5.

The primal problem (P_1) with $\underline{z}_n = [1, \underline{x}_n^T]$ becomes

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

$$\text{s.t.} \quad y_n (\tilde{\underline{w}}^T \underline{z}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0, \quad \text{for } n = 1, 2, \ldots, N$$

The dual problem (D_1) with $\underline{z}_n = [1, \underline{x}_n^T]$ becomes

$$\min_{\underline{\alpha}} \quad \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \underline{z}_n^T \underline{z}_m - \sum_{n=1}^{N} \alpha_n$$

$$= \min_{\underline{\alpha}} \quad \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m (1 + \underline{x}_n^T \underline{x}_m) - \sum_{n=1}^{N} \alpha_n$$

1 ↓ constant

$$\text{s.t.} \quad \sum_{n=1}^{N} \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C, \quad n = 1, 2, \ldots, N$$

This is equivalent to the original problem, since adding a constant doesn't affect the optimal solution

Given above, and given the primal-dual solution $(b^*, \tilde{\underline{w}}^*, \underline{\alpha}^*)$ w.r.t $\underline{z}_n$, we can construct the solution $(b^*, \underline{w}^*, \underline{\alpha}^*)$ w.r.t $\underline{x}_n$:

① b^* : which is the same.

② $\underline{\alpha}^*$: which is the same.

③ $\underline{w}^*$: $\tilde{\underline{w}}^* = \sum_{n=1}^{N} \alpha_n^* y_n \underline{z}_n^* = \sum_{n=1}^{N} \alpha_n y_n \begin{bmatrix} 1 \\ \underline{x}_n \end{bmatrix}$

$\Rightarrow \underline{w}^* = \sum_{n=1}^{N} \alpha_n^* y_n \underline{x}_n$ $= \begin{bmatrix} \tilde{w}_0^* \\ \underline{w}^* \end{bmatrix}$

Since according to K.K.T,

$$\tilde{\underline{w}}^* = \sum_{n=1}^{N} \alpha_n^* y_n \tilde{\underline{x}}_n = \left[\underbrace{\sum_{n=1}^{N} \alpha_n^* y_n}_{\tilde{\underline{w}}_0^*}, \underbrace{\sum_{n=1}^{N} \alpha_n^* y_n \underline{x}_n}_{\underline{w}^*} \right]^T$$

and $\sum_{n=1}^{N} \alpha_n^* y_n = 0 \Rightarrow \tilde{\underline{w}}_0^* = 0$ ∎

6.

Lagrange multiplier : $\begin{cases} \alpha_n \geq 0 : y_n(\underline{w}^T\underline{\Phi}(\underline{x}_n)+b) \geq 1-\xi_n \\ \beta_n \geq 0 : \xi_n \geq 0 \\ \alpha_0 \geq 0 : y_0(\underline{w}^T\underline{\Phi}(\underline{x}_0)+b) \geq 1 \end{cases}$

Given that $y_0 = -1$, all other y_n's are $+1$

$$\Rightarrow L(b, \underline{w}, \underline{\xi}, \underline{\alpha}, \underline{\beta})$$

$$= \frac{1}{2}\underline{w}^T\underline{w} + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\alpha_n\left(1-\xi_n - y_n(\underline{w}^T\underline{\Phi}(\underline{x}_n)+b)\right)$$
$$+ \sum_{n=1}^{N}\beta_n(-\xi_n) + \alpha_0\left(1 - y_0(\underline{w}^T\underline{\Phi}(\underline{x}_0)+b)\right)$$

$$= \frac{1}{2}\underline{w}^T\underline{w} + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\alpha_n\left(1-\xi_n - \underline{w}^T\underline{\Phi}(\underline{x}_n) - b\right) +$$
$$\sum_{n=1}^{N}\beta_n(-\xi_n) + \alpha_0\left(1 + \underline{w}^T\underline{\Phi}(\underline{x}_0) + b\right)$$

