Homework # 6

姓名	學號
操之晴	R13922A04

5.

In Dr. Threshold's setup

$$\mathbf{z}_n^T\mathbf{w}^* = [1, \mathbf{x}_n^T] \cdot [w_0^*, \mathbf{w}_1^*]^T = w_0^* + \mathbf{x}_n^T\mathbf{w}_1^*.$$

This is equivalent to the original constraint $y_n(\mathbf{x}_n^T\mathbf{w}_1^* + b^*) \geq 1 - \xi_n$

With $b^* = w_0^*$

Substitute (b^*, w_1^*, α^*) into the original constraints

The constraint becomes $y_n(\mathbf{x}_n^T\mathbf{w}_1^* + b^*) \geq 1 - \xi_n$ which matches the original formulation.

The objective in the augmented formulation is:

$$\frac{1}{2}\|\mathbf{w}^*\|^2 = \frac{1}{2}(w_0^{*2} + \|\mathbf{w}_1^*\|^2)$$

In the original formulation, the objective is $\frac{1}{2}||w_1^*||^2$

Since $b^* = w_0^*$, substituting does not change the value of the objective function.

The dual variables α^* remain unchanged, as the constraints$\sum_{n=1}^{N} \alpha_n y_n = 0$ and $0 \leq \alpha_n \leq C$ are independent of whether the data is augmented or not.

(b^*, w_1^*, α^*) satisfies the primal constraints and yields the same objective value as the augmented problem. Thus, (b^*, w_1^*, α^*) is a solution to the original problem.

6.

Primal problem:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n$$

$\alpha_n \geq 0\ for \quad y_n(w^\top\phi(x_n)+b) \geq 1-\xi_n.$
$\beta_n \geq 0\ for \quad \xi_n \geq 0.$
$\alpha_0 \geq 0\ for \quad y_0(w^\top\phi(x_0)+b) \geq 1.$

The Lagrangian is:

$$\mathcal{L}(w,b,\xi,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n[y_n(w^\top\phi(x_n)+b)-1+\xi_n] - \sum_{n=1}^{N}\beta_n\xi_n - \alpha_0[y_0(w^\top\phi(x_0)+b)-1].$$

To derive the dual, set the partial derivatives of $\mathcal{L}$ with respect to the primal variables w, b, and ξ to zero:

(1) With respect to w:

$$w - \sum_{n=1}^{N}\alpha_n y_n\phi(x_n) - \alpha_0 y_0\phi(x_0) = 0.$$

$$w = \sum_{n=1}^{N}\alpha_n y_n\phi(x_n) + \alpha_0 y_0\phi(x_0).$$

(2) With respect to b:

$$\sum_{n=1}^{N}\alpha_n y_n + \alpha_0 y_0 = 0.$$

(3) With respect to ξ:

$$C - \alpha_n - \beta_n = 0.$$

Thus, $\alpha_n < C$

Substitute w into the Lagrangian, and simplify using the constraints:

$$\mathcal{L}_D(\alpha) = \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\left(\sum_{n=1}^{N}\alpha_n y_n\phi(x_n) + \alpha_0 y_0\phi(x_0)\right)^\top \left(\sum_{m=1}^{N}\alpha_m y_m\phi(x_m) + \alpha_0 y_0\phi(x_0)\right)$$

$$\mathcal{L}_D(\alpha) = \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(x_n,x_m) - \alpha_0\sum_{n=1}^{N}\alpha_n y_n y_0 K(x_n,x_0) - \frac{1}{2}\alpha_0^2 K(x_0,x_0).$$

Maximize $\mathcal{L}_D(\alpha)$ subject to:

$$\sum_{n=1}^{N} \alpha_n y_n + \alpha_0 y_0 = 0.$$

$$0 \leq \alpha_n \leq C, \alpha_0 \geq 0.$$

Where $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ is the kernel function.
This is the dual form of the one-class soft-margin SVM.

7.

