Chinmay Mittal
(2020CS10336)

Assignment 2 | COL341
Machine Learning

March 30, 2023

Binary SVM

Implementation

All the kernel functions have been implemented in *kernel.py*. I have also added additional helper functions which compute the kernel function of a single vector with a set of vectors. This helper will useful for the inference time when the kernel function has to be computed for all the support vectors with the inference point.
I have implemented the dual optimization problem using the *qpsolvers* package. The implementation for encoding the optimization problem can be found in *svm_binary.py*.
To find the bias I use any training point which has $0 \leq \alpha \leq C$. All training points which have non zero α have been saved as support vectors for inference. I have compared floating points while using a margin of $\epsilon = 1e-4$.

Analysis

Linear Kernel

C	Validation Accuracy
0.01	92.307 %
0.1	92.307 %
1	92.307 %
10	92.307 %

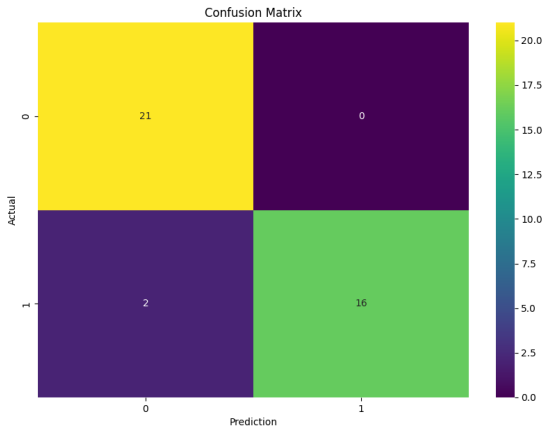

Figure 1: $C = 0.001$

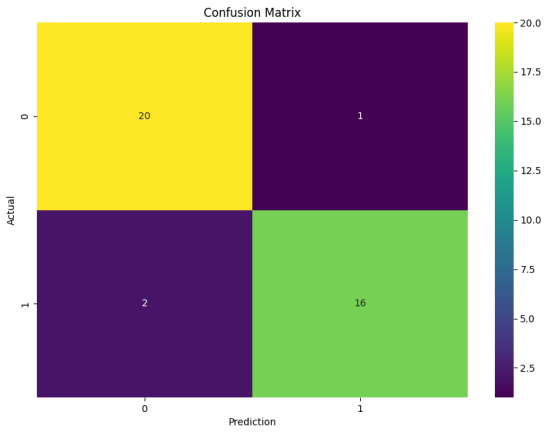

Figure 2: $C = 1$

Increasing value of C implies encouraging smaller values of ϵ and hence larger margins. It is possible that some points are not separable irrespective of how large or small the margin is wheras all the other points are separated irrespective of the margin. In this cases as reported above the validation accuracy remains same irrespective of the value of C.
The choice of hyperparameters for the SVM is extremely sensitive, choosing the right ones can given a very high accuracy compared to other values as seen from the graphs and tables in the subsequent parts.

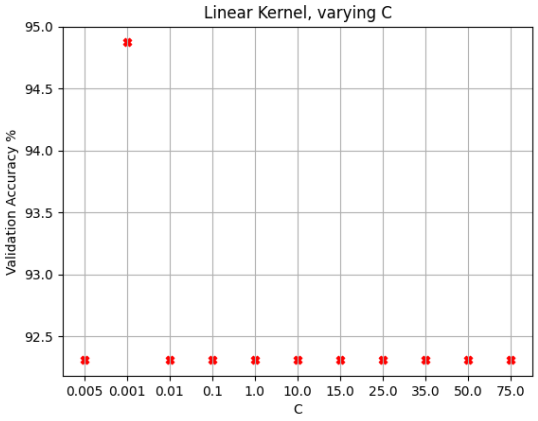

Figure 3: Variation of Validation Accuracy with C

Radial Basis Function Kernel

C	γ	**Validation Accuracy**
0.01	0.1	53.846%
0.1	0.1	53.846 %
1	0.1	53.846 %
10	0.1	53.846 %
0.01	0.01	53.846 %
0.1	0.01	53.846 %
1	0.01	79.48 %
10	0.01	79.48 %
0.01	0.001	53.846 %
0.1	0.001	92.307 %
1	0.001	89.74 %
10	0.001	89.74 %

