CS-584 MACHINE LEARNING

ASSIGNMENT – 4

SUPPORT VECTOR MACHINE

By

JAIMIN SANGHVI(A20344798)

PROBLEM STATEMENT

- A **Support Vector Machine** (**SVM**) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.
- It uses a technique called kernel trick to transform data and then based on these transformations it finds an optimal boundary between possible outputs.
- In simple words, it does extremely complex data transformations, then figures out how to separate your data based on labels or output we have defined.
- In this assignment, we will be focusing on linear (Hard-Soft Margin) SVN as well as non-linear SVM using kernel function.
- Non linear means a boundary that algorithm calculates does not have to be a straight line.
- In the given assignment4, We have to implement an algorithm for support vector machine.
- In addition, We have to generate linearly separable and non linear separable dataset for classification
- To improve high accuracy and performance, I should evaluate an algorithm using K-fold validation.

PROPOSED SOLUTION

- In the given assignment, I have proposed solution in different manners and evaluated model for different datasets.
- The very first task is to generate two datasets: Linearly separable and Non linearly separable
- Next, I have implemented support vector machine algorithm for hard margin and soft margin techniques. As instructed in lecture, I have used QP-Solver libraries to find alpha training data
- I have evaluated an algorithm with K-fold cross validation. In addition, I have changed randomly generated dataset size and data type to evaluate implemented model.
- I have evaluated SVM hard margin on both linearly separable and non linearly separable datasets.
- To improve the results for non linearly separable dataset, I have approached to implement kernel based SVM algorithm using a polynomial and Gaussian kernel function
- To provide solution for last task, I have generated dataset with two classes in which one of the classes has substantially more examples.
- I have implemented three separate program files: 1) for Hard and

Soft margin SVM[HARD_SOFT_MARGIN.py], 2) for Polynomial Kernel SVM[POLY_KERNEL.py] and 3) for Gaussian Kernel SVM [GAUSSIAN_KERNEL.py]

How algorithm works

- I have implemented support vector machine for hard margin in the following steps:
  - ✔ Generate dataset according to given parameter [Linearly separable or Non Linearly separable]
  - ✔ Distinguish training and training data according to k-fold
  - ✔ Calculate parameters for QP-Solver library to find alpha
  - ✔ Find support vector when alpha equals 0
  - ✔ Find weigh w for feature vector X
  - ✔ Find w0 only for support vector sv_X
  - ✔ Project model classifier using linear kernel
  - ✔ Predict y and compute confusion matrix
  - ✔ Compute accuracy, precision, recall and f-measure errors

- I have implemented support vector machine for soft margin in the following steps:
  - ✔ Generate dataset according to given parameter [Linearly separable or Non Linearly separable]
  - ✔ Distinguish training and training data according to k-fold
  - ✔ Calculate parameters for QP-Solver library to find alpha. Do not forget to calculate using penalty C
  - ✔ Find support vector when alpha equals 0 <= alpha <= C
  - ✔ Find weigh w for feature vector X
  - ✔ Find w0 only for support vector sv_X
  - ✔ Project model classifier using linear kernel
  - ✔ Predict y and compute confusion matrix
  - ✔ Compute accuracy, precision, recall and f-measure errors

- I have implemented support vector machine for polynomial-Gaussian kernel in the following steps:
  - ✔ Generate dataset according to given parameter [Linearly separable or Non Linearly separable]
  - ✔ Distinguish training and training data according to k-fold
  - ✔ Calculate parameters for QP-Solver library to find alpha. I have used polynomial and Gaussian function to find P
  - ✔ Find support vector
  - ✔ Find weigh w for feature vector X
  - ✔ Find w0 only for support vector sv_X
  - ✔ Project model classifier using polynomial and Gaussian function
  - ✔ Predict y and compute confusion matrix

- ✔ Compute accuracy, precision, recall and f-measure errors

- In addition, I have evaluated generated data set using Scikit learn libraries and compare it with derived training and testing errors and accuracy

IMPLEMENTATION DETAILS

- I have implemented SVM model algorithms in PyCharm IDE. An implementation consists dynamic argument to run algorithm on different dataset with different parameters
- I have generated linearly separable and non-linearly separable dataset using randomly generating numbers (np.random library)
- The generated dataset is used to classify SVM models
- The given implementations are data-oriented. Hence, User may require necessary changes to run same model for different datasets and parameters
- I have implemented model for K-folds to evaluate model with different fold and achieve best accuracy. In addition, I have calculated confusion matrix as well as precision, recall and f-measure errors
- As per given instruction, I have evaluated my model with two different external datasets named as Iris dataset, Dermitology dataset
- Proble solution for question #3 is attached at the end of document.

DESIGN ISSUES

- It is little bit challenging to implement matrix multiplication without using any library
- Understanding of kernel function is quite difficult
- Calculation of alpha and manage dimension for each matrix
- I do not have idea about data and support vector points plots
- Plotting of decision boundary causes too many errors. It takes quite more time to plot properly.

SOLUTION

- To make mathematical calculations simple, I have used numpy library
- I went through lecture notes and on line stuffs which helped me to understand kernel functions
- I have used QP-Solver library to calculate alpha within few steps
- I went through on line-stuff and learn plotting parameters conceptually which helped me to plot linear-non linear data with linear, polynomial and Gaussian kernel function

INSTRUCTION TO RUN

- I have implemented given problem solution in PyCharm IDE.
- Instruction to run given project file
  i. Load given project in PyCharm IDE
  ii. Run **HARD_MARGIN.py** file for Hard margin support vector classification. During .py execution, it requires some dynamic arguments. The argument instruction is:
    * Please enter type of data: [linear, non-linear] – To choose required dataset type
    * Please enter number of example: [100] – To choose required dataset examples
    * Please enter type of SVM margin : [hard, soft] – To choose SVM margin type
  i. Run **SOFT_MARGIN.py** file for soft margin support vector classification. During .py execution, it requires some dynamic arguments. The argument instruction is:
    * Please enter type of data: [linear, non-linear] – To choose required dataset type
    * Please enter number of example: [100] – To choose required dataset examples
    * Please enter type of SVM margin : [hard, soft] – To choose SVM margin type
  i. Run **POLY_KERNEL.py** file for support vector classification using polynomial kernel based algorithm. During .py execution, it requires some dynamic arguments. The argument instruction is:
    * Please enter type of data: [linear, non-linear] – To choose required dataset type
    * Please enter number of example: [100] – To choose required dataset examples
  i. Run **GAUSSIAN_KERNEL.py** file for support vector classification using Gaussian kernel based algorithm. During .py execution, it requires some dynamic arguments. The argument instruction is:
    * Please enter type of data: [linear, non-linear] – To choose required dataset type
    * Please enter number of example: [100] – To choose required dataset examples

RESULTS AND DISCUSSION

SUPPORT VECTOR MACHINE

Algorithm (Hard-Soft Margin, Kernel based SVM model)

Problem Statement
Find W and W0 to maximize quadratic margin

Steps to solve a Problem
1. Solve the dual for alpha
As per instructed in lecture, I have used cvxopt library to find α

2. Find Support Vector set
To find support vectors, Compare alpha value with 0 for hard margin and with C value for soft margin

For hard margin: support vector is when α > 0
For soft margin: support vector is when 0 <= α < C

3. Compute W and W0
Find W weight factor & W0 intercept point
where:

$$W = \sum_{i=1}^{m} \alpha_i \, y^i x^i$$

$$W0 = \frac{1}{\#sv} \sum_{x_i \in sv} y^i - \left(w^T x^i\right)$$

Classify Test dataset
For given X
For hard margin, If $W^T X + W0 > 0$, then c
For soft margin, If $W^T X + W0 > 1 - S_i$, then c
where, 0<S<=1

Kernel Function

For kernel function, I have used polynomial and Gaussian function in place of linear function,
Polynomial Kernel: $K\left(X^i X^j\right) = \left(X^{i^T} X^j + 1\right)^q$
Gaussian Kernel: $exp\left(\frac{1}{\gamma}\left(X^i - X^j\right)\left(X^i - X^j\right)\right)$

RESULTS

Hard-Margin Support Vector Machine [LINEARLY SEPERABLE DATASET]

On IRIS dataset
Accuracy = 98.8333 %

On Generated Dataset

Accuracy = 97.1428571429 %
Confusion Matrix = [[0 1]
[0 34]]
Precision = [0.0, 1.0]
Recall = [0, 0.97142857142857142]
F-Measure [0, 0.98550724637681153]

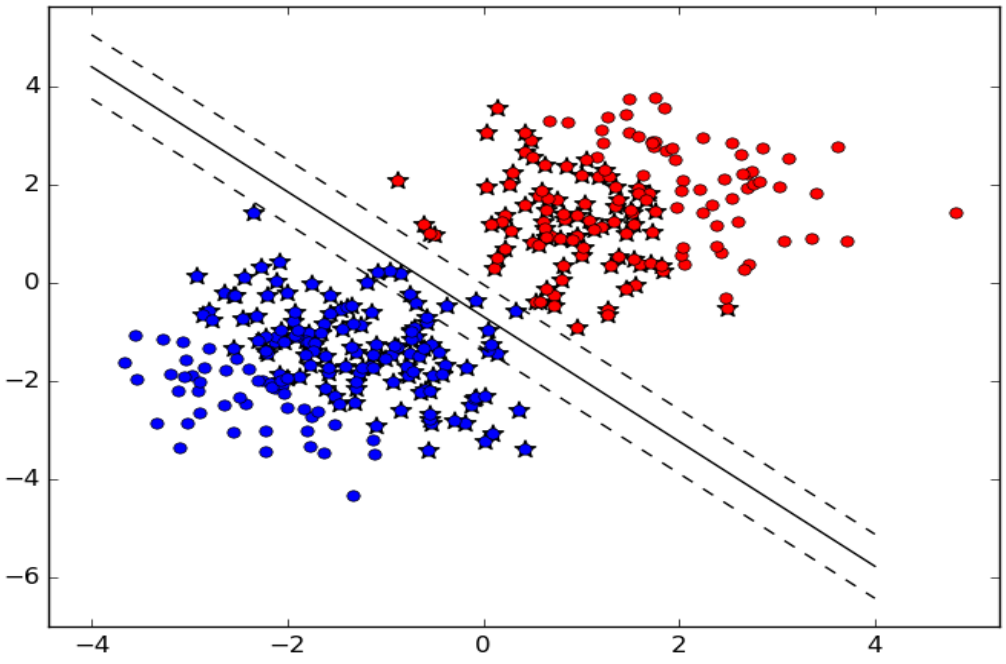

Fig 1. LINEARLY SEPERABLE DATASET WITH HARD MARGIN

Hard-Margin Support Vector Machine [LINEARLY NON-SEPERABLE DATASET]

On IRIS dataset

```
Accuracy =  90.0 %
Confusion Matrix : [[9 1]
                    [0 0]]
Precision =  [0.90000000000000002, 0.0]
Recall =  [1.0, 0.0]
F-Measure [0.94736842105263164, 0]
```

On Generated Dataset

```
Accuracy =  62.8571428571 %
Confusion Matrix : [[22  0]
                    [13  0]]
Precision =  [1.0, 0.0]
Recall =  [0.62857142857142856, 0]
F-Measure [0.77192982456140347, 0]
```

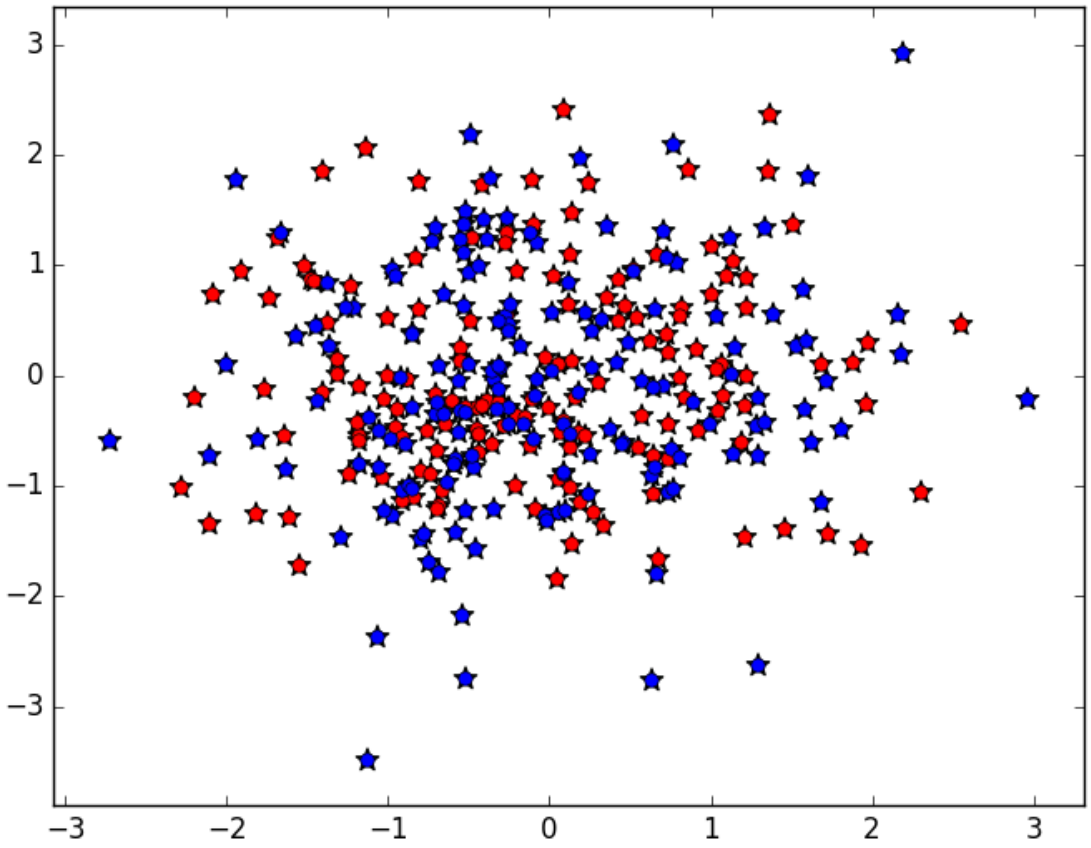

Fig 2. LINEARLY NON SEPERABLE DATASET WITH HARD MARGIN

In above graph, I have plotted generated data point. In addition, I have plotted support vector points with (*) marker. We have calculated support vector point from feature matrix hence it will be overlapped by data points.

CONSLUSION:
The above plots concludes that the model gives better accuracy for linearly separable dataset compared to linearly non separable datasets

Soft-Margin Support Vector Machine [LINEARLY SEPERABLE DATASET]

On IRIS dataset
Accuracy = 98.8333 %

On Generated Dataset
Accuracy = 97.1428571429 %

Confusion Matrix : [[1 0]
[1 33]]
Precision = [1.0, 0.97058823529411764]
Recall = [0.5, 1.0]
F-Measure [0.66666666666666663, 0.9850746268656716]

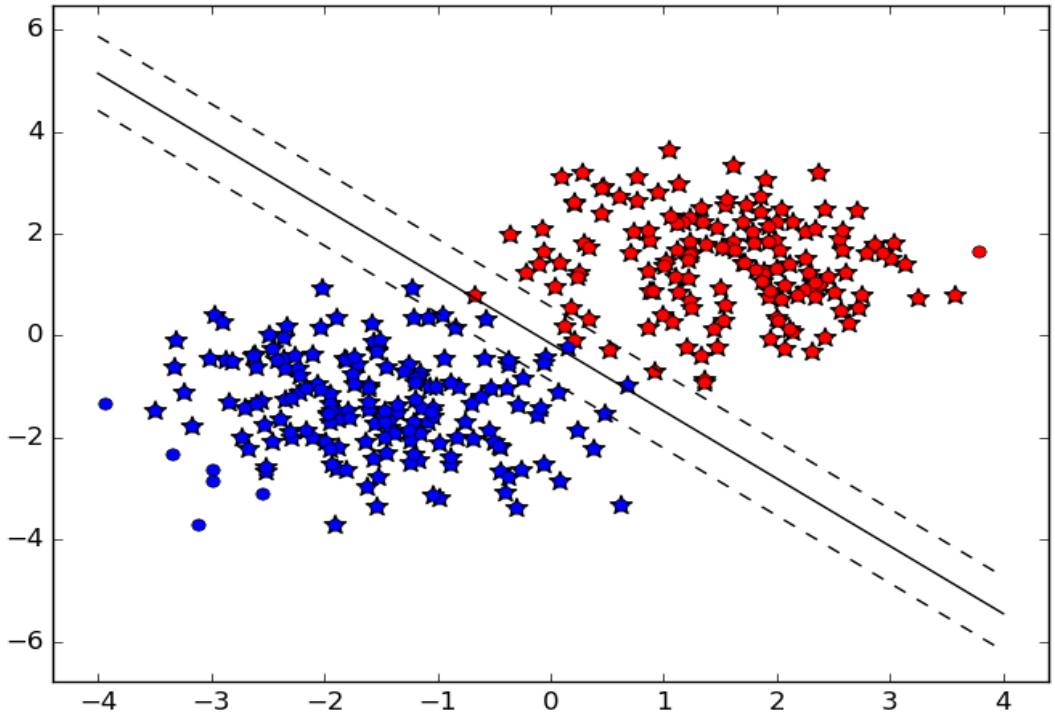

Fig 3. LINEARLY SEPERABLE DATASET WITH SOFT MARGIN

Soft-Margin Support Vector Machine [LINEARLY NON SEPERABLE DATASET]

On IRIS dataset

```
Accuracy =  80.0 %
Confusion Matrix :  [[8 2]
                     [0 0]]
Precision =  [0.80000000000000004, 0.0]
Recall =  [1.0, 0.0]
F-Measure [0.88888888888888895, 0]
```

On Generated Dataset

```
Accuracy =  65.7142857143 %
Confusion Matrix : [[21  0]
                    [12  2]]
Precision =  [1.0, 0.14285714285714285]
Recall =  [0.63636363636363635, 1.0]
F-Measure [0.77777777777777779, 0.25]
```

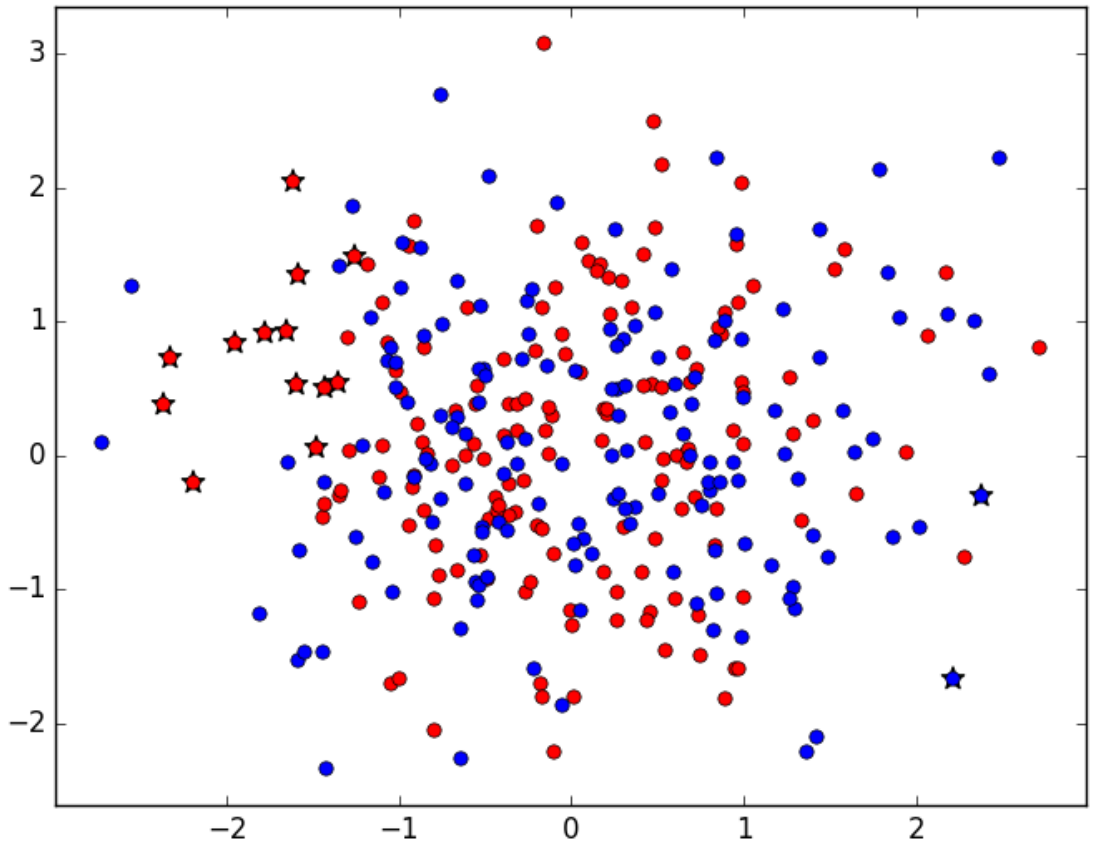

Fig 4. LINEARLY NON SEPERABLE DATASET WITH SOFT MARGIN

In above graphs, I have plotted generated data point. In addition, I have plotted support vector points with (*) marker. We have calculated support vector point from feature matrix hence it will be overlapped by data points.

CONCLUSION:

After evaluating implemented model on randomly generated datasets, I come to conclude that soft margin and hard margin both work better on linearly separated datasets. In addition, soft margin performs better than hard margin in some cases because of noise in data.

I expect soft-margin SVM to be better even when training dataset is linearly separable. The reason is that in a hard-margin SVM, a single outlier can determine the boundary, which makes the classifier overly sensitive to noise in the data.

The whole evaluation come to conclusion that Hard Margin SVM overfits to a particular dataset hence it can be not generalized. Even in a linearly separable dataset, outliers well within the boundaries can effects the margin.

Soft Margin SVM has more versatility because I have control over choosing the support vectors by tweaking the C.

Polynomial based kernel Support Vector Machine [LINEARLY SEPERABLE DATASET]

On IRIS dataset
Accuracy = 94.9333 %

On Generated Dataset
Accuracy = 88.5714285714 %
Confusion Matrix : [[30 0]
[4 1]]
Precision = [1.0, 0.20000000000000001]
Recall = [0.88235294117647056, 1.0]
F-Measure [0.9375, 0.33333333333333337]

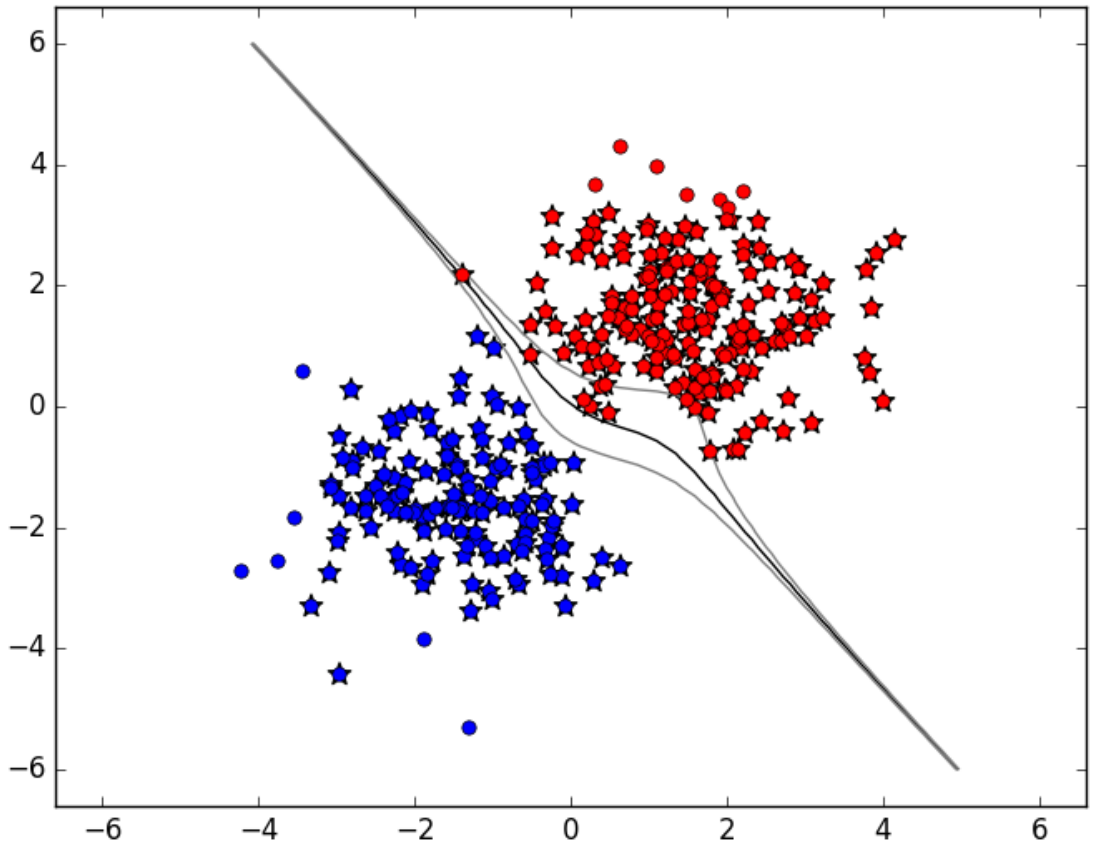

Fig 5. LINEARLY SEPERABLE DATASET WITH POLYNOMIAL KERNEL BASED SVM

Polynomial based kernel Support Vector Machine [LINEARLY NON SEPERABLE DATASET]

On IRIS dataset
Accuracy = 88.9333 %

On generated dataset
Accuracy = 68.5714285714 %
Confusion Matrix : [[22 1]
[10 2]]
Precision = [0.95652173913043481, 0.16666666666666666]
Recall = [0.6875, 0.66666666666666663]
F-Measure [0.80000000000000004, 0.26666666666666666]

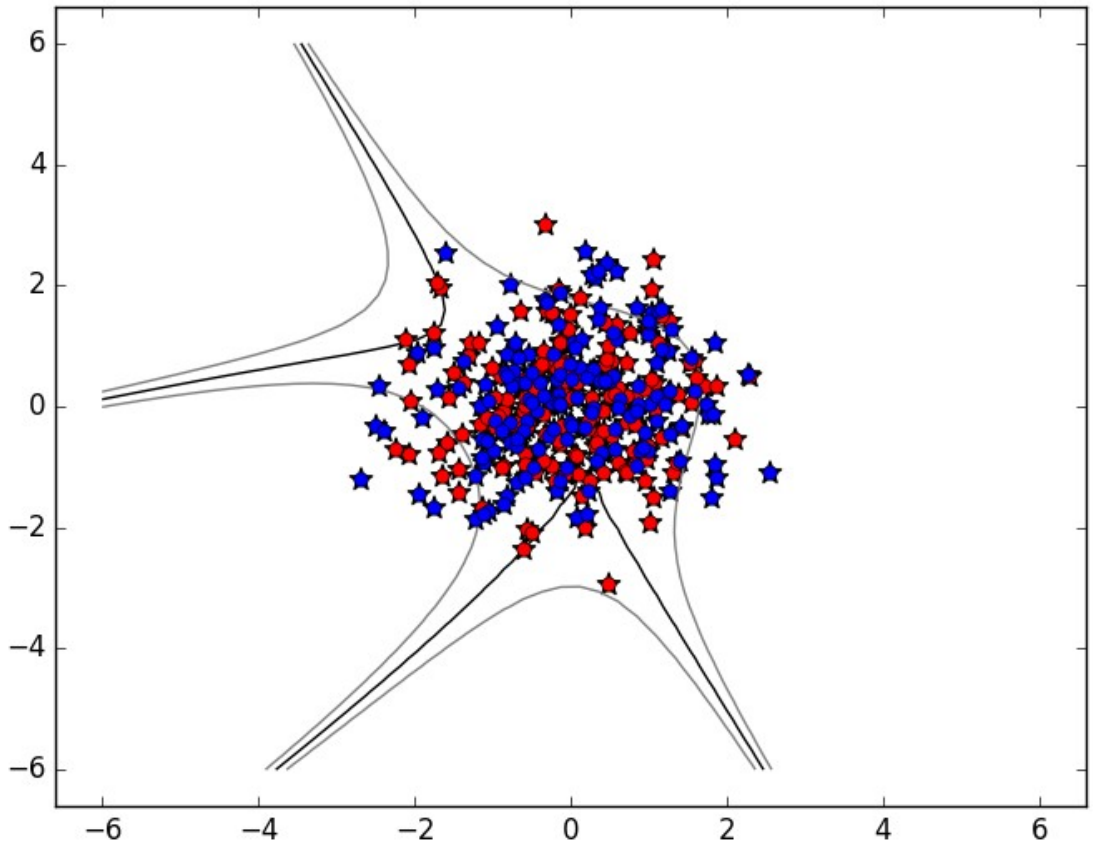

Fig 6.LINEARLY NON SEPERABLE DATASET WITH POLYNOMIAL KERNEL BASED SVM

In above graphs, I have plotted generated data point. In addition, I have plotted support vector points with (*) marker. We have calculated support vector point from feature matrix hence it will be overlapped by data points.

CONCLUSION :

The above graph concludes that the kernel based function performs better compare to linear kernel function on linearly non separable dataset.

Gaussian based kernel Support Vector Machine [LINEARRLY SEPERABLE DATASET]

On IRIS dataset

Accuracy = 76.9333 %

On generated dataset

Accuracy = 91.4285714286 %

Confusion Matrix : [[0 3]
[0 32]]

Precision = [0.0, 1.0]

Recall = [0, 0.91428571428571426]

F-Measure [0, 0.95522388059701502]

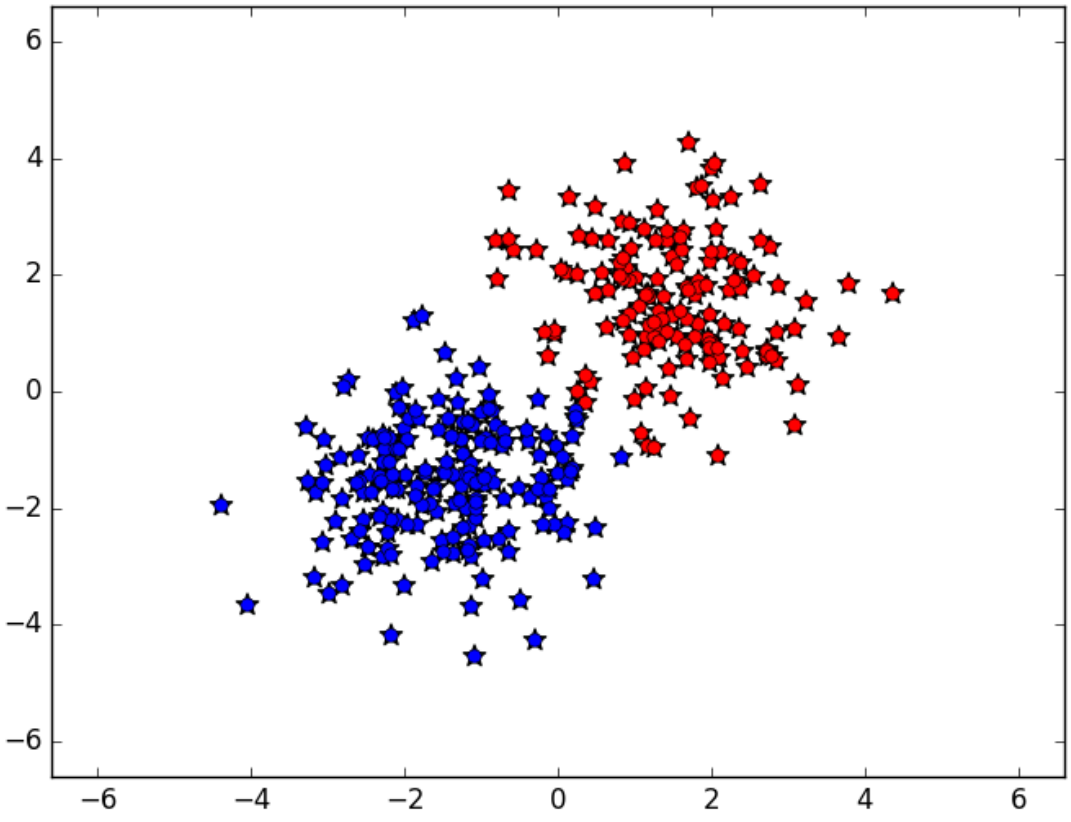

Fig 7. LINEARLY SEPERABLE DATASET WITH GAUSSIAN KERNEL BASED SVM

Gaussian based kernel Support Vector Machine [LINEARLY NON SEPERABLE DATASET]

On IRIS dataset
Accuracy = 84.9333 %

On generated dataset
Accuracy = 68.5714285714 %
Confusion Matrix : [[0 11]
[0 24]]
Precision = [0.0, 1.0]
Recall = [0, 0.68571428571428572]
F-Measure [0, 0.81355932203389836]

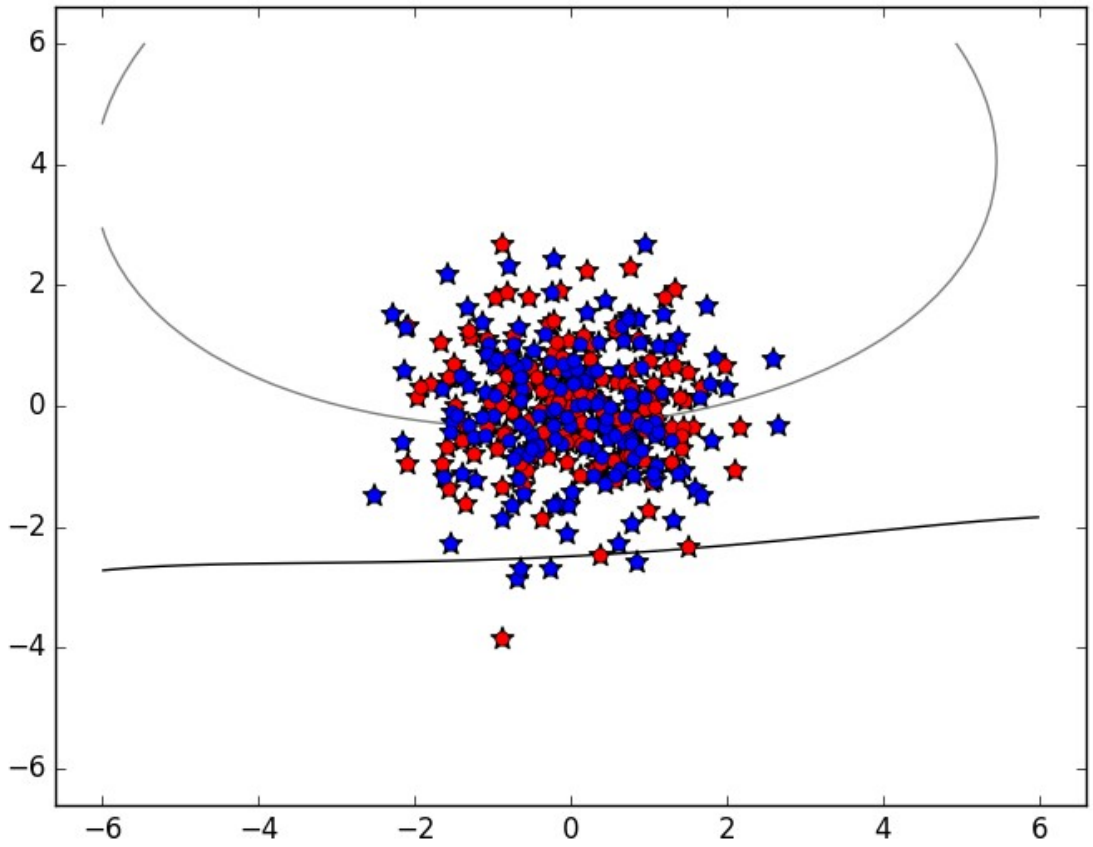

Fig 8. LINEARLY NON SEPERABLE DATASET WITH GAUSSIAN KERNEL BASED SVM

In above graphs, I have plotted generated data point. In addition, I have plotted support vector points with (*) marker. We have calculated support vector point from feature matrix hence it will be overlapped by data points.

CONCLUSION:

The above kernel function charts clearly shows that the linearly not separable dataset treated better using kernel based SVM. The accuracy for linearly non separable dataset in kernel based SVM is much better then linear kernel functions.

In sum up, the only accuracy does not matter for SVM classification and justification of models. However, after evaluating on too many randomly generated dataset as well as on iris dataset, I come to conclude that:

Hard-Soft margin SVM model performs better on linearly separable dataset. On opposite side, polynomial and Gaussian kernel based function performs better on linearly non-separable datasets.

ANSWER #3

Problem #3

Primal objective function of soft margin sum is given by:

$$L_p = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i y^{(i)}(w^T x^{(i)} + w_0) - 1 + \xi_i) - \sum_{i=1}^{m}\beta_i \xi_i$$

$$\frac{\partial L_p}{\partial w_i} = 0, \quad \frac{\partial L_p}{\partial w_0} = 0 \quad \& \quad \frac{\partial L_p}{\partial \xi_i} = 0.$$

For First one

$$\frac{\partial L_p}{\partial w_i} = \frac{1}{2}(2)w - \sum_{i=1}^{m}\alpha_i y^{(i)}(x^{(i)}) = 0$$

$$\Rightarrow w = \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)} \quad (1).$$

For second condition.

$$\frac{\partial L_p}{w_0} = 0 + 0 = \sum_{i=1}^{m}\alpha_i y^{(i)} = 0.$$

$$\Rightarrow \sum_{i=1}^{m}\alpha_i y^{(i)} - \quad (2)$$

Now for 3rd condition

$$\frac{\partial L_P}{\partial \delta_i} = 0.$$

$$\frac{\partial L_P}{\partial \delta_i} = 0 \text{ i.e. } -m\alpha_i - m\beta_i = 0.$$

$$\Rightarrow C - \alpha_i - \beta_i = 0 \quad \text{---(3)}$$

Put value of (1), (2) & (3) eqn in L_P.

$$L_P = \frac{1}{2} C \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)T} \left(\sum_{i=1}^{m} \alpha_i y_i^{(i)} x^{(i)}\right)$$

$$+ C \sum_{i=1}^{m} \delta_i - \sum_{i=1}^{m} \alpha_i y_i^{(i)} \left(\sum_{j=1}^{m} \alpha_j y^{(i)} x^{(i)T} x^{(i)}\right)$$

$$- \sum_{i=1}^{m} \alpha_i y^{(i)} w_0 + \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \alpha_i \delta_i$$

$$- \sum_{i=1}^{m} \beta_i \delta_i$$

For dual primal

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

$$\sum_{i=1}^{m} \alpha_i$$

Maximize the Lp (Dual).

Such that,

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

Now modify constraint

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

$$\Rightarrow C \geq \alpha_i$$

$$\Rightarrow \boxed{0 \leq \alpha_i \leq C}$$ Final constraint.

Hence, Max Lp.

Such that $0 \leq \alpha_i \leq C$

Lpenalty

$$\boxed{\sum_{i=1}^{m} \alpha_i y^{(i)} = 0.}$$

REFERENCES

1. http://www.astro.ufl.edu/~warner/prog/python.html
2. http://www.holehouse.org/mlclass/10_Advice_for_applying_machine_learning.html
3. http://aimotion.blogspot.com/2011/10/machine-learning-with-python-linear.html
4. http://archive.ics.uci.edu/ml/datasets/Iris
5. http://www.cs.umd.edu/~samir/498/SVM.pdf