Solution 1

Step 1

Rearrange the decision boundary equation to match the form $w \cdot \Phi(x) + b = 0$.

$$x_1 + 3x_1x_2 = 6x_2^2 + 8 \quad \rightarrow \quad x_1 + 3x_1x_2 - 6x_2^2 - 8 = 0$$

Step 2

Identify the coefficients that correspond to each component of the basis expansion $\Phi(x) = \big(x_1, x_2, x_1^2, x_2^2, x_1x_2\big)$.

$$\begin{aligned} 0 =& w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + b \\ 0 =& (1)x_1 + (0)x_2 + (0)x_1^2 - 6x_2^2 + 3x_1x_2 - 8 \end{aligned}$$

It follows that each element of the weight vector is:

$$\begin{aligned} w_1 &= 1 \\ w_2 &= 0 \\ w_3 &= 0 \\ w_4 &= -6 \\ w_5 &= 3 \end{aligned}$$

$\therefore$ The weight vector is $w = (1, 0, 0, -6, 3)$

Solution 2 (a)

Step 1

The dimension of $\Phi(x)$, is just the sum of the number of components in the expanded feature vector. Given:

$$\Phi(x) = \left(x_1, \ldots, x_4, x_1^2, \ldots, x_4^2, x_1x_2, \ldots, x_3x_4\right)$$

Each type of term has the following number of elements:

- Original features (t_1): x_1, x_2, x_3, x_4 (4 terms)
- Squared terms (t_2): $x_1^2, x_2^2, x_3^2, x_4^2$ (4 terms)
- Cross-product terms (t_3): $x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, x_3x_4$ (6 terms)

$$dim(\Phi(x)) = t_1 + t_2 + t_3 = 4 + 4 + 6 = 14$$

$\therefore$ The dimension of $\Phi(x)$ is 14.

Solution 2 (b)

Step 1

The decision boundary of the Perceptron has the form:

$$w \cdot \Phi(x) + b = 0$$

Step 2

Now, expand the dot product of the decision boundary.

$$w_1 x_1 + \ldots + w_4 x_4 + w_5 x_1^2 + \ldots + w_{14} x_3 x_4$$

From **Solution 2 (a)** we know that the dimension of $\Phi(x)$ is 14, and it follows that the weight vector w must also have 14 components to match each feature in the expanded space.

$\therefore$ The dimension of w is 14.

Solution 3 (a)

Step 1

When we map to the expanded feature space via $\Phi(x)$, we preserve the linear separability of the dataset. This is because:

- If a dataset is separable in a space $\mathbb{R}^d$, it remains separable when mapped to a higher-dimensional space $\mathbb{R}^D$ where $D > d$.
- The new space actually provides additional flexibility for finding a separating hyperplane. The original linear separator can be represented in the expanded space by setting the weights for all quadratic terms to zero and keeping the weights for the linear terms as they were in the original space.

Hence, when we apply the basis expansion $\Phi(x)$ to our linearly separable dataset, we preserve the linear separability of the dataset. According to the Perceptron convergence theorem, the algorithm will converge in a finite number of steps.

$\therefore$ The Perceptron algorithm will converge when using the given basis expansion on a linearly separable dataset.

Solution 3 (b)

Step 1

We are given:

1. The original dataset is linearly separable in $\mathbb{R}^d$.
2. This means there exists at least one solution using only the original features (i.e., a solution where quadratic term weights are zero).
3. However, in the expanded feature space, there can be many other separating hyperplanes that use the quadratic terms.

Step 2

Note: The Perceptron algorithm does not optimize for having zero weights in particular dimensions, it simply tries to find any separating hyperplane.

The final weight vector depends on:

- The initialization of the weights
- The order in which training examples are presented
- The number of updates made during training

Hence, the Perceptron has no inherent bias toward solutions with zero weights for the quadratic terms.

Note: There is no guarantee that it will find such a solution, even though such a solution could exist.

$\therefore$ The Perceptron algorithm will not necessarily return a weight vector w in which the entries corresponding to quadratic terms in $\Phi(x)$ are zero.

