COL774 - Machine Learning: Assignment - 2

Param Khakhar - 2018CS10362

22 November 2020

Introduction

The following is a writeup for the Assignment-2, Machine Learning (COL774). There are sections corresponding to each of the four questions and within each section, there are sub-sections for each sub-part.

Task -1 : Naive Bayes Classifier

Task - 1.a: Implementing Naive Bayes

The Multinomial Event Model is used for implementing the Naive Bayes Classifier. The pre-processing used is tokenizing the text followed by removal of the punctuations. The accuracy achieved on the **test** set is **60**%, whereas that achieved on the **train** set is **63.12**%.

Task - 1.b: Baseline Models

The following baseline accuracies were achieved:

- **Random Prediction:** The obtained accuracy is **19.9%**, which is consistent theoretically as the probability of guessing right among 5 choices is 0.2 .

- **Majority Class:** The most awarded rating is 5-Stars. So, predicting 5-Stars for all the instances in the test data would result in an accuracy of **43.9%**.

Thus, **Multinomial Naive Bayes** offers an improvement of **0.4 %** as compared to the **Random Prediction** and an upgrade of **16.1%** over the **Majority Class** prediction model.

Task - 1.c: Confusion Matrix

The confusion matrix obtained for the test dataset is as follows:

$$\begin{bmatrix} 12983 & 2160 & 890 & 560 & 1317 \\ 3841 & 2657 & 1180 & 338 & 139 \\ 1280 & 2966 & 3652 & 1184 & 248 \\ 788 & 1875 & 6619 & 16352 & 10555 \\ 1277 & 1180 & 2190 & 10924 & 46563 \end{bmatrix}$$

The highest diagonal value in the above matrix is for 5 Stars. This implies that the model is confidently able to predict the reviews which have a rating of 5-Stars. On the contrary, the model doesn't have much confidence while predicting the reviews with 2 Stars. It confuses the reviews rated at 2 Stars with those rated at 1 Star. This can be observed for the reviews having 4 stars as well as significant number of reviews having rating 4 stars, have been predicted to have 5 Stars.

Task - 1.d: Stemming and Removal of Stop-Words:

Unusually, the accuracy didn't change much on using the provided code for Stemming in which the stop words are also removed. On using both **Porter-Stemmer** and **SnowballStemmer**, the accuracies obtained on the **test** set is **59.54%**. However, the size of vocabulary did change from 186850 to 141448. This might be due to the fact that, the words in the vocabulary are generic and there aren't any words for particular class labels. For example, the word *good* can be used in a rating of 4 stars and 5 stars as well. Likewise, we may have certain words for the reviews with less stars as well. Stemming converts the words in their root form, so that we may have a shorter vocabulary, but root forms of the words may cause result in increased confusion. For example, *finest* occurring in a 5 star review, could get converted to *fine* occurring in a 3 star review.

Task 1.e: Feature Engineering

Using **Bigrams** did improve the test accuracy of the model. However, the vocabulary size turned out to be very large, i.e. beyond the feasible capacity of my system, thus I tried working on a smaller dataset, where the training set comprised of random selection of the 20000 instances from each class, and the test comprised of 10000 instances sampled from the original test set. The accuracy of the model **without bigram** turned out to be **53.01%** whereas, using **bigrams** for the vocabulary resulted in an accuracy of **55.74%**.

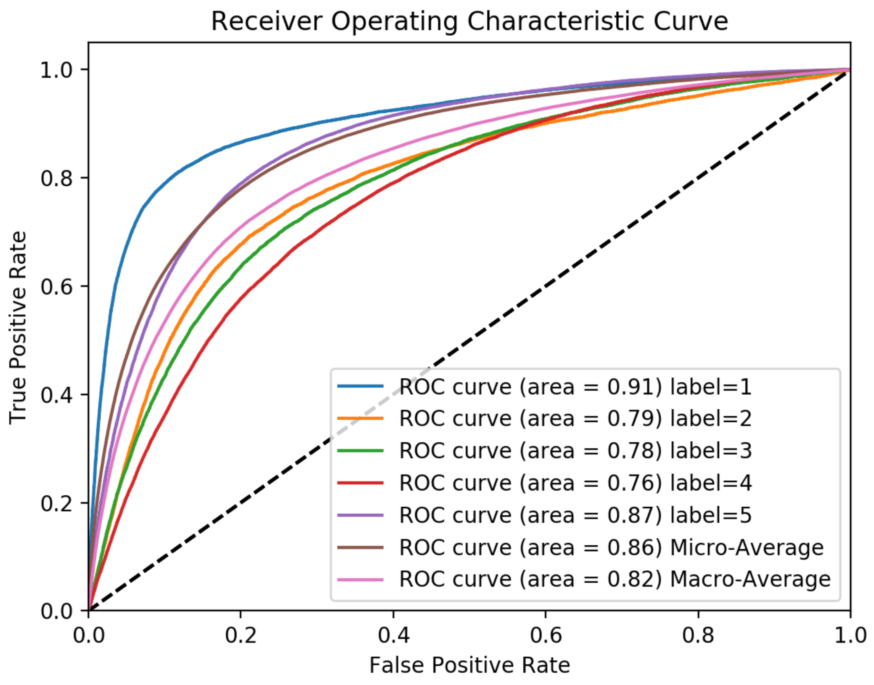
Another possible tweak in the vocabulary could be to select the words less than certain number of occurrences. The following results were obtained due to this on the test set:

Limiting occurrences	**Accuracy**
5	0.565
8	0.611
10	0.615
15	0.608

As evident, there are minute improvements which can be leveraged for future predictions, if the length of the reviews remain more or less the same.

