COL774: Machine Learning
Assignment 2

Devanshi Khatsuriya, 2019CS10344

Note: Refer to README.md for instructions to run code.

1 Text Classification using Naive Bayes

(a) Implemented Naive Bayes algorithm (Multinomial Bag of Words Model with Laplace Smoothing) to classify each 'reviewText' into one of the 5 categories.
Text Processing. Each review was processed by the following steps: tokenization, conversion of each token to lower-case, removal of punctuation characters from each token, removal of non-alphabetic tokens.

- Training Set Accuracy: 70.12 %
- Test Set Accuracy: 66.21 %

(b)
- Test Set Accuracy with a Random Classifier: $\sim$ 20 %
- Test Set Accuracy with Majority Classifier: 66.08 %

The trained model does very well compared to the random classifier, but has approximately the same accuracy as the majority classifier. This is because Class 5 is present in majority in the train set as well as the test set. However, the F1-scores for the trained model will be better the majority classifier.

(c) Confusion Matrix for Test Set Predictions using trained model:

		True Class					
		Class 1	Class 2	Class 3	Class 4	Class 5	Total
Predicted Class	Class 1	18	0	32	41	137	228
	Class 2	3	3	36	128	156	326
	Class 3	1	2	59	473	551	1086
	Class 4	5	2	70	845	2186	3108
	Class 5	19	12	95	782	8344	9252
	Total	46	19	292	2269	11374	14000

Table 1: Confusion Matrix for 1(c)

Class 5 has the highest diagonal entry. Diagonal entry for a class represents the no. of examples of that class that get predicted correctly, which means Class 5 has the largest no. of true-positives. This was probable because Class 5 also has the largest no. of examples, and in general, any model that preforms well overall should also perform well for the majority class (in terms of accuracy). We see that class-wise accuracy (true-positives / true-positives + false-positives) is poor for classes

1, 2, 3 and 4 (7%, 0.1%, 5% and 27% respectively). However, the model achieves high overall accuracy because it has high accuracy for the majority class 5 (90%).

(d) *Further Processing.* After processing each review as in 1(a), further processing was done by removing stop words from each review and further stemming all tokens to their root words.

- Training Set Accuracy: 69.17 %
- Test Set Accuracy: 65.53 %

Both the training set and test set accuracy using the model trained on further-processed data are lesser as compared to 1(a).
Reason. The vocabulary size also decreases from 159,112 in 1(a) to 132,356. Correspondingly the number of parameters (size of vocab.×no. of classes) is also reduced. This may have reduced the ability of model to learn parameters that better fit the data. It is also possible that frequency of stop words and and words without stemming are features that differentiate between the classes better.

(e) An alternative feature on text data that may be used is keeping bi-grams and tri-grams in addition to uni-grams as part of the vocabulary.
Learning a model that took both bi-grams and tri-grams in addition to uni-grams as features resulted in a vocabulary of size over 7 million. This is because features like tri-grams are less likely to be repeated over examples as compared to uni-grams. This model was not viable due to the large number of parameters.
Best Model. After experimenting, the best performing model using additional features was the one learned using uni-grams and bi-grams as features and processed only via tokenization, lower-case, removal of punctuations and removal of non-alphabetic tokens (not stop-word removal and stemming).

- Training Set Accuracy of Best Model: 82.46 %
- Test Set Accuracy of Best Model: 65.96 %

(f) F1-Score for best performing model:

	Class 1	Class 2	Class 3	Class 4	Class 5	Macro F1-Score
F1 Score	0.008264	0.0	0.008936	0.2096	0.8019	0.2057

Table 2: F1-Scores for 1(f)

F1-Scores for classes 1, 2, 3 and 4 are very poor. The model is only likely to perform well on examples of class 5, which is the majority class. Looking only at the test set accuracy, we won't know that the model is only likely to correctly classify examples of the majority class, so the Macro F1-Score is a better metric compared to Test Error for datasets that have some classes as majorities and others as minorities.

