Bayes classifiers** are a family of simple [probabilistic classifiers](http://en.wikipedia.org/wiki/Probabilistic_classifier) that are based on applying [Bayes' theorem](http://en.wikipedia.org/wiki/Bayes%27_theorem) carrying the assumption of the features being independent. Naive Bayes classifiers are practical Bayesian learning methods. This is a simple technique for constructing classifiers, that is, models that assign class labels to problem instances

The naive Bayes classifier applies to learning tasks where an instance x is described by a conjunction of attribute values and where the target function f (x) can take on any value from some finite set.

**Naive Bayes classifier:**

VNB = argmax P (vj) πi (aI,vj) (vj belongs to V)

For some types of probabilistic models, naive Bayes classifiers can be efficiently trained in [supervised way of learning](http://en.wikipedia.org/wiki/Supervised_learning). Even though Naive ayes classifiers have a very naive design with very simple assumptions, they have worked pretty well in most of the complex real-world situations.

Naive Bayes classifiers are highly scalable, requiring a number of parameters and the number of variables in a learning problem.

An advantage of Naive Bayes is that it only requires a small amount of training data to estimate the parameters necessary for classification. Another advantage is that the naive Bayes is cost efficient to be implemented.

**Description of how my code works.**

I have done my project on implementing naive bayes classifier using MATLAB. I have first used the textread function for reading the Vocabulary file and dlmread function to read the data files and the label files.

I have then calculated the probability, that is the MLE by using a for loop for the entire data set and then adding them into a matrix (I have taken a column matrix)

P(Y)

*P (Yk) = no. of docs labelled Y/total no. of docs*

I, then calculate MAP. First, the count of the words in the document is added to the matrix of document IDs and word IDs. And then the MAP is calculated by the given formula.

*P (X|Y) =*

*P*(*Xi*|*Yk*)=(*count* *of* *Xi* *in* *Yk*)+(*α*−1)/(*total* *words* *inYk*)+((*α*−1)∗(*length* *of* *vocab* *list*)))

*α*=1+*β*

*β*=1|*V*|

Once the MAP is calculated, the test data needs to be classified and for this I need to build a matrix for the attributes that are present in the test data, (testinput).

Classification is done by

*Ynew*=*argmax*[ *log*2(*P*(*Yk*))+∑*i*(# *of* *Xnewi*)*log*2(*P*(*Xi*|*Yk*))]

Matrix building is done by using a function named “sparse” which builds a sparse matrix. We consider building a sparse matrix because it does not consume space because of the zeros in the matrix. A transpose of the sparse matrix is taken and it is multiplied with the log of MAP. The maximum value of the product is taken and a confusion matrix is formed between the maximum value and the test label data. This confusion matrix is formed by the function “confusionmat”. Once we get the confusion matrix, we take the sum of the diagonal elements and then divide it by the testsize. This gives the accuracy value. The accuracy values for various values of β are tabulated as follows

|  |  |
| --- | --- |
| Accuracy value | Beta value |
| 78.3477681545636 | 0.00001 |
| 78.8674217188541 | 0.0001 |
| 79.6802131912059 | 0.001 |
| 80.3464357095270 | 0.01 |
| 80.4796802131912 | 0.1 |
| 78.1079280479680 | 1 |

**Accuracies ay various stages**

The accuracy that I got after running the code is 78.46 (β=1/V). When I tried the code for various other Beta (β) values, I got the accuracy going upto 80.47% and 80.34% when the β value is taken between 0.1 and 0.01 respectively. But as we further go on increasing the β value, it is found that the accuracy falls and comes down to 78.10% when β takes the value of 1. It can be observed that on increasing the β value, the accuracy increases then fluctuates on further increase of β and then steadily decreases on further increasing β to 1.

The accuracy obtained when the β value is set to 1/V is most preferred because it’s a stable value, unlike the value obtained (80%) because of smoothing that is obtained at various other values of β.

