**CS584**

**ASSIGNMENT 4**

**SUPPORT VECTOR MACHINE**

**Nivedita Kalele**

**A20329966**

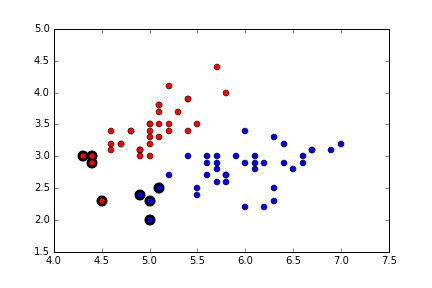
1. Generate a small 2D feature vectors of two classes such that classes are linearly separable:

For this, I have considered iris dataset. First 2 classes i.e. setosa and versicolor are linearly separable. To make dataset 2D only 2 features from this dataset are considered.

For non-linearly separable dataset, versicolor and virginica classes of iris dataset are considered.

1. Implement a linear SVM with hard margins on both datasets:

Support vector plot (linearly separable):



Accuracy for this algorithm is good.

Cvxopt package of python is used to find support vectors.

Output:

pcost dcost gap pres dres

0: -4.9681e-01 -5.9097e-01 1e+02 1e+01 1e+00

1: -3.0112e-02 -1.5904e-03 2e+00 2e-01 2e-02

2: -4.1519e-04 -8.6209e-04 3e-02 3e-03 3e-04

3: 2.5357e-05 -7.7168e-04 8e-04 9e-19 1e-15

4: -1.7980e-04 -2.6976e-04 9e-05 2e-20 4e-16

5: -2.1924e-04 -2.5076e-04 3e-05 3e-20 2e-16

6: -2.4057e-04 -2.4612e-04 6e-06 3e-20 3e-16

7: -2.4518e-04 -2.4524e-04 7e-08 5e-20 3e-16

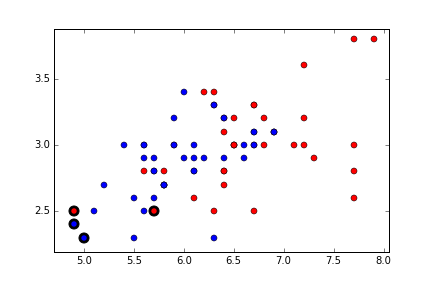
Optimal solution found.

8 support vectors out of 75 points

16 out of 25 predictions correct

Accuracy by my function 0.64

Support Vector plot (non-linearly separable):



Accuracy for this is very bad. This algorithm suffers badly.

Output:

pcost dcost gap pres dres

0: -5.3831e-01 -6.6686e-01 1e+02 1e+01 1e+00

1: -4.4484e-02 -2.0657e-03 2e+00 2e-01 2e-02

2: -1.1008e-03 -8.5918e-04 5e-02 5e-03 6e-04

3: 3.4474e-05 -6.8917e-04 8e-04 1e-05 1e-06

4: -1.4955e-04 -2.5658e-04 1e-04 2e-20 4e-16

5: -1.8615e-04 -2.2719e-04 4e-05 3e-20 3e-16

6: -1.9981e-04 -2.2677e-04 3e-05 8e-20 3e-16

7: -2.2124e-04 -2.2263e-04 1e-06 3e-20 3e-16

8: -2.2212e-04 -2.2213e-04 1e-08 9e-20 3e-16

Optimal solution found.

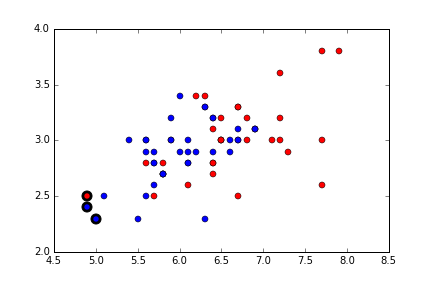
4 support vectors out of 75 points

14 out of 25 predictions correct

Accuracy by my function 0.56

1. Linear SVM algorithm with soft margin :

Support vector marked plot:



Output:

pcost dcost gap pres dres

0: -2.7291e-01 -7.8823e+00 2e+02 1e+01 4e-13

1: -4.9291e-02 -6.6868e+00 1e+01 4e-01 3e-13

2: -5.4765e-03 -6.5301e-01 8e-01 1e-02 6e-14

3: -2.4641e-07 -7.7935e-03 9e-03 1e-04 7e-15

4: -1.3743e-04 -4.5742e-04 3e-04 1e-06 4e-16

5: -1.8859e-04 -2.6029e-04 7e-05 3e-07 2e-16

6: -2.3110e-04 -2.5058e-04 2e-05 2e-08 2e-16

7: -2.3716e-04 -2.3880e-04 2e-06 2e-09 2e-16

8: -2.3798e-04 -2.3801e-04 3e-08 2e-11 3e-16

Optimal solution found.

3 support vectors out of 70 points

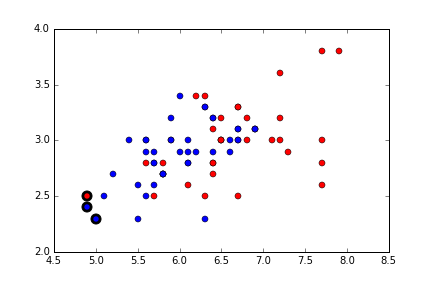
19 out of 30 predictions correct

Accuracy by my function 0.666666666667

By looking at this we can say that for non-linearly separable data algorithm with soft margin performs better than with hard margin.

1. Kernel-based SVM algorithm (Polynomial and Gaussian):

Support vector marked plot:



Output:

pcost dcost gap pres dres

0: -5.3831e-01 -6.6686e-01 1e+02 1e+01 1e+00

1: -4.4484e-02 -2.0657e-03 2e+00 2e-01 2e-02

2: -1.1008e-03 -8.5918e-04 5e-02 5e-03 6e-04

3: 3.4474e-05 -6.8917e-04 8e-04 1e-05 1e-06

4: -1.4955e-04 -2.5658e-04 1e-04 2e-20 4e-16

5: -1.8615e-04 -2.2719e-04 4e-05 3e-20 3e-16

6: -1.9981e-04 -2.2677e-04 3e-05 8e-20 3e-16

7: -2.2124e-04 -2.2263e-04 1e-06 3e-20 3e-16

8: -2.2212e-04 -2.2213e-04 1e-08 9e-20 3e-16

Optimal solution found.

3 support vectors out of 75 points

17 out of 25 predictions correct

Accuracy by my function 0.68

This accuracy is for Gaussian kernel. To look at the accuracy of Polynomial kernel, change the Gaussian variable to False.

For Gaussian Sigma is kept as 5 and for polynomial order is kept as 2.

**Conclusion**:

Hard-margin for linearly-separable data works very well. And for non-separable data soft-margin works well. Also for non-linearly separable data, kernel-based SVM works well. Comparison is done based on the accuracies obtained after implementing these algorithm.