1. Problem Statement

1.1 Support vector machine

In this question:

1.1.1 Find 2 data sets from UCI machine learning database one should be linearly separable the other should be linearly separable.

1.1.2 Implement SVM algorithm to these datasets and both applying the hard margin and the soft margin algorithms.

1.1.3 Applying kernel-based SVM to these datasets.

1.1.4 Implement the scikit-learn package to solve the problem.

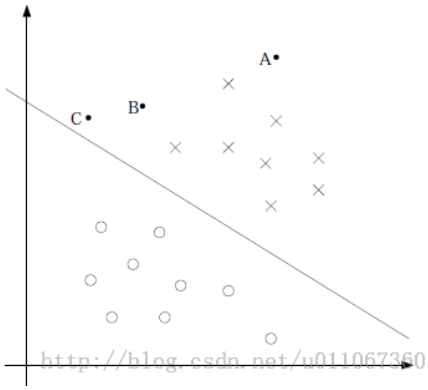
2 Proposed solution

2.1 It is a supervised learning model and related learning algorithms for analyzing data in classification and regression analysis. Given a set of training instances, each training instance being marked as belonging to one or the other of the two categories, the SVM training algorithm creates a model that assigns a new instance to one of the two categories, making it non-probabilistic Meta linear classifier.

2.1.1 Linear classification:

We usually want the process of classification to be a machine learning process. These data points do not need to be points in $\mathbb{R}^2$, but can be any point of $\mathbb{R}^n$ (a hyperplane, the example in 2D space is a straight line). We want to be able to separate these points through an (n-1) dimensional hyperplane, which is often referred to as a linear classifier. There are many classifiers that meet this requirement, but we also want to find the best-class plane, that is, the face that has the largest separation of data points belonging to two different classes, which is also called the maximum interval hyperplane.

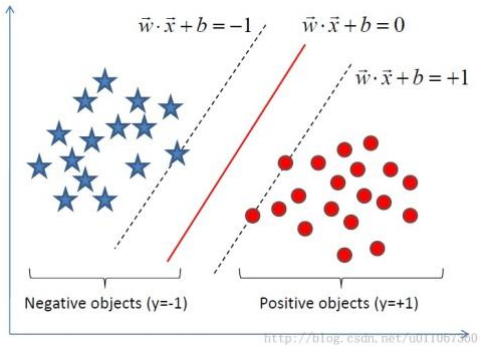
Let's start with one of the following diagrams:



The middle line is wx + b =0, and we emphasize that all points are as far as possible from the middle line. Consider the above three points A, B and C. From the figure we can determine that A is a × category, but C is not sure, and B is still ok. In this way we can conclude that we should be more concerned with the points close to the middle dividing line, so that they are as far away as possible from the middle line, rather than at all points. Because of that, make a part of the point close to the middle line in exchange for another part of the point farther away from the middle line. At the same time, this so-called hyperplane does separate the data points of the two different shapes. The data points on the hyperplane side correspond to y all -1 and on the other side all 1s.

We can make the classification function:

Obviously, if f(x) = 0, then x is the point on the hyperplane. We may wish to require that for all points satisfying f(x)<0, the corresponding y=is equal to -1 and f(x)=>0 corresponds Data point with y=1. As shown below:



The optimal hyperplane can have an infinite number of expressions, ie by arbitrary scaling w and b:

$$w^T \cdot x + b = 1$$

Where x is the point closest to the hyperplane, or the expression that gives the support vector is: y(wx + b) = 1

As mentioned above, we make the two types of points +1, -1, so when there is a new point x need to predict which category belongs to, we can use sgn(f(x)), we can predict, sgn Represents a symbolic function. When f(x) > 0, sgn(f(x)) = +1, when f(x) < 0, sgn(f(x)) = –1. We know that the distance from point x to the hyperplane (β, β_0) is:

$$\gamma = \frac{f(x)}{\|w\|}$$

Maximizing this expression is equivalent to minimizing ||w||, and since ||w|| is a monotonic function, we can add squares to it, and the previous coefficients, familiar students should easily see it. This formula is for convenience.