(g) I experimented with using the 'summary' field by using it individually, and using it on top of 'reviewText'.
I observed that using only summary gives lesser accuracy as compared to 1(a), which might be because summary is very condensed so each summary may contain lesser frequently occurring words and more unique words. However, the prediction accuracy only slightly decreased and the number of parameters reduced significantly (size of vocab. reduced from hundreds of thousands in 1(a) and 1(d) to few thousands (16,320).
Using summary on top of reviewText gives better prediction accuracy:

- Training Set Accuracy: 70.93 %
- Test Set Accuracy: 66.63 %

2 MNIST Digit Classification using SVMs

2.1 Binary Classification

(a) Implemented Linear Kernel Soft-Margin SVM for classifying digits 4 and 5. Used the CVXOPT package to maximize the quadratic dual objective. Used the obtained alpha values to compute parameters w and b of the margin. [Run code for values]

- nSV (no. of support vectors): 150
- Time taken in Finding Optimal Parameters: 211.44 seconds
- Test Set Accuracy: 97.59 %

(b) Implemented Gaussian Kernel Soft-Margin SVM for classifying digits 4 and 5. Used the CVXOPT package to maximize the quadratic dual objective. Used the obtained alpha values and the training examples (X and Y) to make predictions. [Run code for values]

- nSV (no. of support vectors): 1469
- Time taken in Finding Optimal Parameters: 137.61 seconds
- Test Set Accuracy: 99.89 %

(c) Using LIBSVM Package: Linear Kernel Soft-Margin SVM for classifying digits 4 and 5 [Run code for values]

- nSV (no. of support vectors): 150
- Time taken in Training: 0.79 seconds
- Test Set Accuracy: 98.66 %

Gaussian Kernel Soft-Margin SVM for classifying digits 4 and 5 [Run code for values]

- nSV (no. of support vectors): 1459
- Time taken in Training: 3.90 seconds
- Test Set Accuracy: 99.89 %

The time taken by LIBSVM in training the model is significantly smaller as compared to time taken by CVXOPT to find optimal parameters (upto 40x reduction).
However, the test accuracies and number of support vectors of optimal margin are similar, because the optimization problem is same.

2.2 Multi-Class Classification

(a) Trained 45 Gaussian SVM models using CVXOPT optimizer for a one-vs-one classifier that classifies images into digits 0-9. For prediction, used votes from 45 models and additionally score-values ($\text{abs}(w^T x + b)$) to break ties in no. of votes.

- Time taken in Training: $45 \times (\sim 140) = 6300$ seconds ~ 1.5 hours

- Test Set Accuracy: 97.23 %

(b) Trained 45 Gaussian SVM models using LIBSVM for a one-vs-one classifier that classifies images into digits 0-9. For prediction, used votes from 45 models and additionally p-values given to predictions by LIBSVM to break ties in no. of votes.

- Time taken in Training: $45\times(\sim 4) = 180$ seconds $\sim$ 3 minutes
- Test Set Accuracy: 97.24 %

As earlier, the test prediction accuracies are the same, because of the optimization problem being the same. Also, we see a 35x reduction in time taken (computational cost) using LIBSVM.

(c) The confusion matrix for 2.1(a) [Binary Classification, Linear Kernel]:

		Digit Label		
		4	5	Total
Predicted Digit	4	980	2	982
	5	43	849	892
	Total	1023	851	1874

Table 3: Confusion Matrix for Models of 2.1(a)

The confusion matrix for 2.1(b) [Binary Classification, Gaussian Kernel]:

		Digit Label		
		4	5	Total
Predicted Digit	4	981	1	982
	5	1	891	892
	Total	982	892	1874

Table 4: Confusion Matrix for Models of 2.1(a)

Some mis-classified images for 2.1(a) and 2.1(b) are shown in Fig. 1 and Fig. 2.

The confusion matrices for 2.2(a) and 2.2(b) come out to be identical (Table 5).

The most misclassified digits (least recall) are 8 (94.96 %), 2 (95.78 %) and 3 (96.85 %).
The following images are mis-classified by both 2(a) and 2(b) are shown in Fig. 3. We see that some of these digits are written in such a way that they are close to the predicted label as well.

(d) *Cross Validation to Estimate Best Value of C.* Shuffled the training data and divided it into 5 sets. For each value of C in $\{10^{-5}$, $10^{-}3$, 1, 5, 10$\}$, did a 5-fold Cross Validation by taking 4 sets as training data and remaining 1 set as validation data. Computed the test accuracy using best-performing C.V. model out of the 5.
The log(C) vs. Accuracy plot is shown in Fig. 4.

