The University of New Mexico
Computer Science
Ala Jararweh, Julie Hayes

Spring 2022
CS529: Introduction to Machine Learning

Document Classification using Naive Bayes and Logistic Regression

1 Introduction

Document classification algorithms aim to identify the theme or the topic of a text (or document) by transform raw text to numerical features. Moreover, document classification (AKA topic classification) is considered as one of the most crucial tasks in modern machine learning. Document classification can be involved in variety of tasks ranging from simple tasks such as sentiment analysis, spam filtering to more complex tasks such as fraud detection and recommender systems. Probably, the most difficult part of classifying text is Feature Extraction. That is, the process of converting text to vectors that capture the meaning of the text. Many approaches were introduced toward this goal such as Bag of Words(BOW), Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, and Transformer-based models (BERT).

Apparently, the accuracy of a model varies based on how the text is represented. In this project, however, we don't have to deal with this task. But, we did perform some pre-processing techniques to reduce the number of features (vocabulary) from 60,000 feature to 44,000 features. We also implemented Naive Bayes and Logistic Regression to perform document classification.

2 The infeasibility of Model 0

Question 1: *Why would it be difficult to accurately estimate the parameters of this model on a reasonable set of documents (e.g. 1000 documents, each 1000 words long, where each word comes from a 50,000 word vocabulary)?*

Estimating parameters for the Model 0 suggested is combinatorically and computationally exceedingly laborious to intractible. As we'd be trying to understand the probability distribution for each word at each position in the document, we would be counting 1 million words to calculate each of 1 thousand probability distributions. It is not hard to see that for even this comparatively small (by current modern standards) amount of data, few have enough patience, time, or compute power for such a model.

3 Implementation - how the code works

Dealing with text as an input to machine learning models can be somewhat cumbersome due to different reasons such as selecting the most valuable features, identifying the appropriate way to normalize the features, performing the appropriate classification algorithm. Similar to all machine learning tasks, topic classification can be partitioned into multiple stages starting from data preparation to model testing. In this section, we explore various design decisions were considered during each stage of the model.

3.1 Text Preprocessing

Document Preprocessing is an essential phase in many Natural Language Processing (NLP) tasks. Even though the data in this project were already preprocessed to words, we found a few useful ways to further process the data such as stop words. Table 1 shows the number of features after each pre-processing step. We found around 150 features correspond to stop words. Thus, we decided to remove those since they don't add any meaningful information to classification algorithms. Apart from stop word removal,

we performed *stemming*, which is the process of reducing words to their roots. For example, words like *connecting*, *connection* ,and *connected* will be reduced ***connect***. To keep their original counts, all stemmed words that can be reduced to the same root will be aggregated to one vector. In our previous example,

$$V(connect) = \frac{V(connecting) + V(connected) + V(connection) + ...}{Total\ number\ of\ stemmed\ words}$$

Table 1: Reported number of features before and after preprocessing

	w/o preprocessing	Stop Words removal	After Stemming
#OfFeatures	61190	61044	44907

3.2 Feature Normalization

Since we are using word frequencies as features in our models, the feature matrix is expected to be sparse. In other words, the range of values in each feature vector varies widely. Therefore, the distance between values should be normalized. Not only that, feature normalization is also used along with gradient descent to make the model converge faster especially when regularization is applied [1]. In this project, we use three different normalization techniques:

3.2.1 Re-scaling (min-max normalization)

This method is used to scale the range of a vector in [0,1]. The forumal for this method is given below. Where x is the original feature vector, $min(x)$ and $max(x)$ are the minimum and maximum values of that feature vector respectively.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

3.2.2 Standardization (Z-score Normalization)

This method normalize the features such that each feature has a mean of zero and a standard deviation of 1. The equation for this method is given below. Where x is the original feature vector, $\bar{x}$ is the mean of that feature vector, and σ is its standard deviation.