Finally maximize M to be converted to minimize the function under additional constraints:

$$\max \frac{1}{\|w\|} \quad \rightarrow \quad \min \frac{1}{2}\|w\|^2$$

$$\max \frac{1}{\|w\|}, \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

This is a LaGrange optimization problem. The weight vector W and the offset b of the optimal hyperplane can be obtained by the Lagrange multiplier method.

$$L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(w^T x_i + b) - 1)$$

Minimizing (2) can be rewritten as a constrained optimization problem with a differentiable objective function in the following way.

For each $i \in \{1, \ldots, n\}$ we introduce a variable $\zeta_i = \max(0, 1 - y_i(w \cdot x_i - b))$. Note that ζ_i is the smallest nonnegative number satisfying $y_i(w \cdot x_i - b) \geq 1 - \zeta_i$.

Thus we can rewrite the optimization problem as follows

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i + \lambda\|w\|^2$$
$$\text{subject to } y_i(w\cdot x_i - b) \geq 1-\zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i.$$

This is called the *primal* problem.
By solving for the Lagrangian dual of the above problem, one obtains the simplified problem

$$\text{maximize } f(c_1\ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (x_i\cdot x_j) y_j c_j,$$
$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

This is called the *dual* problem. Since the dual maximization problem is a quadratic function of the c_i subject to linear constraints, it is efficiently solvable by quadratic programming algorithms.

Here, the variables c_i are defined such that

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x}_i.$$

Moreover, $c_i = 0$ exactly when $\vec{x}_i$ lies on the correct side of the margin, and $0 < c_i < (2n\lambda)^{-1}$ when $\vec{x}_i$ lies on the margin's boundary. It follows that $\vec{w}$ can be written as a linear combination of the support vectors.

The offset, b, can be recovered by finding an $\vec{x}_i$ on the margin's boundary and solving

$$y_i(\vec{w}\cdot\vec{x}_i - b) = 1 \iff b = \vec{w}\cdot\vec{x}_i - y_i.$$

(Note that $y_i^{-1} = y_i$ since $y_i = \pm 1$.)

3 Implementation details

The core of the problem is to construct the equations in the dual constraint condition to satisfy the matrix required by the quadratic programming function. Use these matrices to calculate the parameter α_i:

P is the product of the Gram matrix and the label matrix;

When the kernel method is used, the Gram matrix becomes a matrix that measures the similarity of two parameters corresponding to the kernel method.

q is the m-row all -1 The parameter matrix of 1 corresponds to the parameter before the independent α_i;

A and ***b*** are used to satisfy the condition of $w_0 = 0$, the matrix A is the transpose of the label matrix, and the value of b is 0;

G and the matrix ***h*** are used to satisfy the inequality constraint:

Under the hard margin condition, only one inequality constraint matrix ***G*** is a diagonal matrix with a diagonal of all -1, and matrix ***h*** is a matrix with m rows all zero.

Under the soft margin condition, due to the dual constraint conditions, the matrix ***G*** and the matrix ***h*** are slightly different, wherein the matrix ***G*** is amplifying a stacking matrix $(2m \times n)$ of diagonal matrices with a front-m behavioral diagonal of all -1 and a diagonal matrix with a post-m behavior of all 1s. The matrix ***h*** becomes a stacking matrix $(2m \times 1)$ of the matrix of the first m rows all zero and the parameter c of the post m behavior controlling slack.

After obtaining α_i, w can be calculated according to the formula:

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

α_i indicates that each point violates the penalty value corresponding to the constraint. The larger the penalty value is, the closer the point is to the boundary or the more important. The n largest α_is are selected according to the formula to

calculate the coordinates of the corresponding support vector, and w_0 is calculated according to the formula:

$$w_0 = \frac{1}{\#\ SV}\sum_{x^{(i)}\in SV}(y^{(i)} - w^T x^{(i)}).$$

w, w_0 is that the slope and intercept of the discriminant function are obtained, and the discrimination can be performed.

Problem:

When calculating the α_i using the quadratic programming function, the matrix type of the parameter needs to satisfy the special matrix type in cvxopt.

When you pour data, data read from different files using different read-in methods will appear in a variety of data types, and you need to unify the data types for calculation.

Unfamiliar use of python.

Instruction:

When generating data, you can change the parameters of the *make_blob* function. Changing the variance of each class produces a linearly separable data set and a linearly inseparable data set.

The parameter c can be changed during use to adjust the slack of the soft margin slack function.

The parameters can be changed to select more/less SVs when SV is selected.

