Document Classification using Naive Bayes and Logistic Regression

Sandeep Reddy Komatireddy (101950696)

Smita Sanjay Deore (101925234)

1 Introduction

Naive Bayes classification uses of Bayes theorem to determine how probable it is that an object belongs to a category. It is called 'naïve' because it uses conditional independence. It means that the algorithm treats the probability of a word appearing in a document as if it is independent of the probability of another word appearing. It doesn't give a concrete answer, but it tells the most likely classification class than others, to which set could belong.

Logistic regression is a process of modeling the probability of a distinct outcome given an input variable. The most common logistic regression model is a binary outcome where results can take up to two values yes/no, true/false, etc. Multinomial logistic regression is a model where results take more than two discrete values. Logistic regression is useful for the analysis of classification problems i.e. while determining if a new sample fits best into a particular category.

2 Problem Definition

The aim is to implement Naive Bayes and Logistic Regression for document classification and apply them to the classic 20 newsgroups dataset. In this dataset, each document is a posting that was made to one of 20 different Usenet newsgroups. The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. This collection has become a popular dataset for experiments in text applications of machine learning techniques, such as text classification and text clustering.[1]

[1] kaggle.com/competitions/cs529-project-2-nb/overview

3 Data Set

The data set contains the following four files:

- vocabulary.txt: It is a list of the words that may appear in documents. The line number is word's d in other files. That is, the first word ('archive') has word-Id 1, the second ('name') has word-Id 2, etc.

- newsgrouplabels.txt: It is a list of newsgroups from which a document may have come. Again, the line number corresponds to the label's id, which is used in the .label files. The first line ('alt.atheism') has id 1, etc.

- training.csv: Specifies the counts for each of the words used in each of the documents. Each line contains 61190 elements. The first element is the document id, the elements 2 to 61189 are word counts for a vectorized representation of words (refer to the vocabulary for a mapping). The last element is the class of the document (20 different classes). All word/document pairs that do not appear in the file have a count of 0.

- testing.csv: It is the same as training.csv except that it does not contain the last element.[2]

4 Naive Bayes Classification

Bayes theorem is given by**:**

$$P(Y_i \mid X_1, X_2, \ldots, X_n) \sim P(X_1, X_2, \ldots, X_n \mid Y_i) * P(Y_i)$$

where P(Y|X) is called the posterior probability
P(Y) is called the prior.
P(X|Y) is called the likelihood

A naive Bayes classifier is based on the Bayes theorem with an assumption of conditional independence between its predictors. Conditional Independence states that A is conditionally independent of B given C if the probability distribution of A is independent of the value of B given the value of C. Features that are Conditional Independence of each other given the class. Thus, we get below:

$$P(Y|X_i) = P(Y) \prod_{d} P(X_{id} \mid Y_j)$$

where d is the number of features

We will estimate P(Y) using MLE i.e., Maximum Likelihood Estimate.

$$P(Y_k) = \#\text{ of documents labeled } Y_k \,/\, \text{Total \# of documents}$$

[2] https://www.kaggle.com/competitions/cs529-project-2-lr/data

We are going to estimate the P(X|Y) using MAP i.e., Maximum a Posteriori Estimate.

$$P(X_i \mid Y_k) = \frac{\left(count\ of\ X_i\ in\ Y_k\right) + (\alpha - 1)}{\left(Total\ words\ in\ Y_k\right) + ((\alpha - 1)(length\ of\ vocaublary\ list))}$$

where

$$\alpha = \beta + 1$$

$$\beta = \frac{1}{|V|}$$

Description of the code:

The original dataset is imported using pd.read_csv() method and this data is split into training and testing for validation purposes. The size of training data is 80% and testing data is 20% of the original data.

We have used two matrices of the size k*d where k is the number of classes and d is the total number of words. These matrices are used to store the values of likelihoods of each word for each class. One of these matrices is further used to store the log_likelihood values which are used to determine the new class for a given document. The original data is filtered so that the index and class columns are removed for training data.

The prior probabilities of each class are calculated and stored in a list. The values for the beta are passed as a list to test the accuracy of the classifier for different values. For actual testing of original testing data, the basic beta value is taken which is 1/ |V|.

We define a function to calculate the log_likelihood values for the above-defined matrix and it returns the values in the form of a matrix. This function takes training data and the beta value as parameters and the model is trained based on this training data.

For validation purposes, we have defined a function to test this model on the testing data which is a part of the original data. This function takes testing data and beta as parameters and returns the accuracy of the model on that particular testing data.

As instructed, this testing is carried out using different values of beta and the accuracies are stored in a list. A graph is plotted with beta values on the X-axis and the corresponding accuracy on the Y-axis, which shows the trend of accuracy w.r.t beta.

Then, there is a function for testing the actual testing data which takes the testing dataset and beta values as parameters. This function returns the row number and predicted class for that row in the form of a list. This is appended to the main list that stores the predicted classes for all rows. The original testing data is imported and the testing function is called. The final result is converted to a CSV file with a header (id, class) and it stores the class for each row of the testing data. This file is used for submissions on Kaggle.

To rank the words based on how much the classifier depends on them, we have defined another function that gives the indices of the words with the highest ranks. Then, the top 100 words with the highest ranks are printed in the form of a list.

[3] http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

5 Logistic Regression Classification

Logistic Regression assumes a parametric form for the distribution P(Y|X), then directly estimates its parameters from the training data.[3] Logistic regression where binary classification is given as :

$$P(Y=1|X) = \frac{1}{1+exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

$$P(Y=0|X) = \frac{exp(w_0 + \sum_{i=1}^{n} W_i X_i)}{1+exp(w_0 + \sum_{i=1}^{n} W_i X_i)}$$

Logistic regression for the conditional likelihood Estimate is given as :

$$\ln\left(P\left(D_y \mid D_x, w\right)\right) = \sum_{j=1}^{N} \ln P(y^i \mid x^i w)$$

$$= \sum_j y^i \left(w_0 + \sum_i^n w_i x^i{}_i\right) - \ln\left(1 + exp\left(w_0 + \sum_i^n w_i x^i{}_i\right)\right)$$

Logistic regression is a regression model and it uses a sigmoid function to model the data. The sigmoid function is:

$$g(z) = \frac{1}{1+e^{-z}}$$

The classifier first multiplies each feature x_i by its weight w_i, sums the weighted features, and at last adds the bias term b. The resultant weighted sum is:

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

The equation of the loss function is:

$$J(\theta) = -\sum_{i=1}^{n}\sum_{k=1}^{K} 1\{y^i = k\} \, logP(y^i = k|x^i;\theta)$$

The learning rate ɳ determines the magnitude of the amount to move in gradient descent. The change we make is the learning rate times the gradient. It is:

$$w^{t+1} = w^t - \eta \frac{d}{dw} f(x;w)$$

[3] http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

Description of the code:

The original dataset is imported using pd.read_csv() method and this data is split into training and testing for validation purposes. The size of training data is 80% and testing data is 20% of the original data.

We have two lists to store the different values of learning rates and penalty rates. The values for each of these variables range from 0.001 to 0.1. The number of times weights are updated is 100.

We have defined a function called train_weights() to update the weights and this function takes training_data, learning rate, and penalty rate as parameters and returns the updated weights.

For validation purposes, we have tested this model on the testing data using a function validation(). The training accuracy can be found using this method.

For the testing part, we defined a function testing() that takes testing_data, learning rate, and penalty rate as parameters. It gives a list as output which contains the id and class of a particular row in each row. Then, the actual testing data is imported and is passed to the testing function. The output of this function is then converted to a CSV file to upload to Kaggle.

To observe the accuracies for different values of learning and penalty rates, the validation function is called multiple times with different parameters and the accuracy of each iteration is stored in a list. Using three lists, a graph is plotted to figure out the trend of accuracy on different learning and penalty rates.

Then, we have defined a function to print the confusion matrix on the validation data that takes validation data, learning rate, and penalty rate as the parameters. This function prints the confusion matrix for the passed learning and penalty rates.

Accuracies under various settings:

For Naïve Bayes classifier:

For beta =1 / |v| = 0.00001634, the accuracy obtained for the original testing data is 0.87127.
For beta = 0.01, the accuracy obtained for the original testing data is 0.88810.

The model has been tested for all other values on validation data and the results are recorded in the answer to Question 2.

For Logistic Regression classifier:

For learning rate=0.01 and penalty rate=0.005, the accuracy obtained on the original testing data is 0.85267.

This model has been also tested on other combinations of learning rate and penalty rate values on the validation data and the results are shown in the answer to Question 3.

[3] http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

Conclusion:

For the Naïve Bayes classifier, the accuracy and beta values are directly related to some extent. As the beta value increases, the accuracy recorded has also increased. For extreme values of beta (high and low values), the accuracy is low when compared to other values. The accuracy obtained by Naïve Bayes is higher than that of Logistic Regression. For this data, the Naïve Bayes classifier seems to be more efficient than the Logistic Regression classifier.

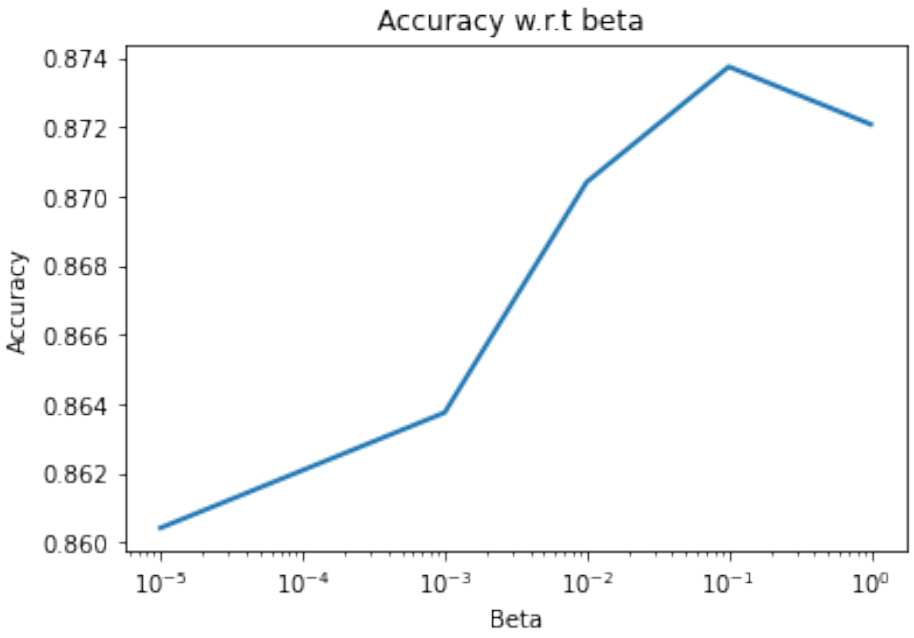
[3] http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

6 Answers to given questions:

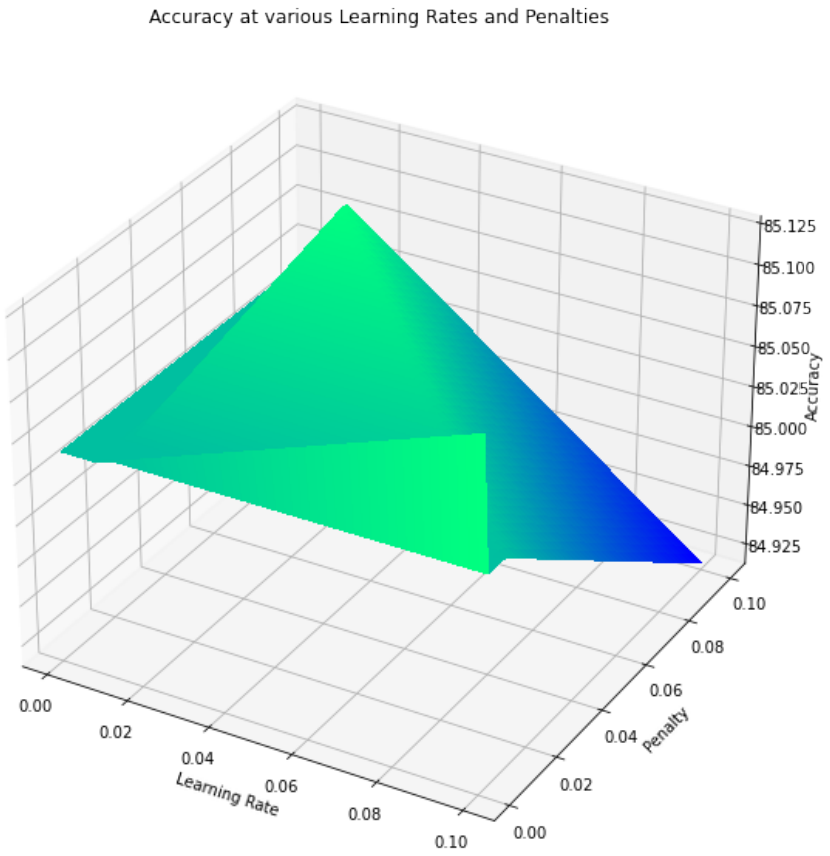
1. It is given that we have to estimate 50,000-dimensional parameters using 1000 samples which is too less. It is difficult to accurately estimate the parameters of this model as the number of samples, in this case, is very small. In this model, we have classified based on word count in each newsgroup, which is a weak feature to classify a document as similar words can be in other newsgroups too with the same count.

2. The Naïve Bayes classifier is re-trained using different values of beta ranging between 0.00001 and 1. The accuracies are plotted w.r.t beta values and the graph is below.

Beta value	**Accuracy**
0.00001	0.8604166666666667
0.0001	0.8620833333333333
0.001	0.86375
0.01	0.8704166666666666
0.1	0.87375
1	0.8720833333333333



3. The Logistic Regression classifier is re-trained under different values of learning rate(eta) and penalty rate (Lamba) ranging between 0.0001 and 0.1. A graph is plotted with accuracy and these two parameters.



4. The overall testing accuracy of the Naïve Bayes classifier is 0.86041667.
The confusion matrix is shown below.

```
[[ 98   0   0   0   0   0   0   0   0   0   0   0   0   1   0   4   0   1   0   5]
 [  0 113   3   4   4   6   0   0   0   0   0   2   3   2   1   0   0   0   0   0]
 [  1   8  92  14   1   8   3   0   1   0   0   1   2   0   2   0   0   0   0   0]
 [  0   5   8  94  10   1   4   0   0   0   0   3   7   0   0   0   1   0   0   0]
 [  0   3   1   5  83   1   1   1   0   0   0   3   7   2   2   0   0   0   1   0]
 [  0   9   3   1   1 100   1   1   0   0   0   0   0   0   1   0   0   0   0   0]
 [  0   2   1   6   1   0  84   8   0   0   0   1   5   2   0   0   0   0   1   0]
 [  0   1   0   0   1   1   2  97   3   0   0   1   1   1   1   0   0   0   0   0]
 [  0   0   0   0   0   0   0   2 146   0   0   0   2   0   0   0   1   0   0   0]
 [  0   0   0   0   0   0   0   0   1 138   2   0   0   0   2   0   1   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 128   1   0   0   0   0   0   0   0   0]
 [  0   0   1   0   1   2   0   0   0   0   0 121   0   0   1   0   1   2   1   0]
 [  0   3   2   5   4   0   3   1   1   0   0   4  86   3   7   0   1   0   0   0]
 [  0   4   0   2   0   0   0   0   2   0   0   1   2 109   1   0   1   0   0   0]
 [  0   1   0   1   1   0   1   1   1   0   0   0   1   0 110   0   0   1   0   0]
 [  1   1   0   1   0   0   0   0   0   0   0   0   0   0   0 117   0   1   0   3]
 [  1   0   0   0   0   0   0   1   0   0   0   1   0   0   0   0 101   0   4   1]
 [  1   0   0   0   0   0   0   0   1   0   0   3   0   0   1   0   0 117   1   0]
 [  0   0   0   0   0   0   0   0   0   0   0   1   0   0   2   0   9   0  72   0]
 [  9   0   0   0   0   1   0   0   0   0   0   1   0   1   0   7   3   1   3  59]]
```