Since $\alpha = 1, b = 0, h(x) = sign(\sum_{n=1}^{N} y_n K(x_n, x))$

The error becomes

$$E_{in}(\hat{h}) = \frac{1}{N}\sum_{i=1}^{N}[[h(x_i) \neq y_i]] = \frac{1}{N}\sum_{i=1}^{N}\left[\left[sign\left(\sum_{n=1}^{N} y_n K(x_n, x_i)\right) \neq y_i\right]\right]$$

If $E_{in}(\hat{h}) = 0$, all prediction must be all correct

$[[sign(\sum_{n=1}^{N} y_n K(x_n, x_i)) = y_i]]$ can be express as $[[(\sum_{n=1}^{N} y_n K(x_n, x_i)) \cdot y_i > 0]]$

When i=N,

$$y_N \sum_{n=1}^{N} y_n K(x_n, x_N) - y_N(y_1 K(x_1, x_N) + y_2 K(x_2, x_N) + \cdots + y_N) > 0$$

Sum up from i=1 to i=N, the result still > 0

Since γ is large enough, $K(x_n, x_m) = \exp\left(-\gamma\left||x_n - x_m|\right|^2\right) \leq \exp\left(-\gamma\epsilon^2\right)$

$$0 < \sum_{i=1}^{N}\left(y_i \sum_{n=1}^{N} y_n K(x_n, x_i)\right)$$

$$\leq [y_1^2 + y_2^2 + \cdots + y_N^2] + [y_1(y_2 \exp(-\gamma\epsilon^2) + \cdots + y_N \exp(-\gamma\epsilon^2)) + \cdots + y_N(y_1 \exp(-\gamma\epsilon^2) + \cdots + y_{N-1}\exp(-\gamma\epsilon^2))]$$

Since $y_i^2 = 1$, $-1 < y_n y_m < 1$, using the smallest value to find the lower bound, that is, assume all $y_n y_m = -1$, we have:

$$0 \leq \sum_{i=1}^{N} y_i^2 + \sum_{i=1}^{N}(-1) * (N-1) exp\left(-\gamma\epsilon^2\right)$$

$$= \sum_{i=1}^{N} y_i^2 + N(-(N-1)\exp(-\gamma\epsilon^2))$$

$$= N - N(N-1)\exp(-\gamma\epsilon^2)$$

From $0 < N - N(N-1)\exp(-\gamma\epsilon^2)$, $1 > (N-1)\exp(-\gamma\epsilon^2)$

By rearranging the terms, we get

$$\gamma > \frac{\ln(N-1)}{\epsilon^2}$$

8.

To prove that $K(x,x') = \exp(2\cos(x-x') - 2)$ is a valid kernel, we need to show that it satisfies the properties of a valid kernel:

A. **Symmetry: $\boldsymbol{K(x,x') = K(x',x)}$**
   Since $\cos(x-x') = \cos(x'-x)$, $K(x,x') = \exp(2\cos(x-x') - 2) = \exp(2\cos(x'-x) - 2) = K(x',x)$
   Hence, $K(x,x')$ is symmetric.

B. **Positive Semi-Definiteness: For any set of points the kernel matrix K, defined by $\boldsymbol{K_{ij} = K(x_i, x_j)}$, must be positive semi-definite.**
   The trigonometric identity: $\cos(x-x') = \cos x\, cosx' + \sin x\, sinx'$
   This suggests that $\cos(x-x')$ can be viewed as the dot product of the feature vectors:

$$\Phi(x) = (\cos x, \sin x)$$

   Therefore,

$$2\cos(x-x') = 2(\cos x\, cosx' + \sin x\, sinx') = 2\langle \Phi(x), \Phi(x')\rangle$$

   Where $\langle \Phi(x), \Phi(x')\rangle$ denotes the inner product of $\Phi(x)\ and\ \Phi(x')$

$$K(x,x') = \exp(2\langle \Phi(x), \Phi(x')\rangle - 2)$$

   Since the exponential of a valid kernel is also a valid kernel, and $2\langle \Phi(x), \Phi(x')\rangle$ represents a valid kernel, $K(x,x')$ is valid.

   Thus, $K(x,x')$ is a valid kernel.

