COL774 Assignment 2

Mridul Gupta (AIZ218322)

Wednesday 06 October 2021

1 Q1

1.1 Q1(a)

- Punctutations were removed from the input data, and words containing numbers like l77t code were also removed.
- The bag of words model was implemented.
- Optimal parameters are stored in a pickle file.
- Train accuracy is 71.25% while test accuracy is 66.54%.
- Also, note that I tried to improve the running time to the point I wrote several different versions of the code, but it just doesn't get better than 30-40 mins. I hope this counts as "reasonable" time.

1.2 Q1(b)

- Expected random prediction accuracy is 20%. Predicting the maximum class, the accuracy will be 66.086%.
- As compared to random prediction, the accuracy gained is 36.08%, but there is no significant gain compared to maximum prediction.

1.3 Q1(c)

- Class "5" has the highest value of the diagonal entry. This is also the class with highest number of samples in the training set.
- It is also seen that classes that have a high prior, pull the decision towards themselves: most of the false predictions belong to class "5",

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	9	0	17	41	161
Actual 2	1	0	24	127	174
Actual 3	2	0	31	427	626
Actual 4	2	0	15	672	2419
Actual 5	8	0	25	615	8604

Figure 1: Confusion matrix for Q1(a)

then to class "4" and so on. The number of training samples from each class is listed below.

- Summing up the column for class "5", we can see 11984 = 85.6% of samples are predicted as belonging to class "5". 13.44% to class "4", 0.8% to class "3", 0% to class "2" and 0.157 to class "1".

Category 1:2529 = 5.05%
Category 2:2638 = 5.28%
Category 3:5634 = 11.27%
Category 4:13267 = 26.53%
Catgeory 5:25932 = 51.86%

1.4 Q1(d)

- Accuracy on test set is 64.77%.
- One would expect the accuracy to improve, but it actually goes down by 1.77%.

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	47	16	49	34	82
Actual 2	11	9	82	110	114
Actual 3	9	11	157	472	437
Actual 4	15	15	171	988	1919
Actual 5	73	47	196	1069	7867

Figure 2: Confusion matrix for Q1(d)

1.5 Q1(e)

Features used was review lengths discretized to four buckets based on boxplot of classwise lengths plotted. Also, since most of the reviews have < 400 words after removal of stop words but some reviews go on to contain around 2500 words. We have a lot of data, so in this step I do not process examples that contain more than 400 words. This can be seen as a meta feature, to avoid overfitting. I tested with n-gram extractions, but it was taking a lot of time to train and I didn't have the resources. So, I ran the n-gram with n=5 on a smaller dataset, and the accuracy decreased. I didn't pursue it further. The accuracy is no still around 65% but there is some speed gain as we are not processing long reviews.

1.6 Q1(f)

The class-wise F1-scores and macro F1-score are:

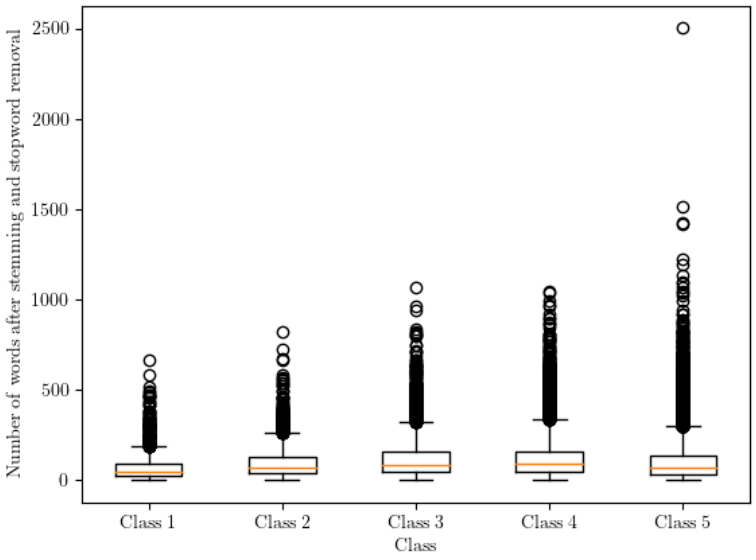

Figure 3: Frequencies of words

Class 1	26.95%
Class 2	4.70%
Class 3	18.77%
Class 4	34.54%
Class 5	80.36%
Macro F1	33.07%

