COL341
Assignment 2

Kushagra Rode

March 30, 2023

Analysis of Binary SVM

Description of the code used to implement the SVM

We have two functions, fit and predict. These belong to the trainer class. The fit function essentially is used to find out the support vectors. For doing this I create the matrices required for solving the dual form of soft margin SVM and then use the qpsolvers library with 'quadprog' as the solver to obtain the alpha's. Now to choose the support vectors, I take a threshold value(10^{-5}) and all those alpha's greater than this threshold corresponds to my support vectors. Finally, I also compute the value of the bias b, here I have made the design decision of choosing the average of each of the biases corresponding to each support vector. The predict function then uses the support vectors and the bias calculated in the fit function to the predict the outputs on the test data. Here I again use the kernel, but this it is calculated for the support vectors and the input feature vectors of the test data. I then use the form of the SVM to predict the value associated to each of the input samples. Positive implies 1 negative implies 0.

Using linear kernel

1. C = 0.01
   Validation accuracy - 92.30 %

Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	16	1
	0	2	20

2. C = 0.1
   Validation accuracy - 92.30 %
   Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	16	1
	0	2	20

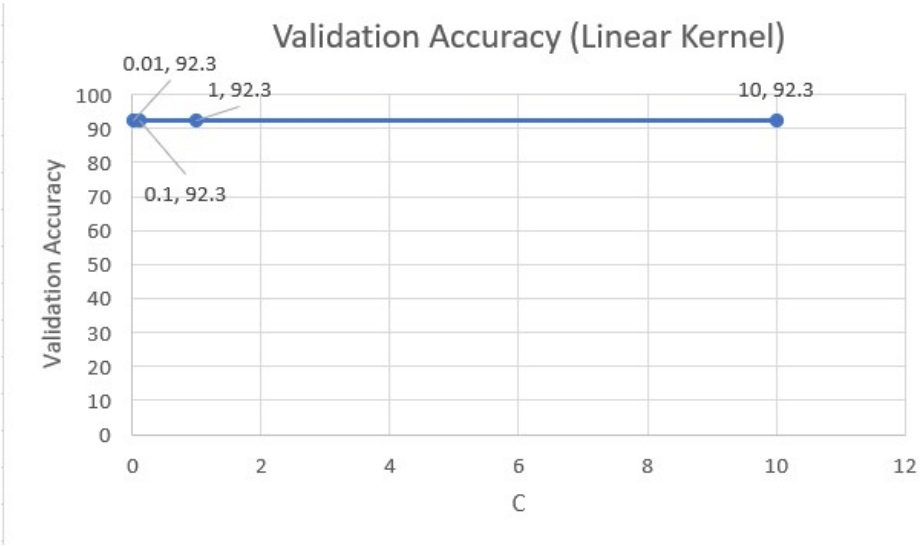
3. C = 1.0
   Validation accuracy - 92.30 %
   Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	16	1
	0	2	20

4. C = 10.0
   Validation accuracy - 92.30 %
   Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	16	1
	0	2	20

Following is the graph of the Validation Accuracy vs the hyperparameter C



We can see from the graph that the validation accuracy doesn't change even if we change the value of C. Although the value of Training accuracy differs for them.

Using RBF Kernel

1. C = 0.01

   (a) $\gamma = 0.1$
   Num of sv = 267
   Validation accuracy - 53.84 %
   Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	0	0
	0	18	21

   (b) $\gamma = 0.01$
   Validation accuracy - 53.84 %
   Num of sv = 260
   Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	0	0
	0	18	21

(c) $\gamma = 0.001$
Validation accuracy - 92.30 %
Num of sv = 208
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	15	0
	0	3	21

2. C = 0.1

(a) $\gamma = 0.1$
Validation accuracy - 53.84 %
Num of sv = 267
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	0	0
	0	18	21

(b) $\gamma = 0.01$
Validation accuracy - 53.84 %
Num of sv = 260
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	0	0
	0	18	21

(c) $\gamma = 0.001$
Validation accuracy - 94.87 %
Num of sv = 184
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	16	0
	0	2	21

3. C = 1.0

(a) $\gamma = 0.1$
Validation accuracy - 53.84 %
Num of sv = 267
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	0	0

