Homework 2

请将完成的 pdf 文件发送至 12031145@mail.sustech.edu.cn 邮箱中，文件命名为学号+姓名。

Part 1:

Consider the dataset consisting of points (x, y), where x is a real value, and $y \in \{-1, 1\}$ is the class label. There are only three points $(x_1, y_1) = (-1, 1)$, $(x_2, y_2) = (1, 1)$, $(x_3, y_3) = (0, -1)$. Let the feature mapping $\boldsymbol{\phi}(u) = [u, u^2]^T$, corresponding to the kernel function $k(u, v) = uv + u^2v^2$.

Question 1 Write down the primal and dual formulations of SVM for this dataset in the transformed two-dimensional feature space based on $\boldsymbol{\phi}(\cdot)$. Note that we assume the data points are separable and set the hyperparameter C to be $+\infty$, which forces all slack variables (ξ) in the primal formulation to be 0 (and thus can be removed from the optimization).

What to submit: primal and dual formulations of optimization objectives with dataset values substituted in. The optimization objectives should include constraints.

Question 2 Next, solve the dual formulation. Based on that, derive the primal solution.

What to submit: Optimal $\alpha_1^*, \alpha_2^*, \alpha_3^*$ in dual solution, and optimal w_1^*, w_2^*, b^* in primal solution.

Part 2:

学习使用 CVXopt 包

Solving QP with CVXopt

$$\begin{aligned} \min_{x} \quad & \tfrac{1}{2} x^\top P x + q^\top x \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b \end{aligned}$$

举例

$$\begin{aligned} \min_{x,y} \quad & \tfrac{1}{2} x^2 + 3x + 4y \\ \text{subject to} \quad & x, y \geq 0 \\ & x + 3y \geq 15 \\ & 2x + 5y \leq 100 \\ & 3x + 4y \leq 80 \end{aligned}$$

转换标准形式，即可调用 CVXopt 包处理。

$$\min_{x,y} \quad \frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix}^{\top}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix}^{\top}\begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -3 \\ 2 & 5 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \preceq \begin{bmatrix} 0 \\ 0 \\ -15 \\ 100 \\ 80 \end{bmatrix}$$

构建对应矩阵：

```
P = matrix([[1.0,0.0],[0.0,0.0]])
q = matrix([3.0,4.0])
G = matrix([[-1.0,0.0,-1.0,2.0,3.0],[0.0,-1.0,-3.0,5.0,4.0]])
h = matrix([0.0,0.0,-15.0,100.0,80.0])
```

Solving with **Linear Quadratic Solver**:

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^{T}\begin{bmatrix} d_1 d_1 \mathbf{x}_1^T \mathbf{x}_1 & d_1 d_2 \mathbf{x}_1^T \mathbf{x}_2 & \cdots & d_1 d_n \mathbf{x}_1^T \mathbf{x}_n \\ d_2 d_1 \mathbf{x}_2^T \mathbf{x}_1 & d_2 d_2 \mathbf{x}_2^T \mathbf{x}_2 & \cdots & d_2 d_n \mathbf{x}_2^T \mathbf{x}_n \\ \vdots & \vdots & \ddots & \vdots \\ d_n d_1 \mathbf{x}_n^T \mathbf{x}_1 & d_n d_2 \mathbf{x}_n^T \mathbf{x}_2 & \cdots & d_n d_n \mathbf{x}_n^T \mathbf{x}_n \end{bmatrix}\boldsymbol{\alpha} + (-\mathbf{1})^T\boldsymbol{\alpha}$$

$$\text{s.t.} \quad \mathbf{d}^T\boldsymbol{\alpha} = 0 \quad \text{and} \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq [\lambda, \lambda, \ldots, \lambda]^T$$

构建对应 P，q，G，h，A，b 矩阵，使用 CVXopt 包求解。

代码部分：需要补充代码及注释（_下划线及#处）

import matplotlib

import matplotlib.pyplot as plt

import numpy as np

from cvxopt import matrix, solvers

from sklearn import svm

'''