①

$$\frac{\partial L}{\partial \xi_n} = 0 \;:\; C - \alpha_n - \beta_n = 0 \Rightarrow \beta_n = C - \alpha_n,$$

$$\alpha_n \le C$$

so we can rewrite the lagrange dual problem as:

$$\max_{\substack{0 \le \alpha_n \le C \\ \beta_n = C - \alpha_n}} \left(\min_{b, \underline{w}} \; \frac{1}{2}\underline{w}^T\underline{w} + \sum_{n=1}^{N} \alpha_n \left(1 - \underline{w}^T \underline{\Phi}(\underline{x}_n) - b\right) + \alpha_0 \left(1 + \underline{w}^T \underline{\Phi}(\underline{x}_0) + b\right) \right)$$

②

$$\frac{\partial L}{\partial b} = 0 \;:\; -\sum_{n=1}^{N} \alpha_n + \alpha_0 = 0, \quad \alpha_0 = \sum_{n=1}^{N} \alpha_n$$

③

$$\frac{\partial L}{\partial w_i} = 0 \;:\; w_i - \sum_{n=1}^{N} \alpha_n \underline{\Phi}(x_{n,i}) + \alpha_0 \underline{\Phi}(x_{0,i}) = 0$$

$$\Rightarrow \underline{w} = \sum_{n=1}^{N} \alpha_n y_n \underline{\Phi}(\underline{x}_n) - \sum_{n=1}^{N} \alpha_n y_0 (\underline{x}_0)$$

We can rewrite the lagrange dual problem:

$$\max_{\substack{0\le\alpha_n\le C \\ \beta_n = C-\alpha_n}} \left(\min_{b,\underline{w}} \frac{1}{2}\underline{w}^T\underline{w} + \sum_{n=1}^{N}\alpha_n\left(1-\underline{w}^T\underline{\Phi}(\underline{x}_n)-b\right) + \sum_{n=1}^{N}\alpha_n\left(1+\underline{w}^T\underline{\Phi}(\underline{x}_0)+b\right) \right)$$

$$= \max_{\substack{0\le\alpha_n\le C \\ \beta_n = C-\alpha_n}} \left(\min_{b,\underline{w}} \frac{1}{2}\underline{w}^T\underline{w} - \underline{w}^T\underline{w} + \sum_{n=1}^{N}\alpha_n \right)$$

$$= \max_{\substack{0\le\alpha_n\le C \\ \beta_n = C-\alpha_n}} -\frac{1}{2}\left\| \sum_{n=1}^{N}\alpha_n\underline{\Phi}(\underline{x}_n) \right\|^2 + \sum_{n=1}^{N}\alpha_n$$

$$\underline{w} = \sum_{n=1}^{N}\alpha_n y_n \underline{\Phi}(\underline{x}_n) - \sum_{n=1}^{N}\alpha_n y_0(\underline{x}_0)$$

$$\alpha_0 = \sum_{n=1}^{N}\alpha_n$$

$$\Rightarrow \min \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m\left(\underline{\Phi}(\underline{x}_n)\right)^T\underline{\Phi}(\underline{x}_m) - \sum_{n=1}^{N}\alpha_n$$

$$\text{s.t. } 0\le\alpha_n\le C$$

$$x \leftarrow QP(Q, \underline{F}, A, \subseteq)$$

$$g_{n,m} = \left(\underline{\bar{\Phi}}(\underline{x}_n)\right)^T \underline{\bar{\Phi}}(\underline{x}_m)$$

$$\underline{P} = -\underline{I}_N$$

$$a_n = \begin{cases} n\text{-th unit direction}, & n = 1, 2 \dots N \\ (-1) \cdot ((n-N)\text{-th unit direction}), & n = N+1, N+2, \dots 2N \end{cases}$$

$$c_n = \begin{cases} 0, & n = 1, 2, \dots, N \\ C, & n = N+1, N+2, \dots, N \end{cases}$$

#

7.