0	18	21

(b) $\gamma = 0.01$
Validation accuracy - 79.48 %
Num of sv = 260
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	10	0
	0	8	21

(c) $\gamma = 0.001$
Validation accuracy - 89.74 %
Num of sv = 97
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	16	2
	0	2	19

4. C = 10.0

(a) $\gamma = 0.1$
Validation accuracy - 53.84 %
Num of sv = 267
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	0	0
	0	18	21

(b) $\gamma = 0.01$
Validation accuracy - 79.48 %
Num of sv = 260
Following is the confusion matrix : -

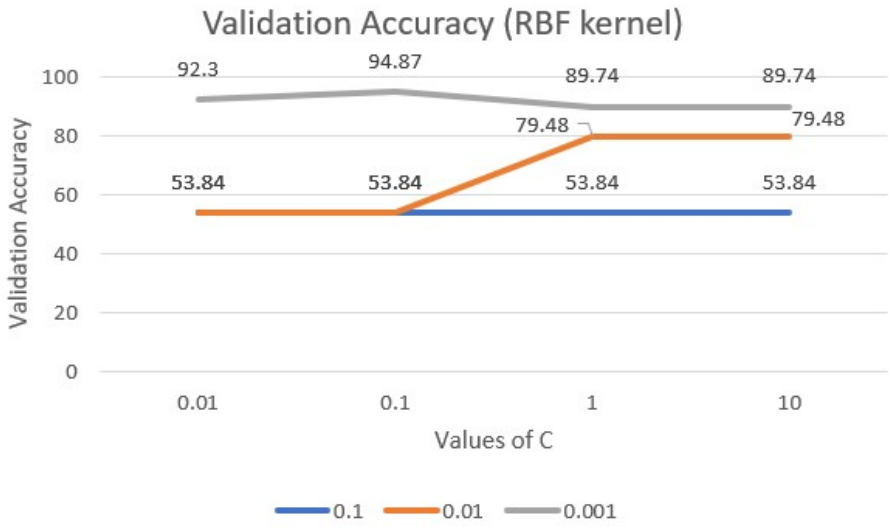
		Predicted	
		1	**0**
Actual	**1**	10	0
	0	8	21

(c) $\gamma = 0.001$
Validation accuracy - 89.74 %
Num of sv = 85
Following is the confusion matrix : -

		Predicted	
		1	**0**
Actual	**1**	16	2

0	2	19

Following is the combined graph of the Validation Accuracy vs the hyperparameter C and the values of gamma. Here each line represents one gamma(legend shown) and the variation for each gamma is shown for all the 4 values of C



From this figure, we can see that when gamma = 0.001, the validation accuracy is the greatest and it decreases on increasing the value of C. For gamma = 0.01, we can see that the accuracy increases somewhat on increasing the value of C. For the case with gamma = 0.1, the accuracy remains constant on increasing C. Although changing the value of C increases the training accuracy. Further the number of support vectors vary significantly on changing the values of C and gamma. Lower number of support vectors generally give a better validation accuracy than the higher ones.

Analysis of Multi-Class SVM

Description of the code used to implement the SVM

For doing this, I have made two new functions,fit_gen and $predict_gen$, these are very similar to the fit and the predict functions, but the difference lies in the fact that the input is filtered according to the need. For example in case of the OVO method, say for each classifier we have two classes 1 and 2, hence we select only those input samples which have their label as either class 1 or class 2.

