CMPE 462 - Project 2
Implementing an SVM Classifier

Student ID1: 2015400069
Student ID2: 2015400177
Student ID3: 2019700087

May 2020

1 Introduction

In this project, we were asked to implement hard and soft margin SVM by using LIBSVM and CVXOPT. First, we loaded the data into the notebook and took the first 150 rows for training and the rest for testing purposes.

2 Hard Margin SVM

In Logistic Regression, we added regularization term (λ) to loss function to prevent coefficients from reaching large values. In SVM, C value is used for regularization. C value can be considered as $1/\lambda$ and instead of adding this term to loss function, loss function is multiplied with C parameter.

The distinction between the hard margin and soft margin comes from the c value. In hard margin SVM, C value must be high to satisfy the requirements for the narrowest margin between support vectors. In our implementation, 10^{10} is used to train hard margin SVM and linear kernel is used for simplicity.

Dataset	C	Kernel	Accuracy(%)
Train	10^{10}	Linear	74.67
Test	10^{10}	Linear	77.50

Table 1: Hard Margin SVM Accuracies for Train and Test Set

As can be seen in the table above, the accuracy on test is higher than the accuracy of train set which is somehow unexpected result. However, there may be one or more of the following reasons behind this situation.

- Since we are using 150 points for training which are relatively low number of points, our model may be underfitted to some degree and behave unstable.
- Since we are not shuffling the data for the sake of assignment evaluation, the test data may be more easier to predict (e.g.having no outliers) compared to train data.

Therefore, in order to obtain more reasonable results in terms of train-test accuracy comparison, we would shuffle the data provided and make it more homogeneous. Also, if it is feasible, we would increase the number of data points to train.

3 Soft Margin SVM

3.1 Observing Performance

As aforementioned in the Hard Margin SVM section, hard margin SVM requires higher C values on the other hand soft margin SVM requires lower C values to form wider margin between support vectors and decision boundary. This difference in margins hard and soft margin SVM models provide error tolerance to the soft margin SVM models.

First, to observe the effect of kernel types in performance, 1 is picked as the fixed C value and models were trained with same data but different kernel types [Linear, Polynomial, Radial Basis, Sigmoid] and results were obtained.

Dataset	Kernel	C	Accuracy(%)
Train	Linear	1	86.67
Train	Polynomial	1	86.00
Train	Radial Basis	1	86.67
Train	Sigmoid	1	82.67
Test	Linear	1	85.00
Test	Polynomial	1	82.50
Test	Radial Basis	1	84.17
Test	Sigmoid	1	84.17

Table 2: Accuracies for training and test datasets for different kernels with $C = 1$

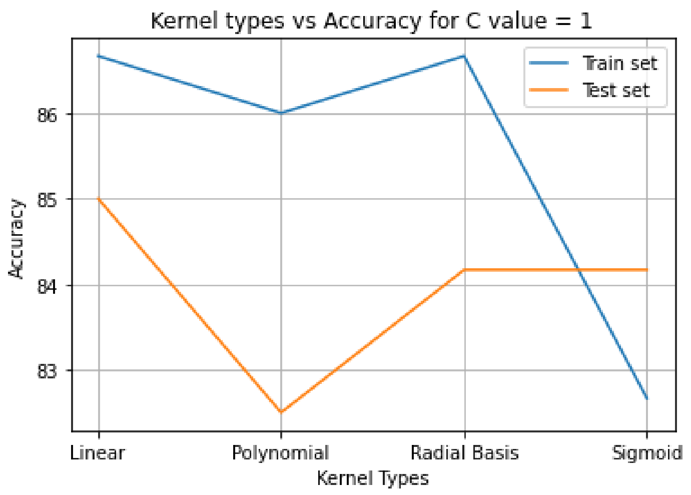

Figure 1: Plot of accuracies for training and test datasets for different kernels with $C = 1$

As can be seen in the table above, different kernel types are trained differently for the same C value, and therefore the test accuracy varies. For C = 1, **linear kernel** has performed better than other three kernel types.

Second, to observe the effect of C value in performance, kernel type is picked as Radial Basis and models were trained with same data but different C values [1,10,25,50,75,100] and results were obtained.

Dataset	Kernel	C	Accuracy(%)
Train	Radial Basis	1	86.67
Train	Radial Basis	10	95.33
Train	Radial Basis	25	97.33
Train	Radial Basis	50	98.00
Train	Radial Basis	75	98.00
Train	Radial Basis	100	99.33
Test	Radial Basis	1	84.17
Test	Radial Basis	10	77.50
Test	Radial Basis	25	78.33
Test	Radial Basis	50	77.50
Test	Radial Basis	75	79.17
Test	Radial Basis	100	78.33

Table 3: Accuracies for training and test datasets for different C values with $kernel$ = Radial Basis

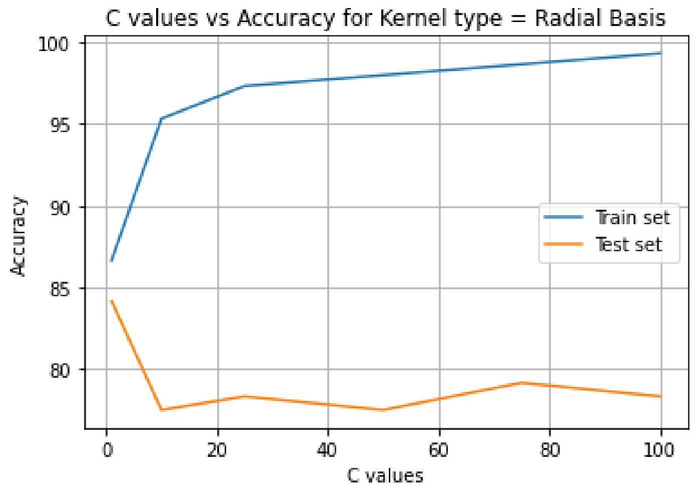

Figure 2: Plot of accuracies for training and test datasets for different C values with $kernel$ = Radial Basis

As can be seen in the table above, with increasing values of C, accuracy values were not increasing linearly. Thus, we can conclude that different accuracy is obtained when the same kernel is trained with different C values.

