CIS5200: Machine Learning Spring 2025

Homework 3

Release Date: March 18, 2025 *Due Date: April 1, 2025*

Name: Haoze Wu
PennKey: haozewu
Collaborators: None

Problem 1: Kernels

1.1 Given that we already have two valid kernels:

$$\begin{aligned} k_1(x,z) &= \phi_1(x)^\top \phi_1(z) \\ k_2(x,z) &= \phi_2(x)^\top \phi_2(z) \end{aligned} \tag{1}$$

and the multiplication of them

$$k(x,z) = k_1(x,z) \cdot k_2(x,z) \tag{2}$$

Then, we have:

$$\begin{aligned} k(x,z) &= k_1(x,z) \cdot k_2(x,z) \\ &= \phi_1(x)^\top \phi_1(z) \cdot \phi_2(x)^\top \phi_2(z) \\ &= (\sum_{i=1}^{n} \phi_1(x)_i \phi_1(z)_i) \cdot (\sum_{j=1}^{m} \phi_2(x)_j \phi_2(z)_j) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} \phi_1(x)_i \phi_1(z)_i \phi_2(x)_j \phi_2(z)_j \\ &= [\phi_1(x)_1\phi_2(x)_1, \phi_1(x)_1\phi_2(x)_2, \cdots, \phi_1(x)_1\phi_2(x)_m, \phi_1(x)_2\phi_2(x)_1, \cdots, \phi_1(x)_n\phi_2(x)_m] \\ &\cdot [\phi_1(z)_1\phi_2(z)_1, \phi_1(z)_1\phi_2(z)_2, \cdots, \phi_1(z)_1\phi_2(z)_m, \phi_1(z)_2\phi_2(z)_1, \cdots, \phi_1(z)_n\phi_2(z)_m] \\ &= \phi(x)^\top \phi(z) \end{aligned} \tag{3}$$

, where $\phi(x) = [\phi_1(x)_1\phi_2(x)_1, \phi_1(x)_1\phi_2(x)_2, \cdots, \phi_1(x)_1\phi_2(x)_m, \phi_1(x)_2\phi_2(x)_1, \cdots, \phi_1(x)_n\phi_2(x)_m]$. Therefore, $k(x,z)$ is a valid kernel. Thus, the multiplication of two valid kernels is also a valid kernel.

1.2 For the given expression $k(x,z) \sum_{r=1}^{d} \alpha_r (k_1(x,z))^r$, we start with the term $(k_1(x,z))^r$. When $r = 1$, we have:

$$(k_1(x,z))^1 = k_1(x,z) \tag{4}$$

, and since k_1 is a valid kernel, $(k_1(x,z))^1$ is also a valid kernel. Then, for $r = r$, we have:

$$(k_1(x,z))^2 = k_1(x,z) \cdot (k_1(x,z))^{2-1} \tag{5}$$

, and since k_1 is a valid kernel and the statement from question 1.1 that $k(x,z) = k_1(x,z) \cdot k_2(x,z)$ is a valid kernel if k_1 and k_2 are valid kernels, then $(k_1(x,z))^2$ is also a valid kernel.

Then, by induction, we can prove that for any $r \geq 2$, $(k_1(x,z))^r = k_1(x,z) \cdot (k_1(x,z))^{r-1}$ is a valid kernel.

Then, consider the full expression of $k(x,z) = \sum_{r=1}^{d} \alpha_r (k_1(x,z))^r$:

$$\begin{aligned} k(x,z) &= \sum_{r=1}^{d} \alpha_r (k_1(x,z))^r \\ &= \alpha_1 (k_1(x,z))^1 + \alpha_2 (k_1(x,z))^2 + \cdots + \alpha_d (k_1(x,z))^d \end{aligned} \tag{6}$$

, and since $(k_1(x,z))^r$ is a valid kernel for any $r \geq 1$, and α_r is a scalar, the sum of valid kernels is also a valid kernel. Thus, the full expression $k(x,z) = \sum_{r=1}^{d} \alpha_r (k_1(x,z))^r$ is a valid kernel.

