CSE 353
HomeWork 7

JeongYoon Lee (114133199)

a) Introduction. Brief summary of what you think the assignment is about

SVM is a model that finding decision boundary line for classification. Training data are classified by decision boundaries, and it is important to set these boundaries well. A good decision boundary should be as far away from the data as possible. And the support vector means the data points closest to this decision boundary, which plays a decisive role in defining the boundary.

This assignment is to implement linear support vector machine using 20 training sample dataset. The training set X has 2 dimension, and the label Y is {-1, 1}. I will find the decision boundary line and the largest margin using SVM model. Also, I will find the support vector which is closest point with decision boundary line when the margin is largest.

b) Method. Brief outline of your (algorithmic) approach

- Find H, Phi, A, c and find q

$$q = \begin{bmatrix} b \\ \vec{w} \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}, \qquad q = quadprog(H, \emptyset, A, C)$$

$$H = \begin{bmatrix} 0 & \vec{0}_{1*d} \\ \vec{0}_{d*1} & I_{d*d} \end{bmatrix}$$

$$\emptyset = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}_{(d+1)*1} \qquad A = \begin{bmatrix} \vdots \\ -y_n, -y_n x_n^T \end{bmatrix}_{N*(d+1)}$$

$$c = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}_{N*1}$$

- Find support vector
  Support Vectors : samples on boundary located the large-margin hyperplane.
  → If (y_n(W.T*x_n + b) ==1), x_n is support vector.
  → I set the tolerance 0.01. So the acceptable range is 0.99 to 1.01

- Margin(W , b) == 1/Size_of_W_vector

  Decision boundary : w^T*x + b = 0
  Two lines along the support vectors : w^T*x + b = 1 , w^T*x + b = -1

c) Experiments. Tables and/or pictures of intermediate and final results that convince us that the program does what you think it does.

1. Implement and apply the linear support vector machine

```
d = 2
N = len(Y)
H = np.eye(d+1)
H[0][0] = 0
Phi = np.zeros(d+1)
c = np.ones(N) * -1

A = X * Y
A = np.append(A, [A[1]], axis = 0)
A[1] = A[0]
A[0] = Y
A = A * (-1)
```

[Make H, Phi, A, c]

```
H
[ 0.00e+00  0.00e+00  0.00e+00]
[ 0.00e+00  1.00e+00  0.00e+00]
[ 0.00e+00  0.00e+00  1.00e+00]

Phi
[ 0.00e+00]
[ 0.00e+00]
[ 0.00e+00]

A
[ 1.00e+00  3.50e-01  8.30e-01]
[-1.00e+00 -5.90e-01 -5.50e-01]
[-1.00e+00 -9.20e-01 -2.90e-01]
[-1.00e+00 -7.60e-01 -7.50e-01]
[ 1.00e+00  3.80e-01  5.70e-01]
[-1.00e+00 -8.00e-02 -5.00e-02]
[ 1.00e+00  5.30e-01  7.80e-01]
[-1.00e+00 -9.30e-01 -1.30e-01]
[-1.00e+00 -5.70e-01 -4.70e-01]
[ 1.00e+00  1.00e-02  3.40e-01]
[ 1.00e+00  1.60e-01  7.90e-01]
[ 1.00e+00  3.10e-01  5.30e-01]
[ 1.00e+00  1.70e-01  6.00e-01]
[ 1.00e+00  2.60e-01  6.50e-01]
[ 1.00e+00  6.90e-01  7.50e-01]
[-1.00e+00 -4.50e-01 -8.00e-02]
[ 1.00e+00  2.30e-01  9.10e-01]
[ 1.00e+00  1.50e-01  8.30e-01]
[ 1.00e+00  5.40e-01  1.00e+00]
[ 1.00e+00  8.00e-02  4.40e-01]

c
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
[-1.00e+00]
```

[Result]

```
sol1 = solvers.qp( H, Phi, A, c)
# print(sol1)
print()
print("q")
print(sol1['x'])
W = [sol1['x'][1],sol1['x'][2]]
b = sol1['x'][0]
```

[Find q ,W and b]

```
q
[ 1.02e-01]
[ 2.86e+01]
[-2.78e+01]
```

[Result] → [b,W1,W2].T

2. Identify which training samples in the dataset are support vectors

```
flag = (Y[i] * ((W[0] * X[0][i]+W[1] * X[1][i]) + b))
if((flag<1.01) and (flag>0.99)) :
  print(">>> x :(",X[0][i] ,",", X[1][i], ') y:',Y[i])
  n = (Y[i] * (W[0] * X[0][i]+W[1] * X[1][i]+b))/np.linalg.norm(W)
  print(">>> margin: ",n)
  print()
```