The macro F1-score and the per class F1-scores is more suited in this kind of dataset as the dataset is imabalanced, and this metric gives insights into the performance per class. An extreme case might be of cancer cell detection from images, where we might have very few examples of the positive class. But it will be crucial to classify them correctly so that a patient gets correct treatment on time and a healthy person doesn't has to go through the treatment.

1.7 Q1(g)

I tried training a Naive Bayes model when summaries were just added as part of the review. Since Naive Bayes is a stable model, and the summaries

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	4	0	9	30	185
Actual 2	1	0	12	45	268
Actual 3	5	1	31	155	894
Actual 4	13	2	59	403	2631
Actual 5	30	6	133	773	8310

Figure 4: Confusion matrix for Q1(g)

are small and very likely to be correlated to the review, the results didn't change much.

Next, I tried implementing a separate Naive Bayes model using just the summaries and calculated actual probabilities from the unnormalized log probabilities of the two models, and just chose the prediction of the one which was confident. The accuracy is still around 65%. But all this does is make the prediction go more towards max prediction as seen from confusion matrix. If I had time, I'd have implemented a Naive Bayes model with some random words dropped out to create a third model and used these in ensemble.

2 Q2

2.1 Q2(a)

2.1.1 i)

Accuracy on training data is 99.28%. Accuracy on test data is 97.26%. There are 309 support vectors in this case.

2.1.2 ii)

Accuracy on training data is 100%, while on test data is 99.02%. There are 1856 support vectors in this case. The accuracy improves as compared to when using linear kernel, but so does the computation time. And in this particular case, the gain in accuracy is not enough to justify the loss in computation time.

2.1.3 iii)

- linear: nSV = 297. Test set accuracy is 97.01%. Training and testing times are 10× faster than my cvxopt implementation.
- gaussian: nSV = 1823. Test set accuracy is 99.26%. Training and testing times are 20× faster than my cvxopt implmentation.
- As noted here, it is not possible to extract weights and bias from the svm model created by libsvm in python. I did not compare them.

2.2 Q2(b)

2.2.1 i)

Test set accuracy is 96.84%. The testing process is very slow though, takes around 15 minutes on the test set of 10^4 samples. Training time is around 70 minutes.

2.2.2 ii)

Test set accuracy is 97.23%. The accuracy is not significantly different from the cvxopt implementation. Without any information about SVM performance on the given dataset other than the cvxopt implementation, the expected accuracy is 96.84%. According to Markov inequality then, $P[\text{acc} \geq 97.23] \leq \frac{96.84}{97.23} = 0.9959$. That is to say, this is not such a rare event.

The testing time is 1.5 minutes. Training time is around 3 minutes. Around 25 times faster.

2.2.3 iii)

The images misclassified by the models shown in figures 7-8 are confusing to my eyes as well. Some of them are written in poor handwriting or incomplete,

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	969	0	1	0	0	3	4	1	2	0
Actual 1	0	1123	3	2	0	1	2	0	3	1
Actual 2	6	0	989	8	2	0	4	9	14	0
Actual 3	0	1	8	978	0	5	0	8	7	3
Actual 4	1	1	3	0	953	0	7	1	2	14
Actual 5	3	0	3	7	1	860	9	1	7	1
Actual 6	7	5	0	0	4	3	937	0	2	0
Actual 7	1	11	14	1	1	0	0	987	2	10
Actual 8	4	0	2	11	3	5	2	4	939	4
Actual 9	5	7	3	9	15	1	1	8	11	949

Figure 5: Confusion matrix for o-v-o SVM using CVXOPT

and trying to complete them one can go two ways, the predicted and the true label.