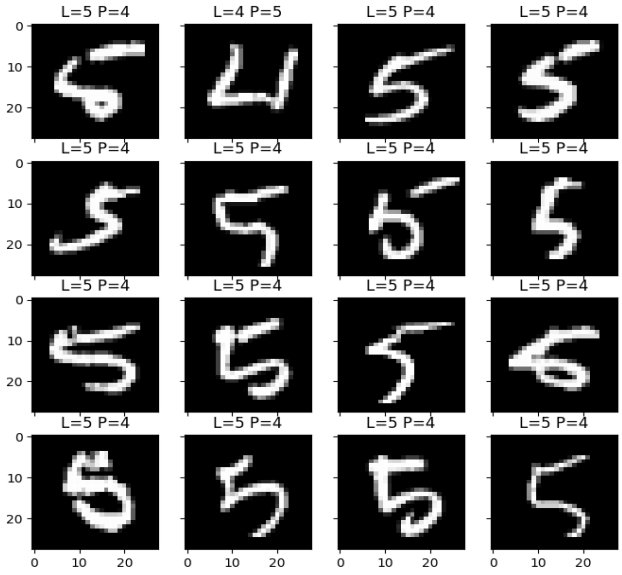

Figure 1: 2.2(c): Some Mis-classified Digits for 2.1(a). L is label and P is predicted digit.

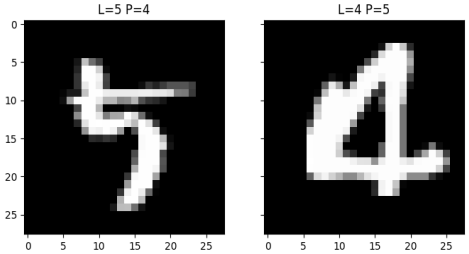

Figure 2: 2.2(c): Mis-classified Digits for 2.1(b). L is label and P is predicted digit.

		Digit Label										
		0	1	2	3	4	5	6	7	8	9	Total
Predicted Digit	0	969	0	1	0	0	3	4	1	2	0	980
	1	0	1121	3	2	1	2	2	0	3	1	1135
	2	4	0	1000	4	2	0	1	6	15	0	1032
	3	0	0	8	985	0	4	0	6	5	2	1010
	4	0	0	4	0	962	0	6	0	2	8	982
	5	2	0	3	6	1	866	7	1	5	1	892
	6	6	3	0	0	4	4	939	0	2	0	958
	7	1	4	19	2	4	0	0	987	2	9	1028
	8	4	0	3	10	2	5	1	3	943	3	974
	9	4	4	3	8	13	4	0	7	14	952	1009
	Total	990	1132	1044	1017	989	888	960	1011	993	976	10000

Table 5: Confusion Matrix for Models of 2.2(a) and 2.2(b)

