COL774 Assignment 2

Siddhant Mago, 2017CS50419

February 2020

1 Text Classification

In this part, I implemented the Naive Bayes algorithm using the bag of words approach.

1.a Implementation

For this part, I did a basic processing of words by splitting them on comma(,), full stop(.) and spaces. I used Laplace smoothing with C = 1 and logarithms to avoid underflow.
I obtained an accuracy of 81.05% on **test data** and an accuracy of 84.93% on **training data**.

1.b Accuracy on random/majority

The accuracies obtained were:

Random prediction: ≈ 50% (average over 5 runs)
Majority prediction: 50.69 %

The accuracy of random/majority baseline is 50% and Naive Bayes text classification gives a huge improvement (≈ 30 %) over that.

1.c Confusion Matrix

The confusion matrix for the my implementation was:

$$\begin{bmatrix} 143 & 34 \\ 34 & 148 \end{bmatrix}$$

(columns are actual class, rows are predicted class)

Both the non diagonal entries (FP = FN) are the same, this means that the classifier is not biased towards any particular class.

1.d Stemming

For this part, I used the tweet tokenizer of nltk library to tokenize the tweets(tweet tokenizer also removes usernames). I also removed stop words, converted the text to lower case and performed stemming (using PorterStemmer of nltk library). However, there was only a marginal increase in the accuracy obtained on the test set. This may be because sometimes stemming can degrade accuracy.

Accuracy on test: 81.6 %

1.e Feature Engineering:

In this part, I used two features; bigrams and length of the tweets.

- After applying bigrams on top of the model, I got an accuracy of **84.67 %** on the test data. While, forming bigrams, I also removed any special characters and stop words from the text. This shows that bigrams gives us a better idea about the context of a text.

- I observed an accuracy of **81.05 %** after adding length of tweets as features, which is slight reduction from the accuracy obtained part d), and is the same as the accuracy observed in part a). This shows that length of tweets may not be a good feature for classification as both positive and negative tweets can be of varying lengths.

So, bigrams help us to improve the overall accuracy of our model, but length of the tweets do not.

1.f TF-IDF:

The feature vector for training was very large and couldn't fit into the memory all at once in dense form. So I used the partial_fit function of model, to train batch wise. I took the **batch size = 400** to train the model. I also used stemmed words after stop words removal which also helped in reducing the number of features.

Percentile	Training time (in s)	Accuracy (test)
100	5500s	49.02 %
1	24.27s	72.98 %
0.1	2.7s	73.53 %
0.01	1.8s	59.8 %

Figure 1: Comparison

Some comments:
- The gaussian NB here gives a very low accuracy (less than our implementation) because for all (100 percentile) the features the gaussian model overfits and thus gives poor training accuracy.
- It is evident that taking a smaller number of features (1 and 0.01) leads to faster training times. Moreover, it also increases the accuracy, this is because now we are only considering those features which give us the maximum information about the document (thus reducing noise) and thus preventing over-fitting.
- On 0.01 %, the accuracy reduces as we are now even ignoring some important features and thus under-fitting the data.

1.g Receiver Operating Characteristic (ROC)

This is the ROC curve of the naive Bayes model with bi-grams as additional features.

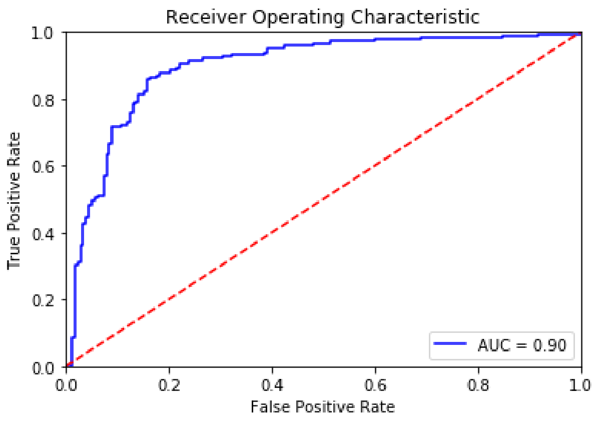

Figure 2: ROC curve:

The area under the ROC curve is 0.9, which means that the classifier is a good one. The area under the ROC curve is a good measure of classifier's strength as AUC is scale invariant, i.e., it measure how well the prediction is ranked and not their absolute value. It is also classification threshold invariant, i.e., it measures the quality of the model's predictions irrespective of what classification threshold is chosen. So having a high area under ROC, suggests that a classifer is a good one. A perfect classifier's ROC has an area of 1.

