Machine Learning Fall 2016

HW5 Writeup

Zihan Ni (zn8ae)

**1.3**

**thetaPos**

love: 0.00079759628516798691

wonderful: 0.0002166985340981517

best: 0.0010616407174724575

great: 0.00097969589365382872

superb: 9.4691796412637707e-05

still: 0.00072839843394336703

beautiful: 0.00026404443230447055

bad: 0.00048438495857233905

worst: 6.1913866885186195e-05

stupid: 6.0092870800327781e-05

waste: 2.913593735773468e-05

boring: 7.101884730947828e-05

?: 0.0020103796776836931

!: 0.00081216425384685422

UNK: 0.9923281434944915

**thetaNeg**

love: 0.00063709436120604201

wonderful: 8.1417809738791315e-05

best: 0.00071240583521442393

great: 0.00052310942757173421

superb: 2.6460788165107175e-05

still: 0.00067169693034502828

beautiful: 0.00014044572179941503

bad: 0.0014125989989680293

worst: 0.0003663801438245609

stupid: 0.00029717500554658831

waste: 0.00016894195520799198

boring: 0.00034806113663333284

?: 0.0031060894415348888

!: 0.0015387966040631559

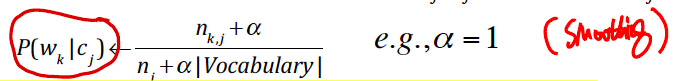
UNK: 0.98996932584018094

\*Note: “love” includes words love, loving, loves, loved

**Explanation of how thetaPos and thetaNeg were computed (use equations if necessary):**

For thetaPos:

First, we accumulate the frequencies of each word in our vocabulary across all the positive documents, which is the first 700 rows in the Xtrain matrix. After we get the accumulated frequencies, we can get a total number of words by simply summing up all of the accumulated frequencies. In order to get the theta for each word with Laplace smoothing, we apply the formula:



Here, we have **α=** 1 and |Vocabulary| = 15. So the equation becomes:

Frequency of word + 1 / Total number of words + 15

We apply the formula for each word in the vocabulary and get thetaPos. Similarly, we can get thetaNeg.

**1.4**

**Multinomial Naive Bayes Classifier (MNBC) test set Accuracy:**

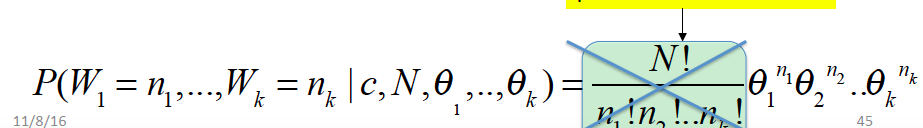
0.675

**sklearn.naive\_bayes.MultinomialNB test set Accuracy:**

0.675

**Explanation of how you test using MNBC (use equations if necessary):**

In order to determin from which class(pos/neg) a test document is from, we need to calculate the joint probability of words in this documents. We apply the formula:



We can leave out the multinomial coefficient part since it is a constant. Moreover, since the thetas are pretty small numbers, we use log likelihood to calculate our document score. In formula, this becomes P = n1 \* log(theta1) + n2 \* log(theta2) … + nk \* log(thetak).

From Xtest, we can get the frequencies for words n1…nk. Thus, we can use the above formula and thetaPos and thetaNeg to calculate the likelihood for positive and negative class. Then, apply MLE we will choose the class with maximum score to be our prediction. Finally, we can calculate the accuracy by comparing the predicted y labels to true lables.

**\*\*MNBC Extra Credit test set accuracy and discussion\*\*:**

First, I have scanned through all the documents in the training corpus and calculate the accumulate frequencies for words(after stemming) other than stop words. Then, I pick 14 top words among the most frequently appered ones. Adding the UNK, the list becomes:

['make', ':', 'get', 'charact', '?',"it'", 'like','ha', 'wa', 'one','movi','thi','hi','film','UNK']

Then I do basically the same classficiation as required except that I will stem all the words and remove stop words when I read the words from documents.