Solution 4 (a)

Step 1

There are five updates in the following order:

- Zero updates on point $x^{(1)}$ (label +1)
- Two updates on point $x^{(2)}$ (label +1)
- Two updates on point $x^{(3)}$ (label -1)
- One update on point $x^{(4)}$ (label -1)

Note: $\alpha = (\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}\alpha^{(4)})$, *where* $\alpha^{(i)} = \alpha^{(i)} + 1$ *when the point is misclassified.*

Step 2

Calculate $\alpha^{(i)}$ for each point.

$$\alpha^{(1)} = \alpha^{(1)} + 1 = \text{updates}_1 = 0$$
$$\alpha^{(2)} = \alpha^{(2)} + 1 = \text{updates}_2 = 2$$
$$\alpha^{(3)} = \alpha^{(3)} + 1 = \text{updates}_3 = 2$$
$$\alpha^{(4)} = \alpha^{(4)} + 1 = \text{updates}_4 = 1$$

$\therefore$ The final value of α is $(0, 2, 2, 1)$.

Solution 4 (b)

Step 1

There are five updates in the following order:

- Zero updates on point $x^{(1)}$ (label +1)
- Two updates on point $x^{(2)}$ (label +1)
- Two updates on point $x^{(3)}$ (label -1)
- One update on point $x^{(4)}$ (label -1)

Note: $b = b^{(1)} + b^{(2)} + b^{(3)} + b^{(4)}$*, where* $b^{(i)} = b^{(i)} + y^{(i)}$ *when the point is misclassified.*

Step 2

Calculate $b^{(i)}$ for each point and find sum.

$$b^{(1)} = b^{(1)} + y^{(1)} = \text{updates}_n \cdot \text{label} = 0 \cdot 1 = 0$$
$$b^{(2)} = b^{(2)} + y^{(2)} = \text{updates}_n \cdot \text{label} = 2 \cdot 1 = 2$$
$$b^{(3)} = b^{(3)} + y^{(3)} = \text{updates}_n \cdot \text{label} = 2 \cdot -1 = -2$$
$$b^{(4)} = b^{(4)} + y^{(4)} = \text{updates}_n \cdot \text{label} = 1 \cdot -1 = -1$$
$$b = b^{(1)} + b^{(2)} + b^{(3)} + b^{(4)} = 0 + 2 + (-2) + (-1) = -1$$

$\therefore$ The final value of b is -1.

Solution 5 (a)

Step 1

The vector α corresponds to the number of training points. In the example from class we have 18 red circles and 18 black triangles, for a total of 36 training points.

$\therefore$ The dimension of α is 36.

Solution 5 (b)

Step 1

The condition $\alpha_i > 0$ is simply the count of support vectors for the SVM and we have 6 support vectors in the example from class.

$\therefore$ There are 6 entries in α that are > 0.

Solution 5 (c)

Step 1

For a kernel SVM, α_i is subject to the following constraint:

$$0 \leq \alpha_i \leq C \quad \forall i$$

$\therefore$ There are 0 entries in α that are < 0.

Solution 5 (d)

Step 1

We have 2 support vectors for the black triangle class and 4 support vectors for the red circle class. The resulting SVM has to optimize the margin more for the red circles and leads to a larger green margin.

$\therefore$ The imbalance of support vectors between the classes leads to a larger green margin.

Solution 6

Python Code

```python
def quadratic_kernel(x, z, c):
    return (np.dot(x, z) + c)**2

def rbf_kernel(x, z, c):
    return np.exp(-np.linalg.norm(x-z)**2/(2*c**2))

def train(x, y, kernel_function, c=1.0, max_iter=1000):
    n_samples = len(x)
    alpha = np.zeros(n_samples)
    b = 0

    ## calculate kernel matrix before looping through dataset
    K = np.zeros((n_samples, n_samples))
    for i in range(n_samples):
        for j in range(n_samples):
            K[i, j] = kernel_function(x[i], x[j], c)

    ## training loop for max iterations
    for _ in range(max_iter):
        mistakes = False
        for i in range(n_samples):
            ## calculate predicted label
            f_xi = 0
            for j in range(n_samples):
                f_xi += alpha[j] * y[j] * K[i, j]
            f_xi += b
            ## update b and alpha if prediction is wrong
            if y[i] * f_xi <= 0:
                alpha[i] += 1
                b += y[i]
                mistakes = True
        ## stop training loop if all predictions are correct
        if not mistakes:
            print("converged!")
            break
    return alpha, b

def classify(x, y, alpha, b, xi, kernel_function, c=1.0):
    f_xi = 0
    for i in range(len(x)):
        f_xi += alpha[i] * y[i] * kernel_function(x[i], xi, c)
    f_xi += b
    return 1 if f_xi >= 0 else -1
```