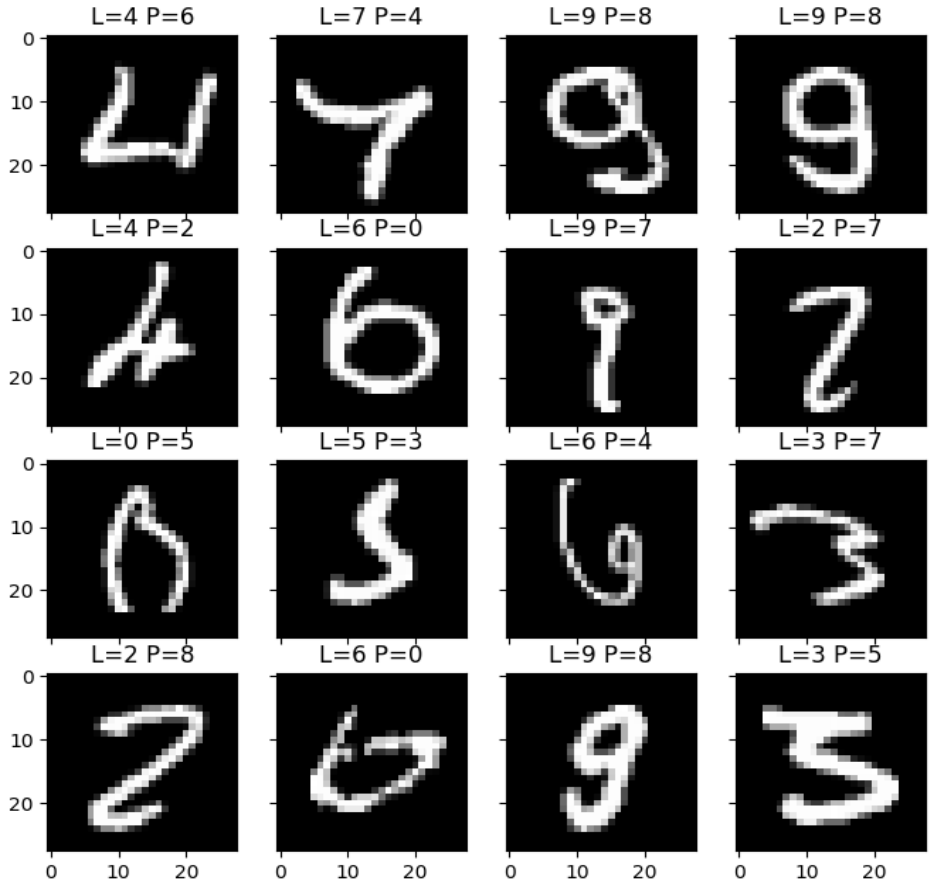

Figure 3: 2.2(c): Some Mis-classified Digits for 2.2(a) and 2.2(b). L is label and P is predicted digit.

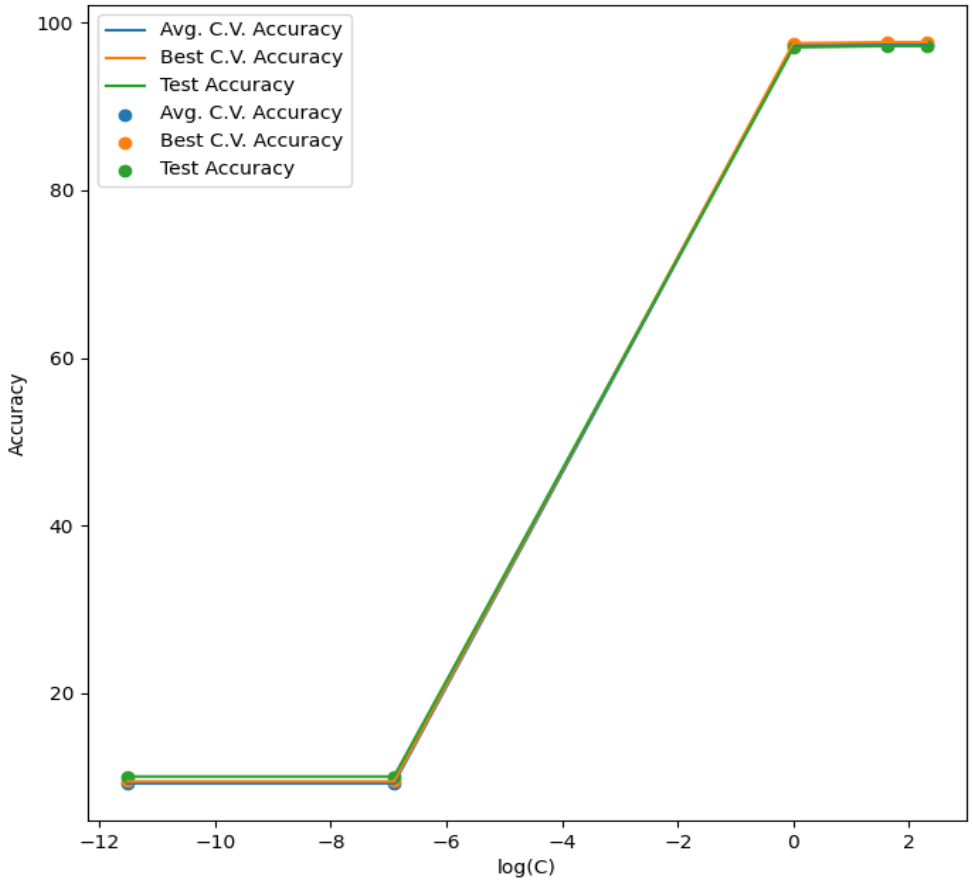

Figure 4: 2.2(d): Cross-Validation and Test Accuracies vs. log(parameter C)