CMPE 462 Machine Learning
Spring 2020 Project II

Galip Ümit Yolcu, 2016400249
Yağmur Ceren Dardağan, 2014400063
Umutcan Uvut, 2009400165

April 28, 2020

1 Introduction

In this project, we are asked to implement Support Vector Machine (SVM) and observe the consequences of applying hard/soft margin SVM, using different kernels and parameter values of C on the given data (data.mat) which has 13 features and 2 labels. For this classification problem, we had 270 samples have been given and we have separated the data as 150 samples for the training and 120 samples for the testing. We have used NumPy, scipy library to take advantage of NumPy arrays and the LIBSVM interface to implement SVM.

Mainly 4 tasks have been given to accomplish, training a hard margin linear SVM, training a soft margin SVM for different values of the parameter C, and with different kernel functions, observing the effects of increasing the values of parameter C on the number of support vectors and finally investigating the changes in the hyperplane for the case of removing one of the support vectors vs. one data point that is not a support vector. All the tasks are explained further in Section 2: Tasks.

2 Tasks

2.1 Task 1: Training a Hard Margin Linear SVM

In the task 1, we are asked to train a hard margin linear SVM using the LIBSVM library's Python interface. Since the library does not include hard margin SVM, we have tried to approximate hard margin SVM by using a very high value of C. A high value of C implies little tolerance against error. We have tried to find a value of C such that, a larger C value does not result in a change in training accuracy. This value is empirically around 70000000. We have used the value 420420420420, which results in a training accuracy of about 75% and a training accuracy of 77.5%. Results are shown in Table Table 1.

C	Training Accuracy	Test Accuracy
100	88.6667% (133/150)	81.6667% (98/120)
1000	90% (135/150)	81.6667% (98/120)
10000	90% (135/150)	81.6667% (98/120)
100000	90% (135/150)	82.5% (99/120)
10000000	82% (123/150)	76.6667% (92/120)
70000000	74.6667% (112/150)	77.5% (93/120)
420420420420	74.6667% (112/150)	77.5% (93/120)

Table 1: Training and test accuracy with different values of the C parameter. After a certain value, increasing C does not effect the decision boundary, which implies that the model has reached the approximate optimal margin, i.e. hard-margin SVM.

2.2 Task 2: Training Soft Margin SVM for Different Values of the Parameter C, and With Different Kernel Functions

In task 2, we have observed the performances of different models using different kernels and parameter values of C when training soft margin SVM. In the LIBSVM library, 4 different kernels are available which are linear, polynomial, radial basis function and sigmoid. For different parameters C we used the values 0.05, 0.5, 5, 50 and 500.

For each kernel, we expect the training accuracy to increase as C increases, because this will result in less tolerated classification errors in the training data. However, we expect the testing accuracy to increase up to a certain value and then start decreasing, signaling overfitting.

This is exactly the situation we see when we look at our results show in Table 2 and Table 3, for training and test accuracies respectively.

C	Linear Kernel	Polynomial Kernel	RBF Kernel	Sigmoid Kernel
0.05	84.66666667%	53.33333333%	76.66666667%	79.33333333%
0.5	86.66666667%	84.66666667%	84.66666667%	84.0%
5	88.66666667%	91.33333333%	91.33333333%	82.66666667%
50	88.66666667%	97.33333333%	98.0%	77.33333333%
500	90.0%	100.0%	100.0%	75.33333333%

Table 2: Training accuracies for different kernels and different values of C. Except the last kernel, accuracies increase as C increases, because the model accomodates less classification error.

C	Linear Kernel	Polynomial Kernel	RBF Kernel	Sigmoid Kernel
0.05	84.16666667%	58.33333333%	77.5%	84.16666667%
0.5	85.0%	79.16666667%	83.33333333%	84.16666667%
5	81.66666667%	80.83333333%	80.0%	82.5%
50	81.66666667%	80.0%	77.5%	72.5%
500	81.66666667%	75.0%	77.5%	74.16666667%

Table 3: Test accuracies for different kernels and different values of C. We see that for low values of C, the accuracy increases for each kernel. After a certain optimal value, accuracies start to decrease, which indicates overfitting.

