CS584: Assignment 4

Pooja Mankani
Department of Computer Science
Illinois Institute of Technology
04/03/2016

Abstract

The report illustrates the implementation of support vector machine on synthetic and real data sets. The 10 fold cross validation for all the implementations is considered and the training error and testing error is evaluated. A custom model is created and the corresponding python built in model is also created using sklearn package in python. The errors of each of the model is compared and analyzed.

1. **Problem Statement**

   The following problems are considered in the implementation:

   Support Vector Machine:

   - Support Vector Machine are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
   - Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.
   - An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.
   - New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.
   - In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.
   - The SVM was implemented and applied on two datasets as follows:
     - Randomly created dataset: A 2D dataset consisting of features that are linearly dependent on each other. The samples are generated as separable and not separable dataset.
     - IRIS dataset: This dataset is extracted from UCI database.

2. **Proposed solution**

   a. **Algorithm Details:**
      - Solve the problem to get alpha
      - Identify the support vector set
      - Compute w and w0
      - Classify based on the values of w and w0.

        If transpose(w) + w0 > 0 then class 1
        else class 2

   b. Soft margin: y(i) (transpose(w) x(i) + w0) > 1 – slack_variable
      - If slack_variable = 0 : satisfy constraint (outside margin)
      - If slack_variable < = 1 : example is inside margin
      - If slack_variable > 1 : example is wrong side

   c. Use cvxopt.qp to get the lagrange multipliers.

d. For qp solver the following matrices are required:

$$p = (y.y.T).(X.X.T)$$

$$q = \begin{bmatrix} -1 \\ . \\ . \\ -1 \end{bmatrix}_{m*1}$$

$$G = \begin{bmatrix} -1 & 0 & \dots & 0 \\ . & & & \\ . & & & \\ 0 & 0 & \dots & -1 \\ 1 & 0 & \dots & 0 \\ . & & & \\ . & & & \\ 0 & 0 & \dots & 1 \end{bmatrix}_{2m*m}$$

$$h = \begin{bmatrix} 0 \\ . \\ . \\ 0 \\ c \\ . \\ . \\ c \end{bmatrix}_{2m*1}$$

$$A = y.T$$

$$b = 0$$

e. Prediction:

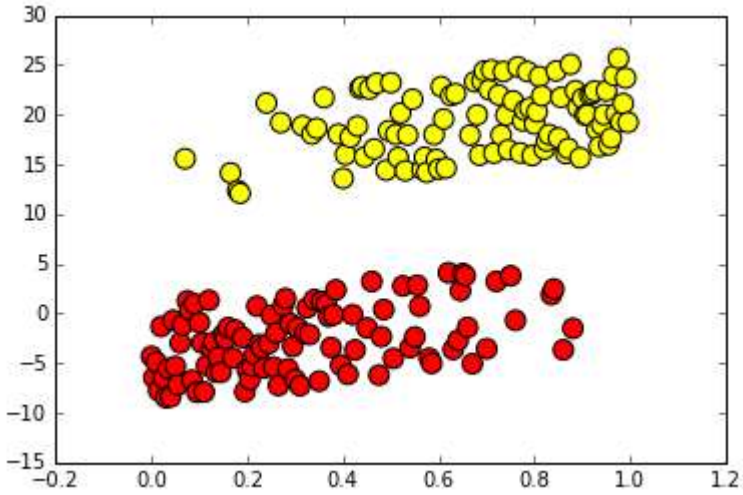
$$if((W.T).X + W_0) > 0 \longrightarrow \hat{y} = 1$$
$$otherwise \longrightarrow \hat{y} = 0$$

3. **Implementation Details:**
    a. **Design issues:** The following design issues were witnessed:
        - The dataset with multiple feature is difficult to plot and cannot be plotted in a two dimensional space since there are multiple features associated to it with a quadratic function.
        - The matrix for polynomial with respect to higher degree for the polynomial kernel is difficult to obtain manually. This is complex since the combination of features is difficult to construct along with the order of the same. For e.g. if the degree is 2 and the number of features is 2 then the various feature combinations would be: **[1 X1 X2 X1^2 X2^2 X1X2].** Evaluating these combinations is difficult and complex.
        - In case multiple records for large data sets are given as a input, for e.g. consider a data with billion records, in this case if we store all the data extracted from the records into variable then there could be a possibility of memory error within your system. This is because it would take up lots of memory to store data within systems.
    b. **Problems faced:** The following issues were faced:
        - Cross validation: The implementation of cross validation to retrieve the training and testing indices was complex to obtain using python array and matrix manipulation. Hence in this case KFold cross validation of sklearn package was used to evaluate the same.
        - Features in higher dimensional space: The multiple combination of features in the higher dimensional space is complex to evaluate using python functions. Since for different degrees and different number of features the combinations become complex. Hence in this case PolynomialFeatures of the sklearn.preprocessing package is used to evaluate the same.
        - The model evaluated by gradient were resulting in high errors hence the data was required to be normalized. Thus the formula was updated such that the difference between the actual and the predicted value was divided by the number of samples.

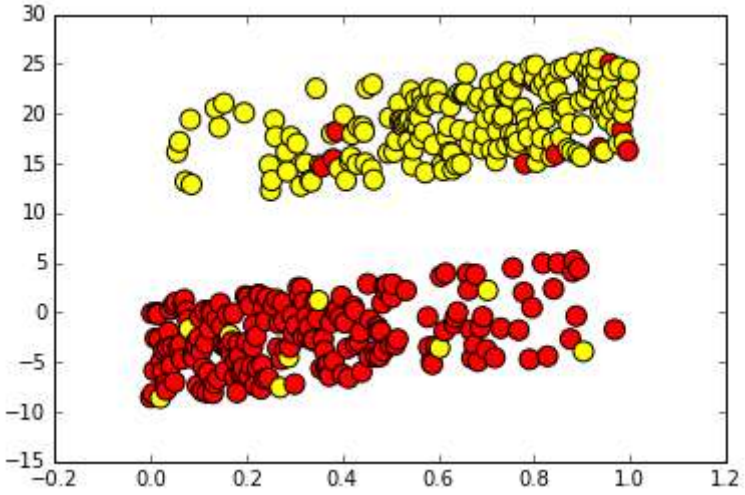
c. **Implementation Details:**

Python Notebook	Analysis	Command
Assignment_4.ipynb	Linear SVM with hard margin and separable data	do_cross_validation(200, False ,'linear', 0, 0,'hard', 10,False,False, verbose=True)
Assignment_4.ipynb	Linear SVM with hard margin and not separable data	do_cross_validation(200, True ,'linear', 0, 0,'hard', 10,False,False, verbose=True)
Assignment_4.ipynb	Linear SVM with soft margin and separable data	do_cross_validation(200, False ,'linear', 0, 0,'soft', 10,False,False, verbose=True)
Assignment_4.ipynb	Linear SVM with soft margin and not separable data	do_cross_validation(200, True ,'linear', 0, 0,'soft', 10,False,False, verbose=True)
Assignment_4.ipynb	Polynomial SVM with soft margin and separable data	do_cross_validation(200, False ,'poly', 2, 0,'soft', 10,False,False, verbose=True)
Assignment_4.ipynb	Polynomial SVM with soft margin and not separable data	do_cross_validation(200, True ,'poly', 2, 0,'soft', 10,False,False, verbose=True)
Assignment_4.ipynb	Gaussian SVM with soft margin and separable data	do_cross_validation(200, False ,'gaussian', 0, 10,'soft', 10,False,False, verbose=True)
Assignment_4.ipynb	Gaussian SVM with soft margin and separable data	do_cross_validation(200, True ,'gaussian', 0, 11,'soft', 10,False,False, verbose=True)
Assignment_4.ipynb	Class with more samples	do_cross_validation(200, True ,'linear', 0, 10,'soft', 10,True,False, verbose=True)
Assignment_4.ipynb	Iris Dataset	do_cross_validation(200, True ,'linear', 0, 10,'soft', 10,False,True, verbose=True)

4. **Results and Discussions:**
   a. **Generate a dataset with 2D features**
      - **Linearly separable dataset**



      - **Not separable dataset**



b. SVM with hard margin

- **Linearly separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[7 0]
[0 13]]

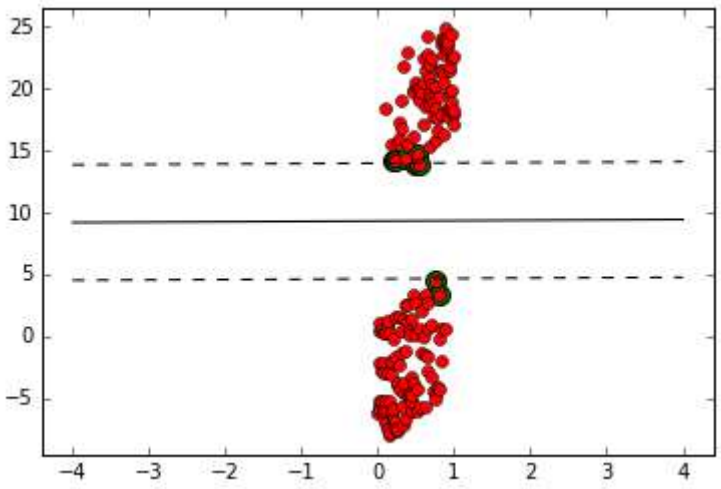
  - **Value of 'w' and 'b' computed:**

Parameter	Value
w	[-0.00654146 0.2141546]
b	-1.99028049685

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

  - **SVM plot:**



  - **Conclusion: For linearly separable data, the hard margin gives 100% accuracy and the SVMs are found near the decision boundary.**

- **Not separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	0.51999999999999991
Precision	0
Recall	0
F1 Measure	0

Confusion Matrix: [[12 0]
[8 0]]

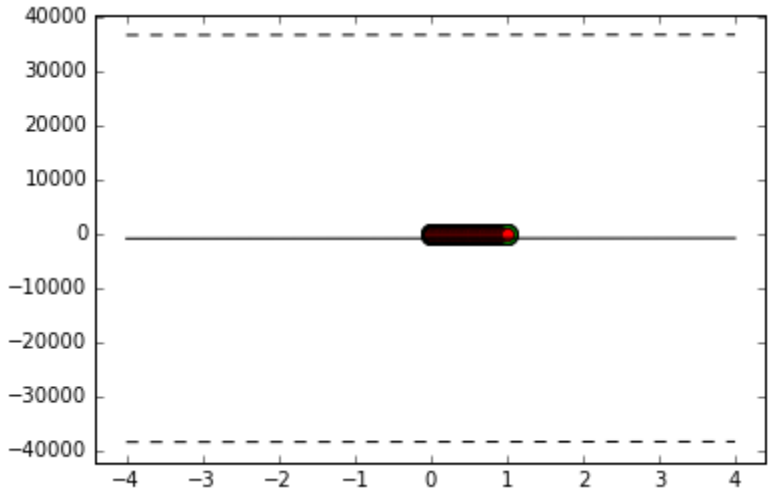
  - **Value of 'w' and 'b' computed:**

Parameter	Value
w	[2.58147717e-04 -2.65836716e-05]
b	-0.0220180908839

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0.60
Precision	0
Recall	0
F1 Measure	0

  - **SVM plot:**



  - **Conclusion: For not separable data, the hard margin gives very bad accuracy. It always tries to classify the data only on one side of the decision boundary.**

c. **SVM with soft margin**

- **Linearly separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[10 0]
[0 10]]

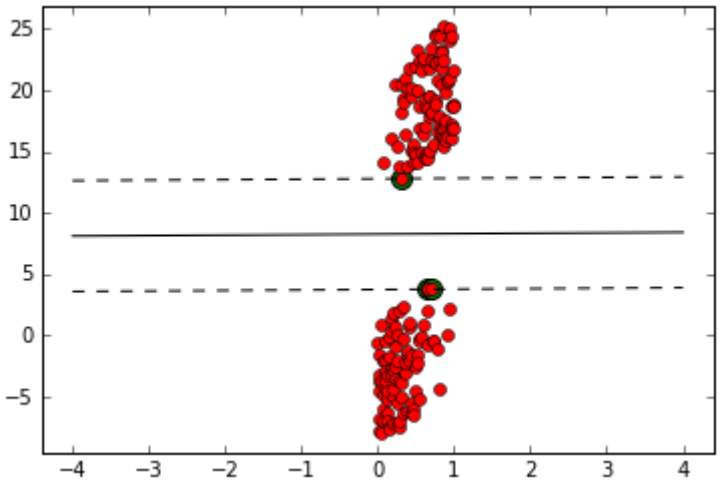
  - **Value of 'w' and 'b' computed:**

Parameter	Value
w	[-0.00844916 0.22115439]
b	-1.82733479834

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

  - **SVM plot:**



  - **Conclusion: For linearly separable data, the soft margin gives 100% accuracy and the SVMs are found near the decision boundary on either sides of the classes.**

- **Not separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	0.90
Precision	1
Recall	0.8462
F1 Measure	0.9167

Confusion Matrix: [[7 0]
[2 11]]

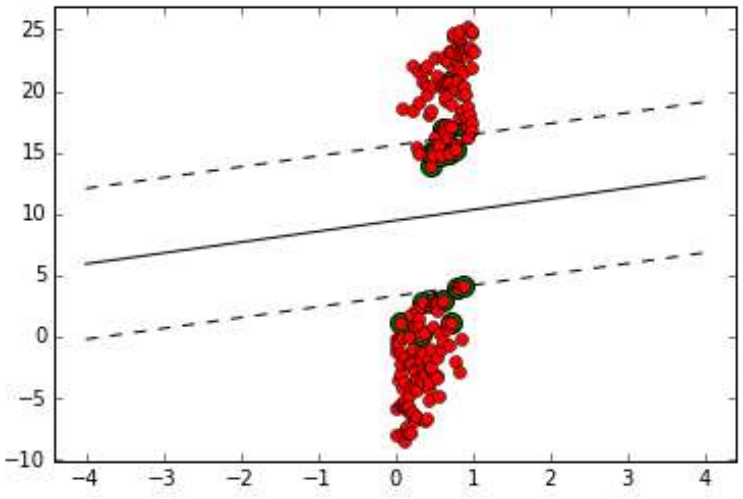
  - **Value of 'w' and 'b' computed:**

Parameter	Value
w	[-0.22929494 0.14767515]
b	-1.11552958294

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0.94999999999999996
Precision	0.97257575757575743
Recall	0.94025641025641027
F1 Measure	0.95443412880026146

  - **SVM plot:**



  - **Conclusion: For not separable data, the soft margin works well and gives a average accuracy of 94%. The best fit slack size is 1^-10. Moreover, in case of soft margin it is observed that the SVMs are obtained in a way the data is separated thus giving accurate results.**

d. Kernel-based SVM algorithm

- **Polynomial kernel for separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[9 0]
[0 11]]

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

  - **Conclusion:** For separable data, the kernel based polynomial margin works well and gives a average accuracy of 100%. The best fit is obtained at polynomial degree = 2 and since there are polynomial features a contour is built which acts as the decision boundary for the prediction.
- **Polynomial kernel for not separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	0.95
Precision	0.9231
Recall	1
F1 Measure	0.96

Confusion Matrix: [[7 1]
[0 12]]

  - **Value of 'b' computed:**

Parameter	Value
b	-0.689562290459

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0.94999999999999996
Precision	0.97071678321678334
Recall	0.93365939615939619
F1 Measure	0.94860582010582029

  - **Conclusion: For not separable data, the kernel based polynomial SVM works well and gives a average accuracy of 94% similar to that of the linear SVM. The SVMs here are spread evenly across the entire dataset thus giving a good prediction. The best fit is obtained at polynomial degree = 2**

e. **Kernel-based SVM algorithm**

- **Gaussian kernel for separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[8 0]
[0 12]]

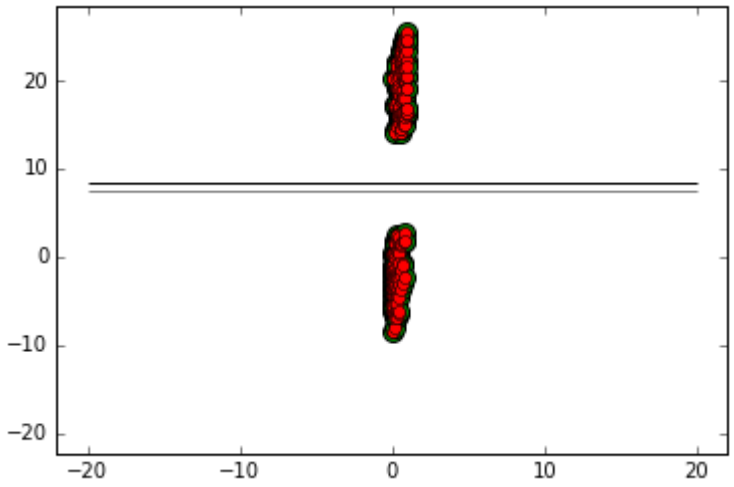
  - **Value of 'b' computed:**

Parameter	Value
b	0.000310096518002

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

  - **SVM plot:**



- **Conclusion: For separable data, the kernel based gaussian SVM works well and gives a average accuracy of 100% at sigma = 10. The best fit is obtained at sigma= 8. Based on the SVM plot we can say that the decision boundary is correctly plotted for both the classes and a a accurate prediction is made.**

- **Gaussian kernel for not separable dataset**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	0.95
Precision	1
Recall	0.9375
F1 Measure	0.9677

Confusion Matrix: [[4 0]
[1 15]]

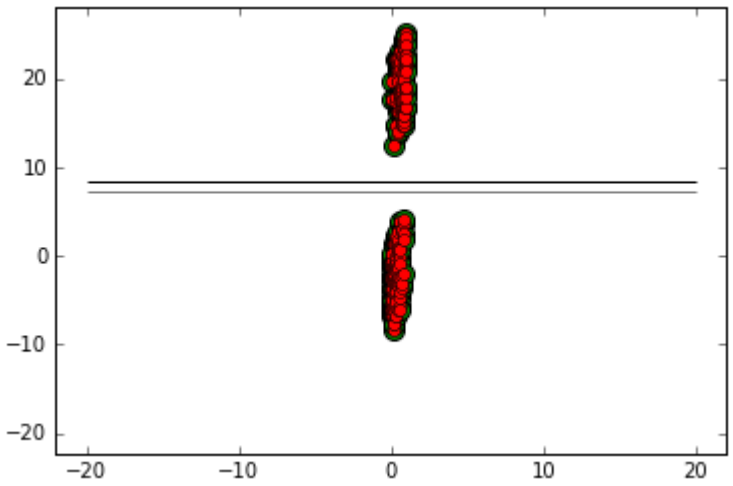
  - **Value of 'w' and 'b' computed:**

Parameter	Value
b	0.0265224712662

  - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0.94999999999999984
Precision	0.97666666666666657
Recall	0.92855769230769225
F1 Measure	0.95026277666626091

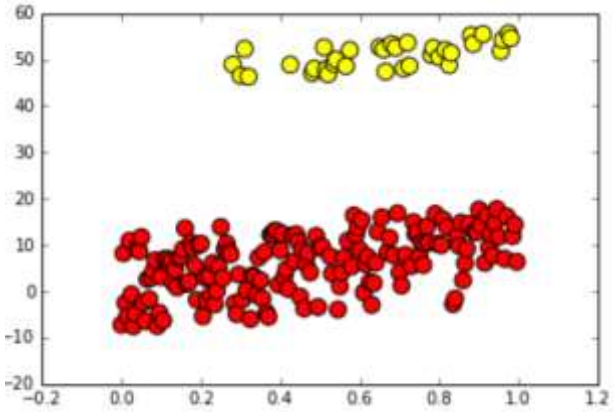
  - **SVM plot:**



- **Conclusion: For not separable data, the kernel based gaussian SVM works well and gives a average accuracy of 95% at sigma = 11. The best fit is obtained at sigma= 11. Based on the SVM plot we can say that the decision boundary is correctly plotted for both the classes and a a accurate prediction is made.**

f. Class with more samples

- In case a class has more samples there are more support vectors created for that class.
- Since there are more support vectors for one class, the decision boundary tends to move towards that class.
- Hence in case though the accuracy is substantially good since the decision boundary tends to be near the class that has more samples the accuracy would affect when large amount of testing data needs to be predicted and due to the decision boundary falling in the dominant class most of the testing samples would be predicted for that class.



- **Overall Parameters computed:**

Parameter	**Result**
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[15 0]
[0 5]]

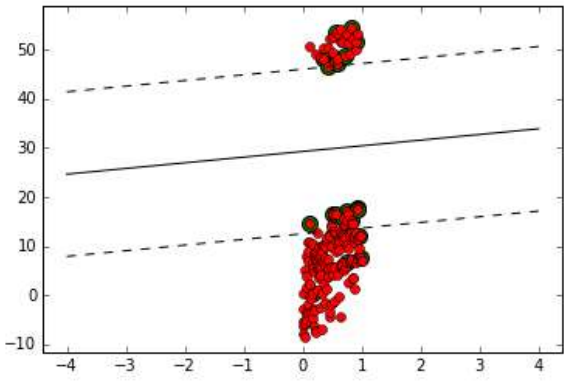
- **Value of 'w' and 'b' computed:**

Parameter	**Value**
w	[-0.00179828 0.07016964]
b	0.0265224712662

- **Average Parameters Computed after 10 folds:**

Parameter	**Result**
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

SVM plot:



IRIS Dataset:

The model is applied on IRIS dataset and 10 fold cross validation was performed for each of the kernel SVM. The following are the results obtained from the IRIS dataset.

- **Linear SVM:**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[6 0]
[0 4]]

  - **Value of 'w' and 'b' computed:**

Parameter	Value
b	1.85529920081
w	[-0.0460345 0.52172253 -1.00316474 -0.4641797]

- **Polynomial SVM:**
  - **Degree: 2**
    - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[6 0]
[0 4]]

- **Degree 3**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	0.84000000000000008
Precision	0.97142857142857153
Recall	0.72583333333333333
F1 Measure	0.80581196581196579

Confusion Matrix: [[6 0]
[2 2]]

Conclusion: The best accuracy is obtained as degree = 2 for polynomial SVM on nominal dataset

- **Gaussian SVM:**
  - **Sigma: 0.1**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	0.71
Precision	0.86666666666666659
Recall	0.72999999999999998
F1 Measure	0.71126984126984127

  - **Sigma: 1**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

  - **Sigma: 10**
  - **Overall Parameters computed:**

Parameter	Result
Accuracy	0.98999999999999999
Precision	0.97499999999999998
Recall	1
F1 Measure	0.98571428571428577

Conclusion:
The best accuracy is obtained at sigma = 1 for Gaussian SVM on nominal dataset

Data Set:

IRIS dataset: http://archive.ics.uci.edu/ml/datasets/Iris

Linear dataset generated using random values and a line function for separable and not separable cases.

References:

https://en.wikipedia.org/wiki/Support_vector_machine

http://cvxopt.org/index.html

https://en.wikipedia.org/wiki/Quadratic_programming

http://cvxopt.org/applications/svm/

http://stackoverflow.com/questions/32543475/how-python-cvxopt-solvers-qp-basically-works