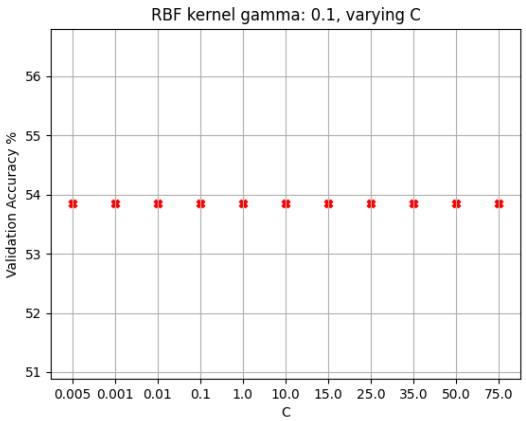

Figure 4: Variation of Validation Accuracy with C, $\gamma = 0.1$

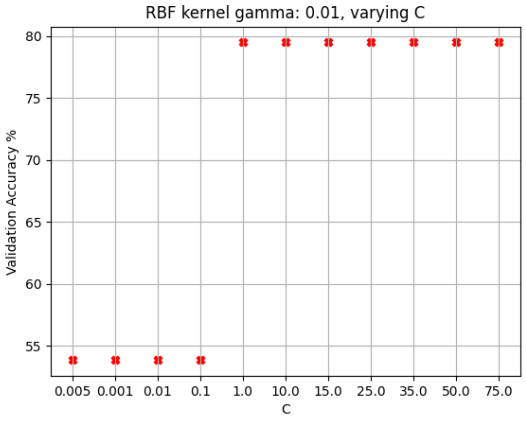

Figure 5: Variation of Validation Accuracy with C, $\gamma = 0.01$

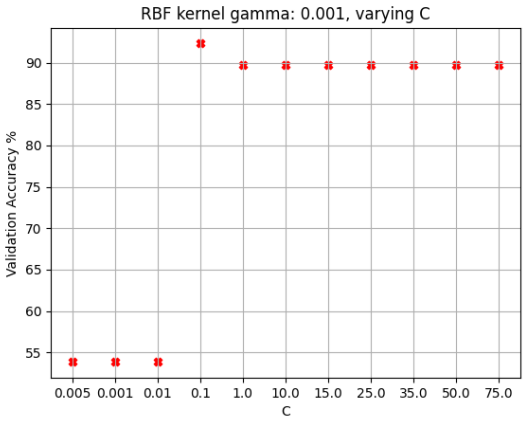

Figure 6: Variation of Validation Accuracy with C, $\gamma = 0.001$

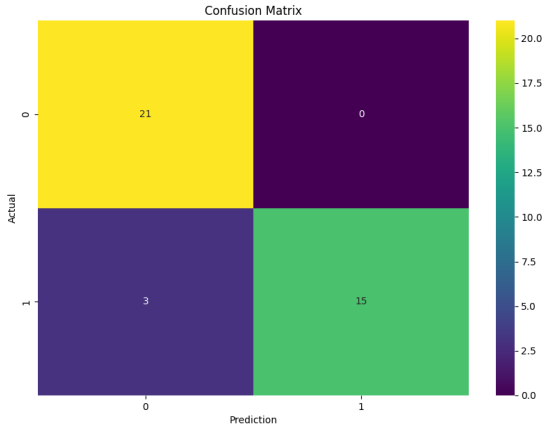

Figure 7: $C = 0.1, \gamma = 0.001$

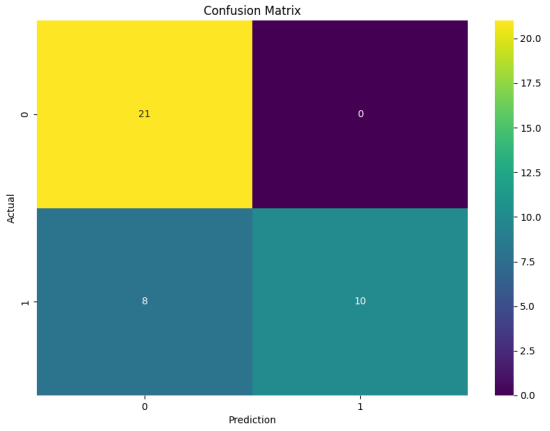

Figure 8: $C = 1, \gamma = 0.01$

The model is extremely sensitive to the value of the hyperparameters. I tried *Grid Search* to find the optimal set of hyperparameters using validation. The value of γ determines the scaling in the RBF function. Too large and too small gamma values can be harmful. The role of C is as described above. Too large C values can cause to model to be biased and too small C values can cause the model to overfit to the training data.

Multi-class Classification

Implementation

The multi-class implementation in *svm_multiclass.py* invoke the implementation of the Binary SVM.
For One vs All, I have trained 1 SVM per class which is trained to differentiate samples of that class from all other samples. While Inference I predict the class for which the corresponding SVM has the highest confidence.
For One vs One, I have trained 1 SVM per pair of classes which is trained to differentiate points of one class from points of the other class. For inference each SVM will provide votes for each class, the class with the most votes will be the final prediction.

One vs One

Radial Basis Function

C	γ	Validation Accuracy
0.1	0.1	23.076 %
1	0.1	69.23 %

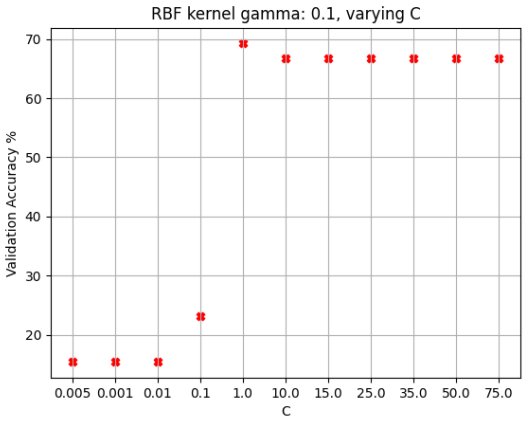

Figure 9: Variation of Validation Accuracy with C, $\gamma = 0.1$

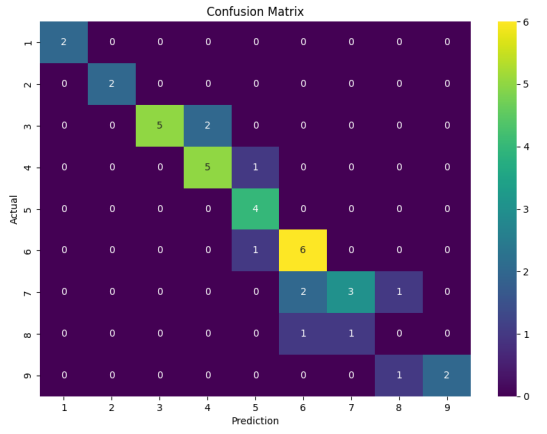

Figure 10: Confusion Matrix $C = 1, \gamma = 0.1$

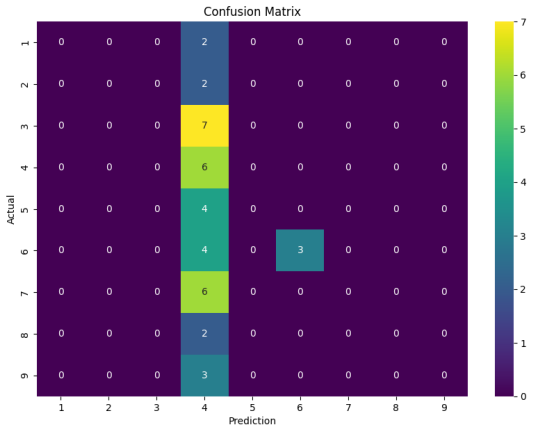

Figure 11: Confusion Matrix $C = 0.1, \gamma = 0.1$

One vs All

Radial Basis Function

C	γ	Validation Accuracy
0.1	0.1	69.23 %
1	0.1	74.35 %

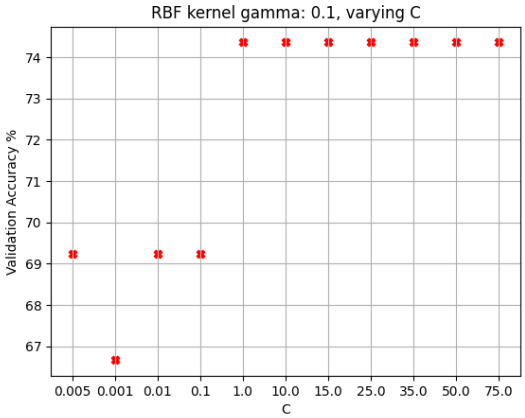