In the Gaussian kernel function, the σ can be changed to change the degree of dispersion of the data distribution.

4 Result and Discussion

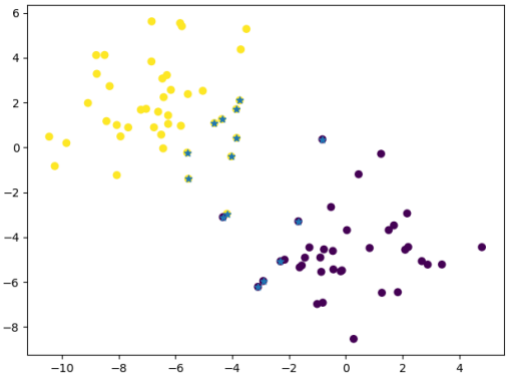
Randomly generate data based on gaussian distribution. There are 100 data in total, randomly separating data in to 80 training data and 20 testing data.

Marked points are recognized as support vector.

4.1 Hard margin:

4.1.1 Linear separable data:

standard deviation of two clusters = [1.0, 1.0]

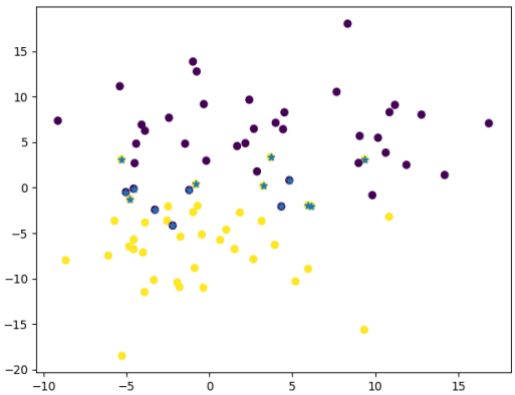



```
hard margin SVM accuracy is : 0.950000
```

4.1.2 Linear inseparable data:

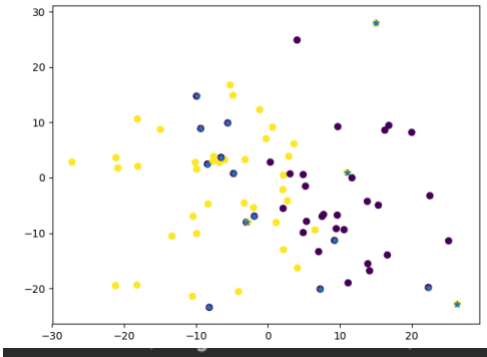
By changing the distribution parameter could make the data inseparable.

standard deviation of two clusters = [5.0, 5.0]




```
hard margin SVM accuracy is : 0.700000
```

standard deviation of two clusters = [10.0, 10.0]

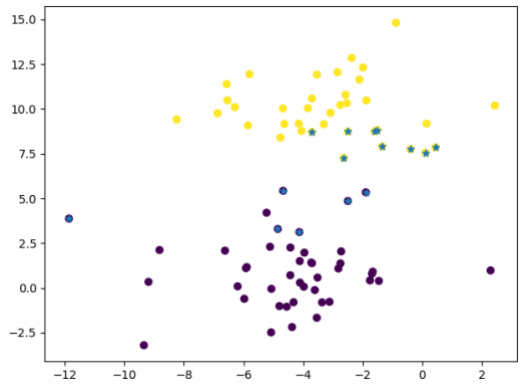


```
hard margin SVM accuracy is : 0.550000
```

4.2 Soft margin *(c = 0.5)*

4.2.1 Linear separable data:

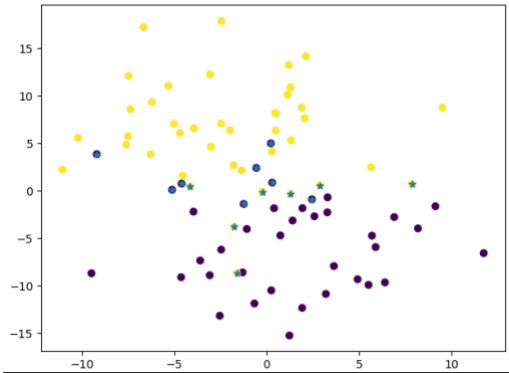
standard deviation of two clusters = [2.0, 2.0]



```
soft margin accuracy is : 1.000000
```