9.

scaled decision stumps $g_{i,\theta}(x) = [x_i \geq \theta]$ outputs 1 if $x_i \geq \theta$, and 0 otherwise

θ is in the range {L+0.5,L+1.5,…,R−0.5}

$$K_{ds}(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{d} \sum_{\theta \in \{L+0.5, L+1.5, \ldots, R-0.5\}} g_{i,\theta}(\mathbf{x}) \cdot g_{i,\theta}(\mathbf{x}')$$

$g_{i,\theta}(x) = 1$ when $x_i \geq \theta$

$g_{i,\theta}(x') = 1$ when $x_i{}' \geq \theta$

$g_{i,\theta}(x) \cdot g_{i,\theta}(x') = 1$ if $\theta \leq \min(x_i, x_i')$ otherwise 0

In the range {L+0.5,L+1.5,…,R−0.5}, the number of θ satisfying $\theta \leq \min(x_i, x_i')$ is

$$\lfloor \min(x_i, x_i') - L + 0.5 \rfloor$$

Denoted as $count_\theta$

For each dimension i, the contribution to the kernel is the number of θ's satisfying the condition. Summing over all dimensions gives:

$$K_{ds}(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{d} \text{count}_\theta$$

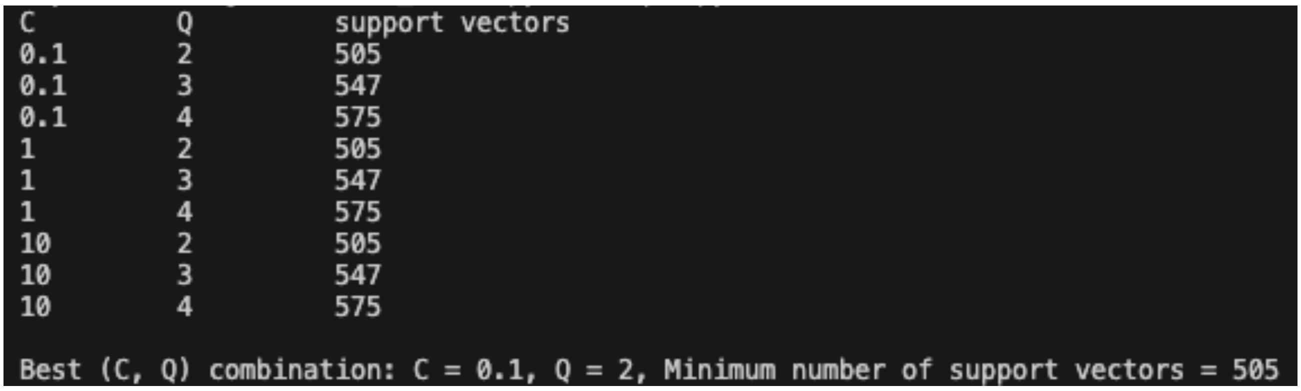
Where $count_\theta = \max(0, \min(x_i, x_i') - L + 1)$

So,

$$K_{ds}(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{d} \max(0, \min(x_i, x_i') - L + 1)$$

10.

C	Q	Support Vectors
0.1	2	**505**
	3	547
	4	575
1	2	**505**
	3	547
	4	575
10	2	**505**
	3	547
	4	575



```
C          Q          support vectors
0.1        2          505
0.1        3          547
0.1        4          575
1          2          505
1          3          547
1          4          575
10         2          505
10         3          547
10         4          575

Best (C, Q) combination: C = 0.1, Q = 2, Minimum number of support vectors = 505
```

In the results, we can see that for the same Q, the number of support vectors remains constant and does not change with C. This indicates that the data distribution is relatively stable, and the model is mainly influenced by the kernel function rather than C. On the other hand, as Q increases, the number of support vectors gradually increases. This is reasonable because higher-degree polynomial kernels make the model more flexible (more easy to overfitting), requiring more support vectors to fit the data.