Figure 12: Variation of Validation Accuracy with C, $\gamma = 0.1$

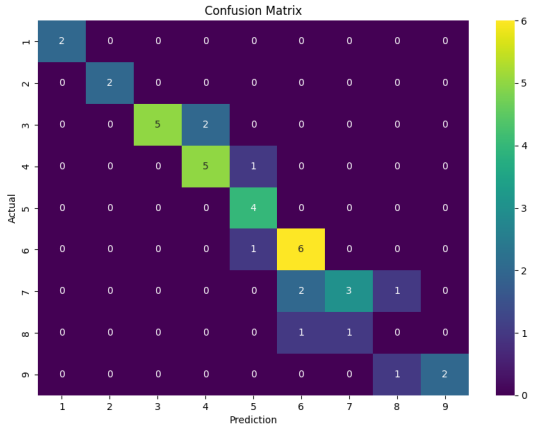

Figure 13: Confusion Matrix $C = 1, \gamma = 0.1$

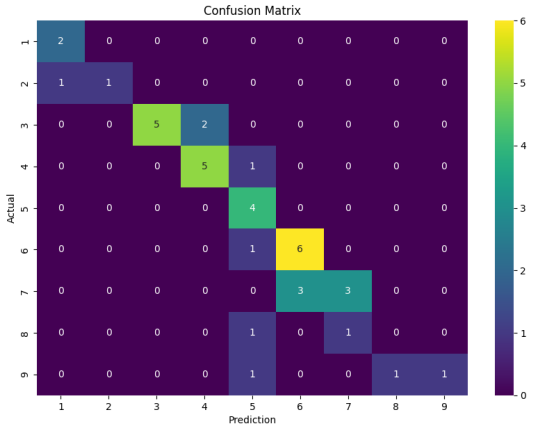

Figure 14: Confusion Matrix $C = 0.1, \gamma = 0.1$

One vs All is sensitive to class imbalance. On the other hand One vs One is more robust to class imbalance. One vs One is computationally expensive and scales badly with an increase in number of classes as compared to One vs All.

Competitive Part

Binary SVM

The following model worked best for me.

Parameter	**Optimum Value**
Kernel	linear
C	0.001
Validation Accuracy	94.871 %

Multiclass SVM

The following model worked best for me.

Parameter	**Optimum Value**
kernel	RBF
C	25
γ	0.001
Validation Accuracy	82.051 %

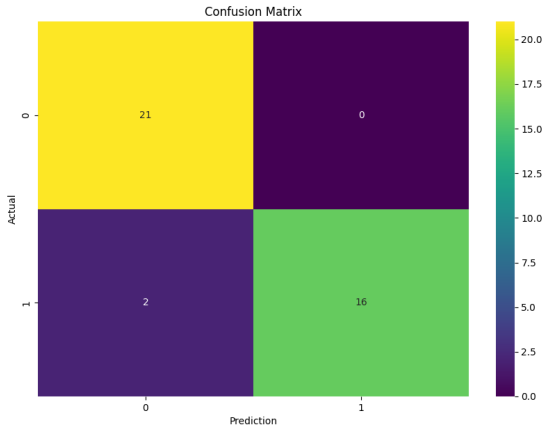

Figure 15: Confusion Matrix for Binary SVM

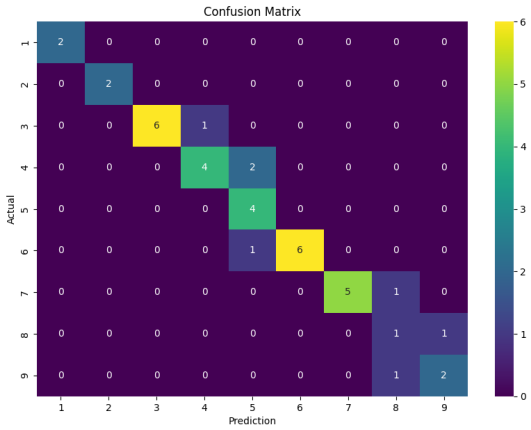

Figure 16: Confusion Matrix for Multiclass SVM

Multiclass SVM is an inherently harder problem then Binary SVM and hence the accuracy drop for the best model.

One can run the code, compute accuracy, plot etc. using the *mymain.py* by changing the variable names appropriately.