4.2.2 Linear inseparable data:

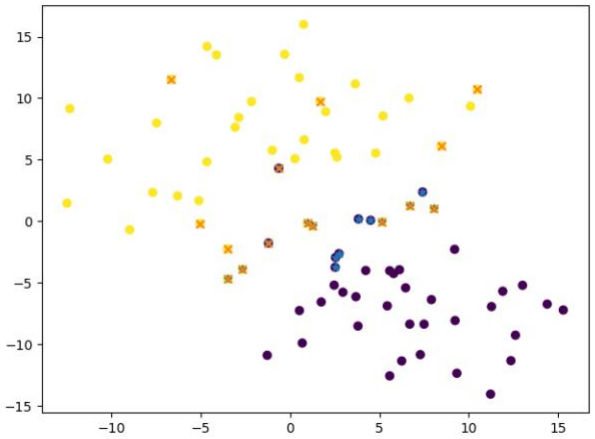
standard deviation of two clusters = [5.0, 5.0]



```
soft margin accuracy is : 0.850000
```

4.3 Soft margin and hard margin

standard deviation of two clusters = [5.0, 5.0]



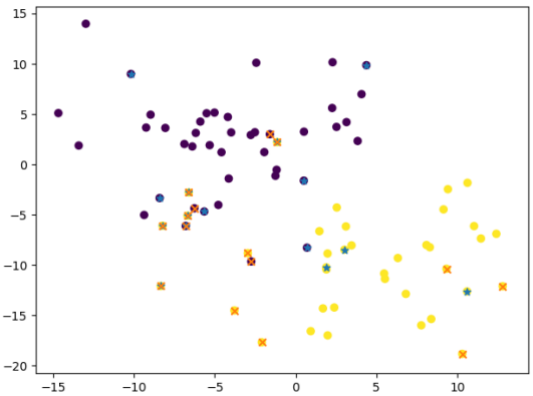
```
soft margin accuracy is : 0.850000
     pcost       dcost       gap    pres   dres
 0: -2.5711e+01 -6.5434e+01  3e+02  1e+01  3e+00
 1: -7.1790e+01 -1.3856e+02  3e+02  1e+01  2e+00
 2: -5.4215e+02 -6.8625e+02  2e+02  7e+00  1e+00
 3: -1.4293e+03 -1.6829e+03  3e+02  7e+00  1e+00
 4: -2.4241e+03 -2.8096e+03  4e+02  6e+00  1e+00
 5: -6.1070e+03 -6.8749e+03  8e+02  6e+00  1e+00
 6: -6.5347e+03 -7.3500e+03  8e+02  6e+00  1e+00
 7: -2.9268e+04 -3.1501e+04  2e+03  6e+00  1e+00
 8: -9.1308e+04 -9.7223e+04  6e+03  6e+00  1e+00
 9: -1.4803e+06 -1.4872e+06  7e+03  6e+00  1e+00
10: -7.5407e+06 -7.5749e+06  3e+04  6e+00  1e+00
11: -1.2838e+09 -1.2852e+09  1e+06  6e+00  1e+00
12: -7.0772e+09 -7.0847e+09  7e+06  6e+00  1e+00
13: -4.7168e+10 -4.7217e+10  5e+07  6e+00  1e+00
14: -6.7192e+10 -6.7261e+10  7e+07  6e+00  1e+00
15: -1.0546e+11 -1.0557e+11  1e+08  6e+00  1e+00
Terminated (singular KKT matrix).
hard margin accuracy is : 0.450000
```

'x' is the SV of the soft margin.

'star' and 'colored' are the SV of hard margin.

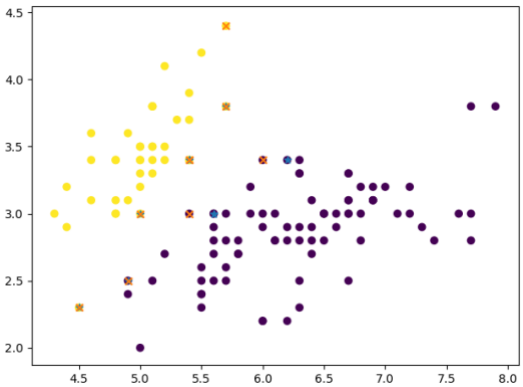
4.4 Kernel method ('x'—Gaussian SV, 'star' --Polynomial SV)

Test on the dataset generated.