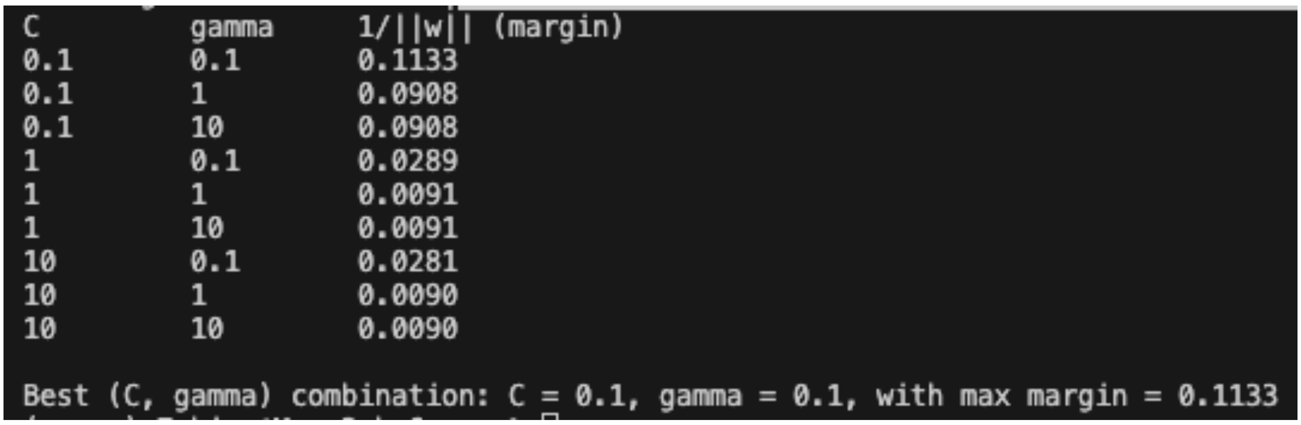
Based on the criterion of the least number of support vectors, the optimal parameters are C=0.1, Q=2, with 505 support vectors. This parameter set likely represents the simplest model.

The code screenshots for Problems 10 to 12 are attached on the next page following Problem 12.

For problems 10 to 12, the full code has been made available on GitHub.
Feel free to check it out if required or interested!

11.

C	γ	1/‖w‖ (margin)
0.1	0.1	**0.1133**
	1	0.0908
	10	0.0908
1	0.1	0.0289
	1	0.0091
	10	0.0091
10	0.1	0.0281
	1	0.0090
	10	0.0090



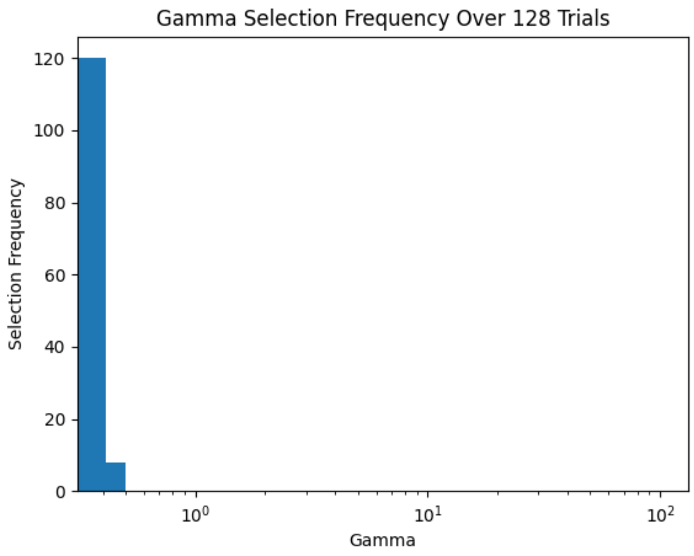
```
C         gamma     1/||w|| (margin)
0.1       0.1       0.1133
0.1       1         0.0908
0.1       10        0.0908
1         0.1       0.0289
1         1         0.0091
1         10        0.0091
10        0.1       0.0281
10        1         0.0090
10        10        0.0090

Best (C, gamma) combination: C = 0.1, gamma = 0.1, with max margin = 0.1133
```

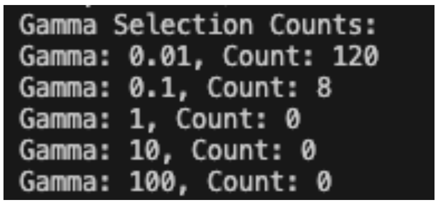
When C is fixed, as γ increases, the margin gets smaller. This is mainly because γ controls the range of influence of the kernel. A large γ makes the model overly focused and easy to overfitting. When γ is fixed, as C increases, the margin gets smaller as well. A large C emphasizes classification accuracy, but it reduces the model's flexibility. The largest margin occurs at C=0.1, γ=0.1, which indicates the model has the best generalization ability.

The code screenshots for Problems 10 to 12 are attached on the next page following Problem 12.

