INF552: Programming Assignment 6 [SVMs]

Yi-Li Chen

Part 1: Implementation

- All imported library

```
import csv
import numpy as np
import matplotlib.pyplot as plt
import cvxopt
```

Part (a) Find the fattest margin line that separates the points in linsep.txt.

- Read the file, 'linsep.txt', into an array.

```
linesp_data = []
with open('linsep.txt') as f:
    file = csv.reader(f)
    for line in file:
        linesp_data.append(line)
linesp_data = np.array(linesp_data).astype('float64')
X = linesp_data[:,:2]
y = linesp_data[:,2]
n_samples, n_features = X.shape
```

- Solve the problem using a Quadratic Programming solver with CVXOPT library. Thus, we need to calculate and prepare the values of matrices for the solver.1

```
K = np.zeros((n_samples, n_samples))
for i in range(n_samples):
    for j in range(n_samples):
        K[i,j] = np.dot(X[i], X[j])
Q = matrix(np.outer(y, y) * K)   # Q = y y X^T X
q = matrix(np.ones((n_samples, 1)) * -1)
A = matrix(y.reshape(1, -1))
b = matrix(np.zeros(1))
G = matrix(np.eye(n_samples) * -1)
h = matrix(np.zeros(n_samples))
solution = solvers.qp(Q, q, G, h, A, b)
```

- Once the solver reach convergence, we can find the support vector by the index of which data point of α is greater than zero.

```python
alphas = np.array(solution['x'])
idx = (alphas > 1e-4).flatten()
alphas = alphas[idx]
sv = X[idx]
sv_y = y[idx]
```

```
---alphas---
[[33.73875192]
 [ 1.29468506]
 [32.4440672 ]]
-----sv-----
[[0.24979414 0.18230306]
 [0.3917889  0.96675591]
 [0.02066458 0.27003158]]
----sv_y----
[ 1. -1. -1.]
```

- Then we can easily obtain the parameters of w and b.

```python
# Retrieve w (exclude alpha = 0)
w = np.zeros(n_features)
for n in range(len(alphas)):
    w += alphas[n]*sv_y[n]*sv[n]
# Retrieve b
b = sv_y[0] - np.dot(sv[0], w)
```

→ $w = [7.25005616 \quad -3.86188932]$

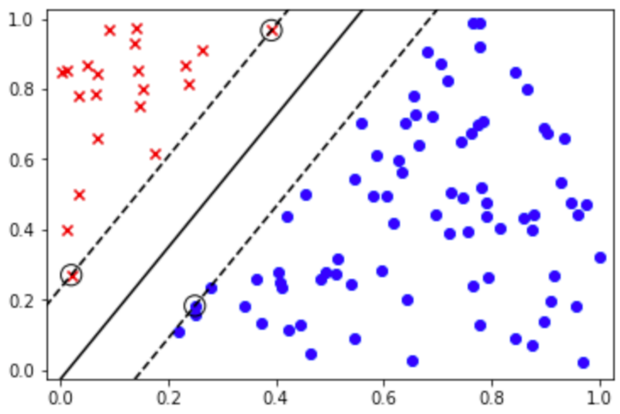
→ $b = -0.1069872903246214$

→ Therefore, we can get the equation of the line:

$$[7.25005616 \quad -3.86188932]^T x - 0.1069872903246214 = 0$$

- Plot the result and circle the support vectors

$$Support\ vectors: \begin{matrix}[[0.24979414 & 0.18230306] \\ [0.39178890 & 0.96675591] \\ [0.02066458 & 0.27003158]]\end{matrix}$$



Part (b) Using a kernel function of your choice along with the same Quadratic Programming solver, find the equation of a curve that separates the points in nonlinsep.txt.