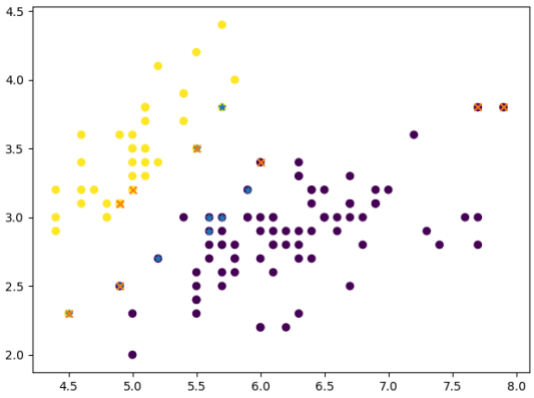
```
poly kernel accuracy is : 0.600000
     pcost       dcost       gap    pres   dres
 0: -2.7489e+01 -7.0471e+01  4e+01  6e-16  2e+00
 1: -3.3528e+01 -3.7097e+01  4e+00  1e-15  3e-01
 2: -3.5876e+01 -3.6389e+01  5e-01  1e-15  2e-02
 3: -3.5881e+01 -3.5888e+01  8e-03  2e-15  2e-04
 4: -3.5881e+01 -3.5881e+01  8e-05  2e-15  2e-06
 5: -3.5881e+01 -3.5881e+01  8e-07  1e-15  2e-08
Optimal solution found.
Gaussian kernel accuracy is : 0.800000
```

Test the iris data set. $\sigma = 0.5$



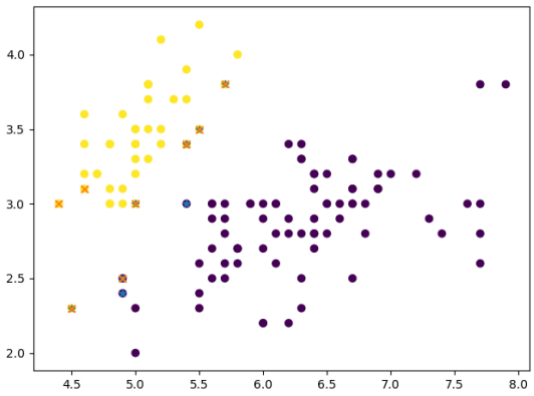
```
poly kernel accuracy is : 1.000000
     pcost       dcost       gap    pres   dres
 0: -7.0730e+00 -2.2337e+01  2e+02  1e+01  2e+00
 1: -4.1364e+00 -2.2029e+01  3e+01  6e-01  1e-01
 2: -7.1972e+00 -1.4445e+01  9e+00  1e-01  2e-02
 3: -9.6608e+00 -1.2603e+01  3e+00  2e-15  9e-16
 4: -1.0889e+01 -1.1575e+01  7e-01  1e-15  8e-16
 5: -1.1269e+01 -1.1410e+01  1e-01  1e-15  9e-16
 6: -1.1372e+01 -1.1379e+01  7e-03  1e-15  9e-16
 7: -1.1378e+01 -1.1378e+01  2e-04  2e-15  9e-16
 8: -1.1378e+01 -1.1378e+01  7e-06  2e-15  8e-16
Optimal solution found.
Gaussian kernel accuracy is : 0.750000
```

By addming the sigma in gaussian kernel, it will raise the accuracy.