For problems 10 to 12, the full code has been made available on GitHub.
Feel free to check it out if required or interested!

12.





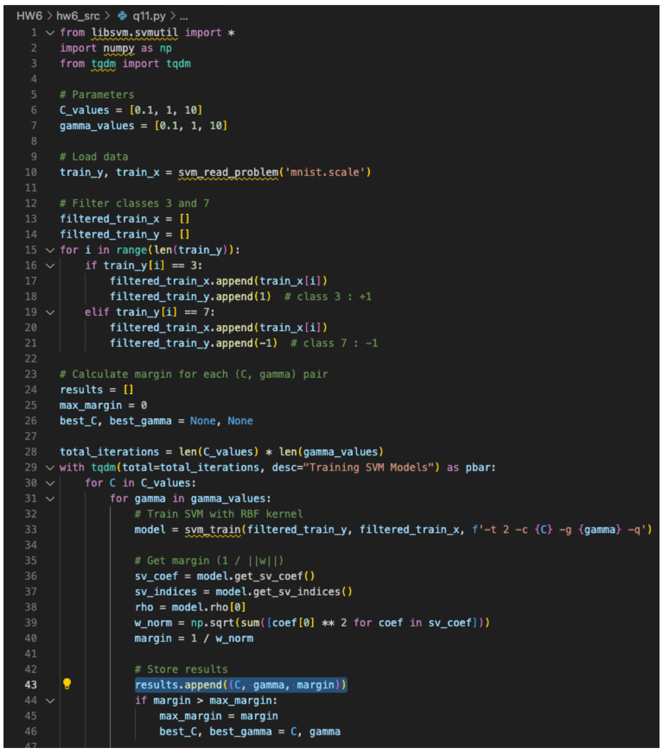
```
Gamma Selection Counts:
Gamma: 0.01, Count: 120
Gamma: 0.1, Count: 8
Gamma: 1, Count: 0
Gamma: 10, Count: 0
Gamma: 100, Count: 0
```

γ=0.01 is the smallest of all the candidates, meaning the RBF kernel covers a larger range. This creates smoother decision boundaries, preventing the model from becoming overly complex due to data details. In most cases, this smoother boundary generalizes better, so it performs better with lower error rates on the validation set.
There are still some choices of γ=0.1. When the validation data has a slight nonlinear characteristic, it might perform slightly better than γ=0.01. However, since it covers a smaller range, the model can sometimes become more complex, making it less stable compared to γ=0.01.
γ=1,10,100 has a very narrow RBF kernel range, causing the model to become overly localized, which means too focused on the details, leading to overfitting. Such models perform poorly on the validation set because they are too rigid and fail to adapt to the validation data.

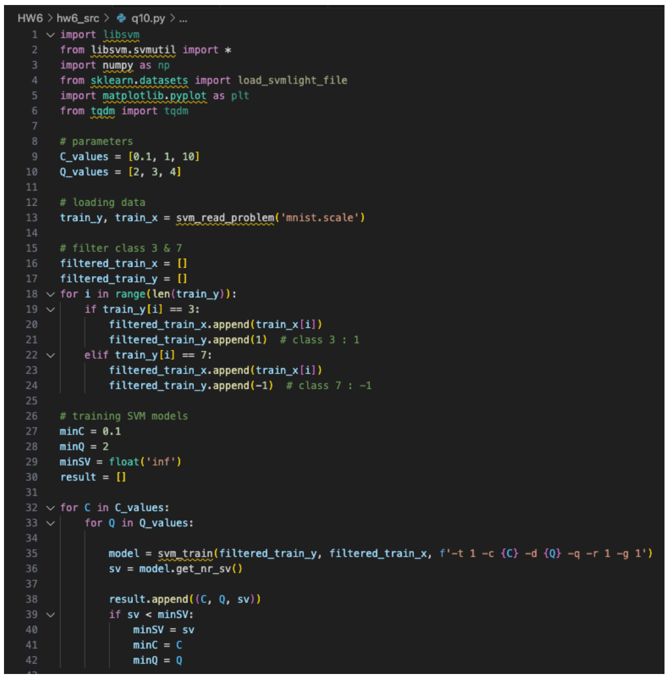
The code screenshots for Problems 10 to 12 are attached on the next page following Problem 12.

