CS 584: Machine Learning
Assignment-4
Support Vector Machine

Name: Priyank P Shah Date:04/23/2016
CWID: A20344797 Term:Spring'16

1. Problem Statement :

- SVM(Support Vector Machine) is a discriminative classifier to identify correct class by separating hyperplane. It is based on a technique that is kernel trick to transform data and then from the transformation is finds an optimum boundary between outputs and classify them.
- In this assignment SVM is performed with Hard margin, Soft margin classification and based on Kernel functions. To be precise Polynomial and Gaussian Kernel functions are used.
- I have generated Separable datasets as well as Non-Separable datasets for Hard margin, Soft margin and Kernel Functions. I have also implemented Kernel functions for external existing dataset.
- To testify the result I have implemented it with the 10 Fold Cross validation too.

2. Proposed Solution:

- I have proposed solution by implementing the task suggested in assignments.
- First of all I have generated different datasets which contains separable and non-separable data.
- I have implemented SVM with the Hard Margin for both Separable and non-separable datasets.
- I have implemented SVM with the Soft margin for both separable and non-separable datasets.
- I have implemented kernel based SVM for generated datasets as well as external datasets.
- In order to find the value of w, w0 and support vector, I have used cvxopt – QP solver. Based on the result of QP solver I have check different condition for hard margin and soft margin.
- For better results and observations I have generated dataset with various sizes and randomized them.
- For the 6th task, I have generated dataset with one class having more examples than other class and observed the result, which I am discuss in Result and observations.
- My work resides in 4 different files named as, HardMargin.ipynb ; Kernel.ipynb ; Kernel Method-External.ipynb and SoftMargin.ipynb.

Algorithm:

Hard Margin:

- Generated Dataset which is either separable or non-separable datasets, using np.uniform and np.random method respectively.
- Derived training and testing datasets.
- Derived W0, W and Support vectore by generating various matrix required by qp solver(P,q,G,h,A,b) from training data and training class.

- Find Support vector when alpha equals .
- find weight w for feature vector x. find w0 for support vectors.
- Projected model classified using linear kernel.
- Predicted class value and computed confusion matrix, precision, f1 measure,a accuracy and recall.

Soft Margin:

- Generated Dataset which is either separable or non-separable datasets, using np.uniform and np.random method respectively.
- Derived training and testing datasets.
- Derived W0, W and Support vector by generating various matrix required by qp solver(P,q,G,h,A,b) from training data and training class. Here dimension value of G and h matrix change as panelty is added in h matrix.
- Find Support vector when alpha is inbetween 0 and C(penalty).
- find weight w for feature vector x. find w0 for support vectors.
- Projected model classified using linear kernel.

  Predicted class value and computed confusion matrix, precision, f1 measure,a accuracy and recall.

Kernel Based SVM:

- Generated Dataset which is either separable or non-separable datasets, using np.uniform and np.random method respectively.
- Derived training and testing datasets.
- Derived W0, W and Support vector by generating various matrix required by qp solver(P,q,G,h,A,b) from training data and training class. While calculating matrix P , I have used Gaussian kernel / polynomial kernel at a time in place of X.Xtrans
- Find Support vector when alpha more than 0.
- find weight w for feature vector x. find w0 for support vectors.
- Projected model classified using linear kernel.
- Predicted class value and computed confusion matrix, precision, f1 measure,a accuracy and recall.
- In missing value dataset, I have used imputer to overcome the the difficulty.

3. Implementation Details:

Instructions:

- Before running Kernel SVM for External dataset, put the datasets into the same folder where the program file resides. I have provided dataset.
- I have developed program in Jupyter which uses format of python notebook. Run the file from any location.
- There are additional dependencies which covers cvxopt and matlibplot for respective work of qp solver to generate W, W0 and support_vector values and Plotting results on graphs.