2.3 Task 3: Observing the Changes in the Number of Support Vectors as the Value of C Increases

In this task, we expect the number of support vectors to decrease as C increases. As we mentioned, a high C value results in less tolerance to classification errors, which results in a narrower margin, which in turn implies a small number of data points in the margin area.

Using linear SVM, our experiments are in accordance with this expectation.

C	# of Support Vectors
1	58
2	56
3	55
4	54
5	54
6	53
7	52
8	52
9	50
10	51
11	51
12	51
13	51
14	50
15	50
16	50
17	50
18	50
19	50

Table 4: Number of support vectors for different values of the C parameter. As C increases, the margin becomes narrower, implying a lower number of support vectors.

2.4 Task 4: Investigating the Changes in the Hyperplane when Removing One of the Support Vectors vs. One Data Point That Is Not a Support Vector

To investigate the changes in the hyperplane, we retrain soft margin SVM with linear kernel and 19 as the value of the parameter C . We defined this SVM as our base model. We found the support vectors and derived the weights and the bias term using the LIBSVM library functions. Then we have chose a candidate support vector and a non-support vector data point to remove from the data. We have retrained the SVM using the same kernel function and the same C value, on the data when one of the support vector point has been removed and have found its support vector points, their weights and the bias. We have followed the same procedure for the non-support vector point and found the resulting model's support vector points, its weights and the bias. We expected the weights of the model to change when we discard a support vector, because this effects the error in the margin region, whereas we expected the weights to remain the same when we discard a non-support vector data point from the training data.

When we investigate the results of 3 different SVM models, we have found that the weights change when we discard a support vector, and they don't change when we discard an irrelevant data point, thus the results are in agreement with our expectations.

For the base model, trained on the full dataset, the weight vector is: [-1.30888701 0.52571551 1.3470139 1.37874173 1.6273681 0.16133491 0.13203498 -1.8521204 -0.02562698 -0.0762774 0.06906356 2.72479396 0.44811908] and the bias term is 2.3785383593196174.

For the second model, trained on the dataset where one of the support vector data points is removed, the weight vector is: [-1.32983847 0.50137912 1.51487358 1.32368342 1.22296991 0.09252357 0.23434143 -1.66513531 -0.08056412 0.29650759 0.01183793 2.98157394 0.42209016] and the bias term is 2.426370193620015.

For the last model, trained on the dataset where one of the non-support vector data points is discarded, the weight vector is [-1.30888701 0.52571551 1.3470139 1.37874173 1.6273681 0.16133491 0.13203498 -1.8521204 -0.02562698 -0.0762774 0.06906356 2.72479396 0.44811908] and the bias term is 2.3785383593196174.

3 Conclusion

In this project, we used the LIBSVM library to train hard margin and soft margin SVM for classification. We have observed the effect of changing the value of the parameter C and the kernel. Moreover, we observed the effect of removing support vector data points from the training dataset.

We reached the best results by using polynomial kernel(with default order 3) with moderate C. We saw that polynomial kernel and RBF kernel managed to make the dataset linearly seperable, because training accuracy reaches 100% with high C.

4 Bonus Problem

We have correctly constructed the needed matrices and vectors according to the course slides for the primal respresentation. However, the QP solver needs an additional constraint of type Ax=b, which is not derived in class. We tried giving A=0 matrix and b=0 vector, so as to make the constraint trivial, satisfied by all $\mathbf{u}$. This resulted in an error saying Rank(A) should not be 1. There is no documentation on the restrictions on Rank(A). Moreover, when we omit A and b, the solver returns wrong solutions and there is no documentation on the default values of A and b. Therefore, we abandon this approach and instead implement the dual representation, using a web page [1] as a guide.

References

[1] *Support Vector Machine: Python implementation using CVXOPT.* URL: `https://xavierbourretsicotte.github.io/SVM_implementation.html`.