CMPE 462 Machine Learning
Department of Computer Engineering
Bogazici University

PROJECT 2
SUPPORT VECTOR MACHINES

Group Name: Hacı Kolonyası
Student ID1: 2015300084
Student ID2: 2015401183

17.05.2020

Contents

1 Hard Margin Linear SVM **1**

2 Soft Margin SVM **2**

3 C & Support Vector Relationship **5**

3.1 Effect of Increase in C 5

4 Changes in Hyperplane **6**

5 Bonus **9**

1. Hard Margin Linear SVM

We used svm_train and svm_predict functions at github source[1] and function parameters can be checked at this file[2].

To train the data with hard margin linear SVM, we set the option as '-c 10000000000 -t 0'. c parameter is about how hard the SVM is. We determined c as too high for hard SVM. t parameter is about kernel type. -t 0 is linear SVM.

Training accuracy is 74.6667% (112/150) and test accuracy is 77.5% (93/120).

2. Soft Margin SVM

We used SVM with different C values(exponential an linear C values) and different Kernel types:

-t:	Kernel Type
0:	linear
1:	polynomial
2:	radial basis function
3:	sigmoid

The option argument is '-c c_value -t kernel_type_number'.

Firstly, we tried exponential C values with each kernel type.

C values = [0.0001, 0.0003, 0.001,0.003,0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000]

training accuracies:

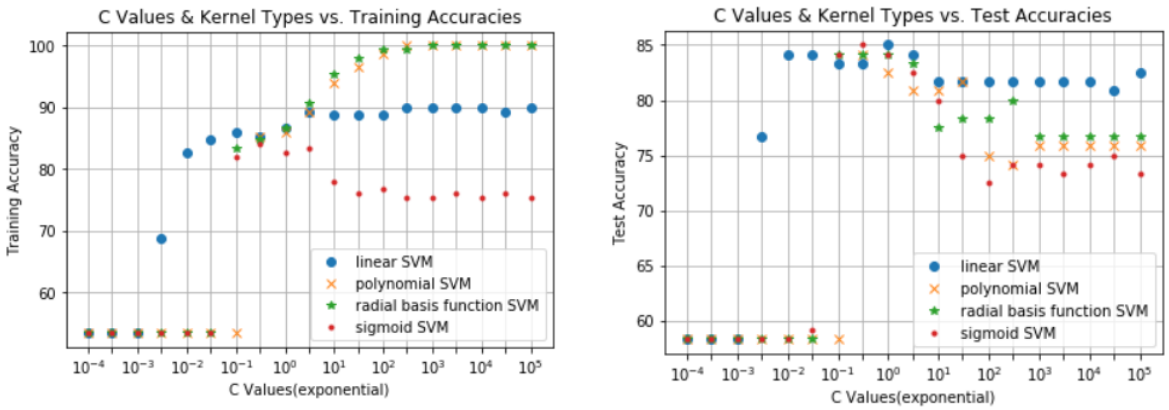
C values:	linear SVM	polynomial SVM	radial basis SVM	sigmoid SVM
0.000100	53.333333	53.333333	53.333333	53.333333
0.000300	53.333333	53.333333	53.333333	53.333333
0.001000	53.333333	53.333333	53.333333	53.333333
0.003000	68.666667	53.333333	53.333333	53.333333
0.010000	82.666667	53.333333	53.333333	53.333333
0.030000	84.666667	53.333333	53.333333	53.333333
0.100000	86.000000	53.333333	83.333333	82.000000
0.300000	85.333333	85.333333	84.666667	84.000000
1.000000	86.666667	86.000000	86.666667	82.666667
3.000000	89.333333	89.333333	90.666667	83.333333
10.000000	88.666667	94.000000	95.333333	78.000000
30.000000	88.666667	96.666667	98.000000	76.000000
100.000000	88.666667	98.666667	99.333333	76.666667
300.000000	90.000000	100.000000	99.333333	75.333333
1000.000000	90.000000	100.000000	100.000000	75.333333
3000.000000	90.000000	100.000000	100.000000	76.000000
10000.000000	90.000000	100.000000	100.000000	75.333333
30000.000000	89.333333	100.000000	100.000000	76.000000
100000.000000	90.000000	100.000000	100.000000	75.333333

Figure 2.1: Training Accuracy Table

test accuracies:

C values:	linear SVM	polynomial SVM	radial basis SVM	sigmoid SVM
0.000100	58.333333	58.333333	58.333333	58.333333
0.000300	58.333333	58.333333	58.333333	58.333333
0.001000	58.333333	58.333333	58.333333	58.333333
0.003000	76.666667	58.333333	58.333333	58.333333
0.010000	84.166667	58.333333	58.333333	58.333333
0.030000	84.166667	58.333333	58.333333	59.166667
0.100000	83.333333	58.333333	84.166667	84.166667
0.300000	83.333333	84.166667	84.166667	85.000000
1.000000	85.000000	82.500000	84.166667	84.166667
3.000000	84.166667	80.833333	83.333333	82.500000
10.000000	81.666667	80.833333	77.500000	80.000000
30.000000	81.666667	81.666667	78.333333	75.000000
100.000000	81.666667	75.000000	78.333333	72.500000
300.000000	81.666667	74.166667	80.000000	74.166667
1000.000000	81.666667	75.833333	76.666667	74.166667
3000.000000	81.666667	75.833333	76.666667	73.333333
10000.000000	81.666667	75.833333	76.666667	74.166667
30000.000000	80.833333	75.833333	76.666667	75.000000
100000.000000	82.500000	75.833333	76.666667	73.333333

Figure 2.2: Test Accuracy Table



(a) Training Accuracy Plot

(b) Test Accuracy Plot

Figure 2.3: Accuracy Plots with different C values and Kernel Types

For all C value and kernel type pairs, training and test accuracies can be checked.

When C value increases(when the model is closer to the hard SVM), training accuracy increases for all kernel types(except sigmoid SVM). Polynomial and radial basis SVM models have 100% training accuracy for high C values.

However, when C value increases(when the model is closer to the hard SVM), test accuracy increases until some c values and then decreases for all kernel types. At the c interval [0.1 1], all kernel types are successful and have high test accuracies. The best C value is 1.0 for linear SVM, 0.3 for polynomial SVM, 0.1, 0.3, 1.0 for radial basis SVM and 0.3 for sigmoid SVM.

Secondly, we tried linear C values at this interval with each kernel type.

C values = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]

training accuracies:

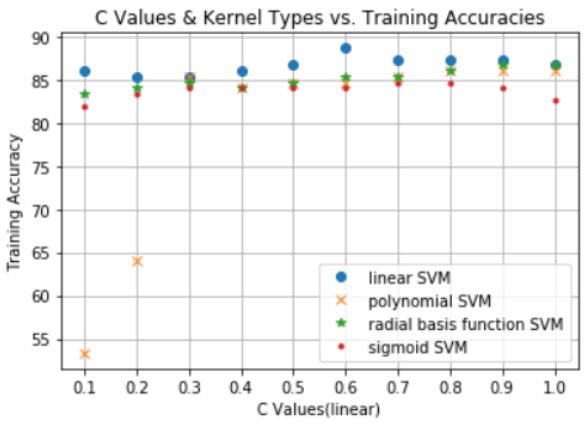
C values:	linear SVM	polynomial SVM	radial basis SVM	sigmoid SVM
0.100000	86.000000	53.333333	83.333333	82.000000
0.200000	85.333333	64.000000	84.000000	83.333333
0.300000	85.333333	85.333333	84.666667	84.000000
0.400000	86.000000	84.000000	84.000000	84.000000
0.500000	86.666667	84.666667	84.666667	84.000000
0.600000	88.666667	84.666667	85.333333	84.000000
0.700000	87.333333	85.333333	85.333333	84.666667
0.800000	87.333333	86.000000	86.000000	84.666667
0.900000	87.333333	86.000000	86.666667	84.000000
1.000000	86.666667	86.000000	86.666667	82.666667

Figure 2.4: Training Accuracy Table

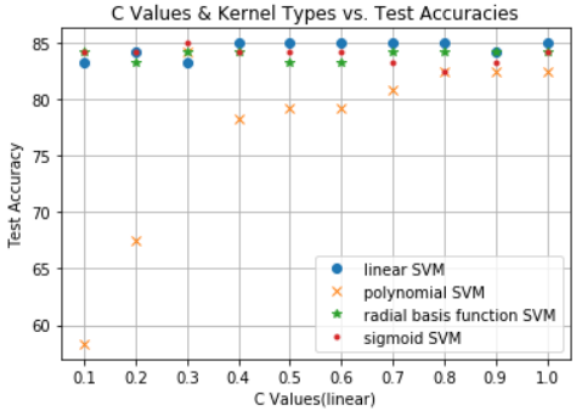
test accuracies:

C values:	linear SVM	polynomial SVM	radial basis SVM	sigmoid SVM
0.100000	83.333333	58.333333	84.166667	84.166667
0.200000	84.166667	67.500000	83.333333	84.166667
0.300000	83.333333	84.166667	84.166667	85.000000
0.400000	85.000000	78.333333	84.166667	84.166667
0.500000	85.000000	79.166667	83.333333	84.166667
0.600000	85.000000	79.166667	83.333333	84.166667
0.700000	85.000000	80.833333	84.166667	83.333333
0.800000	85.000000	82.500000	84.166667	82.500000
0.900000	84.166667	82.500000	84.166667	83.333333
1.000000	85.000000	82.500000	84.166667	84.166667

Figure 2.5: Test Accuracy Table



(a) Training Accuracy Plot



(b) Test Accuracy Plot

Figure 2.6: Accuracy Plots with different C values and Kernel Types

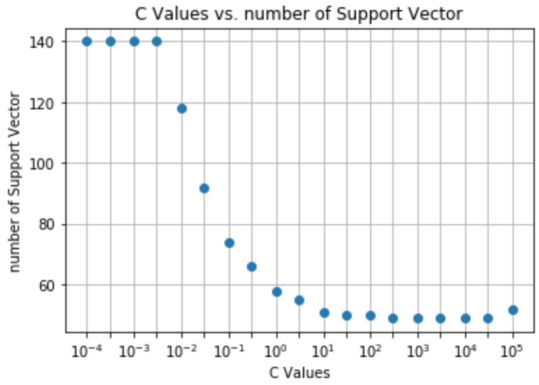
All kernel types (except polynomial SVM with lower c values) have high training and test accuracy. The models with highest test accuracy(85%) are linear SVM with 0.4, 0.5, 0.6, 0.7, 0.8 and 1.0 C values and sigmoid SVM with 0.3 C value.

3. C & Support Vector Relationship

3.1 Effect of Increase in C

For this task we used several penalty terms in increasing order to see the effect of it on the number of support vectors. We used the following vector,resp C values and linear kernel :

$$c_{exp} = [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000]$$



(a) Graph

cvalue	support vector number
0.0001:	140.0
0.0003:	140.0
0.001:	140.0
0.003:	140.0
0.01:	118.0
0.03:	92.0
0.1:	74.0
0.3:	66.0
1:	58.0
3:	55.0
10:	51.0
30:	50.0
100:	50.0
300:	49.0
1000:	49.0
3000:	49.0
10000:	49.0
30000:	49.0
100000:	52.0

(b) Table

Figure 3.1: Number of support vectors as C increases

Conclusion:
As the penalty term increases, the number of the support vectors decreases gradually from 140 to approximately 50. This situation matches with the theory since C is a penalty on slack variables. As C increases we penalize more the violations of the margin. That is why margin gets narrower when C increases and there are fewer support vectors.

4. Changes in Hyperplane

In this task we tried to examine the effect of removing one data point from training set on the hyperplane. For this purpose we used C as 10 and linear kernel.

- First of all we calculated initial weight

  $\mathbf{w} = [-1.3072, 0.5251, 1.3479, 1.3830, 1.6137, 0.1593, 0.1314, -1.8510, -0.0279, -0.0750, 0.0706, 2.7299, 0.4445]$

- Then bias

  $$\mathbf{b} = 2.3752$$

- We combined b and w to create a new vector called $w_{combined}$. We will use this to determine the effect of removing one data point on the location of hyperplane.

  $$w_{combined} = [2.3752, -1.3072, 0.5251, 1.3479, 1.3830, 1.6137, 0.1593, 0.1314, ..., 0.4445]$$

- We calculated margin as reciprocal of the norm of weight vector

  $$\mathbf{margin} = 0.226752003$$

- In the end we determined support vectors by their indices and removed every vector from training set and checked their effects on the following:

  1. L2 norm of the difference vector between $w_{combined}$ and $w_{combined}$ after removing one data point
  2. Margin
  3. the number of support vectors