For problems 10 to 12, the full code has been made available on GitHub.
Feel free to check it out if required or interested!

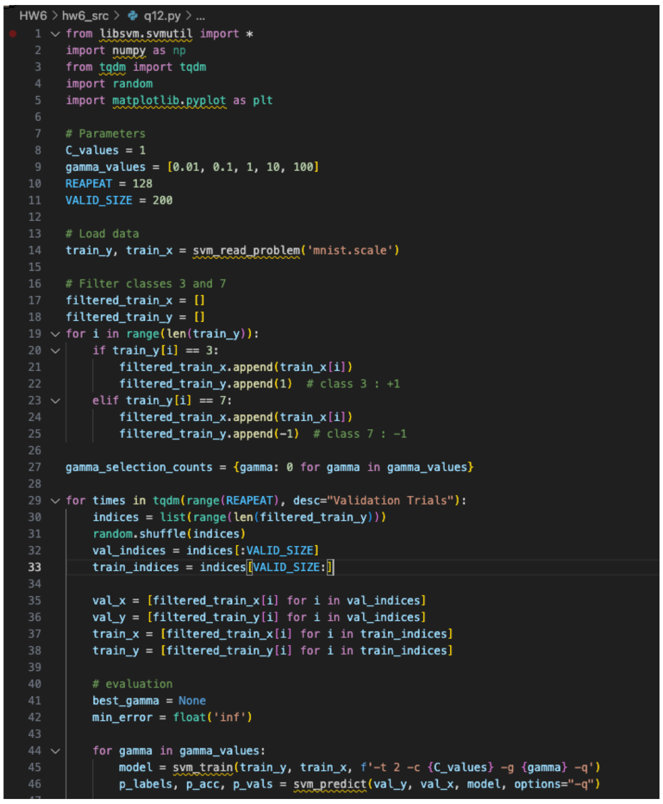
The code screenshots for Problems 10 to 12


HW6 > hw6_src > q11.py > ...

```python
from libsvm.svmutil import *
import numpy as np
from tqdm import tqdm

# Parameters
C_values = [0.1, 1, 10]
gamma_values = [0.1, 1, 10]

# Load data
train_y, train_x = svm_read_problem('mnist.scale')

# Filter classes 3 and 7
filtered_train_x = []
filtered_train_y = []
for i in range(len(train_y)):
    if train_y[i] == 3:
        filtered_train_x.append(train_x[i])
        filtered_train_y.append(1)  # class 3 : +1
    elif train_y[i] == 7:
        filtered_train_x.append(train_x[i])
        filtered_train_y.append(-1)  # class 7 : -1

# Calculate margin for each (C, gamma) pair
results = []
max_margin = 0
best_C, best_gamma = None, None

total_iterations = len(C_values) * len(gamma_values)
with tqdm(total=total_iterations, desc="Training SVM Models") as pbar:
    for C in C_values:
        for gamma in gamma_values:
            # Train SVM with RBF kernel
            model = svm_train(filtered_train_y, filtered_train_x, f'-t 2 -c {C} -g {gamma} -q')

            # Get margin (1 / ||w||)
            sv_coef = model.get_sv_coef()
            sv_indices = model.get_sv_indices()
            rho = model.rho[0]
            w_norm = np.sqrt(sum([coef[0] ** 2 for coef in sv_coef]))
            margin = 1 / w_norm

            # Store results
            results.append((C, gamma, margin))
            if margin > max_margin:
                max_margin = margin
                best_C, best_gamma = C, gamma
```


HW6 > hw6_src > q10.py > ...

```python
import libsvm
from libsvm.svmutil import *
import numpy as np
from sklearn.datasets import load_svmlight_file
import matplotlib.pyplot as plt
from tqdm import tqdm

# parameters
C_values = [0.1, 1, 10]
Q_values = [2, 3, 4]

# loading data
train_y, train_x = svm_read_problem('mnist.scale')

# filter class 3 & 7
filtered_train_x = []
filtered_train_y = []
for i in range(len(train_y)):
    if train_y[i] == 3:
        filtered_train_x.append(train_x[i])
        filtered_train_y.append(1)  # class 3 : 1
    elif train_y[i] == 7:
        filtered_train_x.append(train_x[i])
        filtered_train_y.append(-1)  # class 7 : -1

# training SVM models
minC = 0.1
minQ = 2
minSV = float('inf')
result = []

for C in C_values:
    for Q in Q_values:

        model = svm_train(filtered_train_y, filtered_train_x, f'-t 1 -c {C} -d {Q} -q -r 1 -g 1')
        sv = model.get_nr_sv()

        result.append((C, Q, sv))
        if sv < minSV:
            minSV = sv
            minC = C
            minQ = Q
```