$$x' = \frac{x - \bar{x}}{\sigma}$$

3.2.3 Mean Normalization (L1 Norm)

Mean Normalization is used to scale each feature by the mean value of that feature. Below is the equation for mean normalization. Where x is the original feature vector.

$$x' = \frac{x}{\|x\|_1} = \frac{x}{\sum_{i=1}^{n} |x_i|}$$

3.3 Logistic Regression Classifier

We implemented Logistic Regression classifier as indicated. We used the following equation to train the weights in our classifier:

$$W^{t+1} = W^t + \eta((\delta(Y = y_j) - P(Y|W, X))X - \lambda W^t$$

Where W is the set of weights, η is the step size, $\delta(Y = y_j) = 1$ if the l^{th} training value, Y^l, is equal to y_j, and $\delta(Y = y_j) = 0$ otherwise. λ is the regularization term.

[1] https://en.wikipedia.org/wiki/Feature_scaling

3.4 Naive Bayes Classifier

As indicated, we implemented a Naive Bayes classifier with the following equations:

$P(Y_k) = \frac{\#\text{ docs labeled } Y_k}{\text{total } \#\text{ of docs}}$

$P(Y_k) = \frac{(\text{count of } X_i \text{ in } Y_k)+(\alpha-1)}{(\text{total words in } Y_k)+(\alpha-1)*(\text{length of vocab list})}$

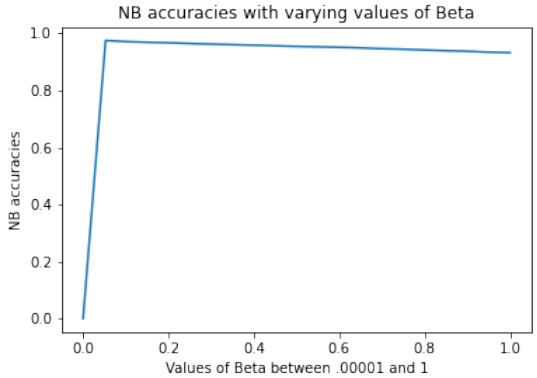
We then use, as specified, this equation to classify the data:

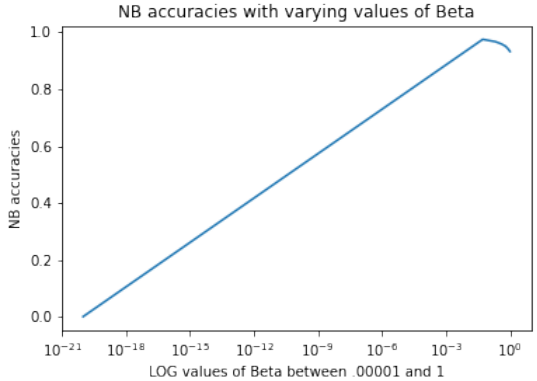
$$Y^{\text{new}} = argmax[\log_2(P(Y_k)) + \sum i(\#\text{of}X_i^{\text{new}}) \log_2(P(X_i|Y_k))]$$

4 Model 1 - Naive Bayes

Question 2: *Re-train your Naive Bayes classifier for values of β between .00001 and 1 and report the accuracy over the test set for each value of β. Create a plot with values of β on the x-axis and accuracy on the y-axis. Use a logarithmic scale for the x-axis. Explain in a few sentences why accuracy drops for both small and large values of β.*

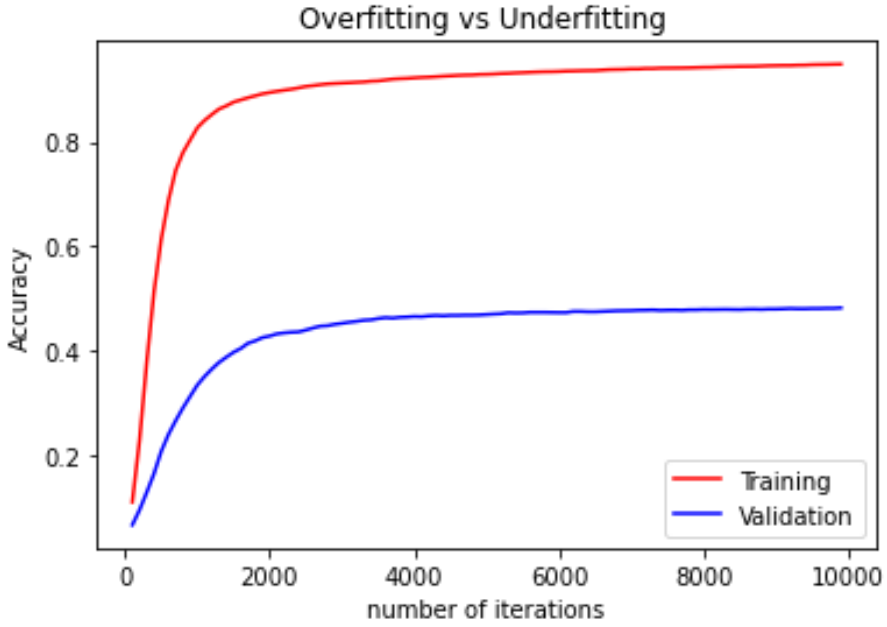
We had to get to very small values of β to show the decrease in accuracy at very small values, so we include below both the trend of accuracy in relation to the analyzed values of β as well as the values of β on a logarithmic scale, both shown below. We can see from the MAP equation that both very small values of beta will cause the numerator to overwhelm the denominator in comparison to the original equation, nearly eliminating the $(1)*(\text{length of vocab list})$ term. Conversely, large values of β will give that term too much influence. Either case will cause the model to not behave as intended and therefore reduce accuracy.





5 Model 2 - Logistic Regression

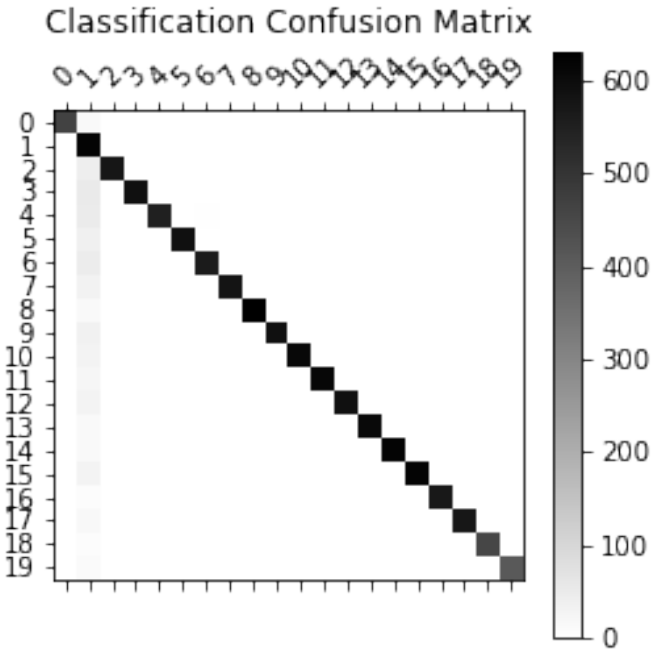
Question 3: *Re-train your Logistic Regression classifier for values of η starting from 0.01 to 0.001, λ = 0.01 to 0.001, and vary your stopping criterium from number of total iterations and report the accuracy over the test set for each value. Explain in a few sentences your observations with respect to accuracies and sweet spots.*

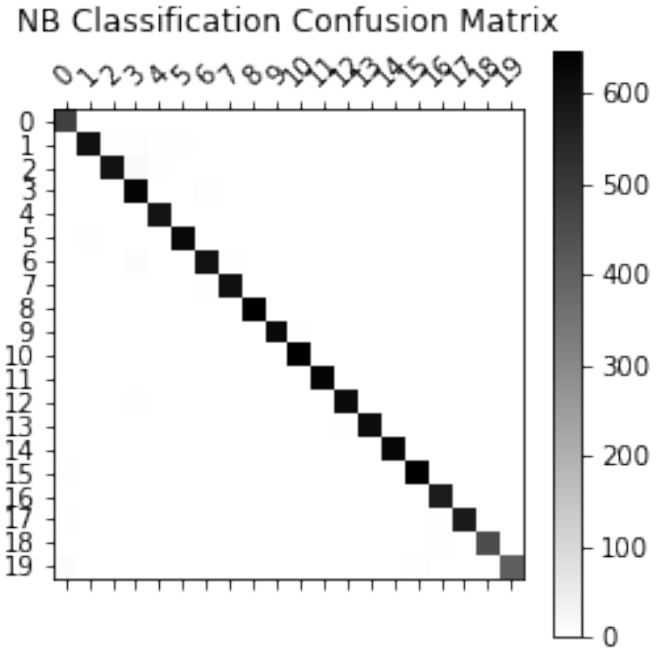


The plot of accuracies above was created with $\eta = .001$ and $\lambda = .001$. Given this accuracy trend, it is clear we are overfitting somehow on our training data. While this plot doesn't give any direct insight into the source of the overfitting, it's possible our λ term may be too small. It seems we didn't quite find the "sweet spot" relationship between η and λ.

6 Analysis of Results

Question 4: *In your answer sheet, report your overall testing accuracy (number of correctly classified documents in the test set over the total number of test documents), and print out the confusion matrix (the matrix C, where c_{ij} is the number of times a document with ground truth category j was classified as category i).*





Question 5: *Are there any newsgroups that the algorithm(s) confuse more often than others? Why do you think this is?*

From the Linear Regression confusion matrix (the one shown first above), it appears that category 2, the computer graphics category. This makes sense as it may include many words associated with games which may have been confused with words in the sports category. Similarly, it may have included words

that were similar to or confused with words in the other computing categories. Finally, there is also the possibility that this category included words generally describing visual displays that were general enough to be repeated in other categories. Interestingly, there doesn't seem to be any confusion in the confusion matrix for Naive Bayes classification.

7 Identifying Important Features

Question 6: *Propose a method for ranking the words in the dataset based on how much the classifier 'relies on' them when performing its classification (hint: information theory will help). Your metric should give high scores to those words that appear frequently in one or a few of the newsgroups but not in all of them. Words that are used frequently in general English (the, of, in, at, etc) should have low scores. Words that appear only extremely rarely in the data sets should also have low scores.*

We used Shannon Entropy (Information), defined as:

$$H(X) = -\sum_{i}^{n} P(x_i) \log P(x_i)$$

to make this classification. We think about Shannon Information as the amount of "surprise" that any bit of information would cause an observer. Yes, this is a very fuzzy definition, but thankfully it is defined mathematically. Words that appear frequently thus have low scores because they offer little 'surprise.' With this metric, words that are extremely rare also have low scores because the metric is a sum over all instances of a word - therefore low instances will result in a lower sum.

Question 7: *Implement your method, and print out the 100 words with the highest measure.*

suvm, syr, much, smear, needs, phobia, ccit, weren, senses, analysis, identify, quotation, distribution, shell, ad, fog, predominately, proper, directing, descriptions, glop, ork, excessive, designate, intriguing, blowing, wtc, mst, delving, degeneration, exons, impressive, ancestry, beginning, freeman, wearer, surveillance, bjectivist, kraken, itc, chemical, teaches, twist, moslem, abolishment, warn, tirades, swamping, loved, traditionally, deters, cards, jzn, attached, mud, undercover, synthetic, fill, florida, madman, brett, repeating, intrusions, instincts, toe, prego, circles, aztec, carols, tri, outrageous, planners, sadiq, sixth, stoning, cake, borrows, banker, dividend, subtly, resorts, equivalently, telepathy, physicist, hoc, affordable, careless, crumenan, sounder, misrepresented, kluge, ncsuvm, ncsu, gdr, hut, torkel, sics, kmldorf, bigbird, rosen