2 Support Vector Machines:

2.i Binary Classification:

In this part I worked on binary classification between ankle boots and tshirts (0 and 9) :

2.i.a Linear Kernel:

For linear kernel. The training time was 24.8s and the accuracies observed were:

Test Accuracy: 100 %
Validation Accuracy: 99.8 %

The number of support vectors obtained were 58. α_i was counted as a support vector if $\alpha_i > 10^{-6}$.

2.i.b Gaussian Kernel:

For the gaussian kernel, only b (intercept) could be calculated at train time but; w_i had to be calculated at test time. So the training time was 50.8s and the accuracies observed were:

Test Accuracy: 100 %
Validation Accuracy: 100 %

The number of support vectors obtained were 878. α_i was considered as a support vector if $\alpha_i > 10^{-6}$.
Though, a gaussian kernel is a more general kernel as it has more parameters, the accuracies obtained in gaussian is only slightly better than linear, this may be because ankle boots and tshirts are very easy to differentiate and thus both the classifiers work well.

2.i.c Sklearn:

Kernel	Linear Kernel		Gaussian Kernel	
	Sklearn	My implementation	Sklearn	My implementation
Training time (s)	0.315s	24.8s	3.42s	50.8s
Accuracy (test)	100 %	100 %	99.8 %	100 %
Accuracy (validation)	99.8 %	99.8 %	100 %	100 %
# of support vectors	57	58	826	878

Figure 3: Comparison

It is evident from the data that the accuracies are almost 100% for all implementations. However, the computational cost (training cost) of sklearn implementations is very less as compared to my implementation. The value of w and b are nearly the same for linear kernel between my implementation and sklearn's implementation.
In gaussian kernel, the $\alpha_{i's}$ obtained are nearly the same for sklearn and my implemenation, however, the value of b is a little different. Possible reasons for this are:

- We have used hard-margin formula for calculating b*. This leads to approximate results as have not included a term to handle the error ϵ_i which leads to a reduction in accuracy
- Also, we have used a general convex optimiser and sklearn uses SVM specific algorithms (and/or convergence criteria), which may lead to slightly different results.

The number of support vectors, depend upon the threshold which might be different across implementations.

2.ii Multiclass classification:

2.ii.a My Implementation:

I implemented the one to one multiclass classifier using the gaussian kernel. It took 2238s to train. In case of ties the class with the higher score was taken. The accuracies obtained were:

Test accuracy: 85.07 %
Validation accuracy: 84.91 %

2.ii.b Sklearn:

I used the one vs one multiclass classifier of sklearn. The accuracies obtained were:

Test accuracy: 88.07 %
Validation accuracy: 87.91 %

Here, we observe that sklearn gives higher accuracy and not only that the training time for sklearn is **185s** compared to **2238s** in my implementation. This is because of parallelization and several other code optimisation in sklearn's implementation.

2.ii.c Confusion matrix:

The confusion matrices for **test data** observed were:

(rows are actual category, columns are predictions)
Sklearn:

$$\begin{bmatrix}
432 & 0 & 5 & 11 & 3 & 0 & 38 & 0 & 10 & 0 \\
1 & 482 & 4 & 9 & 0 & 0 & 4 & 0 & 0 & 0 \\
5 & 0 & 411 & 7 & 37 & 0 & 32 & 0 & 8 & 0 \\
12 & 0 & 3 & 457 & 9 & 0 & 14 & 0 & 5 & 0 \\
3 & 1 & 41 & 13 & 399 & 0 & 38 & 0 & 5 & 0 \\
0 & 0 & 0 & 0 & 0 & 473 & 0 & 16 & 5 & 6 \\
80 & 0 & 55 & 9 & 34 & 0 & 315 & 0 & 7 & 0 \\
0 & 0 & 0 & 0 & 0 & 14 & 0 & 471 & 1 & 14 \\
1 & 0 & 1 & 1 & 2 & 2 & 2 & 2 & 489 & 0 \\
0 & 0 & 0 & 0 & 0 & 11 & 0 & 14 & 1 & 474
\end{bmatrix}$$

My Implementation:

$$\begin{bmatrix}
403 & 0 & 7 & 7 & 0 & 0 & 65 & 0 & 17 & 0 \\
0 & 484 & 6 & 2 & 0 & 0 & 6 & 0 & 2 & 0 \\
0 & 0 & 414 & 3 & 26 & 0 & 44 & 0 & 13 & 0 \\
17 & 10 & 2 & 410 & 6 & 0 & 39 & 0 & 16 & 0 \\
0 & 1 & 55 & 12 & 366 & 0 & 51 & 0 & 15 & 0 \\
1 & 0 & 0 & 0 & 0 & 432 & 0 & 7 & 48 & 12 \\
52 & 1 & 52 & 5 & 20 & 0 & 351 & 0 & 19 & 0 \\
0 & 0 & 0 & 0 & 0 & 48 & 0 & 411 & 6 & 35 \\
1 & 0 & 1 & 0 & 0 & 0 & 3 & 0 & 495 & 0 \\
0 & 0 & 0 & 0 & 0 & 5 & 0 & 6 & 2 & 487
\end{bmatrix}$$