To separate the dataset :

$$h_{\underline{1},0}(\underline{x}_n) = \text{sign}\left(\sum_{m=1}^{N} y_m K(\underline{x}_m, \underline{x}_n)\right)$$

$$= \text{sign}\left(y_n + \sum_{m \neq n} y_m \exp\left(-\gamma \|\underline{x}_m - \underline{x}_n\|^2\right)\right)$$

To separate the given data, we need.

correctly classify each $\underline{x}_n \Rightarrow$

$$y_n\left(y_n + \sum_{m \neq n} y_m \exp\left(-\gamma \|\underline{x}_m - \underline{x}_n\|^2\right)\right) > 0$$

$$\Rightarrow 1 + \sum_{m \neq n} y_m y_n \exp\left(-\gamma \|\underline{x}_m - \underline{x}_n\|^2\right) > 0$$

$$\Rightarrow \left| \sum_{m \neq n} y_m y_n \exp\left(-\gamma \|\underline{x}_m - \underline{x}_n\|^2\right) \right| < 1$$

given $\|\underline{x}_m - \underline{x}_n\| \geq \epsilon$:

$$\Rightarrow \exp(-\gamma \|\underline{x}_m - \underline{x}_n\|^2) \leq \exp(-\gamma\epsilon^2)$$

$$\Rightarrow \left| \sum_{m \neq n} y_m y_n \exp(-\gamma \|\underline{x}_m - \underline{x}_n\|^2) \right| \leq$$

$$\sum_{m \neq n} \left| y_n y_m \exp(-\gamma \|\underline{x}_m - \underline{x}_n\|^2) \right| = \sum_{m \neq n} \exp(-\gamma \|x_m - x_n\|^2)$$

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

so it is sufficient when $(N-1)\exp(-\gamma\epsilon^2) < 1$

$$\xrightarrow{\log} \ln(N-1) - \gamma\epsilon^2 < 0 \Rightarrow \gamma > \frac{\ln(N-1)}{\epsilon^2} \quad \#$$

8.

①

$$\cos(\underline{x}-\underline{x}') = \cos(-(\underline{x}-\underline{x}')) = \cos(\underline{x}'-\underline{x})$$

Let $k'(\underline{x},\underline{x}') = \cos(\underline{x}-\underline{x}') = \cos(\underline{x}'-\underline{x}) = k'(\underline{x}',\underline{x})$

$k'(\underline{x},\underline{x})$ is symmetric ✓

②

a. first.

We now prove that the dot product produce a Gram matrix which is positive semi-definite (PSD)

Consider a set of vectors $\underline{x}_1, \underline{x}_2, \dots \underline{x}_n \in \mathbb{R}^d$, and a matrix G, $G_{ij} = \underline{x}_i \cdot \underline{x}_j$, then for any vector $\underline{a} = [a_1, a_2 \dots a_n]^T \in \mathbb{R}^n$:

$$\underline{a}^T G \underline{a} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j G_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j (\underline{x}_i \cdot \underline{x}_j)$$

$$= \sum_{i=1}^{n} a_i \underline{x}_i \cdot \sum_{j=1}^{n} a_j \underline{x}_j ,$$

Let $\underline{z} = \sum_{i=1}^{n} a_i \underline{x}_i \Rightarrow \underline{a}^T G \underline{a} = \underline{z}^T \underline{z} = \|\underline{z}\|^2 \geq 0$

hence the dot product produce a positive semi-definite gram matrix

b.

back to the original problem,

$$\cos(x - x') = \cos x \cos x' + \sin x \sin x'$$

$$= \langle (\cos x, \sin x), (\cos x', \sin x') \rangle$$

is a dot product in 2D, so its corresponding gram matrix is PSD, and

hence $K'(\underline{x}, \underline{x}') = \cos(\underline{x} - \underline{x}')$ is a valid kernel

Finally, $K(\underline{x}, \underline{x}') = \exp(-2)\exp(2\cos(\underline{x} - \underline{x}'))$