The result I got is 0.615 and 0.615. This result is not as good as the original one because although we are using the most frequent words in the training corpus, the words might not be good indicators of positive or negative attitudes.

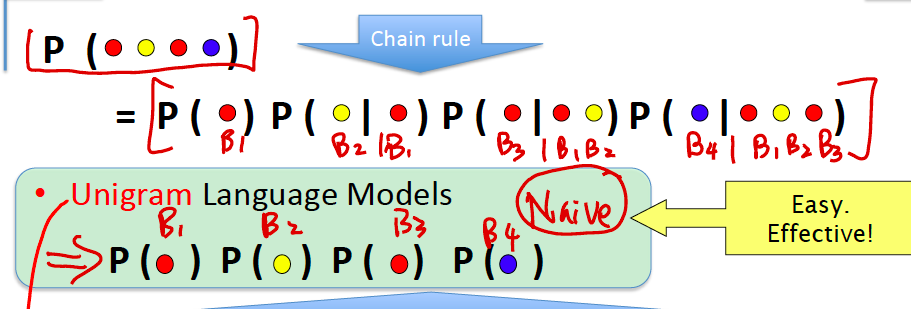
**1.5**

**Multinomial Naive Bayes Classifier Testing with non-BOW test set Accuracy:**

0.675

**Explanation of how you test using MNBC with non-BOW (use equations if necessary):**

Here we do not count the frequencies for words (aka not using BOW). Instead, to calculate the score for a document we just look at each word and refer to the thetaPos and thetaNeg array to sum up the log likelihoods and get scorePos and scoreNeg. The principle is illustrated below:



Similar to the calculation with BOW, multiplying the probabilties might yield a very small number, hence here we add the log likelihood together. At last, we compare the scorePos and scoreNeg and apply MLE, which means we will pick the class with higher score to be our prediction. Finally, we can calculate the accuracy by comparing the predicted y labels to true lables.

**\*\*MNBC non-Bow testing Extra Credit test set accuracy and discussion\*\*:**

Here the test accuracy using extra credit vocabulary is 0.615, which matches the previous two results from extra credit vocabulary. Since we are adding the log likelihood for each word, this should produce the same result as calculating the frequency then multiply the frequencies with the log likelihoods. Indeed, our result verifies that the results are the same.

**1.6**

**thetaPosTrue**

love: 0.349002849002849

wonderful: 0.12393162393162394

best: 0.4928774928774929

great: 0.41452991452991456

superb: 0.06552706552706553

still: 0.37037037037037035

beautiful: 0.1467236467236467

bad: 0.2621082621082621

worst: 0.045584045584045586

stupid: 0.038461538461538464

waste: 0.022792022792022793

boring: 0.05128205128205128

?: 0.5427350427350427

!: 0.2777777777777778

UNK: 0.9985754985754985

**thetaNegTrue**

love: 0.2948717948717949

wonderful: 0.05270655270655271

best: 0.3504273504273504

great: 0.26638176638176636

superb: 0.018518518518518517

still: 0.3247863247863248

beautiful: 0.08974358974358974

bad: 0.4886039886039886

worst: 0.19230769230769232

stupid: 0.15954415954415954

waste: 0.10541310541310542

boring: 0.18518518518518517

?: 0.688034188034188

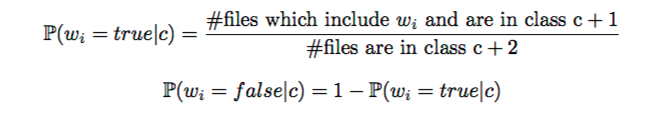
!: 0.39886039886039887

UNK: 0.9985754985754985

\*Note: “love” includes words love, loving, loves, loved

**Explanation of how thetaPosTrue and thetaNegTrue were computed (use equations if necessary):**

We just need to compute the proportion of documents that contains word i in order to get P(wi=true|c). Hence, we set an iterator i and we scan through all rows of Xtrain and see if ith position is non-empty. If yes, then this word appears in this document and we do count+1. According to prior knowledge we know that there are 1400 training documents and 700 of them are positive while 700 are negative. Hence, we know the number of files in each class is 700. Then we can apply the formula:



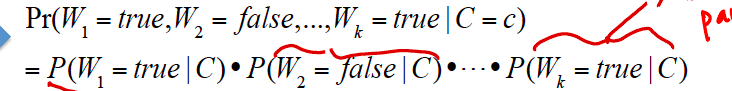
To get the thetaPosTrue and thetaNegTrue. After getting thetaPosTrue and thetaNegTrue, we can get the other haf(False case) easily since there are only two possibilities. Finally, we can calculate the accuracy by comparing the predicted y labels to true lables.

**Multivariate Bernoulli Naive Bayes Classifier (BNBC) test set Accuracy:**

0.69

**Explanation of how you test using BNBC (use equations if necessary):**

In order to get a test document’s score, we just need to check if each word appears in this document. Hence, we can calculate scorePos and scoreNeg by checking if this word presents in the document and if yes we can apply thetaPosTrue and thetaNegTrue. If a word is not present in this document we can look for 1-thetaPosTrue[i] or 1-thetaNegTrue[i]. Similar to before, since multiplying thetas might lead to underflow problem, we add the log likelihoods together to calculate the scorePos and scoreNeg. Then we can apply MLE to determine our prediction class to be the class with maximum sum of log likelihoods.



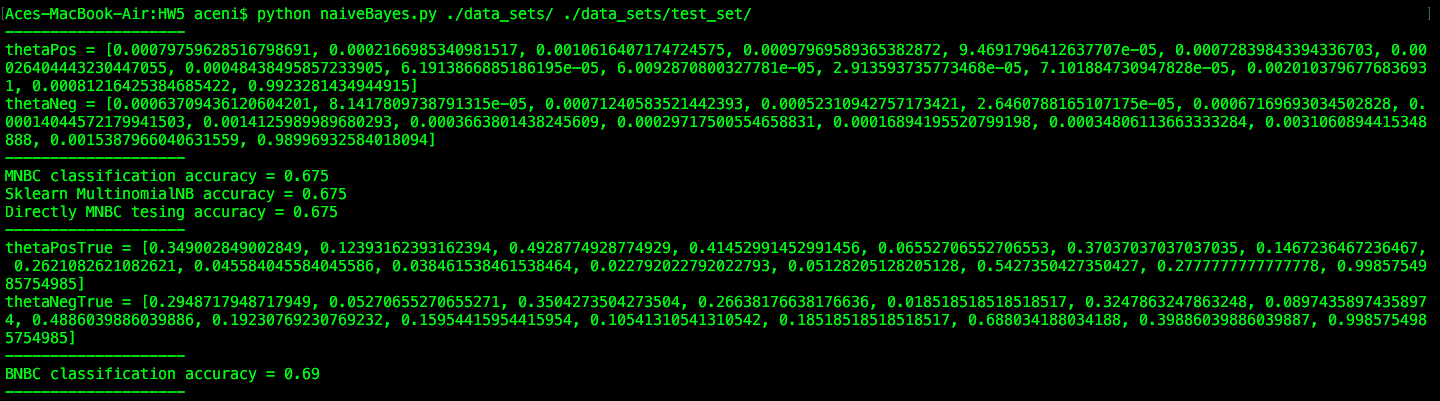
Finally, we can calculate the accuracy by comparing the predicted y labels to true lables.

**\*\*BNBC Extra credit test set accuracy and discussion\*\*:**

Accuracy: 0.5733333333

This score is lower than the MNBC accuracy. It is worth noting that in the required portion, our BNBC accuracy is higher than MNBC accuracy. This difference might due to the aggregate nature of BNBC (thetas do not sum up to 1) and the difference between two vocabularies.

Attachment: Required result screen shot & Extra credit result screen shot

Required: 

Extra Credit:

