Naïve Bayes Classification for Text Classification

Created By:
Arijit Ganguly
1001871460

1 Problem Statement

The data set contains 19997 documents which belong to 20 different classes. We need to train our naïve bayes algorithm on 50% of the data set, i.e., 9998 documents(approximately 500 documents from each class) and use the remaining 50% as the testing set and predict the predicted classes (newsgroups) to calculate the accuracy of the naïve bayes algorithm.

2 Data

The data set contains 19997 documents in total. It is classified into 20 newsgroups.

Each of the documents contains contents related to the news subgroups.

The data contains the below mentioned frequencies:

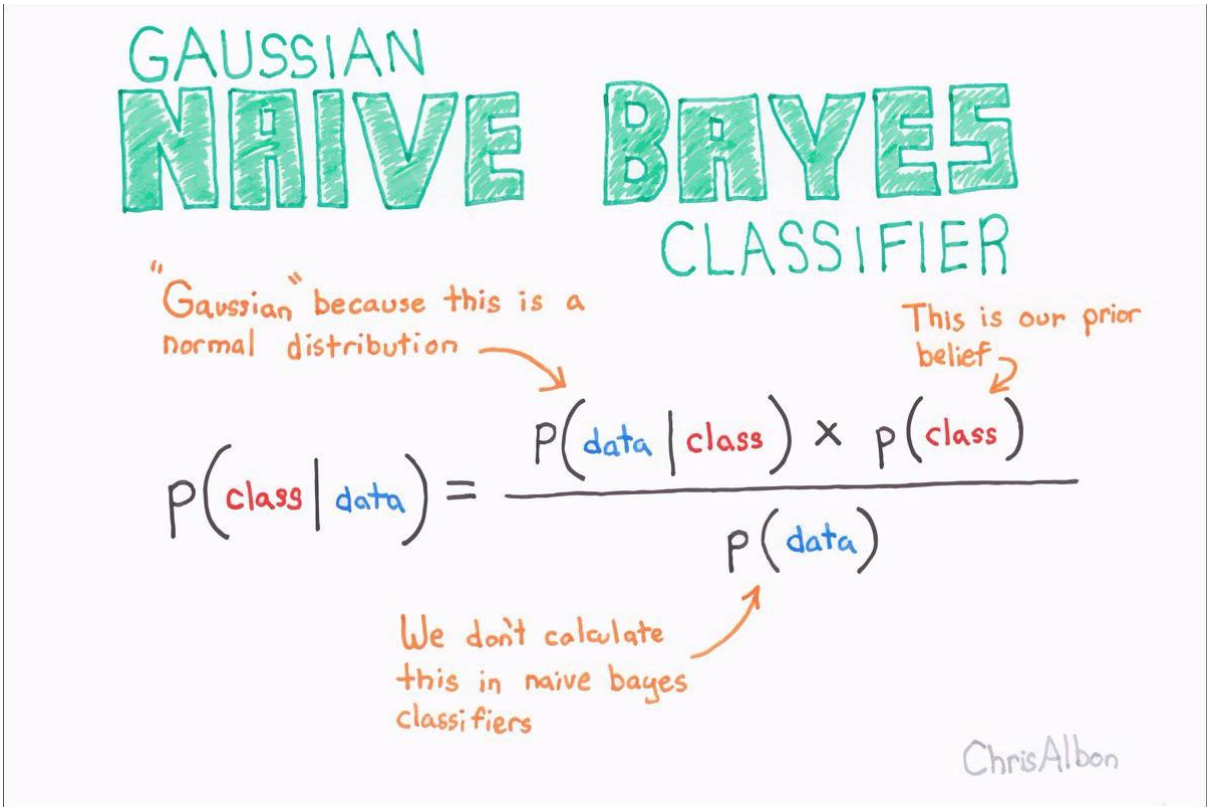
```
alt.atheism:1000
comp.graphics:1000
comp.os.ms-windows.misc:1000
comp.sys.ibm.pc.hardware:1000
comp.sys.mac.hardware:1000
comp.windows.x:1000
misc.forsale:1000
rec.autos:1000
rec.motorcycles:1000
rec.sport.baseball:1000
rec.sport.hockey:1000
sci.crypt:1000
sci.electronics:1000
sci.med:1000
sci.space:1000
soc.religion.christian:997
talk.politics.guns:1000
talk.politics.mideast:1000
talk.politics.misc:1000
talk.religion.misc:1000
```

3 Strategy

The frequency of words in the document can be used to predict the class of the document.

We calculate the number of words in each class by pre-processing the document and then we calculate the probability of each word in the class. We also calculate prior value for each class.

Then using the Bayes theorem, we calculate the predicted class values.



We use the above-mentioned formula in our algorithm to calculate the probability of a test document belonging to a class by calculating the probabilities for each class and the maximum value among that is our predicted value.

Terms Used:

Prior: Probability of predicting the class in the training set.

Class: All the newsgroups available in the dataset are our classes. I use this in the code as **labels** as well.