$\sigma = 2$

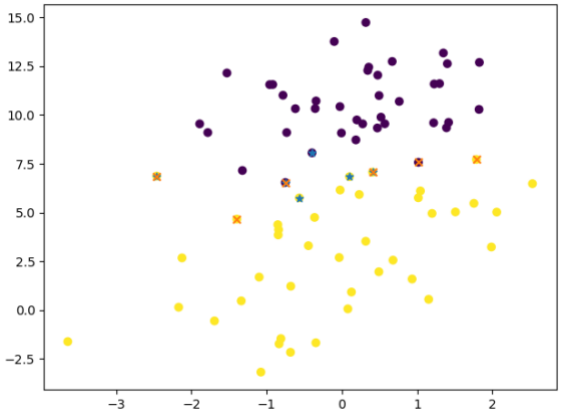
```
poly kernel accuracy is : 1.000000
     pcost       dcost       gap    pres   dres
 0: -1.7854e+01 -4.6912e+01  3e+02  1e+01  2e+00
 1: -2.7246e+01 -5.5519e+01  1e+02  6e+00  8e-01
 2: -5.5698e+01 -8.4105e+01  1e+02  6e+00  7e-01
 3: -1.2954e+02 -1.5698e+02  1e+02  4e+00  5e-01
 4: -1.4039e+02 -1.4573e+02  2e+01  4e-01  5e-02
 5: -1.3828e+02 -1.3925e+02  3e+00  6e-02  7e-03
 6: -1.3808e+02 -1.3812e+02  5e-02  6e-04  7e-05
 7: -1.3810e+02 -1.3810e+02  5e-04  6e-06  7e-07
 8: -1.3810e+02 -1.3810e+02  5e-06  6e-08  7e-09
Optimal solution found.
Gaussian kernel accuracy is : 0.850000
```



$\sigma = 10$

```
poly kernel accuracy is : 1.000000
     pcost       dcost       gap    pres   dres
 0: -6.6774e+01 -1.6849e+02  3e+02  1e+01  2e+00
 1: -1.4456e+02 -2.4526e+02  1e+02  6e+00  1e+00
 2: -1.7762e+02 -2.8305e+02  2e+02  6e+00  1e+00
 3: -3.7776e+02 -4.9659e+02  2e+02  5e+00  1e+00
 4: -1.4212e+03 -1.6060e+03  3e+02  5e+00  9e-01
 5: -2.4068e+03 -2.7871e+03  5e+02  5e+00  9e-01
 6: -4.3024e+03 -5.3152e+03  1e+03  4e+00  7e-01
 7: -5.7714e+03 -6.7371e+03  1e+03  1e+00  2e-01
 8: -5.9273e+03 -5.9671e+03  4e+01  4e-02  7e-03
 9: -5.9320e+03 -5.9324e+03  4e-01  4e-04  7e-05
10: -5.9321e+03 -5.9321e+03  4e-03  4e-06  7e-07
11: -5.9321e+03 -5.9321e+03  4e-05  4e-08  7e-09
Optimal solution found.
Gaussian kernel accuracy is : 1.000000
```

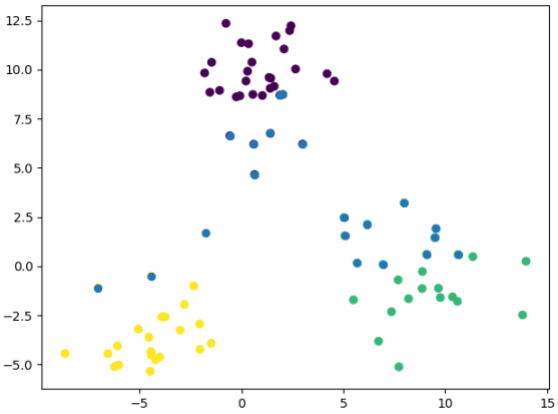
Test on the TestSet.



$\sigma = 2$

```
poly kernel accuracy is : 0.600000
     pcost       dcost       gap    pres   dres
 0: -9.8832e+00 -3.0109e+01  2e+02  1e+01  2e+00
 1: -2.4687e+01 -4.9928e+01  1e+02  4e+00  1e+00
 2: -7.4737e+01 -1.0562e+02  1e+02  4e+00  1e+00
 3: -2.7879e+02 -3.4283e+02  1e+02  3e+00  7e-01
 4: -3.2158e+02 -3.8273e+02  1e+02  1e+00  3e-01
 5: -3.2675e+02 -3.3772e+02  2e+01  2e-01  4e-02
 6: -3.2600e+02 -3.2643e+02  6e-01  5e-03  1e-03
 7: -3.2610e+02 -3.2611e+02  1e-02  5e-05  1e-05
 8: -3.2611e+02 -3.2611e+02  1e-04  5e-07  1e-07
 9: -3.2611e+02 -3.2611e+02  1e-06  5e-09  1e-09
Optimal solution found.
Gaussian kernel accuracy is : 0.800000
```

Test the 3-class data:



5 Reference

A lot of information about python usage is referenced from the CSDN and stack overflow websites.