For cvxopt
0 is usually confused as 6 1 is confused as 2, 3, 4, 6, 3 and 9
2 mostly with 8, 7 and 3.
3 as 2, 7 and 8.
4 as 9.
5 as 6, 3 and 8.
6 as 0.
7 as 2, 1 and 9.
8 as 3
9 as 4 and 8.
Similar trends hold for libsvm implementation as well.

2.2.4 iv)

Value of C that gives best accuracy for 5-fold cross-validation is 5. This is also the value of C that gives the best accuracy for test set.

The graph in figure 9 shows that even though there is some variation, the

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	969	0	1	0	0	3	4	1	2	0
Actual 1	0	1121	3	2	1	2	2	0	3	1
Actual 2	4	0	1000	4	2	0	1	6	15	0
Actual 3	0	0	8	985	0	4	0	6	5	2
Actual 4	0	0	4	0	962	0	6	0	2	8
Actual 5	2	0	3	6	1	866	7	1	5	1
Actual 6	6	3	0	0	4	4	939	0	2	0
Actual 7	1	4	19	2	4	0	0	986	2	9
Actual 8	4	0	3	10	1	5	3	3	942	3
Actual 9	4	4	3	8	13	4	0	9	12	952

Figure 6: Confusion matrix for o-v-o SVM using LIBSVM

5-fold cross-validation accuracies and the test set follow each other closely.

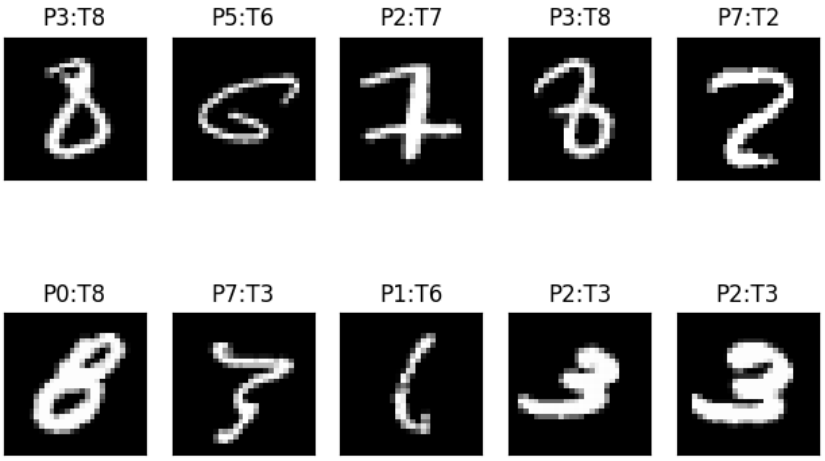

Figure 7: Misclassified images by SVM using CVXOPT

Figure 8: Misclassified imaged by SVM using LIBSVM

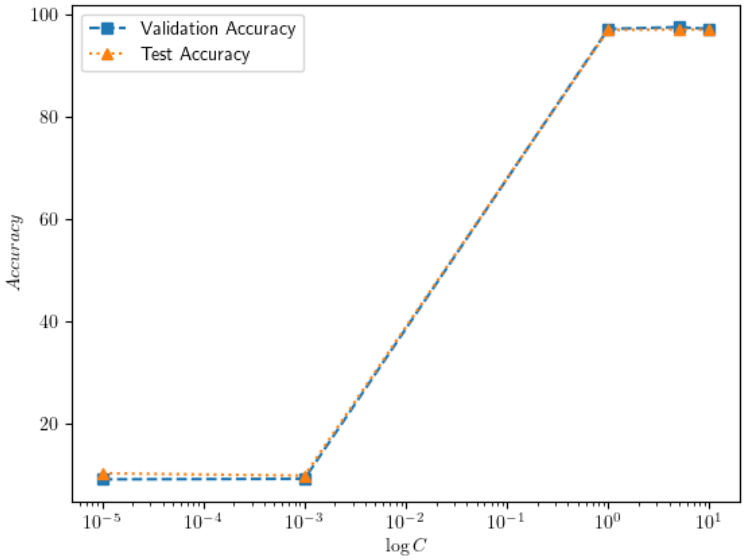

Figure 9: Cross-validation versus test set accuracies