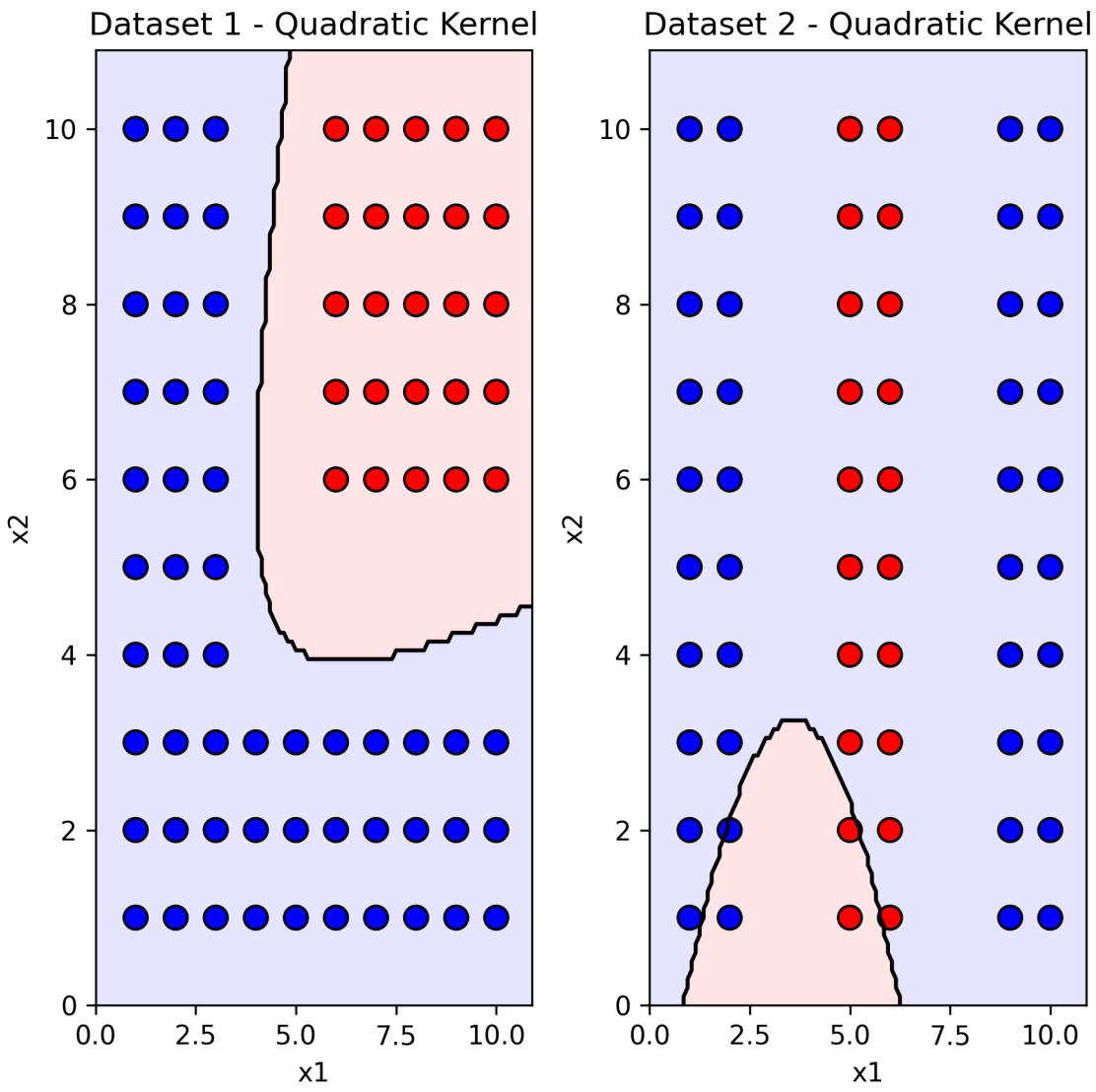

Figure 1: Decision boundary for quadratic kernel

Plots

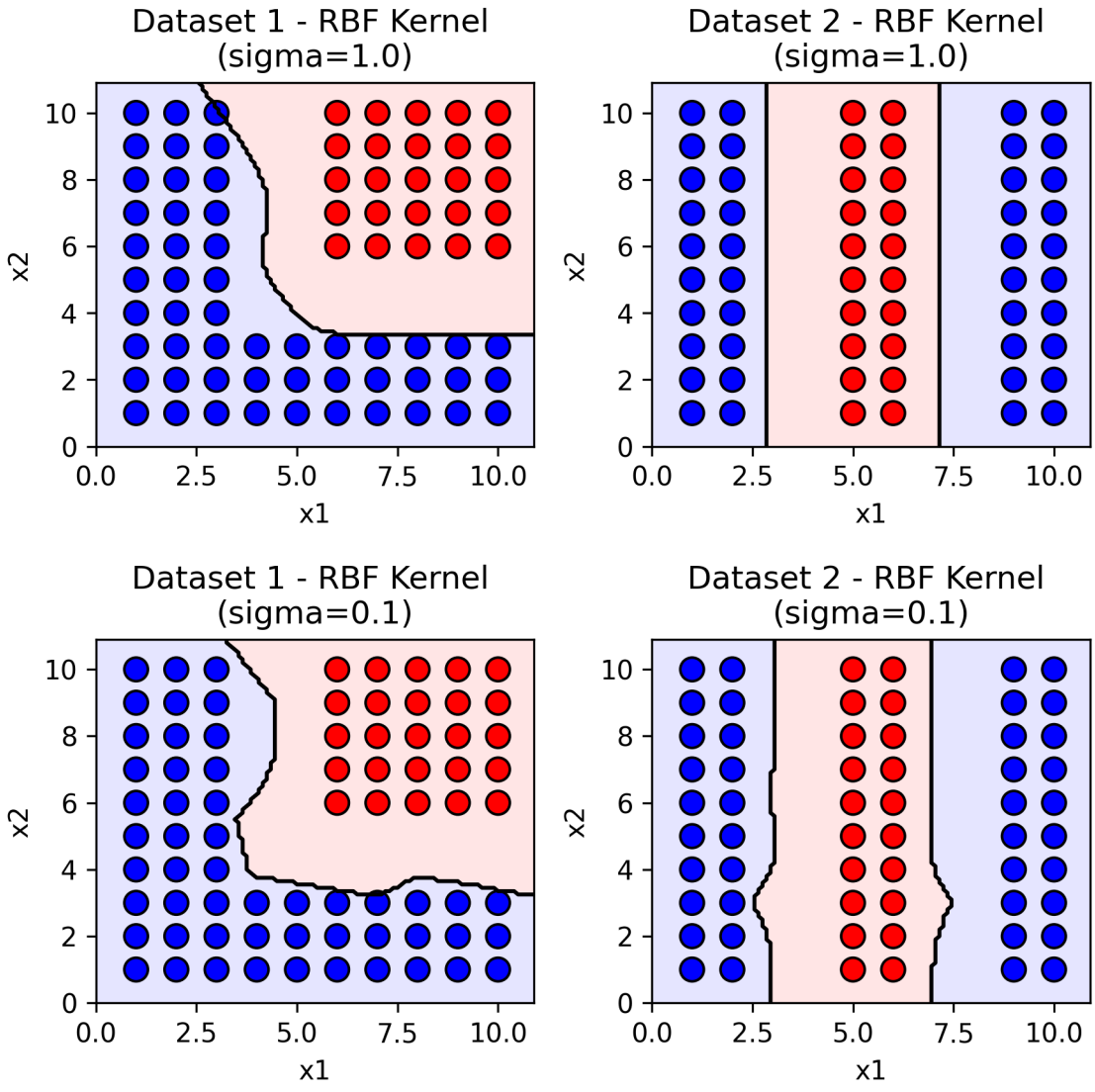

Figure 2: Decision boundary for rbf kernel

Solution 7

Linear SVM

C Value	Training Error (%)	Test Error (%)
0.01	16.32	15.48
0.1	15.91	16.36
1.0	14.25	14.04
10.0	13.56	13.16
100.0	14.05	14.78

Based on the results above it seems that the data is linearly separable.

Quadardic SVM

Training Error (%)	Test Error (%)	Support Vectors
0.17	2.13	17906