Support Vector Machines

Karthik Bangalore Mani
Department Of Computer Science
Illinois Institute of Technology

April 25, 2016

Abstract

The goal of this assignment is to design a Support Vector Machine with Linear, Polynomial and Gaussian Kernels along with Hard and Soft Margins.

Problem Statement

- Given a 2 Dimensional, 2-class Dataset, fit a Support Vector Classifier to the train data. The kernels should be varied and the examples must be classified and error should be measured. The confusion matrix should be constructed and the performance measures such as precision,recall,F-Measure and accuracy should be determined from the confusion matrix.

Proposed Solution

- **Training :**

  -Solve dual for α_i, the lagrange multipliers for all examples:

  -Identify the Support Vector set :

  -SV = $\{X^i \mid \alpha_i > 0\}$

  -Compute W and W_0 :

  -W = $\sum_{i=1}^{m} \alpha_i Y^i X^i$

  -$W_0 = \frac{1}{\#SV} \sum_{X^i \in \{SV\}} Y^i - W^T . X^i$

- **Classification:**
  - **Linear Kernel:**
    -Given X, if $W^T X + W_0 > 0 : Class_1$ else, $Class_2$
  - **Gaussian and Poly Kernel:**
    -Given X, if $\sum_{i=1}^{m} \alpha_i Y^i K(X^i, X) + W_0 > 0 : Class_1$ else, $Class_2$

Implementation Details

1. **Program Design Issues :**
   As the Polynomial order is varied for the polynomial kernel, it takes more time to train the SVM classifier, particularly when there are large number of examples.

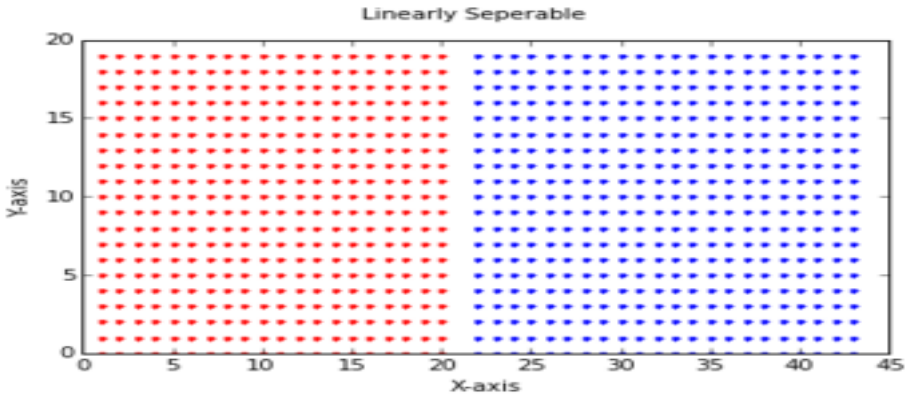
2. **Problems faced :**
   The Lagrange multipliers were not exactly zero, but very close to zero for the support vectors. Hence a threshold of 0.01 was fixed for identifying the support vectors. If the Lagrange multipliers are more than the threshold, then it is counted as a member of the support vector set, else it is not counted as a member of support vector set.

3. **Instructions to use the program**
   Open the SVM.ipynb file in iPython notebook, and execute each cell to get the desired output. The datasets must be placed in the same folder as that of the SVM.ipynb file.

Results and discussion

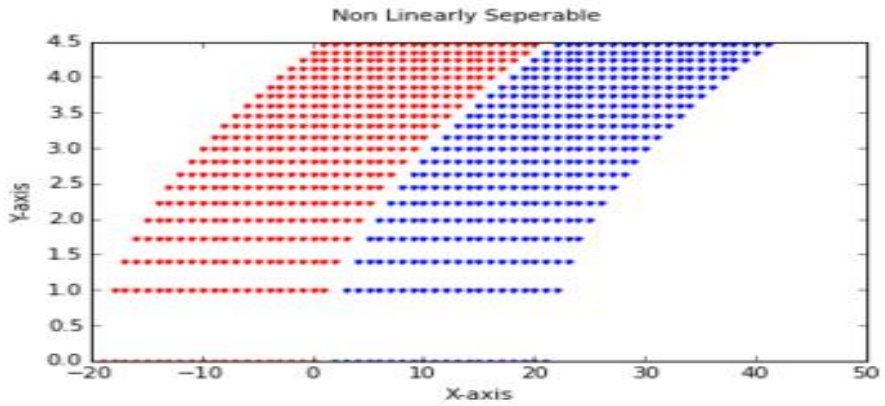
1. **Linearly Separable Dataset :**

   A linearly separable dataset was generated and plotted as follows:

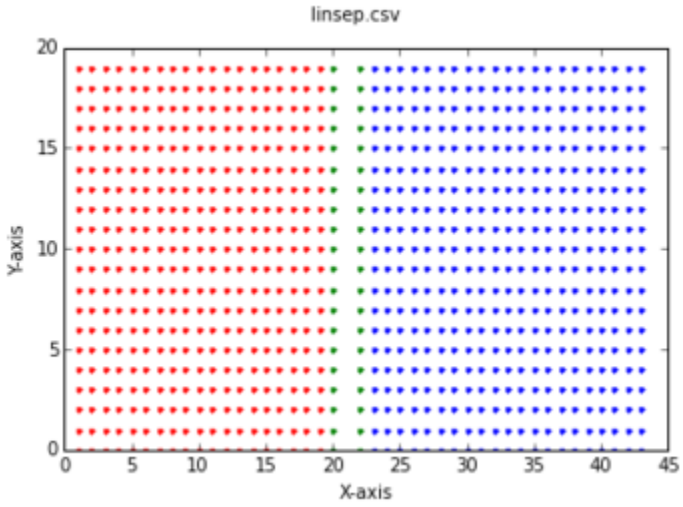


2. **Non Linearly Separable Dataset:**

   A Nonlinearly separable dataset was generated and plotted as follows:



3. **Hard Margin – Linear Kernel :**
   The Support Vectors are highlighted in green color:



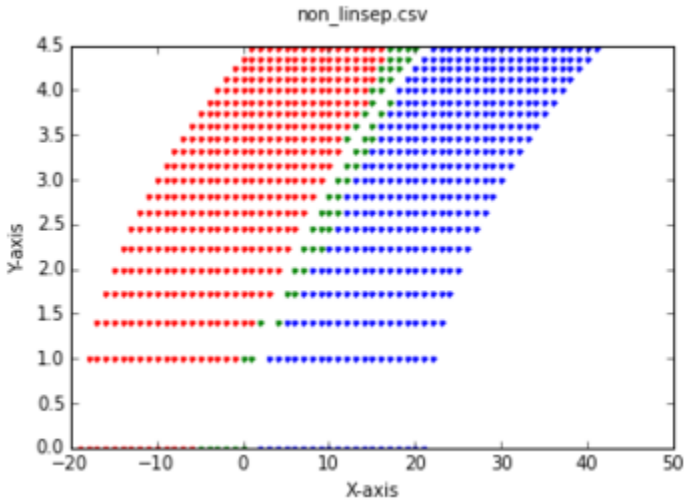
Confusion Matrix:

```
[[ 440.   0.]
[  0.  400.]]
```

Accuracy:

```
1.0
```

Class	Precision	Recall	F-Measure
1	1.0	1.0	1.0
2	1.0	1.0	1.0



Confusion Matrix :

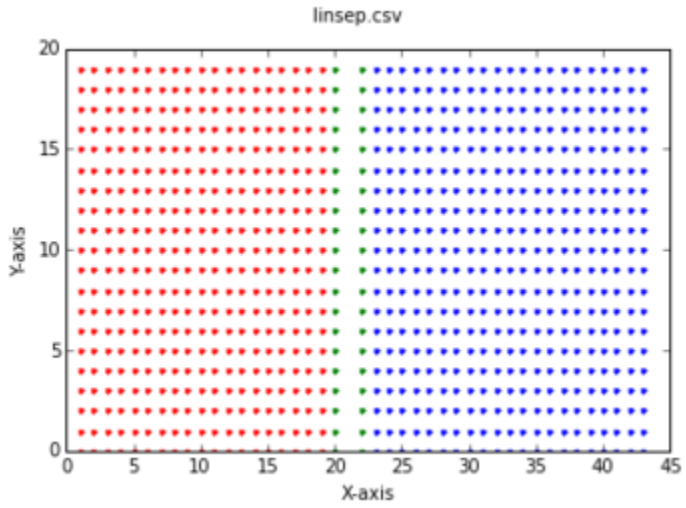
```
[[ 411.    8.]
 [   9.  412.]]
```

Accuracy:

0.979761904762

Class	Precision	Recall	F-Measure
1	0.980906921241	0.978571428571	0.979737783075
2	0.978622327791	0.980952380952	0.979785969084

4. Soft Margin – Linear Kernel :

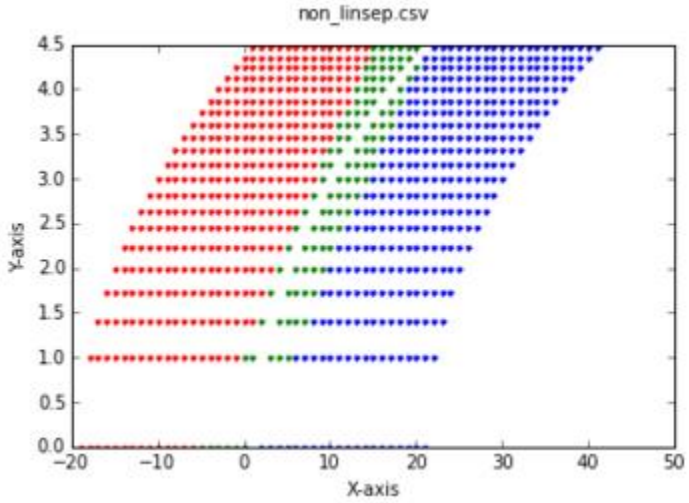


Confusion Matrix:

```
[[ 440.    0.]
[   0.  400.]]
```

Accuracy:
1.0

Class	Precision	Recall	F-Measure
1	1.0	1.0	1.0
2	1.0	1.0	1.0

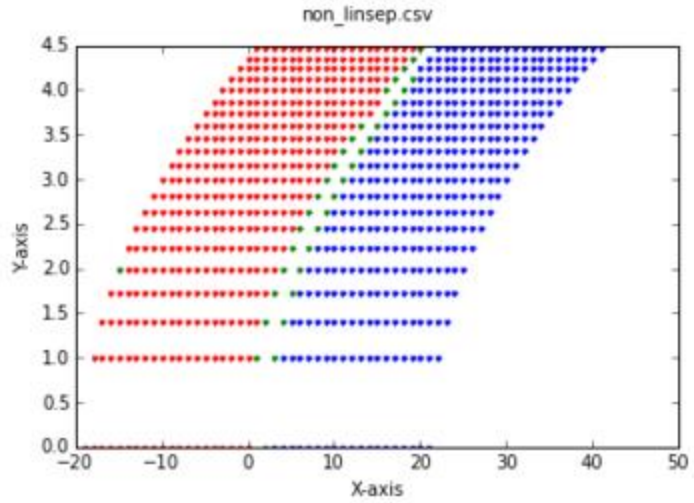


Confusion Matrix :
[[411. 12.]
 [9. 408.]]

Accuracy :
0.975

Class	Precision	Recall	F-Measure
1	0.971631205674	0.978571428571	0.975088967972
2	0.978417266187	0.971428571429	0.974910394265

5. **Polynomial Kernel for Non-Linearly Separable Data :**
   Polynomial Order of 3 was chosen.



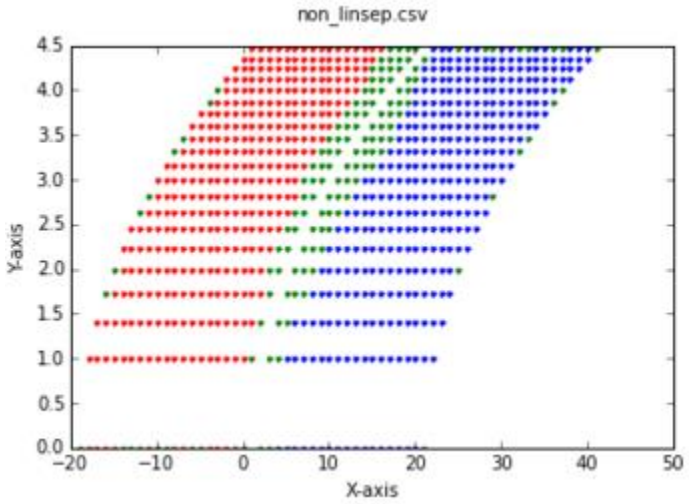

```
Confusion Matrix :
[[ 385.   64.]
 [  35.  356.]]
```

Accuracy:

0.882142857143

Class	Precision	Recall	F-Measure
1	0.857461024499	0.916666666667	0.886075949367
2	0.910485933504	0.847619047619	0.877928483354

6. Gaussian Kernel for Non-Linearly Separable Data :

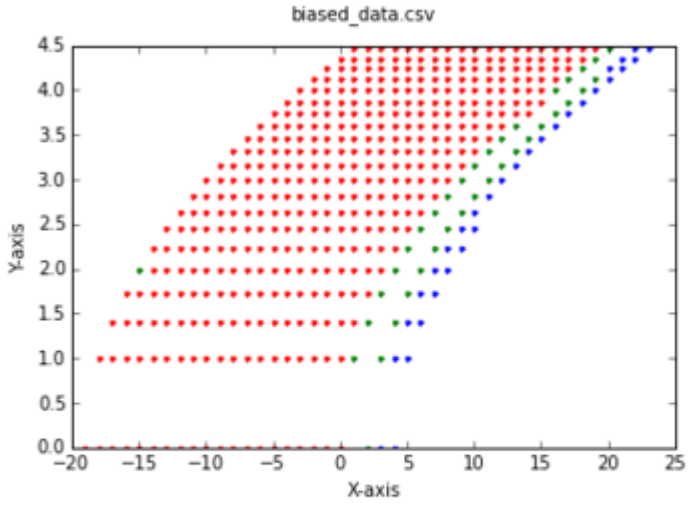


```
Confusion Matrix :
[[ 366.   45.]
 [  54.  375.]]
```

Accuracy:
0.882142857143

Class	Precision	Recall	F-Measure
1	0.857461024499	0.871428571429	0.880866425993
2	0.874125874126	0.892857142857	0.883392226148

7. SVM on a biased dataset, with more number of examples of a particular class:




```
Confusion Matrix :
[[  28.   64.]
 [  21.  356.]]
```

Accuracy:
0.818763326226

Class	Precision	Recall	F-Measure
1	0.304347826087	0.571428571429	0.397163120567
2	0.944297082228	0.847619047619	0.893350062735

$$\min \quad L_p = \frac{1}{2} w^T w + C \sum_{i=1}^{m} s_i$$

$C = 0 \Rightarrow$ Allow any slack value. lots of error

$C = \infty \Rightarrow$ No error. Same as hard margin.

$$\min \begin{cases} L_p = \frac{1}{2} w^T w + C \sum_{i=1}^{m} s_i \\ \text{s.t} \quad s_i > 0 \\ y^{(i)}(w^T x^{(i)} + w_0) > 1 - s_i \end{cases}$$

Now, lets combine the constraints with the objective.

Idea: (1) Penalize for large slack values

(2) Penalize if $y^{(i)}(w^T x^{(i)} + w_0) < 1 - s_i$

$$L_p = \frac{1}{2} w^T w + C \sum_{i=1}^{m} s_i - \sum_{i=1}^{m} \alpha_i \left[y^{(i)}(w^T x^{(i)} + w_0) - 1 + s_i \right] \rightarrow (*) - \sum_{i=1}^{m} \beta_i s_i$$

Unknowns:

w, w_0, s_i

$$\frac{d}{dw} L_p = 0 \quad ; \quad \frac{d}{dw_0} L_p = 0 \quad ; \quad \frac{d}{ds_i} L_p = 0$$

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

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

$$\frac{d}{dw_0} L_p = - \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \Rightarrow \boxed{\sum_{i=1}^{m} \alpha_i y^{(i)} = 0} \rightarrow (2)$$

$$\frac{d}{ds_i} L_p = C - \alpha_i - \beta_i = 0 \Rightarrow \boxed{C - \alpha_i - \beta_i = 0} \rightarrow (3)$$

Substituting (1), (2) & (3) in (6), we get,

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

↳ Same as Hard-margin dual.

Weakness:

- The Gaussian Kernel implementation seems to identify some of the non-support vectors as support vectors.

References

1. **Stackoverflow.com**
2. **Wikipedia.org**