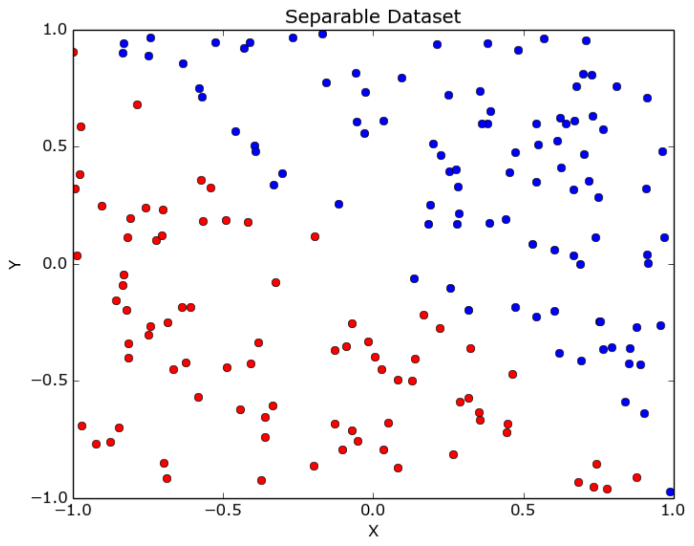
- I have commented the the Non-Separable dataset generation files. By defualt it will generate separable dataset. To check with non-separable dataset comment separable dataset and run.
- Same way in Kernel based SVM implementation I have Commented one kernel. So to check appropriate kernel, comment other one.

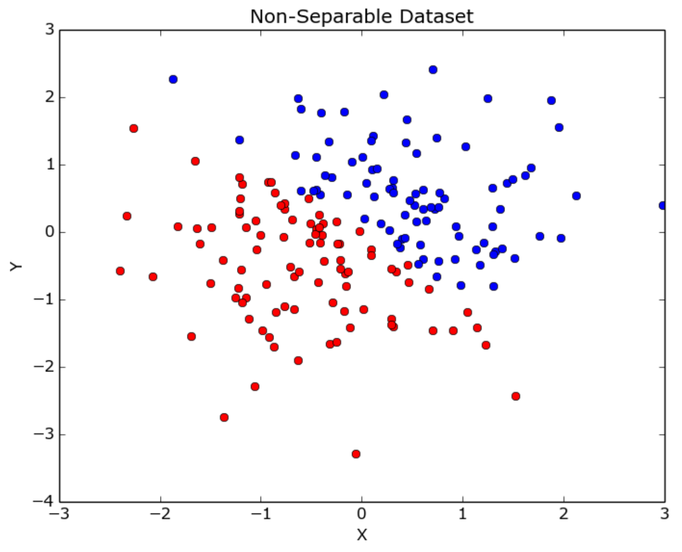
Design Issues:

- installing additional dependency was bit challenging with pycharm IDE, hence I switched to Jupyter.
- Plotting Kernel SVM results on graph and drawing decision boundary was challenging.

4. Results and Discussion:

Note: red marker defines values from class -1 and blue defined value from class 1. Yellow markers are support vectors.





1. Hard-Margin SVM

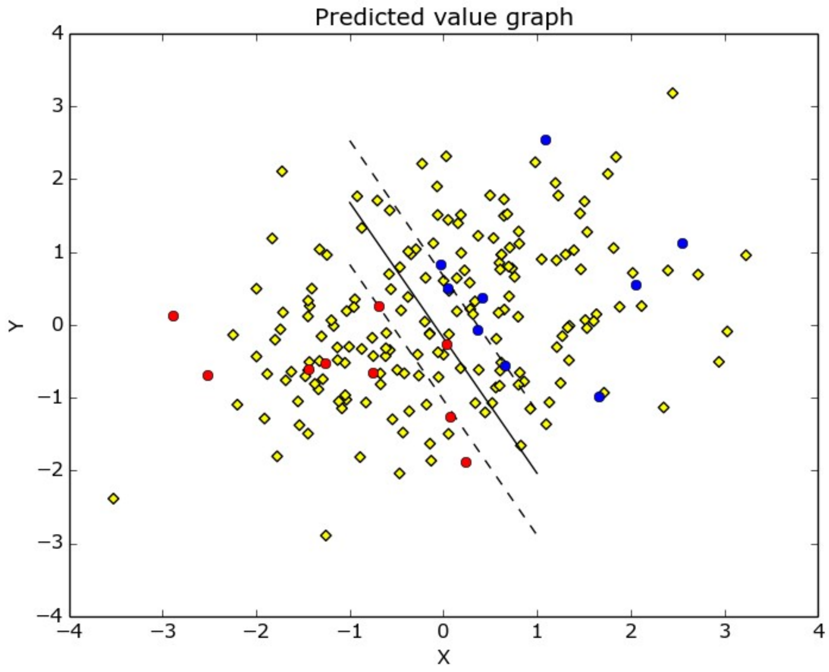
Dataset: Separable

Result:

	Precision	Recall	F1 Measures
Class -1	1.0	1.0	1.0
Class 1	1.0	1.0	1.0

Accuracy: 100.00 %

Confusion Matrix: $\begin{bmatrix} 11 & 0 \\ 0 & 9 \end{bmatrix}$



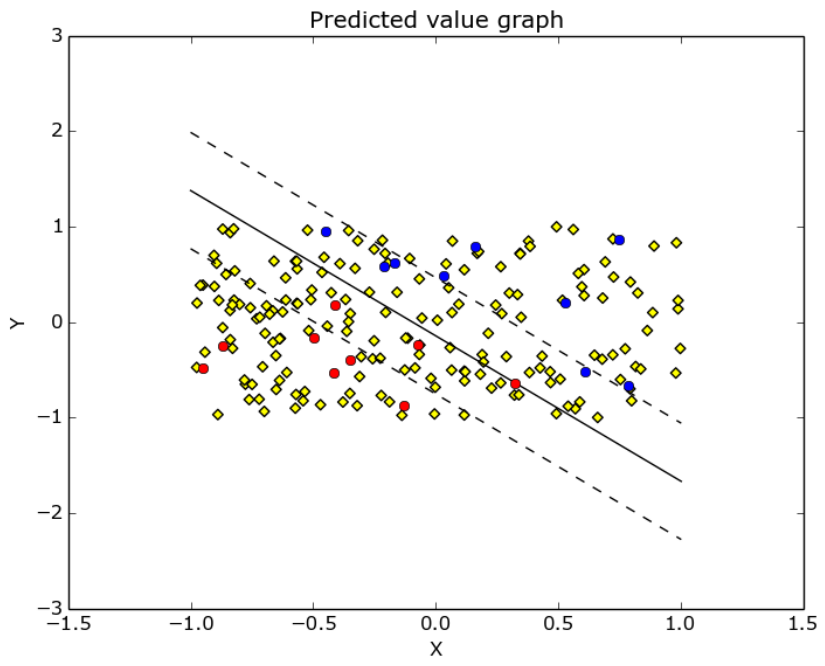
Dataset: Non-Separable

Result:

	Precision	Recall	F1 Measures
Class -1	0.857	1.0	0.923
Class 1	1.0	0.929	0.963

Accuracy: 95.00 %

Confusion Matrix: $\begin{bmatrix} 6 & 0 \\ 1 & 13 \end{bmatrix}$



2. Soft-Margin SVM

Dataset: Separable

Result:

	Precision	Recall	F1 Measures
Class -1	0.8	1.0	0.889
Class 1	1.0	0.833	0.909

Accuracy: 90.00 %

Confusion Matrix: $\begin{bmatrix} 8 & 0 \\ 2 & 10 \end{bmatrix}$

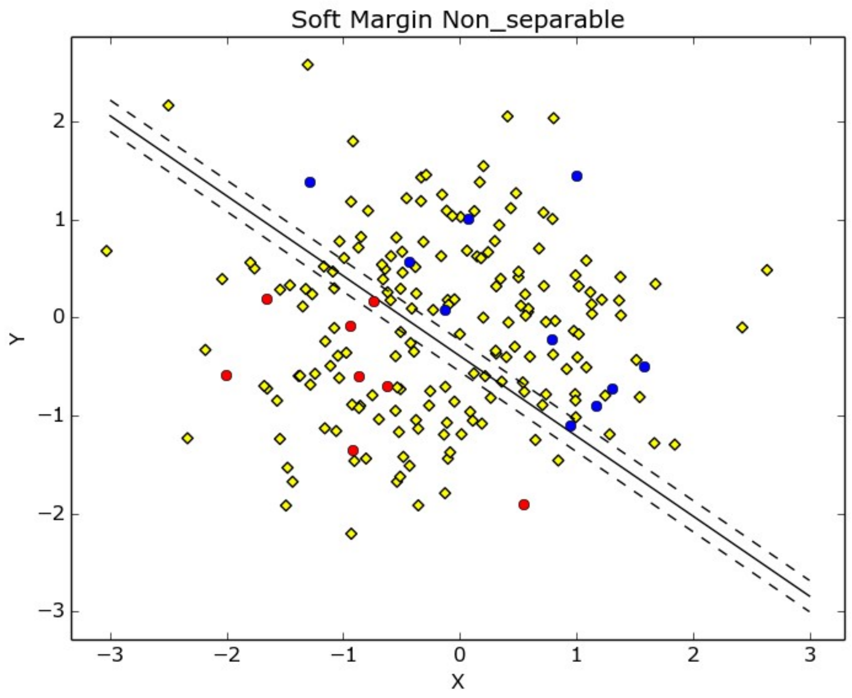
Dataset: Non - Separable

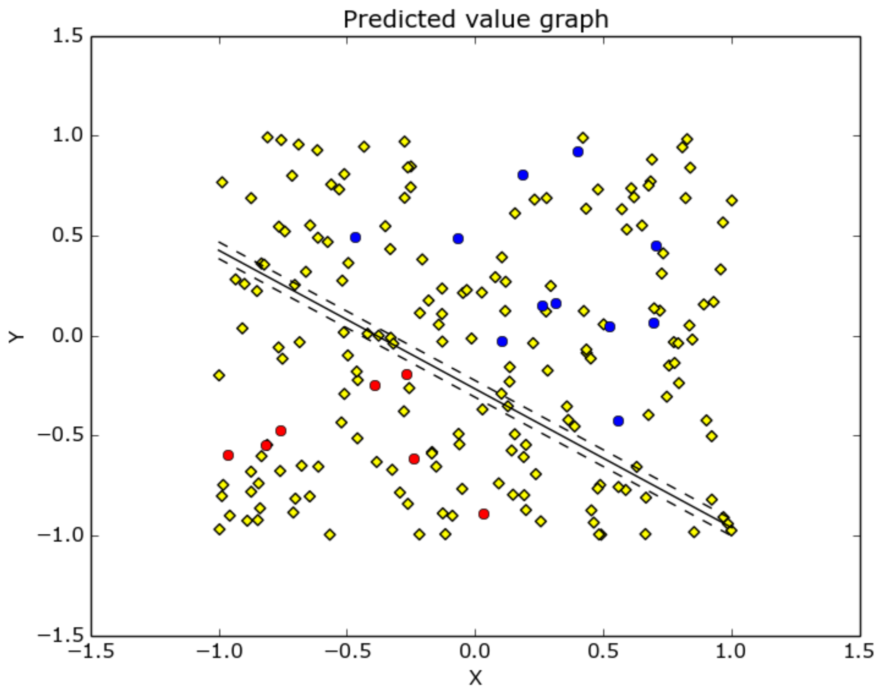
Result:

	Precision	Recall	F1 Measures
Class -1	1.0	0.818	0.9
Class 1	0.818	1.0	0.9

Accuracy: 90.00 %

Confusion Matrix: $\begin{bmatrix} 9 & 2 \\ 0 & 9 \end{bmatrix}$


Soft Margin Non_separable
Y
X
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3


Predicted value graph
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

2. Kernel Based SVM

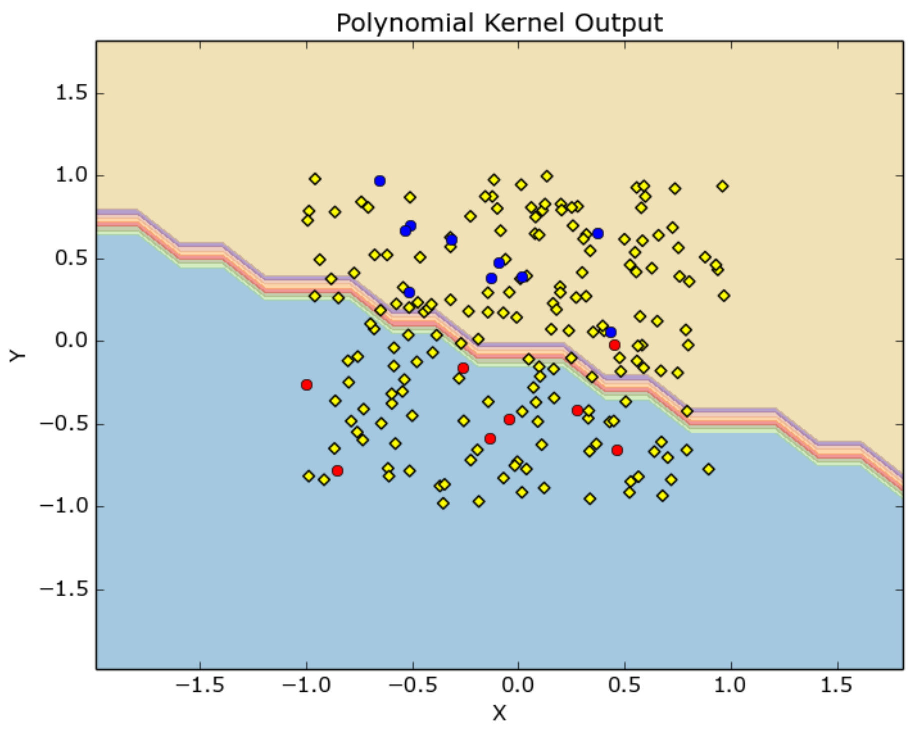
Kernel: Polynomial

Result:

	Precision	Recall	F1 Measures
Class -1	0.889	0.889	0.889
Class 1	0.909	0.909	0.909

Accuracy: 90.00 %

Confusion Matrix: [8 1]
[1 10]



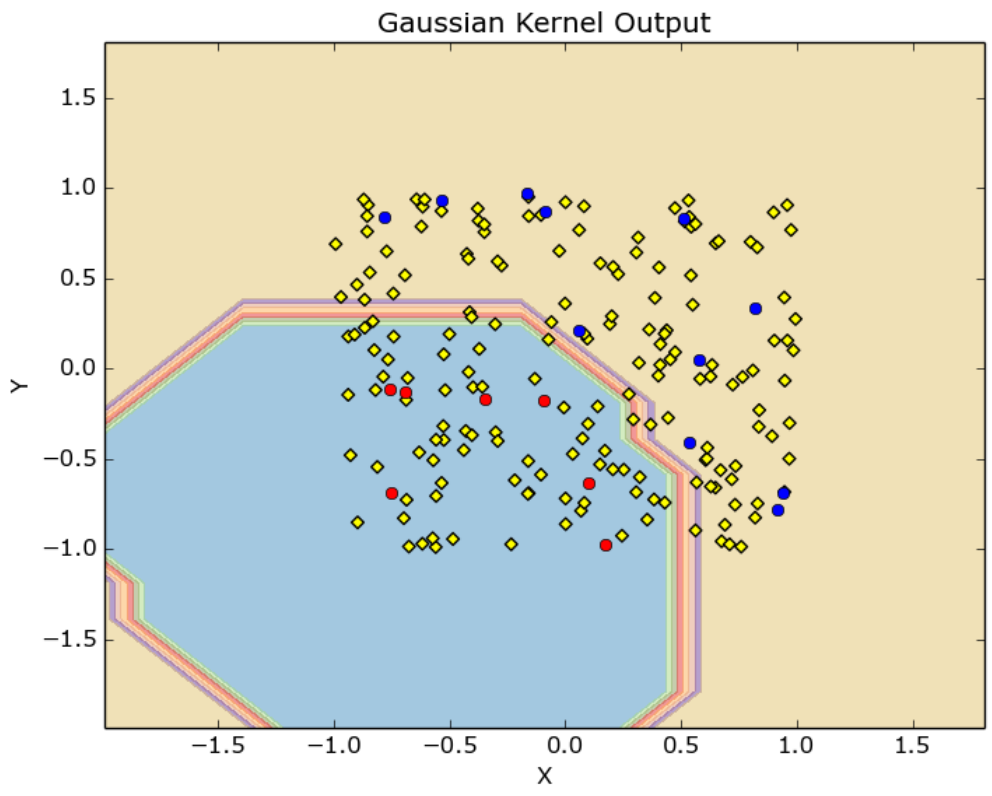
Kernel: Gaussian Kernel

Result:

	Precision	Recall	F1 Measures
Class -1	1.0	1.0	1.0
Class 1	1.0	1.0	1.0

Accuracy: 100.00 %

Confusion Matrix: [8   0]
                    [0 12]

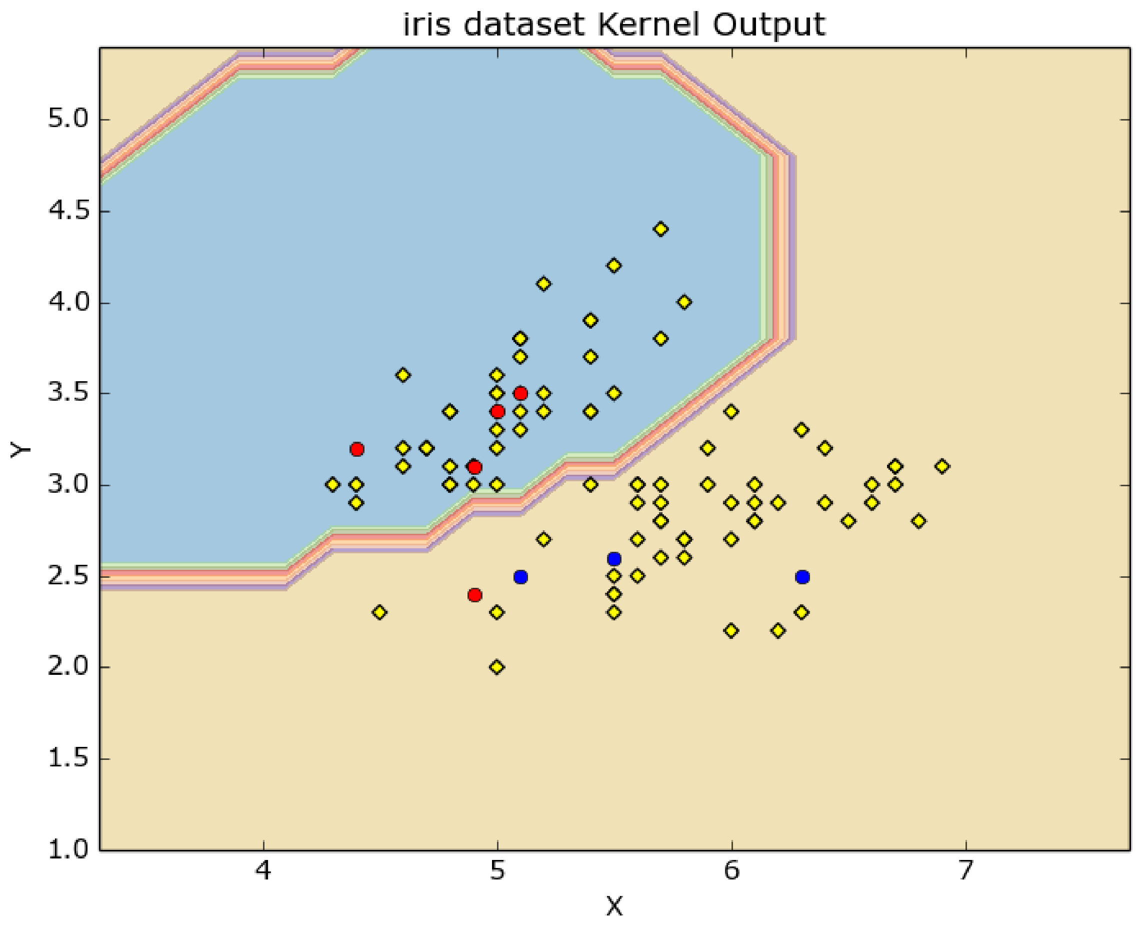


Kernel: Gaussian
Dataset: iris.data (Separable – feaure 0 and 1)
Result:

	Precision	Recall	F1 Measures
Class -1	0.833	1.0	0.909
Class 1	1.0	0.8	0.889

Accuracy: 90.00 %

Confusion Matrix: [5 0]
                    [1 4]

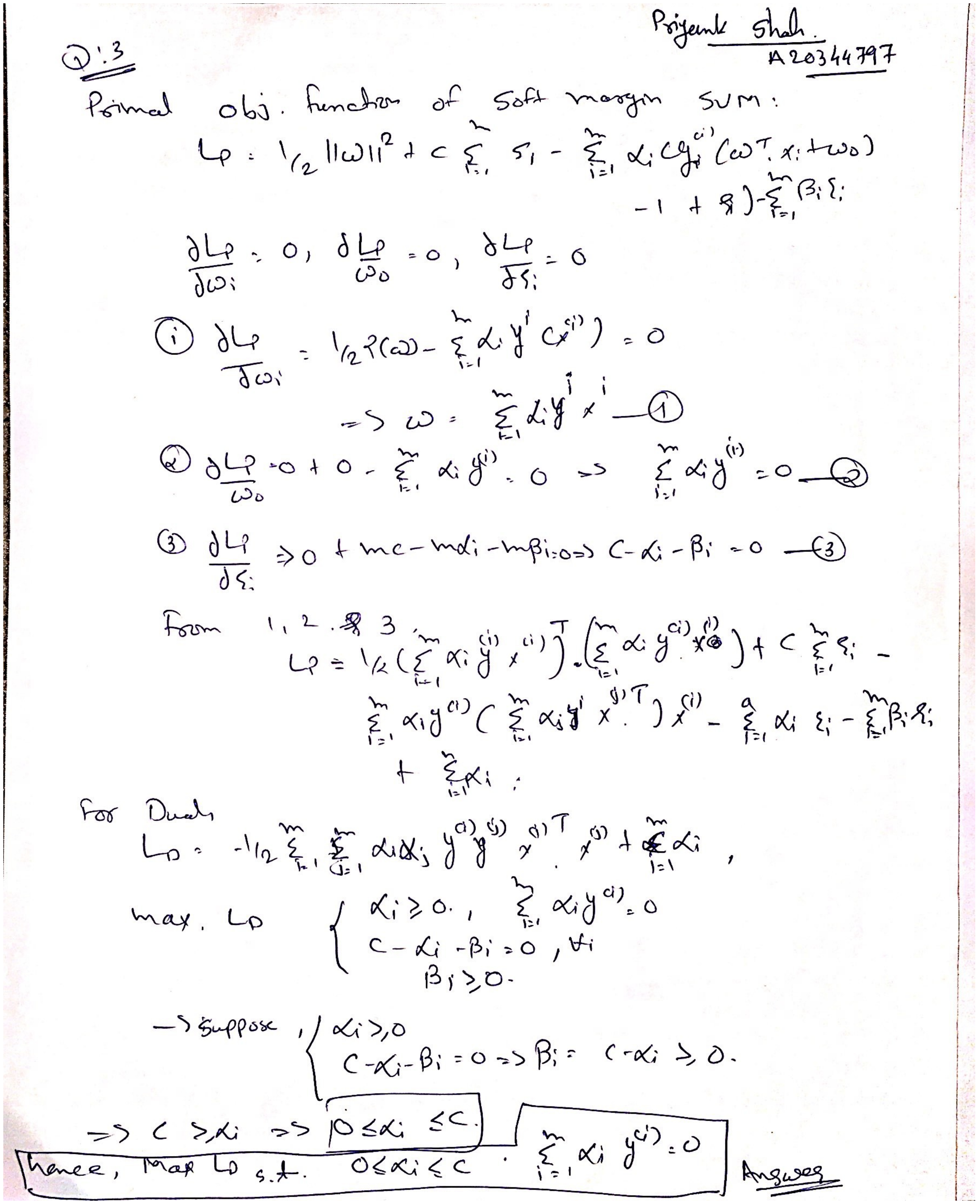


Note:

1.I took the reference from below mentioned website for graph plotting.

2. resultant accuracy varies between 90-100% for all the SVM implementation

Additional Questions:



Priyank Shah.
A20344797

Q:3

Primal obj. function of soft margin SVM:

$$L_P : \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i (y^{(i)}(\omega^T . x_i + \omega_0) - 1 + \xi_i) - \sum_{i=1}^{m} \beta_i \xi_i$$

$$\frac{\partial L_P}{\partial \omega_i} = 0, \quad \frac{\partial L_P}{\omega_0} = 0, \quad \frac{\partial L_P}{\partial \xi_i} = 0$$

(i) $\frac{\partial L_P}{\partial \omega_i} = \frac{1}{2} 2(\omega) - \sum_{i=1}^{m} \alpha_i y^{i} (x^{(i)}) = 0$

$$\Rightarrow \omega = \sum_{i=1}^{m} \alpha_i y^{i} x^{i} \text{ — } (1)$$

(2) $\frac{\partial L_P}{\omega_0} = 0 + 0 - \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ — (2)

(3) $\frac{\partial L_P}{\partial \xi_i} \Rightarrow 0 + mc - m\alpha_i - m\beta_i = 0 \Rightarrow C - \alpha_i - \beta_i = 0$ — (3)

From 1, 2 & 3

$$L_P = \frac{1}{2}\left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}\right)^T . \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}\right) + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i y^{(i)} \left(\sum_{i=1}^{m} \alpha_j y^{j} x^{(j)T}\right) x^{(i)} - \sum_{i=1}^{a} \alpha_i \xi_i - \sum_{i=1}^{m} \beta_i \xi_i + \sum_{i=1}^{m} \alpha_i$$

For Dual

$$L_D = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)T} . x^{(j)} + \sum_{i=1}^{m} \alpha_i ,$$

$$\max . L_D \quad \begin{cases} \alpha_i \geq 0 , & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \\ C - \alpha_i - \beta_i = 0 , \forall i \\ \beta_i \geq 0 . \end{cases}$$

$$\rightarrow \text{Suppose ,} \begin{cases} \alpha_i \geq 0 \\ C - \alpha_i - \beta_i = 0 \Rightarrow \beta_i = C - \alpha_i \geq 0 . \end{cases}$$

$$\Rightarrow C \geq \alpha_i \Rightarrow \boxed{0 \leq \alpha_i \leq C .}$$

$$\boxed{\text{hence, Max } L_D \text{ s.t. } 0 \leq \alpha_i \leq C \cdot \sum_{i=1}^{m} \alpha_i y^{(i)} = 0}$$

Answer

Q:6

Yes, there is a difference in result and accuracy. Reason is there will be significant less data of one class in training which results into bad training of parameter and hence will produce wrong result on testing dataset.

In nutshell, Accuracy decreased and predicted wrong result.

5. Reference:

- `http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html`
- `https://www.quora.com/What-are-Kernels-in-Machine-Learning-and-SVM`
- `http://www.support-vector.net/icml-tutorial.pdf`
- `http://stackoverflow.com/questions/4232455/numpy-load-raises-attributeerror-module-object-has-no-attribute-expr`