HW 5 Solutions

Author:
Fateme Noorzad

StudentID:
810198271

Machine Learning
Winter 2021

Contents

Chapter 1

Question 1

1.1 SVM

In this part we are asked to find the separation lines using SVM. To do so, we need to find the lagrangian function.

We know that for the SVM problem, the lagrangian function is defined as below:

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i[1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)]$$

Based on the given figure, there are three data points in a 2-D space as $\begin{bmatrix}-1\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\end{bmatrix}$, and $\begin{bmatrix}1\\0\end{bmatrix}$. Therefore, $n = 3$, $\mathbf{w} = \begin{bmatrix}w_1\\w_2\end{bmatrix}$. In addition, blue data points are labeled as $y_i = 1$ and the red data point as $y_i = -1$. With all these in mind, the above lagrangian function is as below:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\begin{bmatrix}w_1 & w_2\end{bmatrix}\begin{bmatrix}w_1\\w_2\end{bmatrix} + \sum_{i=1}^{3} \alpha_i[1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)] \\
&= \frac{w_1^2 + w_2^2}{2} + \alpha_1[1 - (-1)(\begin{bmatrix}w_1 & w_2\end{bmatrix}\begin{bmatrix}-1\\0\end{bmatrix} + b)] \\
&\quad + \alpha_2[1 - (1)(\begin{bmatrix}w_1 & w_2\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} + b)] \\
&\quad + \alpha_3[1 - (1)(\begin{bmatrix}w_1 & w_2\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + b)] \\
\rightarrow \mathcal{L} &= \frac{w_1^2 + w_2^2}{2} + \alpha_1[1 + (-w_1 + b)] + \alpha_2[1 - (w_2 + b)] + \alpha_3[1 - (w_1 + b)]
\end{aligned}$$

In order to evaluate the lagrangian coefficients, we find the partial derivations:

$$\begin{cases}
\frac{\partial}{\partial w_1}\mathcal{L} = 0 \rightarrow w_1 - \alpha_1 - \alpha_3 = 0 \rightarrow w_1 = \alpha_1 + \alpha_3 \\
\frac{\partial}{\partial w_2}\mathcal{L} = 0 \rightarrow w_2 - \alpha_2 = 0 \rightarrow w_2 = \alpha_2 \\
\frac{\partial}{\partial b}\mathcal{L} = 0 \rightarrow \alpha_1 - \alpha_2 - \alpha_3 = 0 \rightarrow \alpha_1 + \alpha_3 = \alpha_2
\end{cases}$$

Now based on the definitions of w_1 and w_2 based on lagrangian coefficients, we write the lagrangian function:

$$\mathcal{L} = \frac{(\alpha_1 + \alpha_3)^2 + \alpha_2^2}{2} + \alpha_1[1 + (-(\alpha_1 + \alpha_3) + b)] + \alpha_2[1 - (\alpha_2 + b)] + \alpha_3[1 - (\alpha_1 + \alpha_3 + b)]$$

Now to calculate these coefficients, we use partial derivation as below:

$$\begin{cases} \frac{\partial}{\partial \alpha_1}\mathcal{L} = 0 \rightarrow \alpha_1 + \alpha_3 + 1 - \alpha_1 - \alpha_3 + b - \alpha_1 - \alpha_3 = 0 \rightarrow \alpha_1 + \alpha_3 = 1 + b \\ \frac{\partial}{\partial \alpha_2}\mathcal{L} = 0 \rightarrow \alpha_2 + 1 - \alpha_2 - b - \alpha_2 = 0 \rightarrow \alpha_2 = 1 - b \\ \frac{\partial}{\partial \alpha_3}\mathcal{L} = 0 \rightarrow \alpha_1 + \alpha_3 + 1 - \alpha_1 - \alpha_3 - b - \alpha_1 - \alpha_3 = 0 \rightarrow \alpha_1 + \alpha_3 = 1 - b \end{cases}$$

Therefore, we get:

$$\begin{gathered} 1 - b = 1 + b \rightarrow \boxed{b = 0} \\ \rightarrow \boxed{\alpha_2 = w_2 = 1} \\ \rightarrow \boxed{w_1 = \alpha_1 + \alpha_3 = 1} \\ \rightarrow \begin{cases} \alpha_1 + \alpha_3 = 1 \\ \alpha_1 - \alpha_3 = 1 \end{cases} \quad \rightarrow \boxed{\alpha_1 = 1,\ \alpha_3 = 0} \end{gathered}$$

The last step is to check the four KKT conditions for the calcualted values. If we do so, all the conditions hold. In addition, since it is a convex problem, the evaluated values are the optimum points as well.

Based on the above evaluations, the separating lines can be represented as below:

$$\mathbf{w}^T\mathbf{x} + b = 1,\ \mathbf{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ b = 0 \rightarrow x_1 + x_2 = 1$$

$$\mathbf{w}^T\mathbf{x} + b = -1,\ \mathbf{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ b = 0 \rightarrow x_1 + x_2 = -1$$

1.2 SV

As we know, the data points which their lagrangian coefficinet is non-zero create SVs. Therfore, $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are the ones considered for this task.

$$\mathbf{w} = \alpha_1 y_1 \mathbf{x}_1 + \alpha_2 y_2 \mathbf{x}_2 = 1(-1)\begin{bmatrix} -1 \\ 0 \end{bmatrix} + 1(1)\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Chapter 2

Question 2

2.1 Part 1

The aftermath of the primal problem is the optimal value of $\mathbf{w}$, without gaining any insight about the lagrangian multipliers. This can cause a huge computation cost in classifing a new arrival point in a high-dimensional space. The reason lying behind this fact is that computing $\mathbf{w}^T\mathbf{x} + b$ is an arduous task when d is large.

In fact, to make any predictions regarding the new point represented as $\mathbf{x}$, as was mentioned, we are required to evaluate $\mathbf{w}^T\mathbf{x} + b$, and predict $y = 1$ if and only if this quantity is bigger than zero. However, if we employ $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ to the stated equation, we get:

$$\mathbf{w}^T\mathbf{x} + b = (\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i)^T\mathbf{x} + b = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b$$

Hence, if we find the values of α_i's, which come from solving the dual problem, in order to make a prediction, we have to calculate a quantity which only depends on the inner product of $\mathbf{x}$ and the training dataset's points. In addition, all α_i's are zero except for the support vectors. Thus, many terms in the above summation will be zero, and we are only opt to calculate the inner products between $\mathbf{x}$ and the support vectors (of which there are only small number) in order to make a prediction.

To sum up:

- The dual problem gives significant insight regarding the structure of the classification problem.
- The whole algorithm can be written in terms of only inner products between input feature vectors. Hence, making the problem independent of the input dimensions which makes it cost optimal even when d is large.
- The affine equality constraint can be eliminated from the dual formulation.
- By the last point, the dual problem can be easily cast as a convex quadratic optimization problem whose constrains are only bound constrains.
- The dual problem can be solved efficiently, i.e. via a dual coordinate descent algorithm.
- They provide conditions which help using *Kernel Trick* and other numerical methods to solve the non-linear problems.

2.2 Part 2

- **Hard Margin SVM**:
  In this method, we assume that all positive points lie above the positive plane and all negative points lie below the negative plane and no points lie in between the margin. Mathematically speaking, the optimization problem has the below form:

$$min\ \frac{1}{2}||\mathbf{w}||^2$$
$$s.t.\ 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0 \qquad for\ i = 1, ..., n$$

- **Soft Margin SVM**:
  In this method, we opt to make zero classification error in training, but to make a few errors in necessary conditions. Therefore, the optimization problem can be represent in the methematical form as below:

$$min\ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}|\zeta_i|$$
$$s.t.\ |y_i - w_i x_i| \leq \epsilon + |\zeta_i| \qquad for\ i = 1, ..., n$$

  where ζ is the distance of the misclassified point from its correct hyperplane and C tells us how important ζ is.

As the above explanations regarding both of these methods suggest, hard margin is sensitive to outlier data, while in soft margin, due to the possibility of error, this sensitivity is reduced. (Almost no sensitivity.) Besides, in cases where data cannot be separated by simple lines, soft margin results in better separation, due to the same reason. The below plot can make the stated fact more clear:

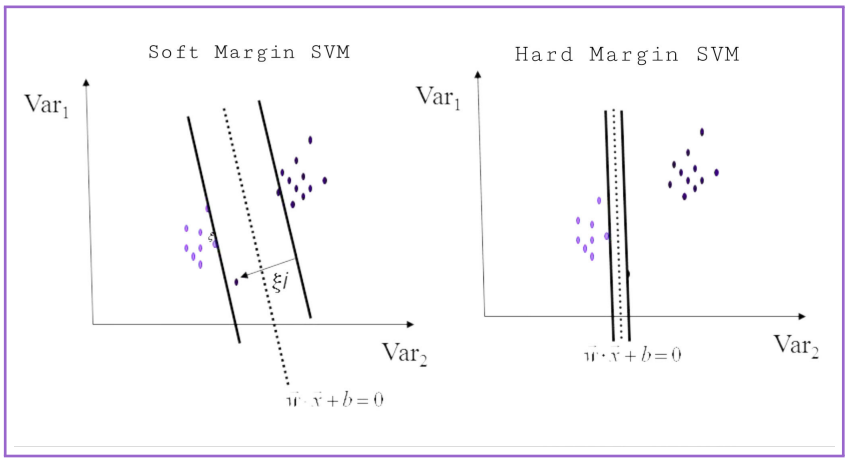

FIGURE 2.1: Soft-Margin and Hard-Margin SVM

2.3 Part 3

As the last part's explanations regarding soft margin SVM suggests, parameter C is our tool to have a control on the soft margin, i.e. it tells us how important ζ should be. In other words, it is a generalization parameter that controls the trade-off

between a low training error and a low test error, which will control sensitiveness to various errors and outlier data.

If the value of C is very high, then we try to minimize the number of misclassified points drastically which results in overfitting. With the same explanation, for small values of C, underfitting takes place.

The below plot gives a good vitualization on the role of C and how its value effects classification.

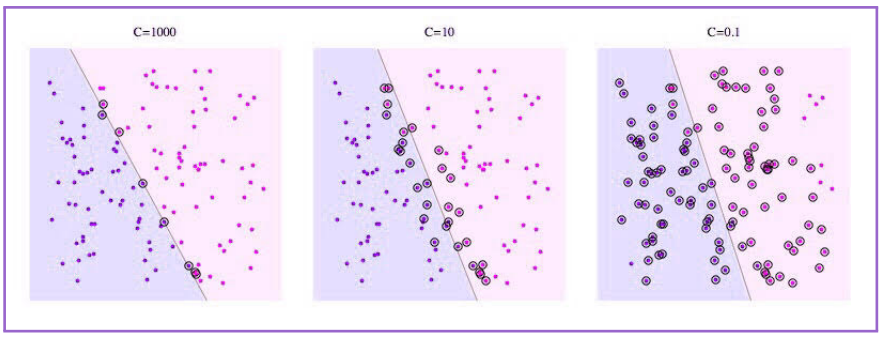

FIGURE 2.2: Classification for Various Values of C

Chapter 3

Question 3

3.1 What Are Kernel Functions?

Kernel Functions are employed to provide a bridge between linearly and non-linearly alogrithms which is built easily. In cases where the boundary between two sets of points are crooked, the usual algorithms will take a long time to converge. Even when they converge it might not be the most optimal boundary.
The main charactristics of Kernel Functions is their distinct approach to this problem. Instead of taking hard route of classifying data in lower dimension by putting a really curvy boundary, Kernel Functions map the data into higher dimensional spaces in the hpe that data is more easily separated there. There are also no constrains on the form of this mapping, which could even lead to infinite-dimensional spaces. The merit about this mapping is that it hardly requires to be computed due to the existance of *Kernel Trick*.

3.2 What Is the Kernel Trick?

The Kernel Trick is a mathematical tool which can be applied to anu algorithm which sloley depends on the dot product between two vectors. Since when we want to transform data to higher dimensions, we want to do this task meticulously, and as mentioned, do not want to calculate the procedure explicitly, dot product-based algorithms are the keys to achieve these goals. By employing the kernel functions, the dot product of data can be directly evaluated by the lower dimension vectors.
Therefore, putting a curved line in the lower dimension will be equivalent to putting a linear boundary in the higher dimension. In addition, there are no constrains on the nature of the input vectors.

3.3 Why Do We Need Kernel Functions?

As was discussed earlier, due to the arduous task of separating data points in lower dimensions, specially when the number of classes and data point increase, a method to trasform data into a higher dimension is mandatory. In a higher dimension space, drawing a simple highperplane can solve the classification problem.

3.4 Proof of $k(x,x)k(y,y) \geq (k(x,y))^2$

For a kernel we have:

$$\begin{bmatrix} k(x,x) & k(y,x) \\ k(x,y) & k(y,y) \end{bmatrix} = \begin{bmatrix} k(x,x) & k(x,y) \\ k(x,y) & k(y,y) \end{bmatrix}$$

Based on the Mercer Theorem, for $K : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$ to be a valid kenel, it is nessesary and sufficient that for any $\{\mathbf{x}_1,\ \mathbf{x}_2,\ ...,\ \mathbf{x}_n\}$ (for all $n < \infty$), the corresponding kernel matrix is symmetric positive semi-definite. Therefore:

$$det\left(\begin{bmatrix} k(x,x) & k(x,y) \\ k(x,y) & k(y,y) \end{bmatrix}\right) \geq 0 \to k(x,x)k(y,y) - (k(x,y))^2 \geq 0 \to \boxed{k(x,x)k(y,y) \geq (k(x,y))^2}$$

3.5 Proof of $\frac{1}{Q}\sqrt{\sum_{i=1}^{Q}\sum_{j=1}^{Q}K(\mathbf{x}_i,\mathbf{x}_j)}$

Based on the definition of mean value, we know:

$$\mu_Q = \frac{\phi(\mathbf{x}_1) + \phi(\mathbf{x}_2) + ... + \phi(\mathbf{x}_Q)}{Q}$$

In addition, the norm of μ_Q is defined as:

$$||\mu_Q|| = \sqrt{\mu_Q^T \mu_Q}$$

If these two are considered together, we get:

$$\begin{aligned} ||\mu_Q|| &= \sqrt{\left(\frac{\phi(\mathbf{x}_1) + ... + \phi(\mathbf{x}_Q)}{Q}\right)^T \left(\frac{\phi(\mathbf{x}_1) + ... + \phi(\mathbf{x}_Q)}{Q}\right)} \\ &= \frac{1}{Q}\sqrt{\sum_{i=1}^{Q}\sum_{j=1}^{Q}\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)} \\ &= \frac{1}{Q}\sqrt{\sum_{i=1}^{Q}\sum_{j=1}^{Q}K(\mathbf{x}_i,\mathbf{x}_j)} \end{aligned}$$

Chapter 4

Question 4

Based on Mercer Theorem, for $K : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$ to be a valid kenel, it is nessesary and sufficient that for any $\{\mathbf{x}_1,\ \mathbf{x}_2,\ ...,\ \mathbf{x}_n\}$ (for all $n < \infty$), the corresponding kernel matrix be a symmetric positive semi-definite one. Therefore, we are required to prove that for the given $K(x, y)$, the kernel matrix is both **symmetric** and **positive semi-definite**.

4.1 $K(x, y) = f(x)K_1(x, y)f(y)$

To prove the stated features, first of all, we need to form the kernel matrix.

$$K(x, y) = f(x)K_1(x, y)f(y)$$

$$\to K = \begin{bmatrix} f(x_1)K_1(x_1, y_1)f(y_1) & \cdots & f(x_1)K_1(x_1, y_n)f(y_n) \\ f(x_2)K_1(x_2, y_1)f(y_1) & \cdots & f(x_2)K_1(x_2, y_n)f(y_n) \\ \vdots & \ddots & \vdots \\ f(x_n)K_1(x_n, y_1)f(y_1) & \cdots & f(x_n)K_1(x_n, y_n)f(y_n) \end{bmatrix}$$

Now if we define a new matrix as below:

$$F(\mathbf{x}) = \begin{bmatrix} f(x_1) & 0 & \cdots & 0 \\ 0 & f(x_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(x_n) \end{bmatrix} = F$$

While keeping in mind that:

$$K_1 = \begin{bmatrix} K_1(x_1, y_1) & \cdots & K_1(x_1, y_n) \\ K_1(x_2, y_1) & \cdots & K_1(x_2, y_n) \\ \vdots & \ddots & \vdots \\ K_1(x_n, y_1) & \cdots & K_1(x_n, y_n) \end{bmatrix}$$

We get:

$$K = FK_1F^T$$

- **Positive Semi-Definite**:

$$\mathbf{z}^T K\mathbf{z} = \mathbf{z}^T FK_1F^T\mathbf{z},\ \mathbf{w} = F^T\mathbf{z} \to \mathbf{z}^T K\mathbf{z} = \mathbf{w}^T K_1\mathbf{w}$$

Based on the fact that $K_1(x,y)$ is a valid kernel, K_1 is a positive semi-definite matrix. Hence, for all $\mathbf{w} \in \mathcal{R}^d$, we can conclude that $\mathbf{w}^T K_1 \mathbf{w} \geq 0$ holds. As a result, for all $\mathbf{z} \in \mathcal{R}^d : \mathbf{z}^T K \mathbf{z} \geq 0$ is clear and thus its positive semi-definiteness.

- **Symmetri**:

$$K^T = (FK_1F^T)^T = (F^T)^T(K_1)^T F^T = FK_1F^T = K$$

Therefore, if $K_1(x,y)$ is a valid kernel, so is $K(x,y)$.

4.2 $K(x,y) = exp(K_1(x,y))$

To prove the stated features, first of all, we need to form the kernel matrix.

$$K(x,y) = exp(K_1(x,y))$$

$$\rightarrow K = \begin{bmatrix} exp(K_1(x_1,y_1)) & \cdots & exp(K_1(x_1,y_n)) \\ exp(K_1(x_2,y_1)) & \cdots & exp(K_1(x_2,y_n)) \\ \vdots & \ddots & \vdots \\ exp(K_1(x_n,y_1)) & \cdots & exp(K_1(x_n,y_n)) \end{bmatrix}$$

While keeping in mind that:

$$K_1 = \begin{bmatrix} K_1(x_1,y_1) & \cdots & K_1(x_1,y_n) \\ K_1(x_2,y_1) & \cdots & K_1(x_2,y_n) \\ \vdots & \ddots & \vdots \\ K_1(x_n,y_1) & \cdots & K_1(x_n,y_n) \end{bmatrix}$$

We get:

$$K = exp(K_1)$$

- **Positive Semi-Definite**:

$$\mathbf{z}^T K \mathbf{z} = \mathbf{z}^T exp(K_1)\mathbf{z} = \mathbf{z}^T \left(\sum_{i=0}^{\infty} \frac{(K_1)^i}{i!} \right) \mathbf{z}$$

Based on the fact that $K_1(x,y)$ is a valid kernel, K_1 is a positive semi-definite matrix. Therefore, $(K_1)^i$ for all $i \in [0,\infty)$ is a positive semi-definite matrix. Hence, for all $\mathbf{z} \in \mathcal{R}^d$, we can conclude that $\mathbf{z}^T(\sum_{i=0}^{\infty} \frac{K_1^i}{i!})\mathbf{z} \geq 0$ holds. As a result, for all $\mathbf{z} \in \mathcal{R}^d : \mathbf{z}^T K \mathbf{z} \geq 0$ is clear and thus its positive semi-definiteness.

- **Symmetri**:

$$K^T = \left(\sum_{i=0}^{\infty} \frac{K_1^i}{i!} \right)^T = \sum_{i=0}^{\infty} \frac{K_1^i}{i!} = K$$

Therefore, if $K_1(x,y)$ is a valid kernel, so is $K(x,y)$.

4.3 $K(x, y) = K_1(x, y) + K_2(x, y)$

To prove the stated features, first of all, we need to form the kernel matrix.

$$K(x, y) = K_1(x, y) + K_2(x, y)$$

$$\rightarrow K = \begin{bmatrix} K_1(x_1, y_1) + K_2(x_1, y_1) & \cdots & K_1(x_1, y_n) + K_2(x_1, y_n) \\ K_1(x_2, y_1) + K_2(x_2, y_1) & \cdots & K_1(x_2, y_n) + K_2(x_2, y_n) \\ \vdots & \ddots & \vdots \\ K_1(x_n, y_1) + K_2(x_n, y_1) & \cdots & K_1(x_n, y_n) + K_2(x_n, y_n) \end{bmatrix}$$

While keeping in mind that:

$$K_1 = \begin{bmatrix} K_1(x_1, y_1) & \cdots & K_1(x_1, y_n) \\ K_1(x_2, y_1) & \cdots & K_1(x_2, y_n) \\ \vdots & \ddots & \vdots \\ K_1(x_n, y_1) & \cdots & K_1(x_n, y_n) \end{bmatrix}, \; K_2 = \begin{bmatrix} K_2(x_1, y_1) & \cdots & K_2(x_1, y_n) \\ K_2(x_2, y_1) & \cdots & K_2(x_2, y_n) \\ \vdots & \ddots & \vdots \\ K_2(x_n, y_1) & \cdots & K_2(x_n, y_n) \end{bmatrix}$$

We get:

$$K = K_1 + K_2$$

- **Positive Semi-Definite**:

$$\mathbf{z}^T K \mathbf{z} = \mathbf{z}^T (K_1 + K_2)\mathbf{z} = \mathbf{z}^T K_1 \mathbf{z} + \mathbf{z}^T K_2 \mathbf{z}$$

  Based on the fact that $K_1(x, y)$ and $K_2(x, y)$ are valid kernels, K_1 and K_2 are positive semi-definite matrices. Hence, for all $z \in \mathcal{R}^d$, we can conclude that $\mathbf{z}^T K_1 \mathbf{z} \geq 0$ and $\mathbf{z}^T K_2 \mathbf{z} \geq 0$ holds. As a result, for all $z \in \mathcal{R}^d : \mathbf{z}^T K \mathbf{z} \geq 0$ is clear and thus its positive semi-definiteness.

- **Symmetri**:

$$K^T = (K_1 + K_2)^T = (K_1)^T + (K_2)^T = K_1 + K_2 = K$$

  Therefore, if $K_1(x, y)$ and $K_2(x, y)$ are valid kernels, so is $K(x, y)$.

4.4 $K(x, y) = K_1(x, y)K_2(x, y)$

To prove the stated features, first of all, we need to form the kernel matrix.

$$K(x, y) = K_1(x, y)K_2(x, y)$$

$$\rightarrow K = \begin{bmatrix} K_1(x_1, y_1)K_2(x_1, y_1) & \cdots & K_1(x_1, y_n)K_2(x_1, y_n) \\ K_1(x_2, y_1)K_2(x_2, y_1) & \cdots & K_1(x_2, y_n)K_2(x_2, y_n) \\ \vdots & \ddots & \vdots \\ K_1(x_n, y_1)K_2(x_n, y_1) & \cdots & K_1(x_n, y_n)K_2(x_n, y_n) \end{bmatrix}$$

While keeping in mind that:

$$K_1 = \begin{bmatrix} K_1(x_1,y_1) & \cdots & K_1(x_1,y_n) \\ K_1(x_2,y_1) & \cdots & K_1(x_2,y_n) \\ \vdots & \ddots & \vdots \\ K_1(x_n,y_1) & \cdots & K_1(x_n,y_n) \end{bmatrix}, \; K_2 = \begin{bmatrix} K_2(x_1,y_1) & \cdots & K_2(x_1,y_n) \\ K_2(x_2,y_1) & \cdots & K_2(x_2,y_n) \\ \vdots & \ddots & \vdots \\ K_2(x_n,y_1) & \cdots & K_2(x_n,y_n) \end{bmatrix}$$

We get:

$$K = K_1 \circ K_2$$

- **Positive Semi-Definite**:

$$if: \; K_1 = \sum_i a_i k_{1,i} k_{1,i}^T, \; K_2 = \sum_i b_i k_{2,i} k_{2,i}^T$$
$$\rightarrow K = \sum_{ij} a_i b_j (k_{1,i} k_{1,i}^T) \circ (k_{2,j} k_{2,j}^T) = \sum_{ij} a_i b_j (k_{1,i} \circ k_{2,j})(k_{1,i} \circ k_{2,j})^T$$

  Based on the fact that $K_1(x,y)$ and $K_2(x,y)$ are valid kernels, K_1 and K_2 are positive semi-definite matrices. Hence, each $(k_{1,i} \circ k_{2,j})$ and $(k_{1,i} \circ k_{2,j})^T$ are positive semi-definite. Besides, both a_i and b_j are positive, thus K is a positive semi-definite matrix.

- **Symmetri**:

$$K^T = (K_1 \circ K_2)^T = (K_1)^T \circ (K_2)^T = K_1 \circ K_2 = K$$

  Therefore, if $K_1(x,y)$ and $K_2(x,y)$ are valid kernels, so is $K(x,y)$.

4.5 $K(x,y) = x^T Ay$

This one is the same as the formula proved in the first section, with $f(x) = x^T$, $K_1(x,y) = A$, and $f(y) = y$. Therefore, $K(x,y)$ is a valid kernel. However, below, its proof can be found:

To prove the stated features, first of all, we need to form the kernel matrix.

$$K(x,y) = x^T Ay$$

$$\rightarrow K = \begin{bmatrix} x_1 Ay_1 & \cdots & x_1 Ay_n \\ x_2 Ay_1 & \cdots & x_2 Ay_n \\ \vdots & \ddots & \vdots \\ x_n Ay_1 & \cdots & x_n Ay_n \end{bmatrix}$$

Now if we define a new matrix as below:

$$F(\mathbf{X}) = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{nn} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix} = F$$

We get:

$$K = FAF^T$$

- **Positive Semi-Definite**:

$$\mathbf{z}^T K\mathbf{z} = \mathbf{z}^T FAF^T\mathbf{z},\ \mathbf{w} = F^T\mathbf{z} \rightarrow \mathbf{z}^T K\mathbf{z} = \mathbf{w}^T A\mathbf{w}$$

  Based on the fact that A is a positive semi-definite matrix, for all $\mathbf{w} \in \mathcal{R}^d$, we can conclude that $\mathbf{w}^T A\mathbf{w} \geq 0$ holds. As a result, for all $\mathbf{z} \in \mathcal{R}^d : \mathbf{z}^T K\mathbf{z} \geq 0$ is clear and thus its positive semi-definiteness.

- **Symmetri**:

$$K^T = (FAF^T)^T = (F^T)^T(A)^T F^T = FAF^T = K$$

  Therefore, if A a positive semi-definite matrix, $K(x, y)$ is a valid kernel.

Chapter 5

Question 5

5.1 Gaussian Kernel

$$
\begin{aligned}
||\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)||^2 &= \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_1) \rangle + \langle \phi(\mathbf{x}_2), \phi(\mathbf{x}_2) \rangle - 2\langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle \\
&= exp(-\frac{\left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\|^2}{2}) + exp(-\frac{\left\| \begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 3 \\ 4 \end{bmatrix} \right\|^2}{2}) \\
&\quad -2exp(-\frac{\left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 4 \end{bmatrix} \right\|^2}{2}) \\
&= 2 - 2exp(-\frac{13}{2}) \\
&= 1.9969 \rightarrow \boxed{d = 1.4131}
\end{aligned}
$$

5.2 Dimension of the New Space

If we take a better look at the given kernel, we have:

$$K(x, y) = (\mathbf{x}^T \mathbf{y} + 1)^2 = (x_1 y_1 + x_2 y_2 + ... + x_d y_d + 1)^2$$

Therefore, the space made by these terms, is $\binom{d+2}{2} = \frac{(d+2)!}{2!d!} = \frac{(d+2)(d+1)}{2}$ dimensional.

Chapter 6

Question 6

6.1 Part 1

- **Generative Approach**:
  This approach creates a joint model of feature vector and classes, represented by $p(X, \omega)$. By employing this joint model, $p(\omega_j|k)$ can be driven, in order to form the classifier. It is essnetial to note that finding $p(X, \omega)$ is an arduous task because it needs lots of samples. Therefore, we may face curse of dimensionality.

- **Discriminatice Approach**:
  This approach creates a model of the form of $p(\omega_j|k)$ from the beggining. In fact, it finds the decision boundaries to form the classifier directly. Therefore, it requires fewer samples.

6.2 Part 2

- **One-Vs-Rest for Multi-Class Classification**:
  It is a heuristic method for using binary classification algorithms for multi-class classification. To do so, it splits the multi-class dataset into multiple binary classification problems. Afterward, a binary classifier is trained on each binary classification problem and predictions are made using the model that is the most cofident. On the grounds of how it works, for a C class classification problem, we are required to train C binary classifiers.

  Based on the above explanations, a possible downside of this approach is that it requires one model to be created and trained for each class. In the case of large datasets (e.g. having millions of rows), slow models (e.g. neural networks), or very large number of classes (e.g. hundreds of classes), this requirement can be an issue. In addition, while finding the decision boundaries, some regions may be left out without being labled.

  This approach is commonly used for algorithms that naturally predict numerical class membership, such as Logistic Regression.

- **One-Vs-One for Multi-Class Classification**:
  It is another heuristic method for using binary classification algorithms (such as Logistic Regression, SVM, and so on) for multi-class classification.

  Like the last one, it splits a multi-class classification dataset into binary classification problems. However, unlike one-vs-rest, it splits the dataset into one dataset for each class versus every other class. As a result, for a C class dataset, we are required to solve $\frac{C(C-1)}{2}$ binary classification problems. This is significantly more than the last strategy, especially when C is large. To make a decision, each binary model predicts one class label and the model with the

most predictions is predicted by this strategy.

Classically, this approach is suggested for SVM and related kernel-based algorithms. The reason laying behind this idea is that the performance of the kernel methods does not scale in proportion to the size of the training dataset and employing subsets of the training data may counter this effect.

- **Linear Machine for Classification**:

  It is a classifier that uses linear discriminant fucntion. In fact, for a C class dataset, it divides the feature space into C decision regions, with $g_i(\underline{X})$ being the largest discriminant if $\underline{X}$ is in region R_i. In the case where R_i and R_j are contiguous, the boundary between them is portion of the hyperplane H_{ij} defined by $g_i(\underline{X}) = g_j(\underline{X})$.

  In this method, it is not the weight vectors themselves but their difference that are essential for us. In fact, it is true that there are $\frac{C(C-1)}{2}$ pairs of regions, they need not all be congiuous, and the total number of hyperplane segments appearing in the decision surface is often fewer than $\frac{C(C-1)}{2}$.

  In addition, it is essential to note that the decision regions for this methos are convex, making the results optimal. However, these restrictions surely limit the flexibility and accuracy of the classifier, as well as making the problem computationally costly. In particular, for good performance, every decision region should be singly connected, and this tends to make linear machine most suitable for problmes in which the conditional densities are unimodel.

6.3 Part 3

It is common to add a block as *feature conditioning* after *feature extraction* and before *classificaion* block. Feature Conditioning is a block which is employed to find a better representation of the input data. In other words, its main goal is to result in a better classification. To achieve this goal, based on the given inputs, we may increase or decrease dimensionality.

In cases where we want to eliminate the repeated, irrelevant, with no information features, clean dataset, remove noise, and making dataset linear, we usually decrease data's dimension.

However, in cases where we want to unfold a complex manifold, and untie intertwined features, to get a simpler classification problem, increasing data's dimension is a wise strategy.

As it is depicted in the below picture, data's are not seprable in one dimesional representation. However, if we increase its dimension to two, they can be classified in a more simple, accurate manner.

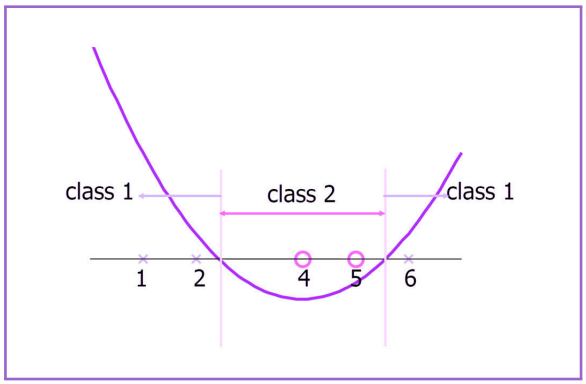

FIGURE 6.1: Increasing Dimension and Better Classification

6.4 Part 4

Suppose we have a binary classification problem where we wish to separate balls from diamonds. This problem is depicted in the below figure.

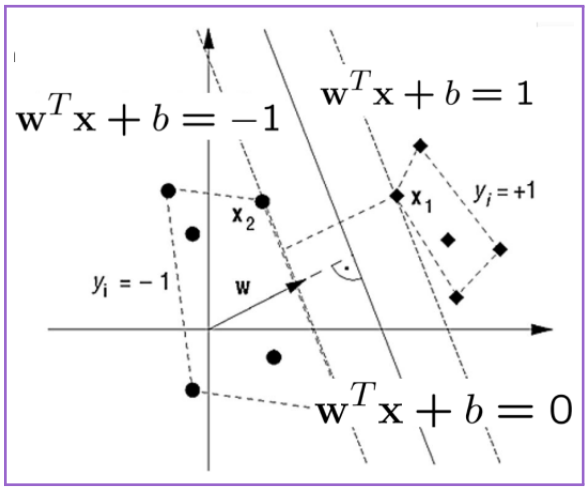

FIGURE 6.2: Diamond and Balls Separation Problem

The optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes, and intersects it half way between the two classes.

6.5 Part 5

We need to solve the below problem:

$$\begin{aligned} &min\ \frac{1}{2}||\mathbf{w}||^2 \\ &s.t.\ 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0 \qquad for\ i = 1, ..., n \end{aligned}$$

The lagrangian is:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

Now to solve this problem, we use partial derivations:

$$\nabla_{\mathbf{w}}\mathcal{L} = 0 \rightarrow \mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = 0 \rightarrow \boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

$$\frac{\partial}{\partial b}\mathcal{L} = 0 \rightarrow \boxed{\sum_{i=1}^{n} \alpha_i y_i = 0}$$

Subistituting above equations in lagrangian, we get:

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i [1 - y_i(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b)] \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i\end{aligned}$$

Note that we have $\sum_{i=1}^{n} \alpha_i y_i = 0$ and that the above result is only a function of α_i. Therefore, we have:

$$max\ W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$s.t.\ \alpha_i \geq 0\ and\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

6.6 Part 6

No, it does not alter soft margins and the only effect is on the margine largrangian coefficients.