4.1 补充说明

(1)题中, 部分同学的答案中优化问题中带有变量 μ_i, 即优化目标为

$$\max_{\boldsymbol{\alpha},\boldsymbol{\mu}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j - \sum_{i=1}^{m} \frac{(\alpha_i + \mu_i)^2}{4C}$$

实际上, $\boldsymbol{\mu} = \mathbf{0}$ 时, 目标函数可以取得最大值, 因此可以直接令 $\boldsymbol{\mu} = \mathbf{0}$, 即得到答案中的形式. 批改作业时, 上式也视为正确.

4.1 补充说明

(2)题中, 部分同学给出的优化问题形式为

$$\begin{aligned}
\min_{\boldsymbol{w},b,\xi_i,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\xi \\
\text{s.t.} \quad & y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \geq 1 - \xi_i \\
& \xi \geq \xi_i \geq 0, i \in [m].
\end{aligned}$$

实际上, 上式中的 $y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \geq 1 - \xi_i$ 可以放缩为答案中的 $y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \geq 1 - \xi$. 两种形式优化的结果相同, 批改作业时, 上式也视为正确.

4.1 补充说明

(3)题中, 部分同学给出的对偶问题形式为

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\
\text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0 \\
& 0 \leq \alpha_i \leq C, \quad i \in [m],\ \boldsymbol{x}_i \in D_+ \\
& 0 \leq \alpha_i \leq kC, \quad i \in [m],\ \boldsymbol{x}_i \in D_- .
\end{aligned}
$$

上式中与答案中的对偶问题等价. 批改作业时, 上式也视为正确.

但是, (3)题中, 部分同学未加说明地使用了题目中没有出现的 C_+ 和 C_- 两个符号，而忽略了 k 倍的条件. 此时会相应扣分.

4.2 Solution

(1) 原问题需要求解 $\boldsymbol{w}, b$, 因此参数量为 $d+1$; 对偶问题需要求解 $\boldsymbol{\alpha}$, 因此参数量为 m.
部分同学的的答案中, 原问题的变量中还考虑了ξ, 即参数量为$d+m+1$, 也视为正确.

(2) 对偶问题的求解代码为:

```python
import cvxpy as cp

def solve_dual(X, y, C):
    m = X.shape[0]
    alpha = cp.Variable(m)
    y_ = y.reshape(-1, 1)
    Q = y_ * y_.T * (X @ X.T)
    loss = 0.5 * cp.quad_form(alpha, Q) - cp.sum(alpha)
    prob = cp.Problem(cp.Minimize(loss),
                      [cp.sum(cp.multiply(y, alpha)) == 0,
                       alpha >= 0,
                       alpha <= C])
    prob.solve()
    return alpha
```

(3) 曲线图为: (图略)

随着 r 的增加, 原问题的求解时间逐渐高于对偶问题的求解时间, 可以大致得出结论, 若样本数量大于维度, 适合求解原问题, 而当维度高于样本数量时, 适合求解对偶问题.

4.3 补充说明

在使用 NumPy 时，一定要利用好其广播的性质，避免使用 `for` 循环（如下代码），否则会大幅增加运算的时间

```
def custom_kernel(X1, X2):
    '''
    :参数 X1: ndarray, 形状(m, d)
    :参数 X2: ndarray, 形状(n, d)
    :返回: 形状为(m, n)的Gram矩阵，第(i,j)个元素为X1[i]和X2[j]之间的核函数值
    '''
    m,d = X1.shape
    n,d = X2.shape
    Gram = np.zeros(shape=(m,n))
    for i in range(m):
        for j in range(n):
            Gram[i,j] = 1/(1+np.linalg.norm(x=X1[i]-X2[j],ord=2)**2)
    return Gram
```