Task 1.f: ROC Curve

The following ROC curves are observed when the input data is tokenized, followed by stopwords removal and stemming,



This is consistent with the confusion matrix as, the AUC values for labels 2 and 4 are low, and those for labels 1 and 5 are high.

Task - 2: Support Vector Machines

In this task, SVM was implemented using the library CVXOPT. Following subtasks were carried out.

Task 2.a: Binary Classification

The last digit of my entry number is 2. So, I classified between the labels 2 and 3 of the MNIST Fashion Dataset. Following statistics were obtained:

- **Linear Kernel:** The accuracies obtained on the **validation** and **test** set are **89.6%** and **91.3%**
- **Gaussian Kernel:** The accuracies obtained on the **validation** and **test** set are **98.2%** and **97.2%**

Task 2.b: Multi-Class Classification

The accuracies for my implementation ($C = 1$, $\gamma = 0.05$) on **Validation Set** is **87.95%**, whereas the **Test Set** is **87.89%**.
The accuracies for *scikit-learn.SVM.SVC* ($C = 1$, $\gamma = 0.05$) on **Validation Set** is **87.91%**, whereas the **Test Set** is **88.01%**.

Thus, the accuracies are approximately, the same for my implementation and sklearn's. However, there are differences in the training time, and inference time. The following table represents the time in seconds, for both the implementations:

Task	**My implementation**	**Scikit Learn**
Training	1407.61s	252.85s
Validation Prediction	34.78s	30.93s
Test Prediction	67.54s	61.15s

The scikit learn implementations, training time is much less as compared to my implementation. A particular reason for this could be the quadratic solver used by scikit learn's implementation. Currently, my implementation is using CVXOPT for solving the constraint optimization. I have tried to use vectorized operations, as far as possible instead of iterations. However, sklearn implementation might be using a faster operations. The auxiliary libraries used by me are Pandas, and Numpy, which might affect the speed as well.

The confusion matrix for the classification of labels - 2 vs labels - 3 is as follows:

$$\begin{bmatrix} 492 & 8 \\ 79 & 421 \end{bmatrix}$$

The performs decently while making predictions. The diagonal values are higher denoting higher true positives and true negatives and eventually a better performance.

The confusion matrix for the multi-class classification task is as follows:

$$\begin{bmatrix}
424 & 0 & 4 & 12 & 2 & 0 & 73 & 0 & 0 & 0 \\
0 & 483 & 0 & 2 & 2 & 0 & 0 & 1 & 0 & 0 \\
5 & 4 & 407 & 3 & 45 & 0 & 57 & 1 & 2 & 0 \\
11 & 8 & 7 & 453 & 13 & 0 & 9 & 0 & 1 & 0 \\
3 & 0 & 37 & 8 & 395 & 0 & 30 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 473 & 0 & 12 & 2 & 10 \\
46 & 5 & 37 & 17 & 38 & 0 & 324 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 0 & 16 & 0 & 471 & 2 & 14 \\
10 & 0 & 8 & 5 & 5 & 5 & 7 & 1 & 489 & 1 \\
0 & 0 & 0 & 0 & 0 & 6 & 0 & 14 & 0 & 475
\end{bmatrix}$$

The SVM is able to perform decently, as is evident from the values of the diagonal elements.

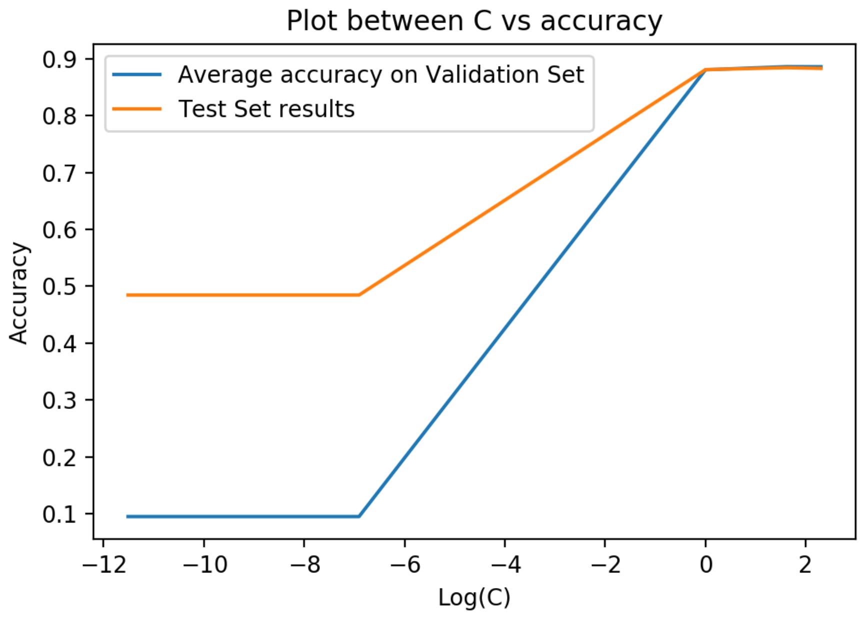
On varying the hyperparameter C, the following accuracies were obtained on performing a 5 - Fold Cross Validation, on the concatenated Train and Validation set.