Problem 2: SVM

2.1 Given the hard-margin SVM optimization problem is:

$$\begin{aligned} \min_{w} \quad & w^\top w \\ \text{s.t.} \quad & y_i w^\top x_i \geq 1, \forall i \end{aligned} \tag{7}$$

, we then try to kernalize it with the given kernel $k(x,z)$:

$$k(x,z) = \begin{cases} 1 & \text{if } x = z \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

First, using the Representer Theorem, we write the weight w as a linear combination of the training data:

$$\begin{aligned} \min_{\alpha} \quad & (\sum_{i=1}^{m} \alpha_i x_i)^\top (\sum_{j=1}^{m} \alpha_j x_j) \\ \text{s.t.} \quad & y_i (\sum_{j=1}^{m} \alpha_j x_j)^\top x_i \geq 1, \forall i \end{aligned} \tag{9}$$

, which can be simplified as:

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j x_i^\top x_j \\ \text{s.t.} \quad & y_i \sum_{j=1}^{m} \alpha_j x_j^\top x_i \geq 1, \forall i \end{aligned} \tag{10}$$

Then, we can apply the kenerl trick by replacing the inner product $x_i^\top x_j$ with the kernel $k(x_i, x_j)$:

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j k(x_i, x_j) \\ \text{s.t.} \quad & y_i \sum_{j=1}^{m} \alpha_j k(x_j, x_i) \geq 1, \forall i \end{aligned} \tag{11}$$

Notice that the kernel function $k(x, z)$ is actually an indicator function that returns 1 iff $x = z$ and 0 otherwise. Thus, we may rewrite the objective function and the restriction as:

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j \mathbb{1}[x_i = x_j] \\ \text{s.t.} \quad & y_i \sum_{j=1}^{m} \alpha_j \mathbb{1}[x_i = x_j] \geq 1, \forall i \end{aligned} \tag{12}$$

Recall that the classifier is trained on a dataset such that all x_1 are distinct, i.e., $x_i \neq x_j$ for all $i \neq j$. We then have:

$$\mathbb{1}[x_i = x_j] = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Thus, the optimization problem can be further simplified as:

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i^2 \\ \text{s.t.} \quad & y_i \alpha_i \geq 1, \forall i \end{aligned} \tag{14}$$

2.2 To find the optimal solution to the optimization problem of the kernalized hard-margin SVM, we first notice the restriction that

$$y_i \alpha_i \geq 1, \forall i \tag{15}$$

and the objective function is to minimize $\sum_{i=1}^{m} \alpha_i^2$. Thus, the optimal solution is to set $\alpha_i = \frac{1}{y_i}$ for all i. Then, the optimal solution to the optimization problem is $\alpha_i^\star = \frac{1}{y_i}$ for all i. Considering the fact that $y_i \in \{-1, 1\}$, the optimal solution is simply $\alpha_i^\star = y_i$ for all i.

2.3 Using the optimal $\alpha^\star$ learnt, we can write the classifier as:

$$\begin{aligned} f(x) &= \text{sign}\left(\sum_{i=1}^{m} \alpha_i^\star k(x_i, x)\right) \\ &= \text{sign}\left(\sum_{i=1}^{m} y_i k(x_i, x)\right) \\ &= \text{sign}\left(\sum_{i=1}^{m} y_i \mathbb{1}[x = x_i]\right) \end{aligned} \tag{16}$$

2.4 For the loss of the classifier $f(x)$, using the hinge loss function, we have:

$$\begin{aligned} \ell(f(x_j), y_j) &= \max(0, 1 - y_j f(x_j)) \\ &= \max(0, 1 - y_j \text{sign}\left(\sum_{i=1}^{m} y_i \mathbb{1}[x_j = x_i]\right)) \\ &= \max(0, 1 - y_j y_j) \\ &= 0 \end{aligned} \tag{17}$$

2.5 For any unseen data point x, the classifier $f(x)$ is:

$$f(x) = \text{sign}\left(\sum_{i=1}^{m} y_i \mathbb{1}[x = x_i]\right) \tag{18}$$

, which would simply output the label of the training data point x_i which is exactly the same as x, and 1 if there is no such training data point.

