

CS584 – Assignment 4

Support Vector Machines

A20364639

Yung Chi Liu

● Problem statement

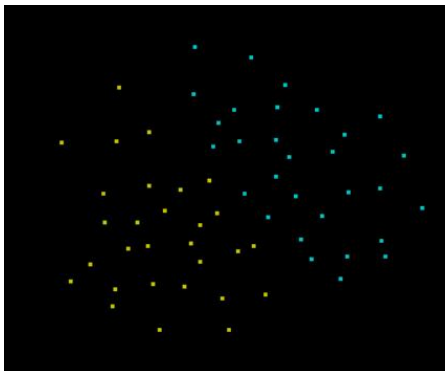
In this assignment I will implement Support Vector Machine algorithm. First, use SVM with hard margins and apply to dataset and plot the result. Second, is use SUM with soft margins and test the data. Finally, the last algorithm is to implement kernel-based SVM algorithm using a polynomial and Gaussian kernel function (Radial Basis Function) and test the same data sets. More test the data set with 1 class had more examples.

1. Generate Data sets:

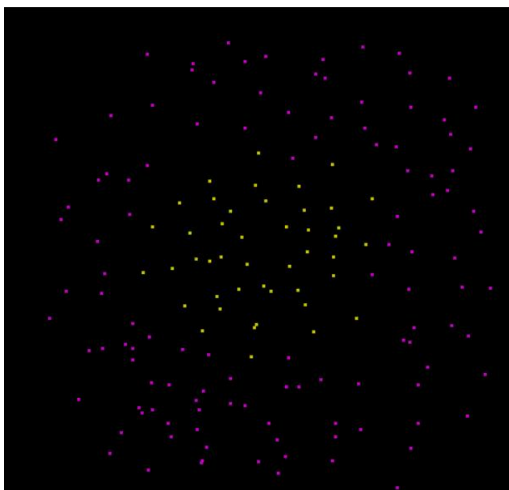
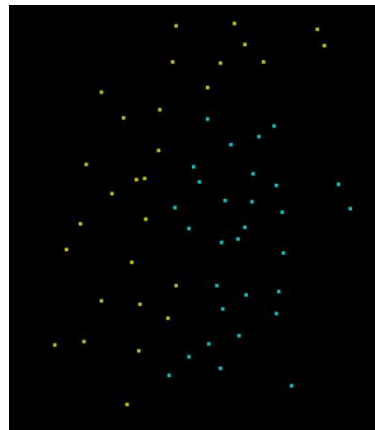
I generate data sets by libsvm. It use mouse to generate data by click, and then modify the data to fit my program.

Data generate by libsvm:

Linear separate data :



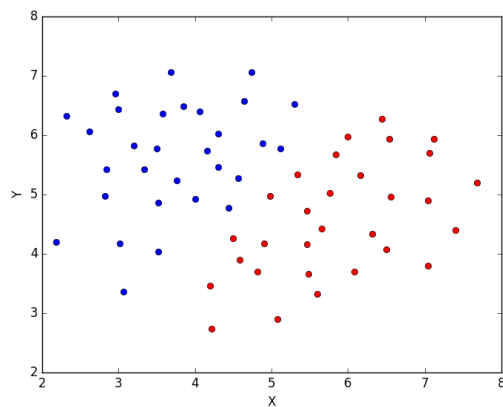
Non-linear separate data:



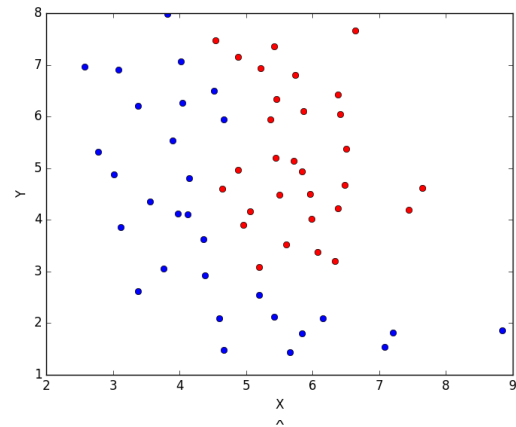
Circle like data:

Plot in python (modified data)

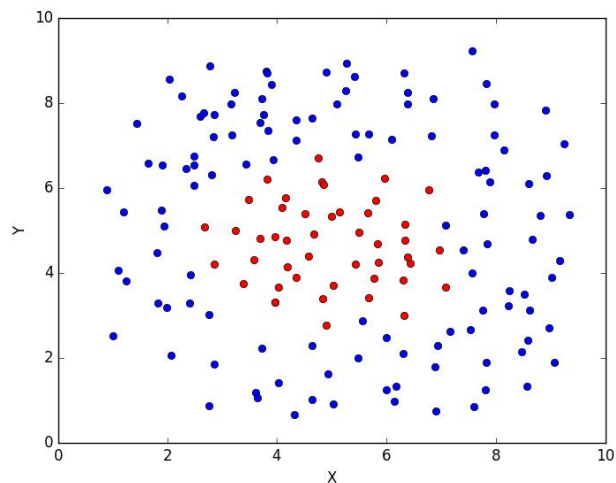
Linear separate data:



Non-linear separate data:



Circle data:



2. Hard margin SVM

● Implantation:

Algorithm:

1. Solve Dual for α_i
2. Identify support vector set
3. Compute W , W_0
4. Classification: $(W^T X + W_0) > 0 \rightarrow C0$
 $(W^T X + W_0) < 0 \rightarrow C1$

Program:

1. Solve Dual for α_i (Lagrange Mutipier) :

To solve α_i , I use

solution = cvxopt.solvers.qp(P, q, G, h, A, b)

which fit the equation

$$\begin{aligned} -L_D &= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} - \sum_{i=1}^m \alpha_i \\ &= \frac{1}{2} \alpha (y y^T \circ X X^T) \alpha + [-1 \dots -1] \alpha \end{aligned}$$

Which $\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$, $X = \begin{bmatrix} X^{(1)T} & \dots \\ \vdots & \\ X^{(m)T} & \dots \end{bmatrix}$

$$P = y y^T \circ X X^T, q = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}, A = y^T, b = 0, G = -1 \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}, h = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

2. Compute W_0 and Support vector:

$$SV = \{x^i | \alpha^i > 0\}$$

$$W_0 = \frac{1}{\#SV} \sum_{x \in SV} (y^i - W^T x^i)$$

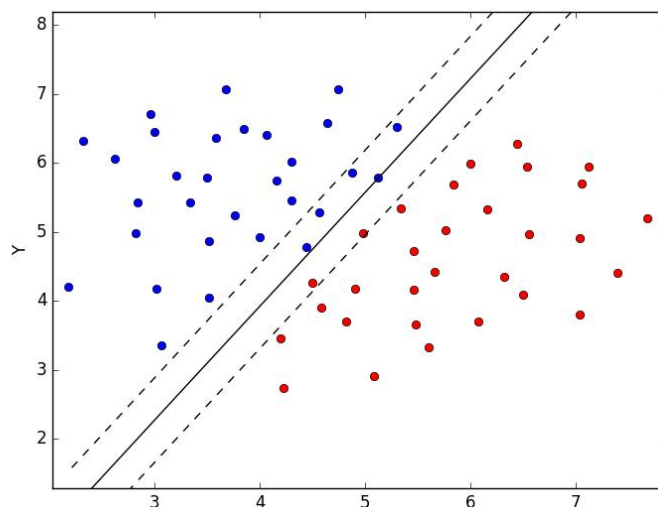
3. Compute W :

$$W = \sum_{i=0}^n \alpha_i y^{(i)} x^{(i)}$$

● Result:

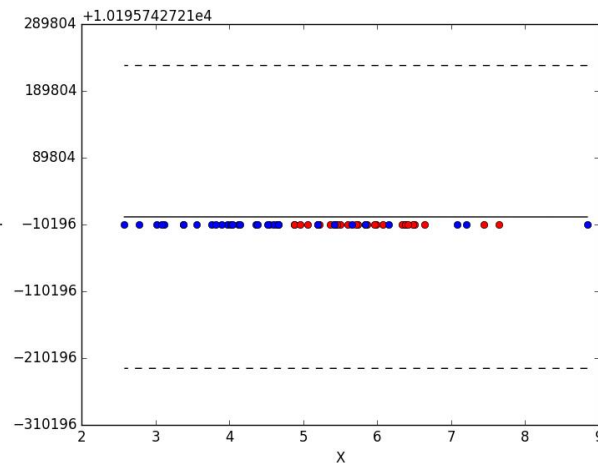
Plot:

Linear separate data:



10 Fold Validation accuracy: 0.966666666667

Non-linear separate data:



10 Fold Validation accuracy: 0.266666666667

As the result hard margin could not work on non-linear separate data.

3. Derive LD from Lp with soft margin SVM:

$$\begin{aligned}
 \textcircled{1} \quad \frac{\partial L_P}{\partial W} &= \frac{1}{2} \cdot 2 \cdot W - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0 \\
 &\Rightarrow W = \frac{\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}}{\sum_{i=1}^m \alpha_i} \\
 \textcircled{2} \quad \frac{\partial L_P}{\partial W_i} &= - \sum_{i=1}^m \alpha_i y^{(i)} = 0 \\
 &\Rightarrow \sum_{i=1}^m \alpha_i y^{(i)} = 0 \\
 \textcircled{3} \quad \frac{\partial L_P}{\partial \xi_i} &= C - \alpha_i - \beta_i \\
 &\Rightarrow C - \alpha_i - \beta_i = 0 \\
 L_P &= \frac{1}{2} \left(\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)T} \right) \left(\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \right) + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i y^{(i)} \left(\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)T} \right) x^{(i)} \\
 &\quad - \sum_{i=1}^m \alpha_i y^{(i)} W_0 + \sum_{i=1}^m \alpha_i - \sum_{i=1}^m \alpha_i \xi_i - \sum_{i=1}^m \beta_i \xi_i \\
 (LD) &= \frac{-1}{2} \sum_{i=1}^m \sum_{j=2}^m \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} + \sum_{i=1}^m \alpha_i \quad \text{same as hard margin.} \\
 \begin{cases} \max L_D \\ \text{s.t. } \alpha_i \geq 0, \sum_{i=1}^m \alpha_i y^{(i)} = 0, C - \alpha_i - \beta_i = 0, \forall i, \beta_i \geq 0. \end{cases} \\
 \begin{cases} \max L_P \\ \text{s.t. } 0 \leq \alpha_i < C, \sum_{i=1}^m \alpha_i y^{(i)} = 0. \end{cases}
 \end{aligned}$$