[Find support vector and each margin]

```
----- (b) Support vectors -----
>>> x :( 0.76 , 0.75 ) y: 1.0
>>> margin:  0.02510477823107326

>>> x :( 0.08 , 0.05 ) y: 1.0
>>> margin:  0.025104778382102136

>>> x :( 0.69 , 0.75 ) y: -1.0
>>> margin:  0.02510477842152438
```

[Support vectors and Distance between each support vector points and decision boundary line]

3. Compute the largest margin you achieved from your SVM.

```
margin_through_w = np.linalg.norm(W)
margin_through_w = 1/margin_through_w
print("margin through W : " , margin_through_w)
print()
```

[Calculate margin through W vector size]

```
margin through W :  0.025104778432969318
```

[Margin using W in my svm model]

```
----- (c) Largest Margin -----
x :( 0.69 , 0.75 ) y: -1.0
largest margin :  0.02510477842152438
```

[Largest Margin among support vectors that I found in the (c)-2]

>>> The largest margin that I found with support vectors is almost same with margin that I found with W. This difference exist because I set tolerance 0.01 when I found support vectors.

4. Visualize the training data, highlight the support vectors, plot the decision boundary line and the two lines along the support vectors.

```
Wx = ax.get_xlim()
Wm = W[0]/W[1]
Wy = [((-1)*Wm*num)-b/W[1] for num in Wx]
```

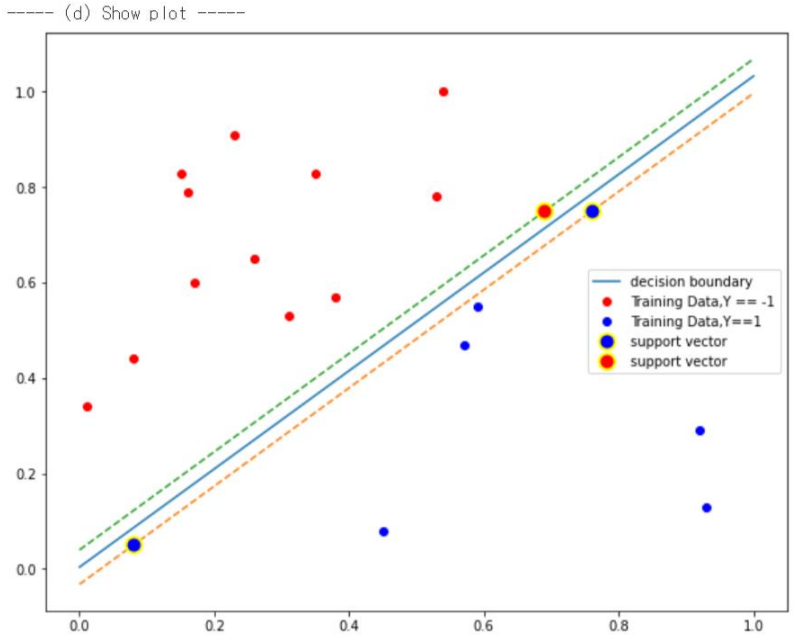
[Plot decision boundary line]

```
Wy = [((-1)*Wm*num)+(1-b)/W[1] for num in Wx]
plt.plot(Wx, Wy, linestyle = 'dashed')
```

[Plot line along the support vectors: w^T*x + b = 1]

```
Wy = [((-1)*Wm*num)+(-1-b)/W[1] for num in Wx]
plt.plot(Wx, Wy, linestyle = 'dashed')
```

[Plot line along the support vectors: w^T*x + b = -1]



d) **Discussions and Conclusions. Any design decisions you had to make and your experimental observations. What do you observe about the behavior of your program when you run it? Does it seem to work the way you think it should? Play around a little with different setting to see what happens. Note, your open-ended exploration is highly valued.**

Since our given data in this assignment was linear separable data, it was possible to divide two data classes by a single decision boundary line. In this case, we does not include any outliers, but it has possibility that overfitting may occur when used in other dataset with this way. Therefore, next time, I want to try soft margin with data with more vague boundaries. Overfitting and underfitting should be carefully considered in the range that allows outliers.

In the SVM library provided by Python sklearn, when data is not linearly separable, other non-linear boundaries can be found by putting “poly” option as the kernel parameter value. This can also be solved through the feature transform we did in the last assignment without using sklearn's library.