C	**Time in s**	**Average accuracy for all the Folds**
1e-05	4601.25s	0.09489
0.001	5114.70s	0.09489
1	1472.15s	0.8808
5	1712.94s	0.8862
10	1708.95s	0.8861

The value of C = 5, gives the best results, however, there aren't much differences between the results of C = 5 and C = 10. Following are the test set accuracies obtained for different values of C:

C	**Time in s**	**Accuracy**
1e-05	1389.26s	0.484
0.001	1444.84s	0.484
1	465.56s	0.881
5	416.44s	0.884
10	415.69s	0.883

A trend similar to the average accuracies on the validation set is observed for the accuracies on test set, for the different values of C. The graph representing the different values would be:



The optimal results are obtained for C = 5, for both test and validation sets. Thus, using validation sets for tuning hyper-parameters of the model in general can result in the better generalization of our model to the unseen data.

Task - 3: SVMs vs Multinomial Naive Bayes

The following table represents the statistics regarding the time and accuracies obtained on training the respective models. The **train-validation set ratio is 3:1.**

- **Multinomial Naive Bayes:** There aren't any hyper-parameters to be optimized for this learning algorithm. The model is thus trained on the entire training set, and tested on the entire test set. The training time for the model was **0.932s**, and the accuracy on the test set was **0.589**.

- **SVM (Liblinear):** The hyperparameter tuned for this model is **C** and the loss used which could be **hinge** or **hinge squared**, which is **defaulted to 1**, but for our task, the values of **C** selected are $\{0.001, 0.01, 0.1, 1.0, 10\}$. The following table represents the respective training times for different value of **C** where the loss is **hinge squared** and the corresponding accuracies on the validation set.

C	**Time in s**	**Accuracy**
1e-03	30.437s	0.610
0.01	14.711s	0.662
0.1	14.689s	0.673
1	45.646s	0.666
10	323.879s	0.648

  Following is the table for the values of $\mathbf{C} = \{0.001, 0.01, 0.1, 1.0, 10\}$ and where the loss is **hinge** loss.

C	**Time in s**	**Accuracy**
1e-03	30.437s	0.610
0.01	50.719s	0.612
0.1	185.76s	0.639
1	569.268s	0.653
10	2153.114s	0.653

  There is a significant difference in the training time for hinge squared loss vs hinge loss. Also, it was observed that using hinge loss at times didn't result in convergence within 1000 iterations. Therefor, the field **max_iter** had to be changed.

  The training time of the model with the best hyper-parameter **C = 0.1**, and **hinge-squared loss** was **20.139s**, and the resulting accuracy on the test set was **0.675**

- **SVM (SGD):** The hyperparameter tuned for this model is the regularization parameter **alpha**, which is **defaulted to 0.0001**, but for our task, the values of **alpha** selected are $\{1e-08, 1e-07, 1e-06, 1e-$

$05, 1e-04, 0.001, 0.01, 0.1\}$. The following table represents the respective training times for different value of **alpha** and the corresponding accuracies on the validation set.

The training time of the model with the best hyper-parameter **alpha = 1e-06** was **20.49s**, and the resulting accuracy was **0.663**

alpha	**Time in s**	**Accuracy**
1e-08	121.72s	0.610
1e-07	42.07s	0.642
1e-06	17.33s	0.663
1e-05	8.89s	0.655
1e-04	6.19s	0.633
1e-03	5.68s	0.581
1e-02	5.36s	0.498
1e-01	10.54s	0.440

A summary of the three learning algorithms is represented in the table below:

Learning Algorithm	**Time in s**	**Accuracy**
Multinomial Naive Bayes	0.932s	0.589
SVM-Liblinear	20.14s	0.675
SVM-SGD	20.49	0.663

Remarks:

- The training time for both **Liblinear** and **SGD** are approximately equal for the best set of hyper-parameters. However, for **SGD**, the time taken by suboptimal hyperparameters is significantly less as compared to the **Liblinear SVM**.

- The accuracy is higher for the **Liblinear** as compared to the **SGD** on the validation set and the test set as well (best hyper-parameters).