HW6 > hw6_src > q12.py > ...

```python
from libsvm.svmutil import *
import numpy as np
from tqdm import tqdm
import random
import matplotlib.pyplot as plt

# Parameters
C_values = 1
gamma_values = [0.01, 0.1, 1, 10, 100]
REAPEAT = 128
VALID_SIZE = 200

# Load data
train_y, train_x = svm_read_problem('mnist.scale')

# Filter classes 3 and 7
filtered_train_x = []
filtered_train_y = []
for i in range(len(train_y)):
    if train_y[i] == 3:
        filtered_train_x.append(train_x[i])
        filtered_train_y.append(1)  # class 3 : +1
    elif train_y[i] == 7:
        filtered_train_x.append(train_x[i])
        filtered_train_y.append(-1)  # class 7 : -1

gamma_selection_counts = {gamma: 0 for gamma in gamma_values}

for times in tqdm(range(REAPEAT), desc="Validation Trials"):
    indices = list(range(len(filtered_train_y)))
    random.shuffle(indices)
    val_indices = indices[:VALID_SIZE]
    train_indices = indices[VALID_SIZE:]

    val_x = [filtered_train_x[i] for i in val_indices]
    val_y = [filtered_train_y[i] for i in val_indices]
    train_x = [filtered_train_x[i] for i in train_indices]
    train_y = [filtered_train_y[i] for i in train_indices]

    # evaluation
    best_gamma = None
    min_error = float('inf')

    for gamma in gamma_values:
        model = svm_train(train_y, train_x, f'-t 2 -c {C_values} -g {gamma} -q')
        p_labels, p_acc, p_vals = svm_predict(val_y, val_x, model, options="-q")
```

13.

Primal SVM Problem:

Objective : $\min_{w,b} \frac{1}{2}\|w\|^2$

subject to : $y_n(w^T x_n + b) \geq 1$

Lagrangian : $\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} \alpha_n (1 - y_n(w^T x_n + b))$

objective : $\frac{1}{2}\|w\|^2$

constraint using $\alpha_n \geq 0$

dual problem : $\max_{\alpha \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$

Dual SVM Problem:

Objective : $\min_{\alpha} \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m y_n y_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n$

subject to : $\sum_{n=1}^{N} \alpha_n y_n = 0$, $\alpha_n \geq 0$

Lagrangian : $\mathcal{L}(\alpha, \lambda) = \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m y_n y_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n - \sum_{n=1}^{N} \lambda_n \alpha_n$

objective : $\mathcal{L}(\alpha, \lambda) = \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m y_n y_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n$

λ_n constraint using $\alpha_n \geq 0$

$$\min_{\alpha} \begin{cases} \infty, & \text{if constraint violated } \alpha_n < 0 \\ \text{objective value}, & \text{if constraint satisfied } \alpha_n > 0 \end{cases}$$

Derivation of Lagrange Dual:

$\min_{\alpha} \max_{\lambda \geq 0} \mathcal{L}(\alpha, \lambda) \geq \min_{\alpha} \mathcal{L}(\alpha, \lambda')$

$\min_{\alpha} \max_{\lambda \geq 0} \mathcal{L}(\alpha, \lambda) \geq \max_{\lambda \geq 0} \min_{\alpha} \mathcal{L}(\alpha, \lambda)$

Lagrange dual for dual problem : $\max_{\lambda \geq 0} \min_{\alpha} \mathcal{L}(\alpha, \lambda)$

Lagrange for primal problem : $\min_{w,b} \max_{\alpha \geq 0} \mathcal{L}(w,b,\alpha)$

$\therefore$ The order of max and min are reversed, and the optimization is performed on different variables in the primal & dual problems.

Based on theoretical explanations and GPT's responce, the dual of the dual problem isn't always equivalent to the primal problem.