Homework 3: Naive Bayes Classification

Aditya Shastry

October 7, 2016

1 Bag of Words

1. The size of vocabulary for the entire training set is 252165.

2. The top 10 words for both the positive and negative classes are common determiners like a, an, the etc. So, these will not help to distinguish the positive and negative classes.
These constructs will be the most used words in most of the english language text.

2 Word Probabilities and Pseudocounts

1. The function is implemented in the code nb.py

2. Below are the listed probabilities from the code:
Word fantastic given pos label 0.000427899015832
Word fantastic given neg label 0.0
Word boring given pos label 0.0
Word boring given neg label 0.000791452314998
From the above probabilities the word Fantastic has a higher probability for positive class and the word Boring has a higher probability for the negative class. This is the usual expectation as Fantastic usually refers to something good and Boring refers to something bad.

3. As seen above, the probability for Fantastic in negative label is 0 and same for Boring in positive label. So, if we are estimating probability of a phrase or a sentence in the test data and the word Fantastic occurs in the negative test data set, then the whole probability becomes 0. This will lead to wrong conclusions. So, it would be a mistake to leave the probabilities at 0.

4. The function is implemented in the code nb.py.

3 Prior and Likelihood

1. Likelihood = $P(w_{d1}, w_{d2}, ..., w_{dn}|y_d) = \prod_{i=1}^{n} P(w_{di}|y_d)$
Taking log on both sides:
$\log(\text{likelihood}) = \sum_{i=1}^{n} P(w_{di}|y_d)$

2. The function is implemented in the code nb.py.

3. The function is implemented in the code nb.py.

4 Normalization and the Decision Rule

1. Normalizer = $P(w_d) = \frac{count(w_d)+\alpha}{\sum_{i=1}^{|V|} count(w_{di})+\alpha|V|}$
Here α is the pseudocount. Here, $count(w_d)$ is the total number of times the word occurs in the corpus. And $\sum_{i=1}^{|V|} count(w_{di})$ is the total number of words in the corpus.

2. $P(y_d|w_d) = \frac{P(w_d|y_d)P(y_d)}{P(w_d)}$
Taking log on both sides:
$log(P(y_d|w_d)) = log(\frac{P(w_d|y_d)P(y_d)}{P(w_d)}) = log(P(w_d|y_d)P(y_d)) - log(P(w_d))$
$= log(P(w_d|y_d)) + log(P(y_d)) - log(P(w_d))$

3. For finding the best posterior, we maximise on the parameter y_d. As the normalizer, w_d, is not dependent on y_d, it will not affect the value of y_d which best maximises the posterior. Hence, the log normalizer need not be calculated.

4. The function is implemented in the code nb.py.

5. The function is implemented in the code nb.py.

5 Evaluation

1. After implementing the function in the code, the classifier's accuracy is 86.96%.

2. The function is implemented in the code nb.py.
According to the results, the highest accuracy was 87.152% for pseudocount value 0.25.

3. One of the wrongly classified files were 10006_7.txt.
By a simple read, it seems like a positive review. But the review has words and phrases like 'disappointed', 'down note', 'really' 'cheesy', 'crap', 'not a top ten or twenty'. These bring down the score of posterior as these words have a higher probability for the negative class. So, just assigning probabilities might not be just enough.
If we notice the reviews, the negative words/phrases are being said only about a few things portrayed in the movie like the oriental culture or the fashion and music in the movie. If we implement a method to assign weights to the likelihood based on the significance of the word on the label, then there is a possibility of correcting some of the mis-classifications.

6 Exploratory Analysis

1. Range of LR function is $[0, \infty)$

2. The function is implemented in the code nb.py.

3. Likelihood ratio for 'fantastic' is 4.068 and likelihood ratio for 'boring' is 0.216.
Likelihood ratio of few of the top 10 words from both labels are : 'the' - 1.036, 'and' - 1.188, 'a' - 1.029, 'of' - 1.089, 'to' - 0.945, 'is' - 1.126.
By simple numerical comparisons, we notice that fantastic is a much better indicator of positive and boring is a much better indicator of negative than the words in top 10.

4. In this case, if LR of a word is very close to 1, it means the likelihood of the word in positive label is almost the same as the likelihood of the word in negative label. So, these words are not very indicative of the class. So, these words are not important for the classifier.
If LR $\gg 1$, then the numerator is very high compared to the denominator. This means the likelihood of the word for positive label is very high compared to that for negative label. So, this word is highly indicative of positive class. Eg. fantastic for positive class. These words are important to the classifier.
If LR $\ll 1$, then the denominator is very high compared to the numerator. This means the likelihood of the word for negative label is very high compared to that for positive label. So, this word is highly indicative of negative class. Eg. boring for negative class. These words are important to the classifier.

7 Bonus

2. Below are the 2 classifiers which will help better classify movie reviews:
   i) Adjectives for cinema terms: For a few terms used constantly in any cinema reference like cinematography, direction, cast, screenplay etc, the adjectives used for these terms can tell a lot about the movie. Eg. good cinematography, bad screenplay, fantastic cast, abysmal directing etc.
   ii) Names mentioned in the movie: There are a few actors, directors, and writers whose names are enough. Eg. Martin Scorcese, Leonardo Dicaprio, Aaron Sorkin etc. If these names are present in the movie review, then there is a higher chance of this being a positive review than a negative.

3. In case of multi-class classification, we will not be able to have static keys for the labels in the dictionaries. So, we will have to initialize empty dictionaries and add the labels in the update model function of the code.

4. The normalizer is just probability of the word, which depends on the counts of the individual words (with or without pseudocounts). So, the classifier being binary or multi-class doesnt change the normalizer.

5. The classify function will have to record all the unnormalized log posteriors in a list, compute the max value and return the corresponding label. The current code will not be able to handle this.