The confusion matrices for **Validation data** observed were:

Sklearn:

$$\begin{bmatrix}
212 & 0 & 1 & 8 & 0 & 0 & 26 & 0 & 3 & 0 \\
0 & 237 & 3 & 7 & 0 & 0 & 2 & 0 & 1 & 0 \\
5 & 0 & 205 & 3 & 18 & 0 & 13 & 0 & 5 & 0 \\
6 & 0 & 0 & 228 & 6 & 0 & 9 & 0 & 1 & 0 \\
1 & 1 & 24 & 8 & 200 & 0 & 15 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 241 & 0 & 2 & 1 & 5 \\
34 & 0 & 28 & 3 & 19 & 0 & 165 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 8 & 0 & 230 & 1 & 11 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 2 & 244 & 0 \\
0 & 0 & 0 & 0 & 0 & 6 & 0 & 8 & 1 & 235
\end{bmatrix}$$

My Implementation:

$$\begin{bmatrix} 201 & 2 & 1 & 4 & 0 & 0 & 38 & 0 & 4 & 0 \\ 0 & 240 & 2 & 2 & 0 & 0 & 3 & 0 & 3 & 0 \\ 2 & 0 & 207 & 1 & 13 & 0 & 14 & 0 & 12 & 0 \\ 11 & 7 & 0 & 197 & 6 & 0 & 20 & 0 & 9 & 0 \\ 0 & 2 & 30 & 5 & 185 & 0 & 19 & 0 & 9 & 0 \\ 0 & 0 & 0 & 1 & 0 & 225 & 0 & 1 & 17 & 6 \\ 19 & 0 & 27 & 1 & 11 & 0 & 184 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 24 & 0 & 199 & 6 & 21 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 & 2 & 245 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 4 & 5 & 239 \end{bmatrix}$$

Frome the above confusuion matrices, we observe that the classes t-shirts and shirts (0 and 6) are often mis-classified as each other, which makes sense as they're very similar in appearance.

2.ii.d 5-Fold Validation

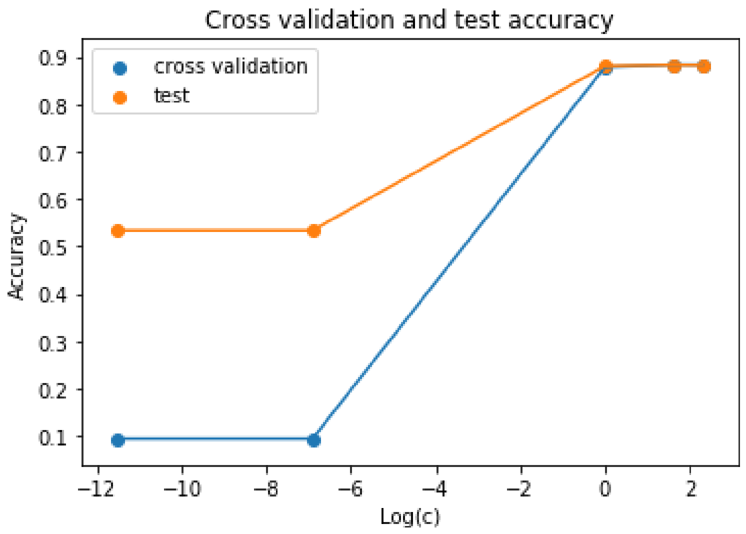

Figure 4: Cross validation accuracy

The accuracies obtained were:

C	Test accuracy	Cross Validation accuracy
10^{-5}	53.39 %	9.46 %
10^{-3}	53.39 %	9.46 %
1	88.08 %	87.83 %
5	88.3 %	88.33 %
10	88.24 %	88.31 %

Figure 5: Comparison

The value of C = 5 gives the best cross-validation accuracy. At this value, the test accuracy is also the highest. So value of C = 5, is the best among the given 5 values. A smaller value of C means that the classifier increases the margin without caring about mis-classification of data points, thus giving a very low validation accuracy. A larger value of C means a smaller margin is taken in order to prevent mis-classification, so it gives higher accuracy. So, the optimal value of C depends on the scale of the data, how far or how close are the data points from the seprator.