4. Implement soft margin SVM:

- **Implantation:**

Use same algorithm as hard margin SVM, add a Slack variables ϵ_i

The old model:

$$d(x) = W^T X + W_0 > 0$$

the new model:

$$d(x) = W^T X + W_0 > 1 - \epsilon_i$$

solve Dual for α_i , I use solver.qp(P, q, G, h, A, b) same as hard margin.

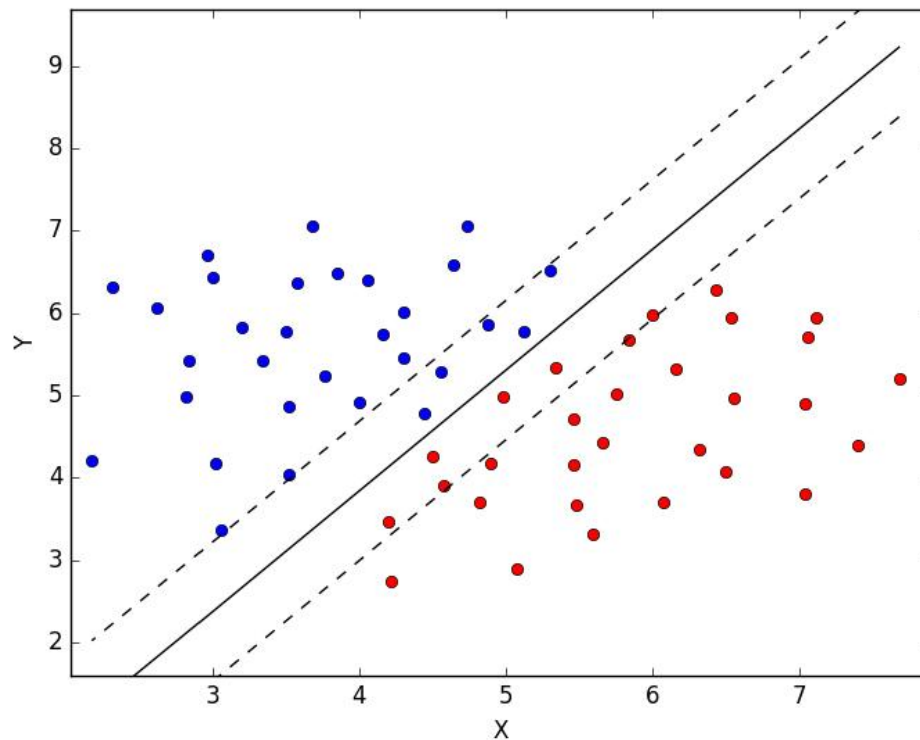
However, the G and h is different.

$$\begin{cases} \min_x \frac{1}{2} X^T P X + q^T \\ \text{s. t. } A x = b, G x \leq h \end{cases}$$

$$G = \begin{bmatrix} -1 \\ \vdots \\ -1 \\ - \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad h = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ c \\ \vdots \\ c \end{bmatrix}, \quad \text{set } c = 1$$