**Answers to the questions 1-7**

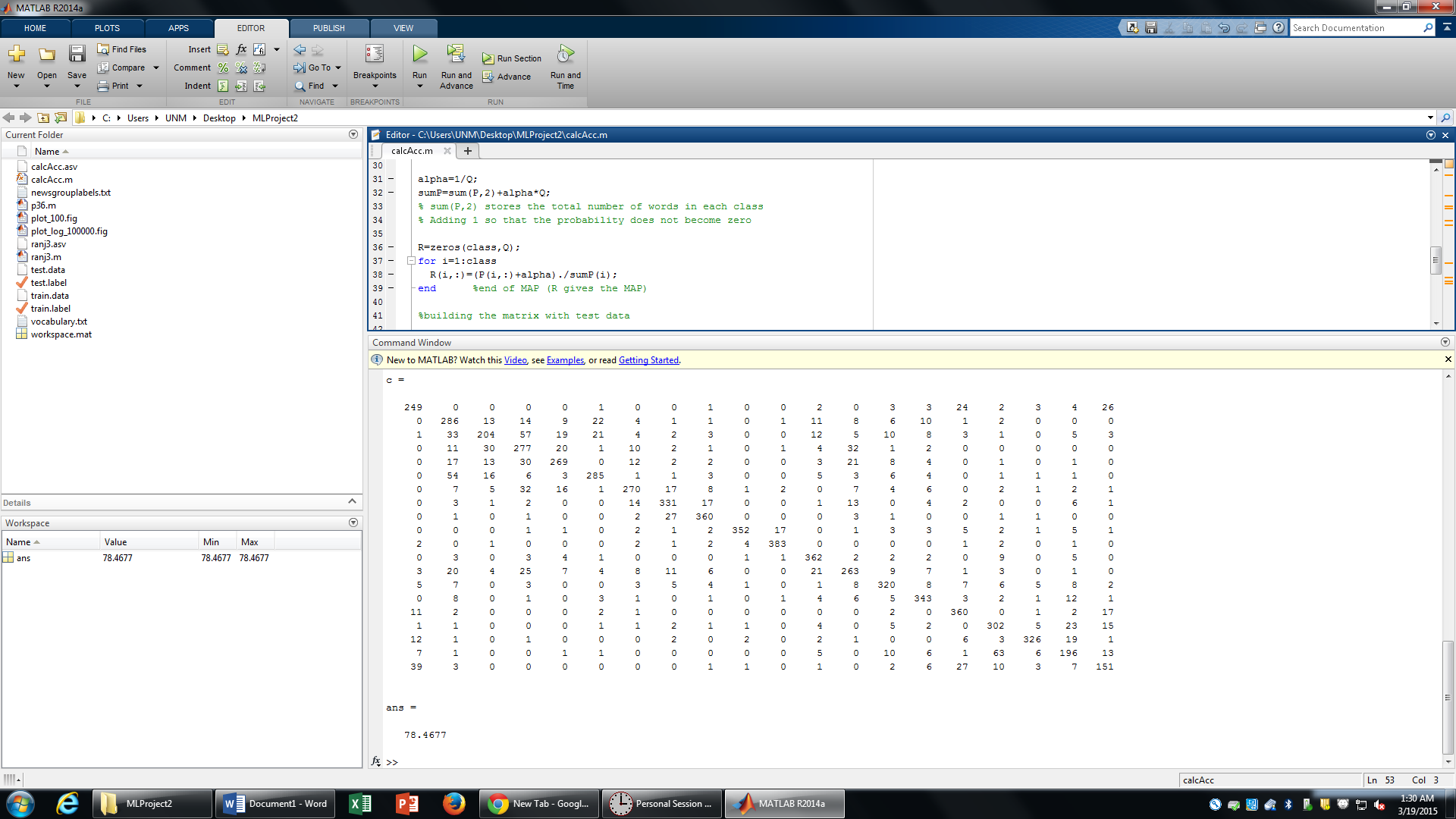
**1.** According to the question, the assumption is that each word in a document is occurring just once. So for the given 1000 documents, estimating the parameters accurately gets difficult because the vocabulary list is too big (50,000). Hence this small document set may result in overfitting. This is the reason that I could think of keeping the data size and overfitting problem in mind.

**2.** The no. of correctly classified documents are 5889 and the total number of documents are 7505.

Hence the accuracy becomes (5889/7505)\*100=78.467 The confusion matrix obtained is

Confusion matrix

C=



**3.** Yes, there are a few newsgroups where the algorithm has got confused, and has classified incorrectly. We can see in the confusion matrix that the diagonal values are high. This diagonal values gives the number of correctly classified documents. Apart from the diagonal values, there are few other values in the matrix, which show the misclassification.

If a confusion matrix shows (2, 8) 78, it means that in 2nd row and 8th column, the number of wrongly classified elements are 78.

Similarly, by seeing the obtained confusion matrix, we can say that,

Column2, row3- the number of misclassified documents are 33

Column2, row6- the number of misclassified documents are 54

column1, row20- the number of misclassified documents are 39

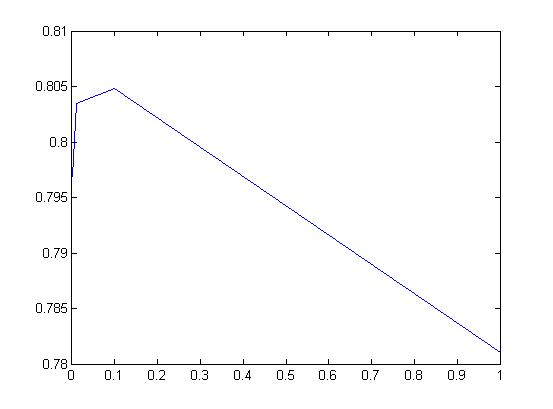
column17, row19- the number of misclassified documents are 63

column13, row4- the number of misclassified documents are 32

**4.** The accuracy values for various values of β are tabulated as follows

|  |  |
| --- | --- |
| Accuracy value | Beta value |
| 78.3477681545636 | 0.00001 |
| 78.8674217188541 | 0.000100000000000000 |
| 79.6802131912059 | 0.00100000000000000 |
| 80.3464357095270 | 0.0100000000000000 |
| 80.4796802131912 | 0.100000000000000 |
| 78.1079280479680 | 1 |

And the plot is



This is the plot of accuracy on y axis with β value (ranging from 0.00001 to 1) on y axis

It is observed that as we increase the β value, the accuracy first increases and then decreases. We see such a fluctuating behaviour because when the β value is small, the probability of the words that occur for a very small number of times, getting neglected is very low. And hence we can say that, for small values of β, the probability of these words tends to increase.   
As the β value increases, the classifier may ignore the words that occur for a very small number of time. This may be the reasons for the decrease in the accuracy rate.

Also, the use of dirichlet distribution, increases the accuracy and the highest value that we have got is a very unstable value. Hence it decreases immediately on further increase of β value.

**5.** Here, I implement a new method where I take the probability values of each word from the matrix (20X161188). Once the probabilities are calculated, I sort the array and get the top 100 probability values. Once I get the values, with the help of its index, I map the indices with the word list, and get the top 100 most probable words from the vocabulary list. I feel, this method is still probabilistic, since we are going about finding the words by calculating the probability.

**6.** The implementation of this method is done by inserting the following piece of code

a=sum(P)/61188;

[b ix]=sort(a,100)

[at,ct]=sort(b(:),'decend');

for i=1:100

dt(i)=ct(i)

end

VB(dt);

This will give the list of top 100 most probable words. The list is as follows.



**7.** Database bias is something that happens when a testing dataset contains some words which are not present in the training set (new words). There are certain words that are present in both training and testing data sets These set of words make the data sets a biased one (training set). Due to the presence of these new words, which are not present in the training set, and classifying the test data is not done efficiently and this makes the data sets a biased one. By looking at the given data, I feel there are some words which may not be present in the data set. For example, *leehian,* *voyetra,* *hadam* are some of the words that I could find in the given data set which are generally not present in the longer in use now. Hence, a dataset which is biased with many new words, cannot really classify efficiently.

**References**

**References that I had taken for the Report**

<http://en.wikipedia.org/wiki/Naive_Bayes_classifier>

<https://web.stanford.edu/class/cs124/lec/naivebayes.pdf>

<http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf>

<http://www.cs.cmu.edu/~mccallum/textbeyond/papers/fuka.pdf>

<https://piazza.com/class/i55s4m0ttv7ck?cid=40>

<http://www.academia.edu/9040601/NAIVE_BAYES_CLASSIFIER_WITH_MODIFIED_SMOOTHING_TECHNIQUES_FOR_BETTER_SPAM_CLASSIFICATION>

I referred the Carnigie Melon University’s research papers and Home works for answering the reason for the fluctuation of the accuracy with the changing β values and also for 7th question.

**References for writing the code:**

I referred to the p36.m code that was given by the professor for the calculation of MAP.

I also referred to the concepts and formulae that was posted on piazza.