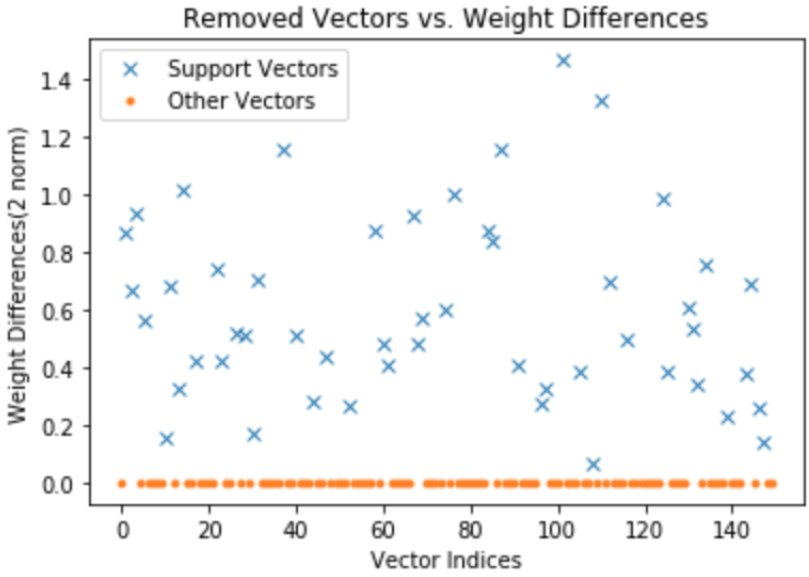


Figure 4.1: L2 norm difference of b and w combined vector

Conclusion1:
As we can see from graph when we removed a vector which is not a support vector L2 norm of difference between two vectors is zero or almost zero. However if we remove one support vector it changes our decision boundary and its location and difference is not zero anymore.

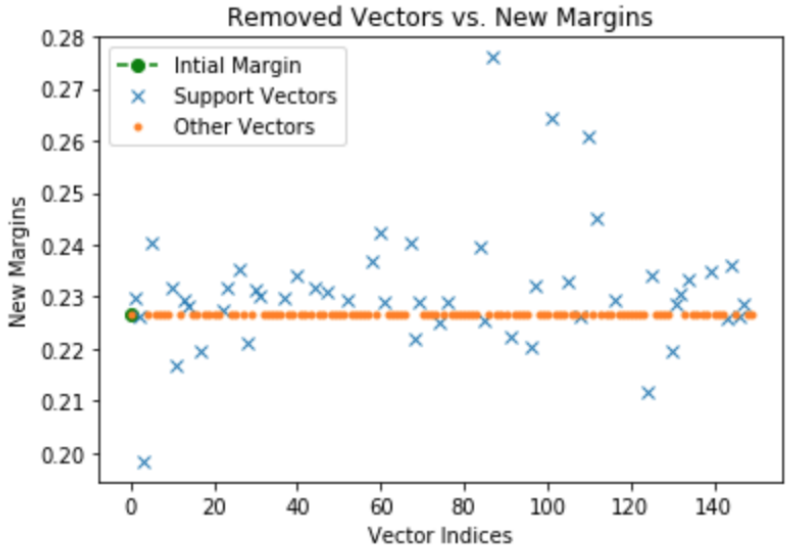


Figure 4.2: Margins before and after removing one data point

Conclusion2:
As we can see from graph when we removed a vector which is not a support vector margin does not change or change with a difference smaller than 0.00001. So removing non-support vector does not have an effect on margin. In contrast If we remove a vector which is a support vector margin changes drastically. It most of the time increases and in several occasions decreases.