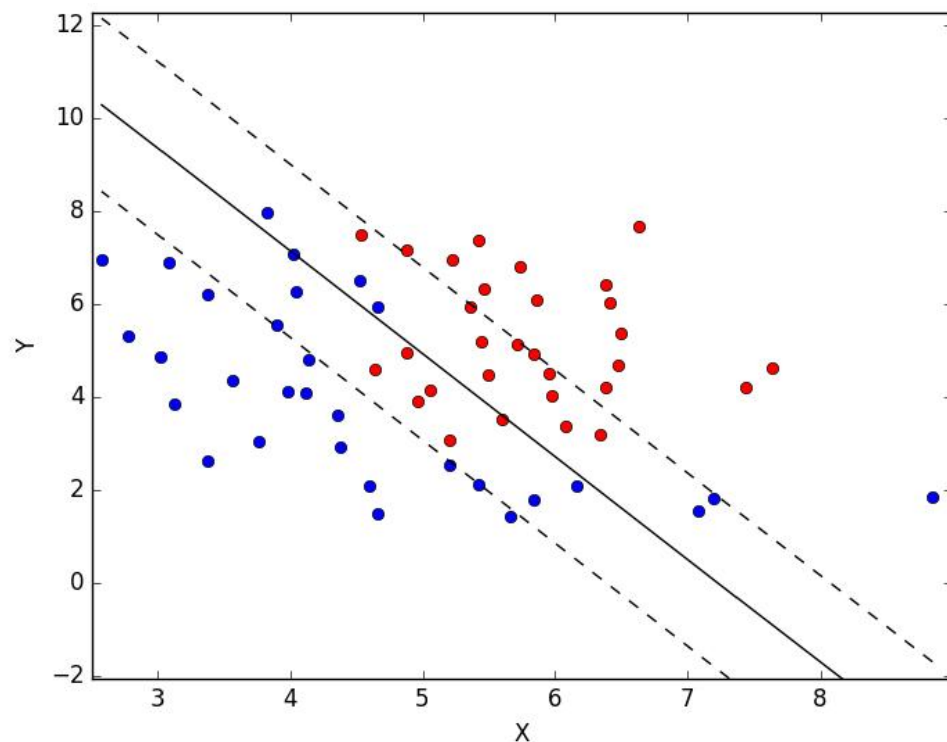
- **Result:**

Linear separate data:



10 Fold Validation accuracy: 1.0

Non-linear separate data:



Non-linear separate data:

10 Fold Validation accuracy: 0.783333333333

4. Polynomial and Gaussian kernel functions:

Kernel functions:

Polynomial: $K(x^{(i)}, x^{(j)}) = (x^{(i)T}x^{(j)} + 1)^q$

Gaussian : $K(x^{(i)}, x^{(j)}) = \exp\left(\frac{-1}{r}(x^{(i)} - x^{(j)})^T(x^{(i)} - x^{(j)})\right)$

Use Kernel function brings X to higher dimension.

■ Solve Dual for W_0

For linear SVM:

$$W_0 = y^{(i)} - W^T x^{(i)}$$

For Kernel SVM:

$$\begin{aligned}
W_0 &= y^{(i)} - W^T \phi(x^{(i)}) \quad \phi: \text{basis function} \\
\rightarrow W_0 &= \frac{1}{\#SV} \sum_{x^{(i)} \in SV} y^{(i)} - W^T \phi(x^{(i)}) \\
&= \frac{1}{\#SV} \sum_{x^{(i)} \in SV} y^{(i)} - \sum_{x^{(j)} \in SV} \phi(x^{(j)T}) \phi(x^{(i)}) \\
&= \frac{1}{\#SV} \sum_{x^{(i)} \in SV} y^{(i)} - \sum_{x^{(j)} \in SV} K(x^{(j)}, x^{(i)})
\end{aligned}$$

■ Classification:

$$\begin{cases} \sum_{x^{(i)}} \alpha_i y^{(i)} K(x^{(i)}, x) + W_0 > 0 \rightarrow C1 \\ \text{otherwise} \rightarrow C0 \end{cases}$$

■ Result:

Polynomial Kernel SVM accuracy:

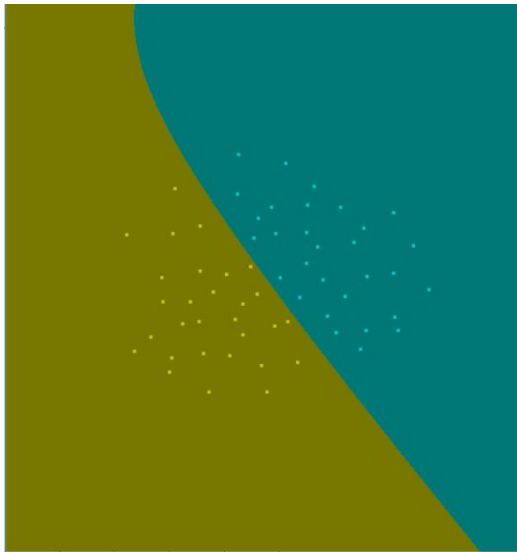
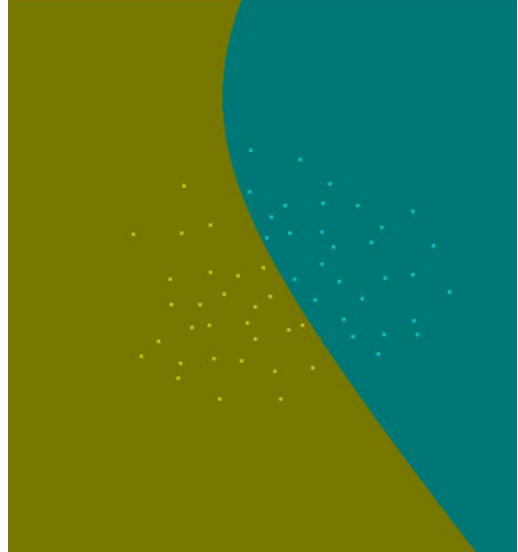
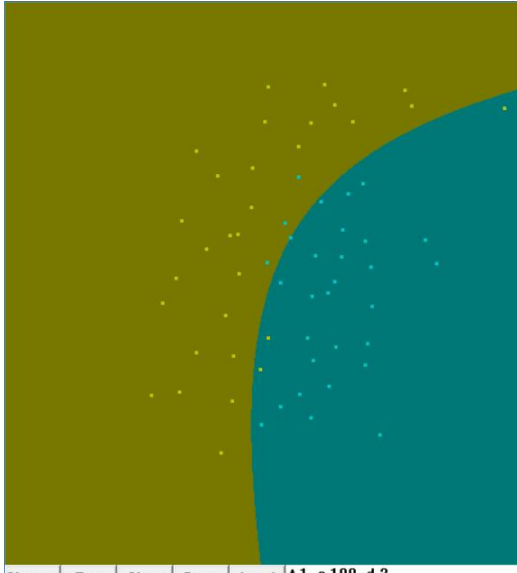
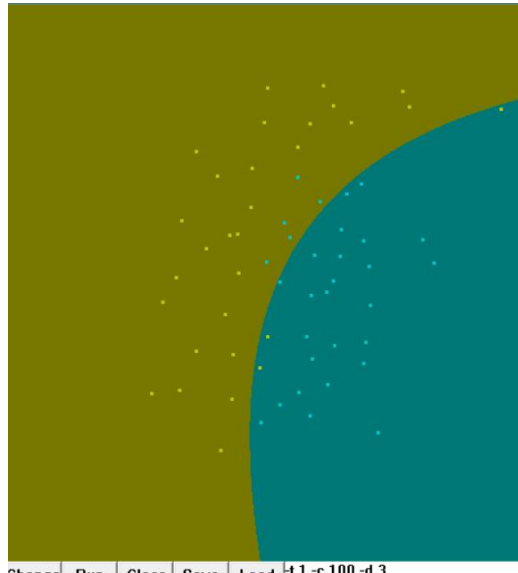
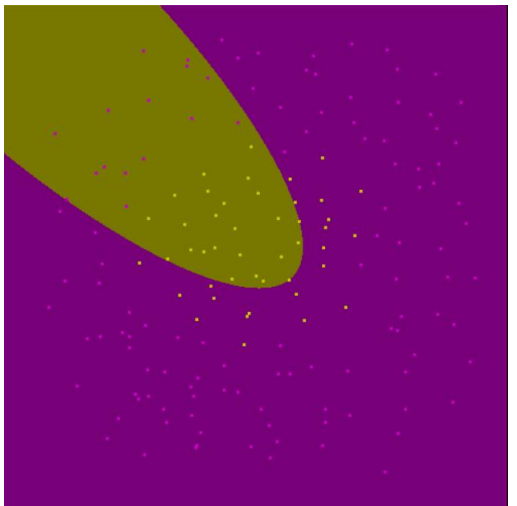
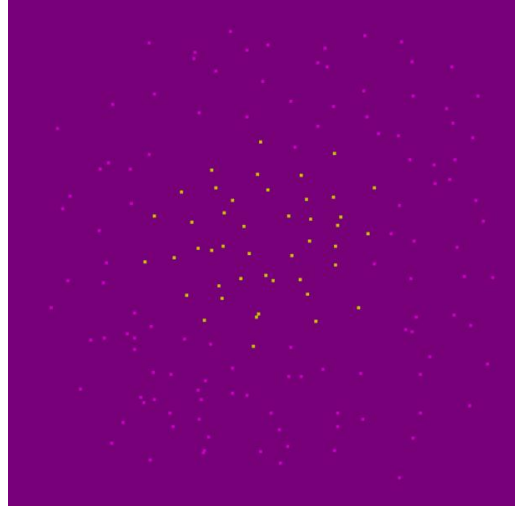
	Linear Sep	Non linear	circle
q = 2	0.583333333333	0.5	0.129779411765
q = 3	0.65	0.516666666667	0.716176470588

As the result, the accuracy is more accurate. However, the result is wrong compare to the test from libsvm.

One reason could W0 which I derived with fault equation, or my coding part with Kernel trick is wrong.

Libsvm test plot:

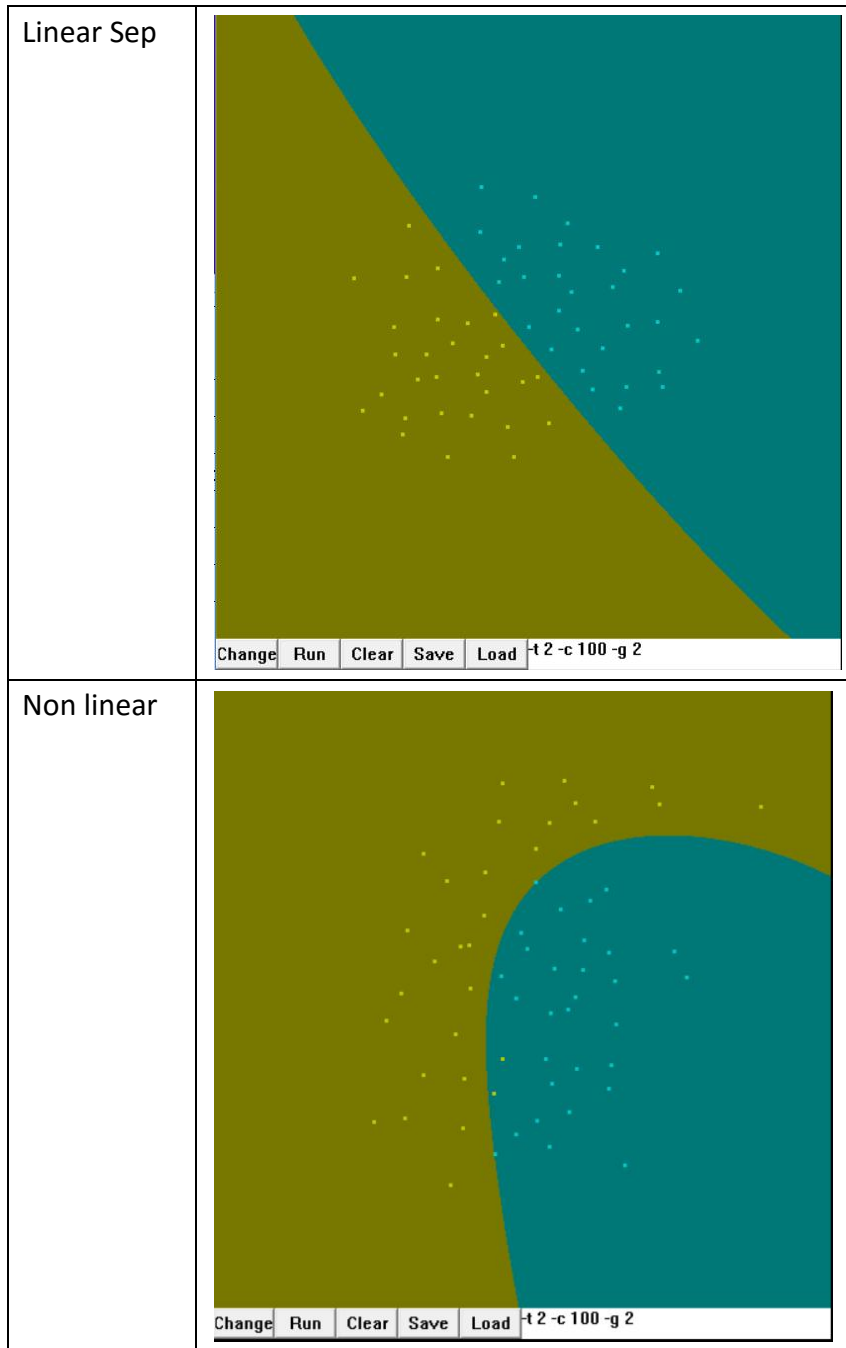
	q = 2	q = 3
--	-------	-------

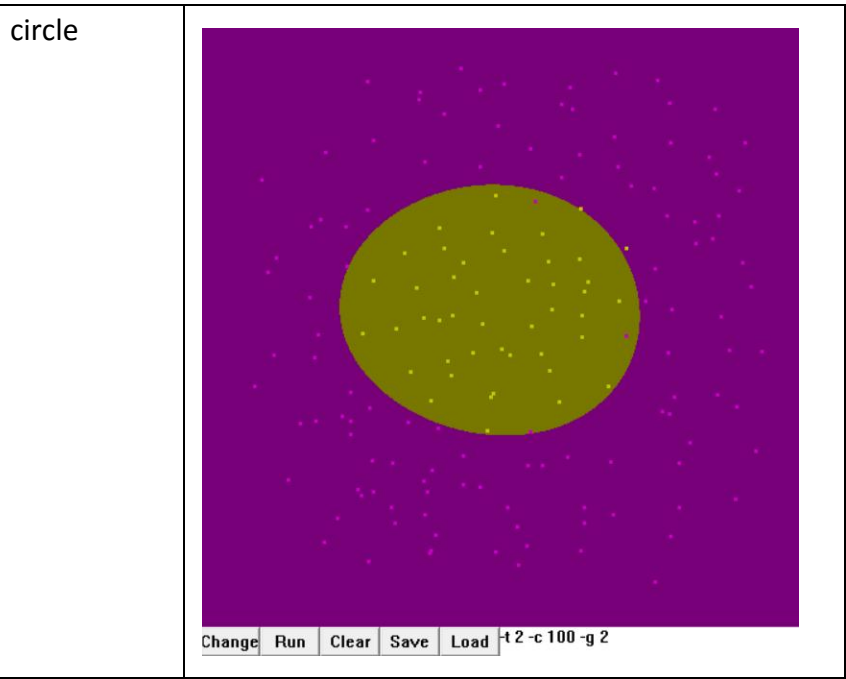
<div>Linea</div> <div>r Sep</div>	 <div> <div>Change</div> <div>Run</div> <div>Clear</div> <div>Save</div> <div>Load</div> <div>t1 -c 100 -d 2</div> </div>	 <div> <div>Change</div> <div>Run</div> <div>Clear</div> <div>Save</div> <div>Load</div> <div>t1 -c 100 -d 3</div> </div>
<div>Non</div> <div>linear</div>	 <div> <div>Change</div> <div>Run</div> <div>Clear</div> <div>Save</div> <div>Load</div> <div>t1 -c 100 -d 2</div> </div>	 <div> <div>Change</div> <div>Run</div> <div>Clear</div> <div>Save</div> <div>Load</div> <div>t1 -c 100 -d 3</div> </div>
<div>Circle</div>	 <div> <div>Change</div> <div>Run</div> <div>Clear</div> <div>Save</div> <div>Load</div> <div>t1 -c 100 -d 2</div> </div>	 <div> <div>Change</div> <div>Run</div> <div>Clear</div> <div>Save</div> <div>Load</div> <div>t1 -c 100 -d 3</div> </div>

radial basis function with $r = 2$:

Linear Sep	Non linear	circle
1.0	0.7666666666667	0.554779411765

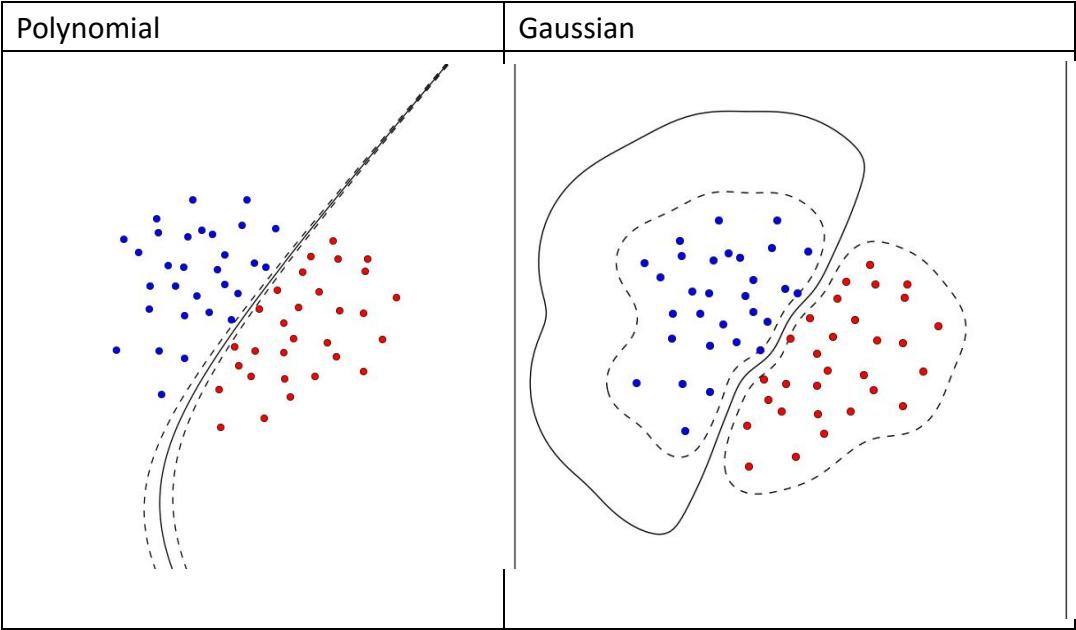
The result is also not corrects, it far more reach the test from libsvm



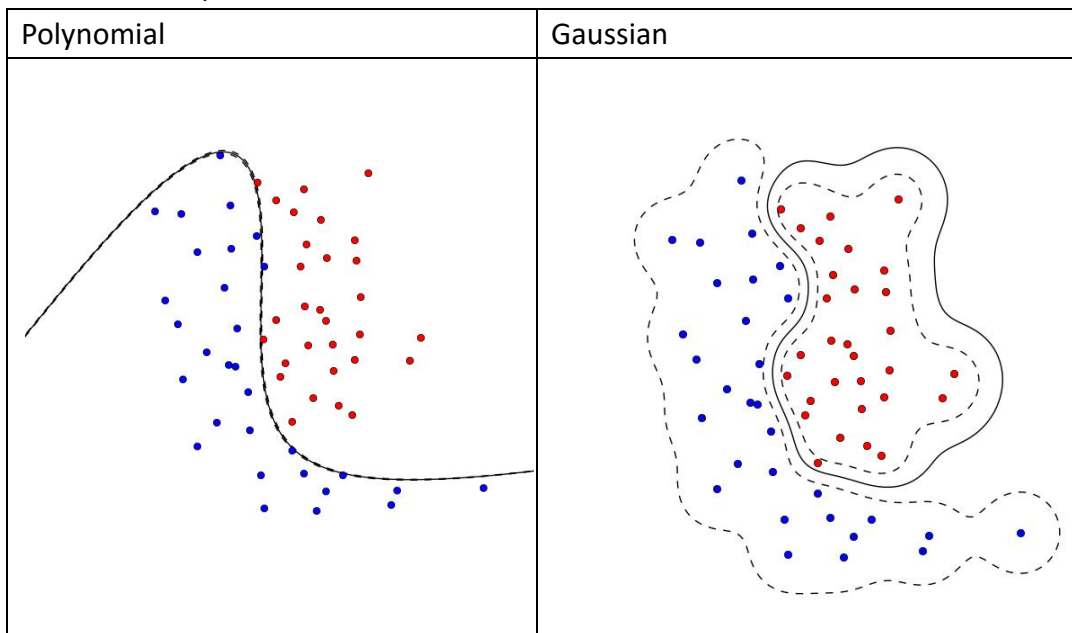


Plot by python SVC

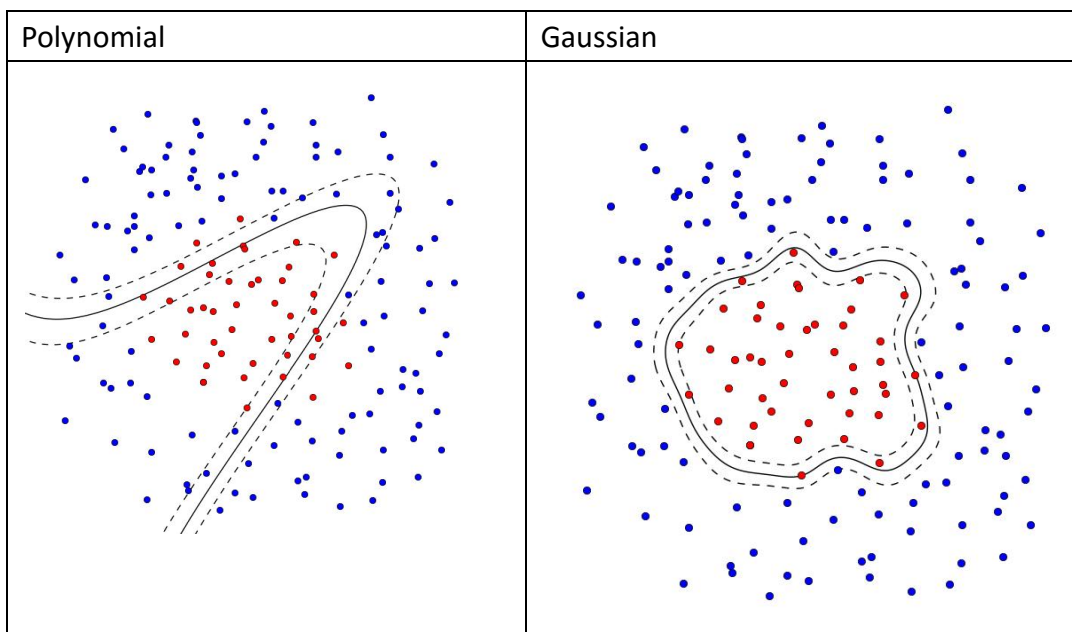
Linear separate data:



Non-Linear separate data:



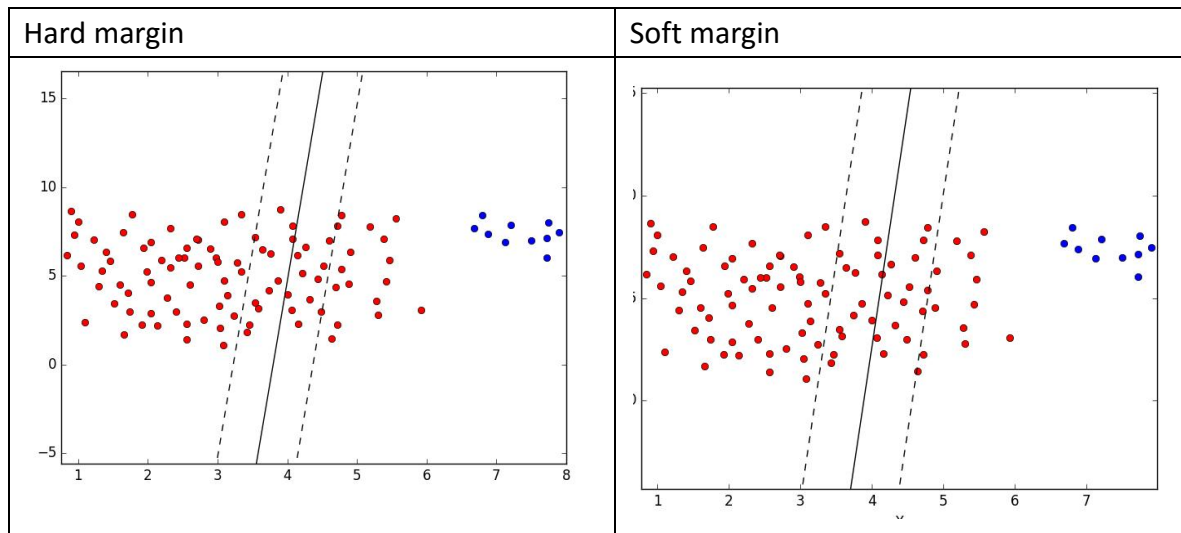
Circle like data:



- Test with 1 class with more data:

In this question I test hard margin and soft margin with linearly separable data.

The data ratio of 2 classes is 10:1



As the result, if one class had more data than one the other. The boundary line will close to the side which had more data. One way to eliminate this situation is to compute the rate of classes. And select the training data base on this ratio, such that it could avoid this problem happened.

● **Reference:**

1. data set UCI website. <http://archive.ics.uci.edu/ml/>
2. Gady Agam. Lecture note. Discriminative learning
3. Python plot website.
<http://www.cnblogs.com/wei-li/archive/2012/05/23/2506940.html>