- Read the file, 'nonlinsep.txt', into an array.

```python
nonlinesp_data = []
with open('nonlinsep.txt') as f:
    file = csv.reader(f)
    for line in file:
        nonlinesp_data.append(line)
nonlinesp_data = np.array(nonlinesp_data).astype('float64')
X = nonlinesp_data[:,:2]
y = nonlinesp_data[:,2]
n_samples, n_features = X.shape
```

- Define the RBF as my kernel function and set gamma as 0.01

```python
def RBF_kernel_function(x1, x2, gamma=0.01):
    return np.exp(-gamma * np.linalg.norm(x1 - x2) ** 2)
```

→ The kernel function that I used is

$$K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$$

- Solve the problem with the same Quadratic Programming solver as above. Thus, we need to calculate and prepare the values of matrices for the solver as well.

```python
K = np.zeros((n_samples, n_samples))
for i in range(n_samples):
    for j in range(n_samples):
        K[i,j] = RBF_kernel_function(X[i], X[j])
Q = matrix(np.outer(y, y) * K)  # Q from kernel function
q = matrix(np.ones((n_samples, 1)) * -1)
A = matrix(y.reshape(1, -1))
b = matrix(np.zeros(1))
G = matrix(np.eye(n_samples) * -1)
h = matrix(np.zeros(n_samples))
solution = solvers.qp(Q, q, G, h, A, b)
```

- Again, once the solver reach convergence, we can find the support vector by the index of which data point of α is greater than zero.

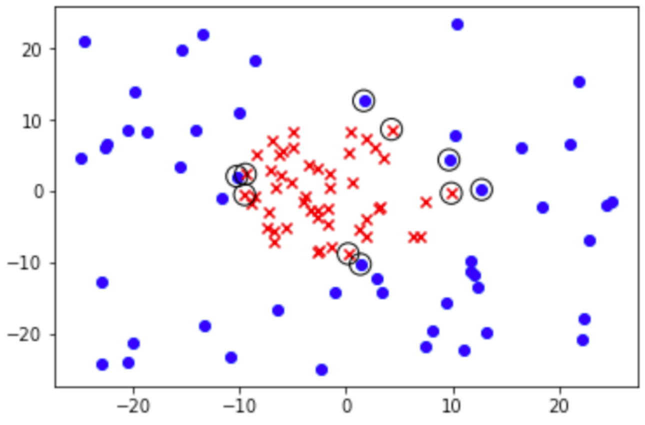
```python
alphas = np.array(solution['x'])
idx = (alphas > 1e-4).flatten()
```

```
alphas = alphas[idx]
sv = X[idx]
sv_y = y[idx]
```

```
---alphas---
[[  4.90610978]
 [154.79716375]
 [  6.06570051]
 [ 11.36810383]
 [  9.05592846]
 [ 16.36480164]
 [141.11823703]
 [  9.22492878]
 [ 12.26624767]
 [  7.21879402]]
-----sv-----
[[ 12.74780931   0.19913032]
 [-10.260969     2.07391791]
 [  1.66404809  12.68562818]
 [  1.3393313  -10.29098822]
 [  9.67917724   4.3759541 ]
 [ -9.53754332  -0.51895777]
 [ -9.46760885   2.36139525]
 [  0.20162846  -8.81260121]
 [  9.90143538  -0.31483149]
 [  4.27289989   8.67079427]]
----sv_y----
[ 1.  1.  1.  1.  1. -1. -1. -1. -1. -1.]
```

- Plot the result and circle the support vectors

$$
\mathit{Support\ vectors}: \begin{matrix}[[\ 12.74780931 & 0.19913032] \\ [-10.2609690 & 2.07391791] \\ [\ 1.66404809 & 12.68562818] \\ [\ 1.3393313 & -10.29098822] \\ [\ 9.67917724 & 4.3759541\ ] \\ [-9.53754332 & -0.51895777] \\ [-9.46760885 & 2.36139525] \\ [\ 0.20162846 & -8.81260121] \\ [\ 9.90143538 & -0.31483149] \\ [\ 4.27289989 & 8.67079427]]\end{matrix}
$$



With the help of Quadratic Programming solver, it is really not that hard to finish this assignment. Although it is the first time for me to use the CVXOPT library, it is still not hard if I prepare all the matrices well that the solver needed. Once it converges, we can easily obtain the w and b that professor have taught us in the courses.