3.2 Relationship Between C and Support Vectors

To observe the relationship between the c values and the number of support vectors, different models were trained with the Linear kernel and different C values [1, 5, 10, 100, 300].

Kernel	C	Number of SVs
Linear	1	58
Linear	5	54
Linear	10	51
Linear	100	50
Linear	300	49

Table 4: Number of support vectors with respect to C

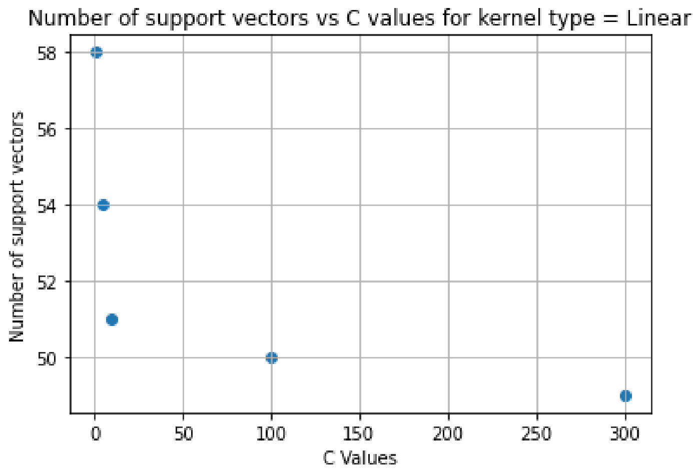

Figure 3: Plot of number of support vectors with respect to C

As C value increases, the number of support vector decreases. The reason behind this behavior can be explained in the following way. As we know from lectures, the C parameter is used to determine how much we want to avoid misclassifying each training example. High C can be considered as we are allowing fewer outliers and vice versa. If we increase the C parameter, actually we define greater penalty on violation of the constraint, and the model will try to decrease the number of violations. Since support vectors can be defined as the data points that lie closest to the decision surface (or hyperplane), there will also be fewer support vectors since the model will change to reduce the number of violations. Also, we have found out that after some number, increasing C value more does not affect the number of support vectors. For instance, number of support vectors are same for both C value 300 and 1000.

3.3 Hyperplane

In SVM, decision boundary described as a hyperplane and it depends on the values that were used while training the model. But, the hyperplane is only affected by the points that are support vectors. The points that are not support vectors are not affect the hyperplane.

We trained the model with Linear kernel and 1 as the C value. After training the model, w vector, norm of w, b value and the number of support vectors were calculated. After getting the initial calculations, the model was trained two more times to prove that the hyperplane will be affected by points that are support vectors and will not be affected with the points that are not support vectors.

To achieve this goal, one point which is a support vector and one point which is not a support vector is removed from both training and test separately. After deletion, calculations were made for new hyperplane.

Operation	Kernel	C	$\|\|w\|\|$	b	# of Support Vectors
NoOp	Linear	1	2.48	1.185	58
One Support Vector Deleted	Linear	1	2.66	1.311	56
One Point Deleted	Linear	1	2.48	1.185	58

Table 5: Changes in hyperplane with different operations.

As can be seen in the table, deleting a point which is a support vector from the dataset was affected the hyperplane.

The reason behind this change in hyperplane can be explained with the fundamentals of SVM. SVM is an algorithm that is trying to find the best hyperplane that separetes multiple classes by maximizing the distance (margin) between each other. To calculate the margin between these classes, SVM uses some points which are called as **support vectors** that are closer to the hyperplane than other points.

Since, SVM only uses support vectors to calculate the margin and the hyperplane. After finding the points which are support vectors, the remaining data will become useless for the SVM. Even if we remove all of the points which are not support vectors, SVM will find the same hyperplane.

4 Hard Margin SVM with CVXOPT

As bonus part of the project, we are expected to implement hard margin SVM with using CVXOPT QP Solver. To achieve this task, we had to come up with a different notation that was used in the lecture slides. QP Solver was defined as $u* < -QP(Q, p, A, c)$.

To adapt this equation into the CVXOPT QP Solver, we treated this equation as $u* < -QP(P, q, G, h)$. After QP Solver finished its calculations, we had to extract w vector and b term from the $u*$. As mentioned in the slides, the very first term in the $u*$ vector is corresponds to bias (b) and the rest corresponds to the w vector. After extracting these terms, we use these terms to make predicitons in our toy dataset. To make predictions, $g(x) = sign(X^T w + b)$ was used as the hypothesis function.

In conclusion, we managed to find the same w vector and b term which were given in the lecture slides and make predictions with %100 accuracy by using CVXOPT QP Solver.