0.0.1 The One vs One (OVO) method

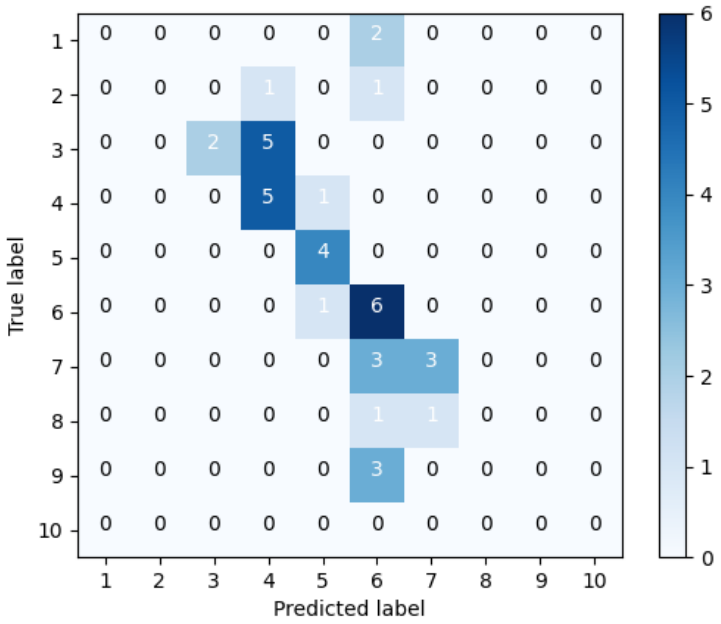
This method basically makes use of $\binom{n}{2}$ binary classifiers, where n is the number of classes in which the data is to be classified. Each classifier classifies the data into either class 1 or class 2. And like this we obtain a prediction on each of the classifiers. Finally for finding out the correct overall class, I make use of majority voting method which the, my output will be the class which will be given by majority of the classifiers.

0.0.2 The One vs All (OVA) method

Here there is one classifier for each class. Each classifier has to predict if the input sample belongs to that class or it belongs to the other. Here I have made a slight modification in getting the results from the $predict_gen$ function, I actually use the sigmoid function to get the probability of the input sample belonging to each class whichever has the highest probability, I predict that class as my output for this input sample. Analysis is based on the RBF Kernel

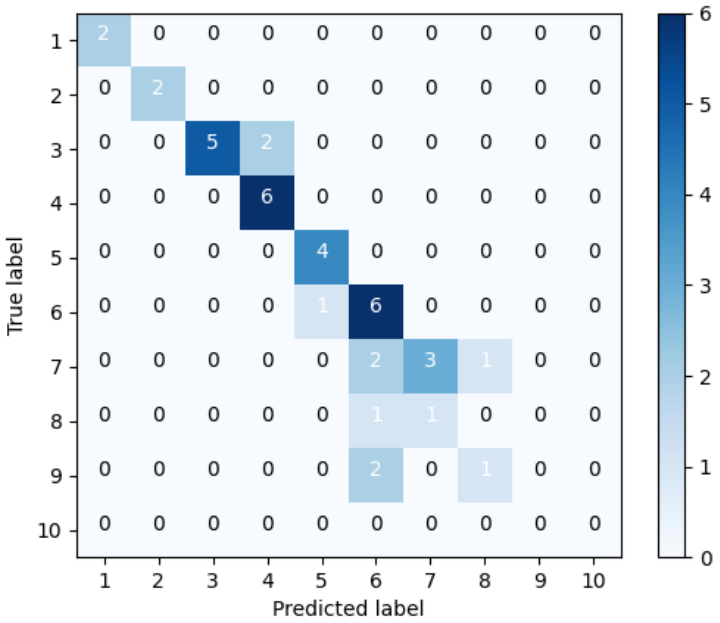
OVO method

1. C = 0.1
   Validation accuracy - 51.28 %



We can clearly see that we have a lot of confusion for class 3 with class 4, this is the major source of misclassification along with this class 6 is confused with class 7, hence the lower validation accuracy.

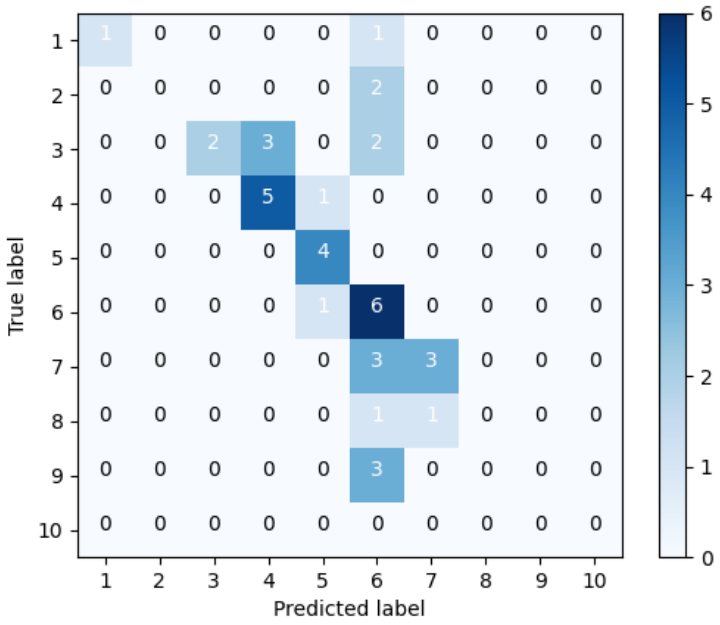
2. C = 1.0
   Validation accuracy - 71.79 %



   Here, the confusion matrix is mostly diagonal major confusions in classes 3,7 and 8. This is the reason for better validation accuracy.

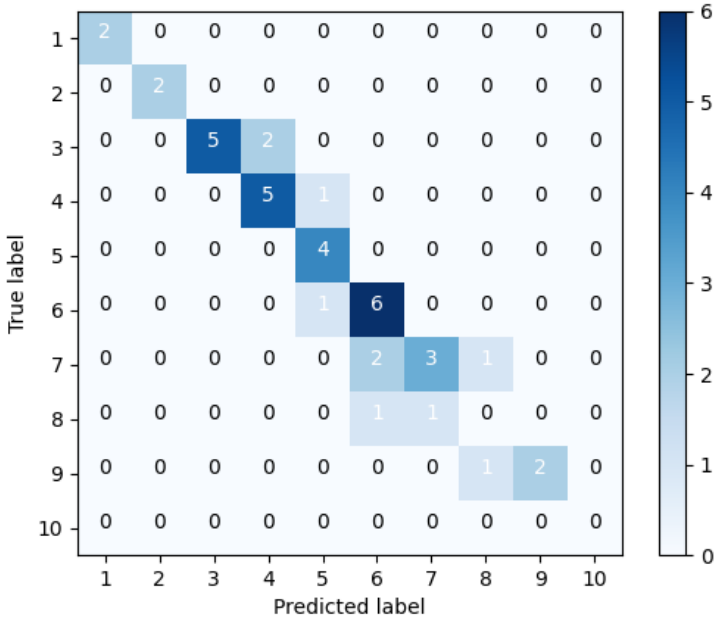
OVA method

1. C = 0.1
   Validation accuracy - 53.84 %



We can clearly see that we have a lot of confusion for class 2 with class 6, further class 3 is also misclassified here, this is the major source of misclassification along with this class 7 is confused with class 6, hence the lower validation accuracy.

2. C = 1.0
   Validation accuracy - 74.35 %



In both the methods above, we can clearly see that on increasing C we get a greater validation accuracy. This is because larger C implies a narrower margin hence resulting in more closely fitting the training data

Best Parameters

For finding the best parameters, I have considered a few things, firstly the validation accuracy which is a measure of the model's generalizability. Secondly, we should also consider model simplicity, i.e the choice of the kernel. I also took a look at the bias for each of the support vectors and my aim is to keep all the biases as close as possible.

Binary Classification

I tested with all the kernels and a wide range of hyperparameters and found the linear, RBF and laplacian kernels to have the highest validation accuracies. Among this linear is the simplest and RBF is the most complex. Although, the value of C and gamma vary for the three kernels. Between laplacian and RBF, the value of C at which maximum validation accuracy is achieved is higher as compared to the RBF one. A higher C implies a narrower boundary which tends to fit the training data more.
Laplacian lies somewhat in between which maybe provides us the best of both worlds, simplicity along with generalization ability. Further, the number of support vectors comes out to be lesser in the laplacian case than the RBF and hence it prevents it somewhat guarantees that overfitting is not taking place. Hence, I would prefer to choose the laplacian kernel over the rbf kernel. This decision is taken in consideration with the type of dataset that we have which is of very high dimensionality but with fewer samples.
So the final best values are -

1. C = 1
2. $\gamma = 0.0009765$
3. kernel = Laplacian

Multi-class Classification

Again I tested with all the kernels in a similar way. Tested with both methods of classification. For me, the OVO method worked in a better way in terms of validation accuracy. The OVA method produces similar accuracy though it has a

lower training accuracy. Hence it may not actually capture the data very well. So the final best values are -

1. C = 10
2. $\gamma = 0.0078125$
3. kernel = RBF