$= \exp(-2) \sum_{n=0}^{\infty} \frac{(2\cos(\underline{x} - \underline{x}'))^n}{n!}$, which is also a valid kernel since valid kernel raised to positive integer powers is also a valid kernel, and the sum of the valid kernel is also a valid kernel

Hence $K'(\underline{x}, \underline{x}') = \exp(2\cos(\underline{x} - \underline{x}') - 2)$ is a valid kernel #

9.

for each dimension i :

$$\sum_{j=1}^{k} g_{i,\theta_j}(\underline{x})\, g_{i,\theta_j}(\underline{x}') = \sum_{j=1}^{k} \left[\min(x_i, x_i') > \theta_j\right]$$

since the thresholds are spaced evenly by 1 :

$$\Rightarrow \sum_{j=1}^{k} \left[\min(x_i, x_i') > \theta_j\right] = \left\lceil \min(x_i, x_i') - (L + 0.5) \right\rceil$$

$$\Rightarrow K_{ds}(\underline{x}, \underline{x}') = \sum_{i=1}^{d} \sum_{j=1}^{k} g_{i,\theta_j}(\underline{x})\, g_{i,\theta_j}(\underline{x}')$$

$$= \sum_{i=1}^{d} \left\lceil \min(x_i, x_i') - (L + 0.5) \right\rceil$$

To prove that it's a valid kernel, we need to prove that it's both symmetric & positive semi-definite

① since $\min(x_i, x_i') = \min(x_i', x_i)$, it's trivial.

to see that $K_{ds}(\underline{x}, \underline{x}') = K_{ds}(\underline{x}', \underline{x})$

hence it's symmetric

②

since $K_{ds}(\underline{x}, \underline{x}')$ can be expressed as the dot product of $\underline{\Phi}_{ds}$: $K_{ds}(\underline{x}, \underline{x}') = \left(\underline{\Phi}_{ds}(\underline{x})\right)^T \left(\underline{\Phi}_{ds}(\underline{x}')\right)$,

where $\underline{\Phi}(\underline{x}) = \left[[x_1 > \theta_1], [x_1 > \theta_2] \dots, [x_1 > \theta_K], \dots [x_d > \theta_K] \right]$

and given the proof of previous problem, the dot product is positive semi-definite, and hence $K_{ds}(\underline{x}, \underline{x}')$ is positive semi-definite

Given above, $K_{ds}(\underline{x}, \underline{x}') = \sum_{i=1}^{d} \left[\min(x_i, x_i') - (L + 0.5) \right]$

is a valid kernel #

Problem 10

At Q = 2, C ∈ {0.1, 1.0, 10.0}, it has the smallest number of support vector : 505. I think that when the degree increases, it creates the more complex decision boundary, and hence it needs more support vectors to define the boundary. Besides, according to the lecture, C determines the trade-off between low error on the training set and large margin, and when the margin is large, it may cause more data points to become support vectors. But in this experiment, it seems that the value of C has no effect. I think it may be because the degree of the kernel dominates the number of the support vectors.

number_of_sv

Q \ C	0.1	1.0	10.0
2	505	505	505
3	547	547	547
4	575	575	575

Figure 1:

```
Codeium: Refactor | Explain | Generate Docstring | ×
def _experiment(y_train_filtered, x_train_filtered, C_candidates, Q_candidates):
    min_num_sv = {'C': 0, 'Q \\ C': 0, 'Number of Support Vectors': float('inf')}
    num_sv_list = []

    for _, q in enumerate(Q_candidates):
        for _, c in enumerate(C_candidates):

            # soft SVM, kernel = (γu^Tv + coef0)^degree -> (1 + x_n^Tx_m)^Q
            model = svm_train(y_train_filtered, x_train_filtered, f'-s 0 -c {c} -t 1 -d {q} -g 1 -r 1')
            num_sv = {'C': c, 'Q \\ C': q, 'Number of Support Vectors': model.get_nr_sv()}

            if min_num_sv['Number of Support Vectors'] > num_sv['Number of Support Vectors']:
                min_num_sv = num_sv

            num_sv_list.append(num_sv)

        # return num_sv_list

    return num_sv_list, min_num_sv
```