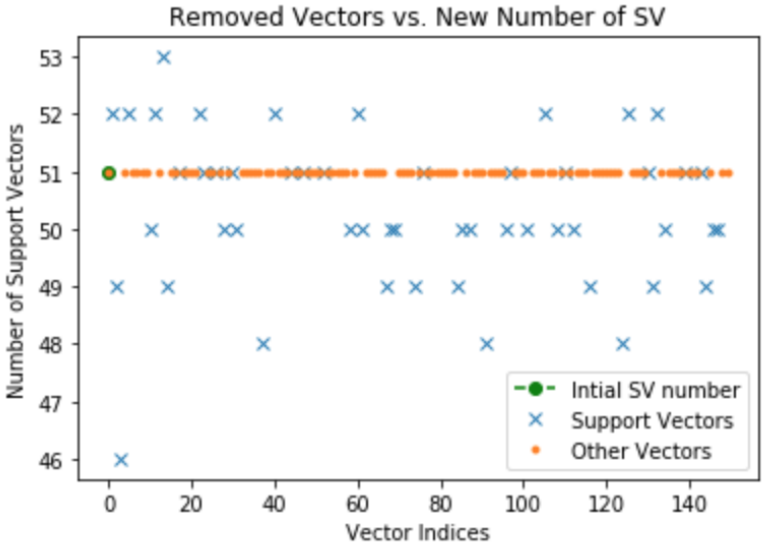


Figure 4.3: The number of suppor vectors before and after removing one data point

Conclusion3:
Our initial number of support vectors is 51 when we remove one data point which is not support vector then this number 51 does not change. In contrast when we remove one data point which is a support vector then number of support vectors changes. It may increase decrease or stay same depending on the support vector. This indicates that our hyperplane changes indeed.

5. Bonus

In bonus we implemented primal solution of hard margin SVM with CVXOPT. For this purpose we used the following matrices.

Q, P, A, c matrix forms for the QP solver:

$$Q = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}_{(d+1)x(d+1)} \quad p = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{(d+1)} \quad A = \begin{bmatrix} -y_1 & -y_1 x_1^T \\ -y_2 & -y_2 x_2^T \\ \vdots & \vdots \\ -y_N & -y_N x_N^T \end{bmatrix}_{Nx(d+1)} \quad c = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}_N$$

For the toy data set:

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3x3} \quad p = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_3 \quad A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 2 \\ -1 & -2 & 0 \\ -1 & -3 & 0 \end{bmatrix}_{4x3} \quad c = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}_4$$

We used solvers.qp function and took u vector ([bias, w1, w2]).

- bias: -1
- weights: [1, -1]
- margin: 0.707

The data points are estimated correctly.

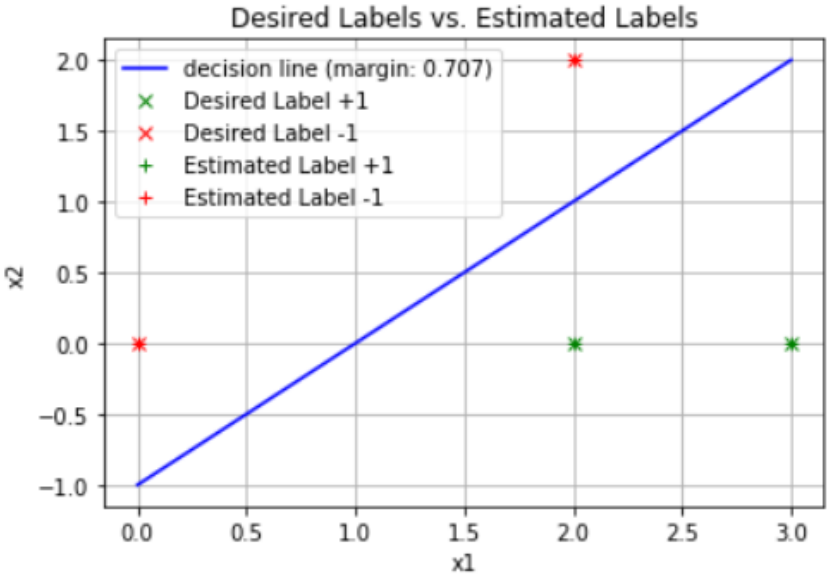


Figure 5.1: Desired and Estimated Points

Bibliography

[1] GitHub. 2020. Cjlin1/Libsvm. [online] Available at: <https://github.com/cjlin1/libsvm/tree/master/python> [Accessed 17 May 2020].

[2] Csie.ntu.edu.tw. 2020. LIBSVM – A Library For Support Vector Machines. [online] Available at: <https://www.csie.ntu.edu.tw/~cjlin/libsvm/> [Accessed 17 May 2020].

[3] Baytaş, İ., 2020. Support Vector Machines Slide.CMPE 462 Machine Learning Bogazici University.

[4] coefficients, M., 2020. Matlab Libsvm - How To Find The W Coefficients. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/10131385/matlab-libsvm-how-to-find-the-w-coefficients> [Accessed 17 May 2020].