The overall testing accuracy of the Logistic Classifier is 0.85041667.
The confusion matrix is shown below.

```
[[ 86   0   0   0   0   0   0   0   0   0   0   0   0   0   0  13   0   4   1   5]
 [  1 120   4   5   0   3   0   0   0   0   0   3   0   0   2   0   0   0   0   0]
 [  1  10  96  12   3   8   1   0   0   0   1   0   0   0   0   1   0   0   0   0]
 [  0   4   4 108   5   0   2   0   0   0   0   1   3   0   0   4   0   2   0   0]
 [  0   1   0   3  92   0   0   1   0   0   0   4   0   0   1   4   0   4   0   0]
 [  0   7   2   1   1 101   0   0   0   0   0   2   0   0   0   2   0   1   0   0]
 [  0   1   2   9   3   0  75   7   0   0   1   1   3   2   0   5   0   2   0   0]
 [  0   0   0   0   1   1   1  96   2   0   0   3   0   0   0   1   0   4   0   0]
 [  0   0   0   0   0   0   2   3 144   0   0   0   0   0   0   1   0   1   0   0]
 [  0   0   0   0   0   0   0   0   0 139   1   0   1   0   0   2   0   0   1   0]
 [  0   0   0   0   0   0   0   0   0   0 128   1   0   0   0   0   0   0   0   0]
 [  0   2   0   0   0   1   1   0   0   0   0 123   0   0   0   0   0   3   0   0]
 [  1   5   1   8   5   0   0   0   0   0   2   9  83   1   1   1   0   3   0   0]
 [  0   2   0   1   0   1   0   0   0   0   0   2   2 108   0   4   1   1   0   0]
 [  0   3   0   0   0   0   0   0   0   0   0   0   0   0 108   3   0   4   0   0]
 [  0   1   0   1   0   0   0   0   0   0   0   0   0   0   0 120   0   1   0   1]
 [  0   0   0   0   0   0   0   0   0   0   0   1   0   0   1   2 103   2   0   0]
 [  0   0   0   0   0   0   0   0   0   0   1   0   0   0   1   1   0 120   1   0]
 [  0   0   0   0   0   0   0   0   0   1   0   1   0   1   1   3   8   6  62   1]
 [  8   0   0   0   0   1   1   0   0   0   0   1   0   0   0  35   3   4   3  29]]
```

5. It is observed that sub-topics in the group “comp” which belongs to labels 2 to 6 get highly confused with each other and topics with “comp” and “sci.electronics (13)” gets confused with each other. The main reason for this is that all of these subtopics have a large number of similar words. Some of the other groups like talk.religion.misc(20), alt.atheism(1), and soc.religion.christian(16) confuse each other as most of the vocabulary in these groups is the same and all of these groups belong to the same kind of speech.

6. We have defined a function rank() to find the rank of the words. It calculates the rank of each word based on the entropy values of its probability. It gives high scores to those words that appear frequently in a few documents and it also neglects the common words that are used frequently in English. If the value returned is high, the classification strongly relies on that word. If the value returned is less, then the classification is less dependent on that word.

7. The implementation of the above method results in the top 100 words that are shown below.

```
['vijay', 'iheirent', 'sinnokf', 'sheoak', 'ucnv', 'redgum', 'pethybridge', 'ped
antic', 'psd', 'sourcing', 'tranmit', 'austrailian', 'mmunic', 'ation', 'jaehyun
g', 'steenking', 'corpus', 'christi', 'engga', 'grandfathered', 'regulatory', 'c
ondensing', 'aerodynamic', 'droplets', 'condensate', 'hotwell', 'clocking', 'mid
get', 'stich', 'electricans', 'vcb', 'cob', 'mpaul', 'acq', 'hct', 'oddball', 'd
atasheets', 'sprinkling', 'ballast', 'misrepresents', 'bimetalic', 'mustafa', 'd
ripping', 'smu', 'zsc', 'chuckling', 'uww', 'frankh', 'wpa', 'hpcuhe', 'lowers',
'cussed', 'toggled', 'regrettable', 'stain', 'adrift', 'pspice', 'clamping', 'im
pure', 'vender', 'eleceng', 'routh', 'krouth', 'ibmmail', 'profs', 'usfmctmf', '
eld', 'facsimile', 'mora', 'trygon', 'smd', 'elevators', 'tachistoscope', 'flour
escent', 'sealing', 'barrels', 'giggling', 'elevator', 'susannah', 'witchcraft',
'dehydrators', 'dehydrator', 'coastline', 'pinpoint', 'cathode', 'pulsed', 'glea
ming', 'icbo', 'ingenious', 'mdgoodma', 'sgs', 'dnewman', 'farmland', 'infertile
', 'opamps', 'relativly', 'equipments', 'itr', 'heatsinks', 'arrevola']
```