Please install the package first

complete the code and add relevant comments

y is equal to d in slides, C is equal to lambda in slides.

```
'''
def custom_svm_fit(x, y, C):
    # slove alpha
    n_samples, n_features = x.shape
    #
    K = np.zeros((n_samples, n_samples))
    for i in range(n_samples):
        for j in range(n_samples):
            K[i, j] = _____
    #
    P = matrix(_____ * K)
    #
    q = matrix(_____)
    #
    tmp1 = np.diag(_____ (n_samples) * _____)
    tmp2 = _____ (n_samples)
    G = matrix(np.vstack((tmp1, tmp2)))
    #
    tmp3 = _____ (n_samples)
    tmp4 = _____ (n_samples) * C
    h = matrix(np.hstack((tmp3, tmp4)))
    #
    A = matrix(y, (1, n_samples), 'd')
    b = matrix(0.0)
    #
    solution = solvers.qp(P, q, G, h, A, b)
```

```
    #
    alphas = np.ravel(solution['x'])
    # calculate w and b
    _____
    _____
    _____
    print(w, b)
    return w, b


# Please describe this function
def plot_data(x, y):
    x_min = 4
    x_max = 7
    y_min = 2
    y_max = 5
    colors = ["steelblue", "orange"]
    plt.figure(figsize=(8, 8))
    plt.scatter(x[:, 0], x[:, 1], c=y.ravel(), alpha=0.5,
                cmap=matplotlib.colors.ListedColormap(colors), edgecolors="black")
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.show()


# Please describe this function
def plot_result(x, y, w ,b):
    x_min = 4
    x_max = 7
    y_min = 2
```

```
    y_max = 5
    xx = np.linspace(x_min, x_max)
    a = -w[0] / w[1]
    yy = a * xx - (b) / w[1]
    margin = 1 / np.sqrt(np.sum(w ** 2))
    yy_neg = yy - np.sqrt(1 + a ** 2) * margin
    yy_pos = yy + np.sqrt(1 + a ** 2) * margin
    plt.figure(figsize=(8, 8))
    plt.plot(xx, yy, "b-")
    plt.plot(xx, yy_neg, "m--")
    plt.plot(xx, yy_pos, "m--")
    colors = ["steelblue", "orange"]
    plt.scatter(x[:, 0], x[:, 1], c=y.ravel(), alpha=0.5,
                cmap=matplotlib.colors.ListedColormap(colors), edgecolors="black")
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.show()


if __name__ == "__main__":
    hw2_x = np.load('hw2_x.npy')
    hw2_y = np.load('hw2_y.npy')
    #
    plot_data(hw2_x, hw2_y)
    #
    w, b = custom_svm_fit(hw2_x, hw2_y, C=10)
    print("W:", w)
    print("b:", b)
    plot_result(hw2_x, hw2_y, w, b)
    #
```

clf = svm.SVC(kernel="linear", C=10)

clf.fit(hw2_x, hw2_y.ravel())

w = clf.coef_[0]

b = clf.intercept_

print("W:", w)

print("b:", b)

print("Finished")

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from cvxopt import matrix, solvers
from sklearn import svm

'''
Please install the package first
complete the code and add relevant comments
y is equal to d in slides, C is equal to lambda in slides.

'''
def custom_svm_fit(x, y, C):
    # slove alpha
    n_samples, n_features = x.shape
    #
    K = np.zeros((n_samples, n_samples))
    for i in range(n_samples):
        for j in range(n_samples):
            K[i, j] = _____
    #
    P = matrix(_____ * K)
    #
    q = matrix(_____)
    #
    tmp1 = np.diag(_____ (n_samples) * _____)
    tmp2 = _____ (n_samples)
    G = matrix(np.vstack((tmp1, tmp2)))
```

```
    #
    tmp3 = _____ (n_samples)
    tmp4 = _____ (n_samples) * C
    h = matrix(np.hstack((tmp3, tmp4)))
    #
    A = matrix(y, (1, n_samples), 'd')
    b = matrix(0.0)
    #
    solution = solvers.qp(P, q, G, h, A, b)
    #
    alphas = np.ravel(solution['x'])
    # calculate w and b
    _____
    _____
    _____
    print(w, b)
    return w, b


# Please describe this function
def plot_data(x, y):
    x_min = 4
    x_max = 7
    y_min = 2
    y_max = 5
    colors = ["steelblue", "orange"]
    plt.figure(figsize=(8, 8))
    plt.scatter(x[:, 0], x[:, 1], c=y.ravel(), alpha=0.5,
                cmap=matplotlib.colors.ListedColormap(colors),
edgecolors="black")
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.show()


# Please describe this function
def plot_result(x, y, w ,b):
    x_min = 4
    x_max = 7
    y_min = 2
    y_max = 5
    xx = np.linspace(x_min, x_max)
    a = -w[0] / w[1]
    yy = a * xx - (b) / w[1]
```

```
72.     margin = 1 / np.sqrt(np.sum(w ** 2))
73.     yy_neg = yy - np.sqrt(1 + a ** 2) * margin
74.     yy_pos = yy + np.sqrt(1 + a ** 2) * margin
75.     plt.figure(figsize=(8, 8))
76.     plt.plot(xx, yy, "b-")
77.     plt.plot(xx, yy_neg, "m--")
78.     plt.plot(xx, yy_pos, "m--")
79.     colors = ["steelblue", "orange"]
80.     plt.scatter(x[:, 0], x[:, 1], c=y.ravel(), alpha=0.5,
81.                 cmap=matplotlib.colors.ListedColormap(colors),
    edgecolors="black")
82.     plt.xlim(x_min, x_max)
83.     plt.ylim(y_min, y_max)
84.     plt.show()
85.
86.
87. if __name__ == "__main__":
88.     hw2_x = np.load('hw2_x.npy')
89.     hw2_y = np.load('hw2_y.npy')
90.     #
91.     plot_data(hw2_x, hw2_y)
92.     #
93.     w, b = custom_svm_fit(hw2_x, hw2_y, C=10)
94.     print("W:", w)
95.     print("b:", b)
96.     plot_result(hw2_x, hw2_y, w, b)
97.     #
98.     clf = svm.SVC(kernel="linear", C=10)
99.     clf.fit(hw2_x, hw2_y.ravel())
100.      w = clf.coef_[0]
101.      b = clf.intercept_
102.      print("W:", w)
103.      print("b:", b)
104.      print("Finished")
```