3.1 Method Used:

3.1.1 Pre-processing

This function helps us to process all the documents and create vectors for each documentation. It performs the below mentioned steps:

1. Lowers the case of content in the document
2. Removes all punctuation marks
3. Removes all the digits
4. Removes all remaining single letters
5. Removes the Stop words

3.1.2 Create Doc Vector

Once the document has been pre-processed, we get only the required words. We use this function to count the occurrence of each word and add it to a dictionary where the key is the word, and the value is the frequency of occurrence. We will use the resultant vector to build the probabilities.

3.1.3 Calculate Prior

This function predicts the probability of each class in the total dataset. It generates a list for each class which is then used in the Bayes theorem to calculate the final probabilities

3.1.4 Calculate Probabilities

This function calculates the probabilities for each word in that class. The calculated probabilities is used to in the Bayes Theorem implementation while predicting the class of the new document.

3.2 Process

Steps:

1. We traverse through all documents to create a vector with the frequency of each word for each one
2. Simultaneously we build the dictionary for each class counting the frequency each word appears for that class.
3. Once everything is completed, we calculate the prior values and the probability of each word appearing in that class.
4. Then we move to testing phase, we start creating the vectors for the remaining documents.
5. Once we get the frequencies of all the documents, then we need to retrieve the probabilities for each of the words from the class vector list and calculate the probability of the document being that part of the class by finding the product of all the words, present in test document vector, in that class.
6. Once the probability for that document is calculated, we multiply it to the prior to finally get the probability the document belongs to the class.
7. We repeat the process for each class and take the max probability as the predicted class.
8. Then we repeat this process for each document in the testing part and create accuracy scores

4 Results

In the first pass of the project, I used only the direct probabilities that was calculated and there was an issue with the calculation.

Issue #1

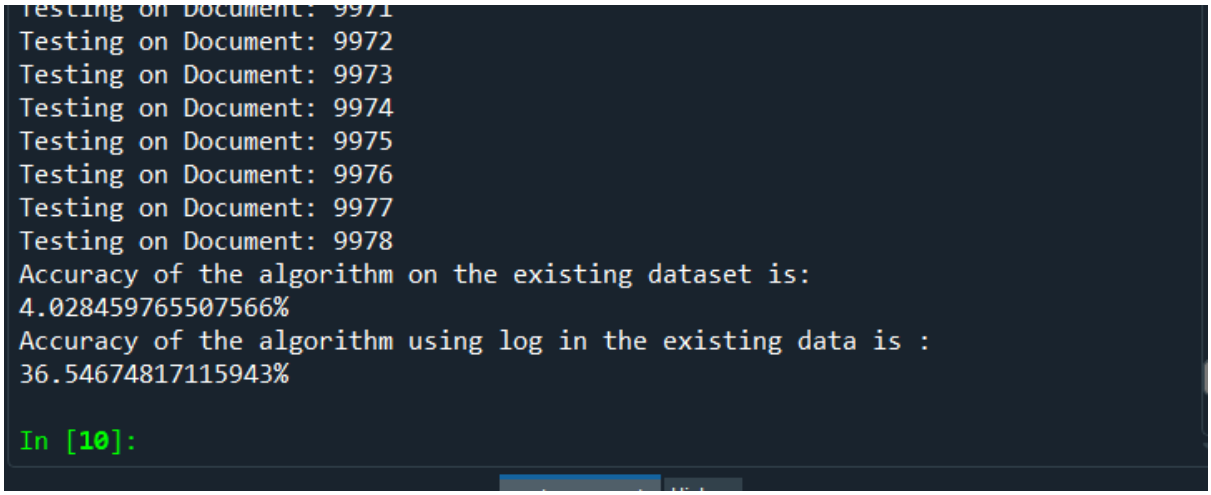
Finding the product of the probabilities of each word in the document was creating so small that eventually the computer started reading them as 0.0 after some number of calculations.

This started classifying my documents as the first class by default as all classes went to 0.0

Solution:

Use of Log function when calculating the probabilities. This reduced the number of zeros values that I was getting.

Algorithm classification jumped from 4% accuracy to 36%



Issue #2

If a word is available in the test document but not available in the class dictionary, then the word is ignored.

Solution:

Implemented Multinomial Naïve Bayes to add an alpha value to each count of the word and then add the product of number of missing words and the alpha to the total count of the class and re-calculates the probabilities.

****Calculation is taking too long to predict the classes****

Issue #3

The words even after pre-processing creates generates words like ‘aa’, ‘aaa’ as words. Code is designed to use all words as features and is using these words as well if they are present in the test document.

Solution:

Some of the words have been added to the stop words list.

****Permanent solution required. Not sure how these are generated. Even if they are how to prevent them?”**

Issue #4

Code is designed to use all the words found in the document however some of them can be very bad predictors.

Solution:

Using on the words with higher frequencies or words related to them as the frequencies(antonyms and synonyms)

5 References

5.1 My github repository for the code and project report

Github

5.2 Youtube Link that explains using frequency of words in naïve bayes in email classification

https://www.youtube.com/watch?v=O2L2Uv9pdDA

5.3 Image reference for used for the Bayes formula

https://becominghuman.ai/naive-bayes-theorem-d8854a41ea08