It means there is almot no generalization to unseen data points, and the classifier is simply memorizing the training data points by the optimal $\alpha^\star$ learnt.

Problem 3: Decision Trees

3.1 For a 1-leaf decision tree, the best result it can achieve is to predict the majority class, i.e., outputing -1 here regardless of the input. When doing so, the error rate is 0.25.

3.2 For a 2-leaf decision tree, we can split the data into two groupd once based on any of the features.

If we split the data based on x_1, the best we could reach is predict -1 when $x_1 = 0$ and 1 when $x_1 = 1$. Then, the error rate is 0.25.

If we split the data based on x_2, the best we could reach is predict -1 when $x_2 = 1$ and 1 when $x_2 = 0$. Then, the error rate is 0.25.

If we split the data based on x_3, the best we could reach is predict -1 when $x_3 = 0$ or $x_3 = 1$. Then, the error rate is still 0.25.

This 2-leaf decision tree performs no better than the 1-leaf decision tree, and the error rate is still 0.25. The reason why the 2-leaf decision tree performs no better than the 1-leaf decision tree is that the data is not linearly separable, and the 2-leaf decision tree can only yield a linear decision boundary, which is not able to separate the data perfectly.

3.3 The entropy is given as:

$$H[D_{<s}] = -p_{<s} \log p_{<s} - (1 - p_{<s}) \log(1 - p_{<s}) \tag{19}$$

For the 1-leaf decision tree, the fraction of the data points for label -1 is $p_{<s} = \frac{3}{4}$, thus the entropy is:

$$H = -\frac{3}{4} \log \frac{3}{4} - \frac{1}{4} \log \frac{1}{4} \approx 0.8113 \tag{20}$$

3.4 For all the possible splits for a 2-leaf decision tree, we have: If we split the data based on x_1, then entropy is:

$$\begin{aligned} H[D_{<s}]_1 &= 0 - 1 \log 1 = 0 \\ H[D_{>s}]_1 &= -\frac{1}{2} \log \frac{1}{2} - \frac{1}{2} \log \frac{1}{2} = 1 \\ H_1 &= \frac{4}{8} \cdot 0 + \frac{4}{8} \cdot 1 = 0.5 \end{aligned} \tag{21}$$

If we split the data based on x_2, then entropy is:

$$\begin{aligned} H[D_{<s}]_2 &= -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1 \\ H[D_{>s}]_2 &= 0 - 1\log 1 = 0 \\ H_2 &= \frac{4}{8}\cdot 1 + \frac{4}{8}\cdot 0 = 0.5 \end{aligned} \tag{22}$$

If we split the data based on x_3, then entropy is:

$$\begin{aligned} H[D_{<s}]_3 &= -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} = 0.8113 \\ H[D_{>s}]_3 &= -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} = 0.8113 \\ H_3 &= \frac{4}{8}\cdot 0.8113 + \frac{4}{8}\cdot 0.8113 = 0.8113 \end{aligned} \tag{23}$$

Spliting over x_1 or x_2 yields the same lower entropy, which is 0.5, and spliting over x_3 yields an entropy of 0.8113.

3.5 Comparing the entropy of the 1-leaf decision tree and the 2-leaf decision tree, noticing that the both the entropys of spliting over x_1 and x_2 decrease from 0.8113 to 0.5, while the entropy of spliting over x_3 remains the same. That means spliting over x_1 or x_2 is the best choice for the 2-leaf decision tree, and the best split is to split the data based on x_1 or x_2. Noticing that the labels were generated by utlizing x_1 and x_2 and has noting to do with x_3, the best split is to split the data based on x_1 or x_2, which is reflected by the entropy.