CIS5200: Machine Learning Spring 2025

Homework 3

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

- HW3 will count for 10% of the grade. This grade will be split between the written (30 points) and programming (20 points) parts.
- All written homework solutions are required to be formatted using LaTeX. Please use the template here. Do not modify the template. This is a good resource to get yourself more familiar with LaTeX, if you are still not comfortable.
- You will submit your solution for the written part of HW3 as a single PDF file via Gradescope. The deadline is **11:59 PM ET**. Contact TAs on Ed if you face any issues uploading your homeworks.
- Collaboration is permitted and encouraged for this homework, though each student must understand, write, and hand in their own submission. In particular, it is acceptable for students to discuss problems with each other; it is not acceptable for students to look at another student's written Solutions when writing their own. It is also not acceptable to publicly post your (partial) solution on Ed, but you are encouraged to ask public questions on Ed. If you choose to collaborate, you must indicate on each homework with whom you collaborated.
- **Optional Question**: We have added one optional question in this homework for those interested in trying some more challenging problems. These questions are optional and come with no extra credit.

Please refer to the notes and slides posted on the website if you need to recall the material discussed in the lectures.

1 Written Questions (30 points)

Problem 1: Kernels (10 points)

Consider two kernel functions k_1 and k_2 and their feature maps ϕ_1 and ϕ_2 respectively. In particular,

$$k_1(x, z) = \phi_1(x)^\top \phi_1(z) \text{ and } k_2(x, z) = \phi_2(x)^\top \phi_2(z).$$

In the midterm we worked out two algebraic operations that preserve validity of a kernel:

1. $k(x, z) = \alpha \cdot k_1(x, z)$ for any constant $\alpha \geq 0$
2. $k(x, z) = k_1(x, z) + k_2(x, z)$.

In this problem, we will work out a few more.

1.1 (5 points) Show that $k(x, z) = k_1(x, z) \cdot k_2(x, z)$ is a valid kernel. Specifically, design a feature map ϕ using ϕ_1, ϕ_2 such that $k(x, z) = \phi(x)^\top \phi(z)$.

1.2 (5 points) Show that $k(x, z) = \sum_{r=1}^{d} \alpha_r (k_1(x, z))^r$ is a valid kernel assuming $\alpha_r \geq 0$ for all $r \in \{1, \ldots, d\}$.

Hint: Use the algebraic operations from the midterm and 1.1.

Optional (zero credit) Show that $k(x, z) = \min(x, z)$ is a valid kernel over input space $\mathcal{X} = [0, 1]$.

Problem 2: SVM (12 points)

Consider running hard-margin kernel-SVM with the following kernel:

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

Assume that the training dataset $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is such that all x_i are distinct (that is, for all $i \neq j$, $x_i \neq x_j$) and $y_i \in \{-1, 1\}$.

2.1 (2 points) Recall the hard-margin SVM objective:

$$\min_{\mathbf{w}} \mathbf{w}^\top \mathbf{w} \text{ s.t. } y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$$

Write the kernalized version of this objective. You can assume that the optimal solution is in the span of the training points. Make sure to substitute the value of k to simplify the objective.

2.2 (5 points) Using the kernalized version, find the optimizer α_* of the optimization problem for the given dataset.

2.3 (2 points) Using α_* from above show that the classifier learned is:

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

Here $\mathbb{1}[x = x_i]$ is an indicator function that returns 1 if $x = x_i$ and 0 otherwise.

Recall: The sign function is given by $\text{sign}(x) = \begin{cases} +1 & \textit{if } x \geq 0 \\ -1 & \textit{otherwise} \end{cases}$.

2.4 (1 point) What is the error of the classifier on the training dataset?

2.5 (2 points) What is the prediction on any x not seen in the training dataset? What does this say about the generalization error of this classifier?

Problem 3: Decision Trees (8 points)

Recall that when building a decision tree for classification problems, we used the notion of node purity instead of classification error when deciding how to grow the decision tree. Consider 3-dimensional binary data $x \in \mathbb{R}^3$ where the label $y \in \{-1, 1\}$ is generated from the first two features as $f(x) = x_1 \wedge \neg x_2$. In other words, the entire dataset of all possible examples can be written as:

x_1	x_2	x_3	y
0	0	0	-1
0	0	1	-1
0	1	0	-1
0	1	1	-1
1	0	0	1
1	0	1	1
1	1	0	-1
1	1	1	-1

3.1 (1 point) What is the lowest possible classification error that a 1-leaf decision tree (i.e. only the root node predicting a constant value) can achieve?

3.2 (2 points) Consider all three possible splits for growing to a 2-leaf decision tree. Does there exist a split that achieves lower classification error than a 1-leaf decision tree? If so, what is it? If not, why is that?

3.3 (1 points) What is the entropy of the 1-leaf decision tree?

3.4 (3 points) Consider all three possible splits (based on x_1, x_2, or x_3) for growing to a 2-leaf decision tree. Find the split that results in the lowest possible entropy, and calculate the corresponding entropy of the split.

3.5 (1 points) Looking at the results from the previous parts, explain why entropy serves as a more informative measure of split quality than classification error in this particular.

2 Programming Questions (20 points)

Use the link here to access the Google Colaboratory (Colab) file for this homework. Be sure to make a copy by going to "File", and "Save a copy in Drive". As with the previous homeworks, this assignment uses the PennGrader system for students to receive immediate feedback. As noted on the notebook, please be sure to change the student ID from the default '99999999' to your 8-digit PennID.

Instructions for how to submit the programming component of HW 3 to Gradescope are included in the Colab notebook. You may find this PyTorch linear algebra reference and this general

PyTorch reference to be helpful in perusing the documentation and finding useful functions for your implementation.