**MULTINOMIAL**

I decided to use what the lecture slides called the "somewhat more subtle version" of smoothing when I added smoothing to prevent over-fitting; this “more subtle version” is where I used an alpha instead of one. My first chosen alpha was 0.5. I also used log probabilities instead of just probabilities to prevent underflow.

My first test run was an accuracy test. I wrote a function to pick the best category for a message feature based on my Multinomial Classifier, then ran it on twenty message features per category. In a test set of 20 categories, this meant I scored 400 features. 378 features were classified correctly, for a total of 93.25% correct classifications.

I then attempted to test different alphas to see how I could improve my accuracy, with these results:

|  |  |
| --- | --- |
| Alpha | Accuracy |
| 0.7 | 92.5% |
| 0.6 | 92.75% |
| 0.5 | 93.25% |
| 0.4 | 93.75% |
| 0.3 | 94.5% |

Unsurprisingly, accuracy improves as I used smaller alphas (and thus closer to over-fitting). I will leave my alpha at 0.5 for now and adjust the alpha when I reach the k-fold section, where changing the alpha is not as explicitly overfitting.

\*Note on the multinomial code: I took “(tab-separated, one per line)” to mean one set-of-twenty-messages-from-the-same-newsgroup per line. Thus, I have twenty lines of twenty tab-separated newsgroup guesses, rather than one newsgroup guess per each of four hundred lines.

\*Additional note on the multinomial code: I used an old CS124 homework assignment as a guide.

**K-FOLD**

I started the k-fold on the multinomial classifier, without feature selection, and with alpha = 0.5. I got an average accuracy of 84%. I then decided to test new alphas to try to improve this accuracy, yielding this graph:

This trend was the same as the trend before using k-fold; the lower the alpha, the higher the accuracy. However, 0 is too small and leads to over-fitted data and 1 is, according to lecture slides, simplistic; so I will continue to use my original of 0.5 as an alpha, since it is exactly between 0 and 1.

Average accuracy for multinomial without feature selection and alpha 0.5: 82%

Average accuracy for multivariate without feature selection and alpha 0.5: 83%

I am calculating error as % error = 100% - % accurate.

Error for multinomial classifier without feature selection and alpha 0.5: 18%

Error for binomial classifier without feature selection: 17%

**TWCNB**

I started out with simply CNB. I continued to use my original decision of 0.5 as an alpha, as explained above. As explained in the handout, I use the regular Naïve Bayes Multinomial code but changed two variables:  
numOccurrencesWordInDocsOfSameClass became numOccurrencesWordInDocsOfOtherClass

numOfWordsInDocsOfSameClass became numOfWordsInDocsOfOtherClass   
Then returned -1 times the natural log of the probability, to mirror the way the handout was now taking the negative sum of probabilities rather than the positive sum in order to give classes that “this word appears very infrequently in all other classes” a high score instead of a low one. This increased our accuracy to 95.5% instead of 93.25%.

Then I implemented WCNB.