Figure 2:

```
Codeium: Refactor | Explain | Generate Docstring | ×
def main():
    start_time = time.time()

    y_train, x_train = load_LIBVSM('./mnist.scale')

    train_filter = np.where(np.isin(y_train, [3, 7]))[0]

    x_train_filtered = x_train[train_filter]
    y_train_filtered = y_train[train_filter]

    # convert to binary label (3 -> 1, 7 -> -1)
    y_train_filtered = np.where(y_train_filtered == 3, 1, -1)

    # print(y_train_filtered)
    # print(x_train_filtered)

    Q_candidates = [2, 3, 4]
    C_candidates = [0.1, 1, 10]

    num_sv_list, min_num_sv = _experiment(y_train_filtered, x_train_filtered, C_candidates, Q_candidates)

    print(f'\nAt C = {min_num_sv['C']}, Q = {min_num_sv['Q \\ C']}, number of support vectors has the smallest

    construct_table(num_sv_list)

    end_time = time.time()
    execution_time = end_time - start_time

    print(f"Execution time: {execution_time} seconds")
```

Figure 3:

Problem 11

At C = 0.1, $\gamma = 10.0$, it has the largest margin : 0.09079308469711390.

1. With respect to C, we can find that as C increase, margin becomes smaller, which matches the conclusion of the lecture : large C penalizes errors heavily, and gives less importance on maximizing the margin.

2. With respect to γ, we can find that when C $\in \{1.0, 10.0\}$, as γ increase, margin becomes smaller, I think it's because large γ has sharp Gaussian, and hence creates highly localized effects. Moreover, large C enforces the tight fit by penalizing errors heavily, which amplify the impact of large γ on reducing the margin width. However, when C = 0.1, since C is small, it allows some errors, hence it can capture the localized structure within the larger margin. So when consider about γ, we should also consdier its interaction with C.

margin

gamma \ C	0.1	1.0	10.0
0.1	0.04528230614518350	0.02096813856308830	0.020645792843444700
1.0	0.09077972081339700	0.009077996937389160	0.008983567876401830
10.0	0.09079308469711390	0.009079335957839600	0.008982792405547940

Figure 4:

```
Codeium: Refactor | Explain | ×
def compute_gaussian_kernel_margin(sv, coef, gamma):
    """
    w^Tw = alpha^TQalpha
    To calculate kernel matrix Q : Q_{i,j} = K(x_i, x_j)
                                           = exp(-gamma||x_i - x_j||^2)
    ||x - x'||^2 = ||x||^2 + ||x'||^2 - 2x^Tx'
    """

    # Compute the squared norms of each sample
    sq_norms = np.sum(sv ** 2, axis=1).reshape(-1, 1)

    # Compute pairwise squared distances and then apply rbf
    kernel_matrix = np.exp((-gamma) * (sq_norms + sq_norms.T - 2 * np.dot(sv, sv.T)))

    # w^Tw = alpha^TQalpha
    norm_w_squared = np.dot(coef, np.dot(kernel_matrix, coef))
    margin = 1 / np.sqrt(norm_w_squared)

    return margin
```

Figure 5:

```
def _experiment(y_train_filtered, x_train_filtered, C_candidates, gamma_candidates):
    max_margin = {'C': 0, 'gamma \\ C': 0, 'margin': 0}
    margin_list = []

    for _, r in enumerate(gamma_candidates):
        for _, c in enumerate(C_candidates):

            print(f'\n gamma = {r}, c = {c}:\n')

            model = SVC(kernel='rbf', C=c, gamma=r)
            model.fit(x_train_filtered, y_train_filtered)

            print(f'model computed\n')

            sv = model.support_vectors_
            coef = model.dual_coef_[0]

            margin = {'C': c, 'gamma \\ C': r, 'margin': compute_gaussian_kernel_margin(sv, coef, r)}

            print('margin computed\n')

            if max_margin['margin'] < margin['margin']:
                max_margin = margin

            margin_list.append(margin)

    return margin_list, max_margin
```

Figure 6:

Problem 12

($\gamma = 0.01$) is the most frequently selected, which means that it has the smallest E_{val} at the most time. I think it's because according to the lecture, large γ results in sharp Gaussian, which may cause overfitting, and hence results in high E_{val}.

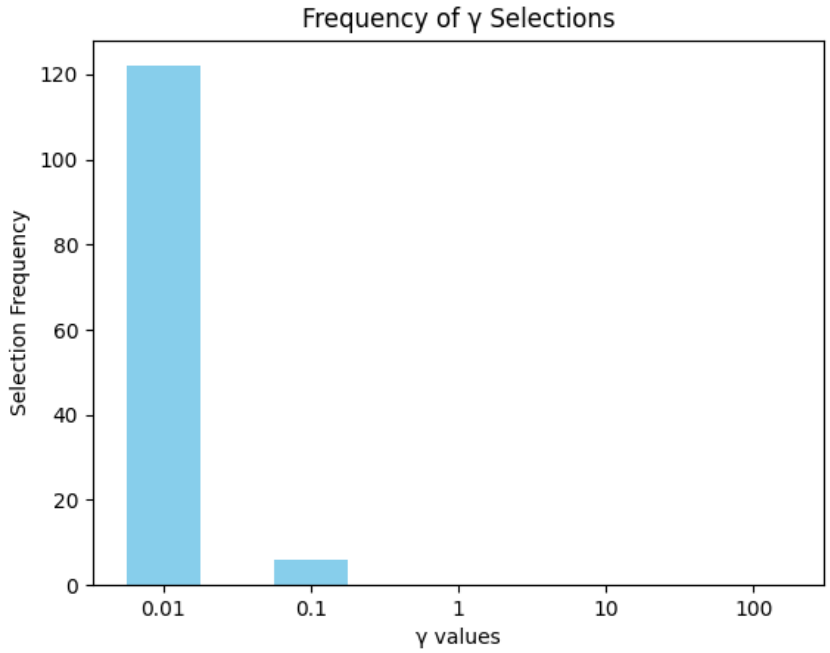

Figure 7:

```python
def experiments(y_train_filtered, x_train_filtered, c, gamma_candidates, times = 128):
    frequency = {
        0.01: 0,
        0.1: 0,
        1: 0,
        10: 0,
        100: 0
    }

    for _ in range(times):
        # frequency[best_gamma] += 1
        frequency[_experiment(y_train_filtered, x_train_filtered, c, gamma_candidates)] += 1

    return frequency

Codeium: Refactor | Explain | Generate Docstring | ×
def _experiment(y_train_filtered, x_train_filtered, c, gamma_candidates):

    # randonly samples 200 examples for validation
    x_sub_train, x_val, y_sub_train, y_val = train_test_split(x_train_filtered, y_train_filtered, test_size=200)

    best_gamma = 0
    best_acc = 0

    # find the best gamma
    for r in gamma_candidates:

        model = svm_train(y_sub_train, x_sub_train, f'-s 0 -c {c} -t 2 -g {r}')
        _, p_acc, _ = svm_predict(y_val, x_val, model)

        if best_acc <= p_acc[0]:
            best_acc = p_acc[0]
            best_gamma = r

    return best_gamma
```

Figure 8:

```python
def plot_bar_chart(frequency):

    keys = list(frequency.keys())
    values = list(frequency.values())

    plt.bar(range(len(keys)), values, width=0.5, color='skyblue')

    plt.xlabel('Y values')
    plt.ylabel('Selection Frequency')
    plt.title('Frequency of Y Selections')
    plt.xticks(range(len(keys)), keys)

    plt.savefig('./hw6_12.png')
```

Figure 9: