CSCI 145 Problem Set 2

September 3, 2025

Submission Instructions

Please upload *your* work by **11:59pm Monday September 8, 2025.**

- You are encouraged to discuss ideas and work with your classmates. However, you **must acknowledge** your collaborators at the top of each solution on which you collaborated with others and you **must write** your solutions independently.
- Your solutions to theory questions must be written legibly, or typeset in LaTeX or markdown. If you would like to use LaTeX, you can import the source of this document (available from the course webpage) to Overleaf.
- I recommend that you write your solutions to coding questions in a Jupyter notebook using Google Colab.
- You should submit your solutions as a **single PDF** via the assignment on Gradescope.

Grading: The point of the problem set is for *you* to learn. To this end, I hope to disincentivize the use of LLMs by **not** grading your work for correctness. Instead, you will grade your own work by comparing it to my solutions. This self-grade is due the Friday *after* the problem set is due, also on Gradescope.

Problem 1: Single Value Functions

Consider a supervised learning problem with n labels $y^{(1)}, \ldots, y^{(n)} \in \mathbb{R}$. In class, we explored the linear function class that predicted a weighted combination of the input points. In this problem, we'll consider the function class that outputs a single real number $m \in \mathbb{R}$ for all points.

A single number that best fits the data is known as its *central tendency* in statistics. Here, we will derive different central tendencies from an empirical risk minimization perspective.

Part A: ℓ_2-norm

Consider the ℓ_2-norm loss function

$$\mathcal{L}(m) = \sum_{i=1}^{n} (y^{(i)} - m)^2.$$

Show the optimal value m^* is the average.

Part B: ℓ_∞-norm

Consider the ℓ_∞-norm loss function

$$\mathcal{L}(m) = \max_{i \in \{1,\ldots,n\}} |y^{(i)} - m|.$$

Derive the value m^* that minimizes this loss.
Hint: Think about the minimization problem directly rather than using derivatives.

Part C: ℓ_1-norm

Consider the ℓ_1-norm loss function

$$\mathcal{L}(m) = \sum_{i=1}^{n} |y^{(i)} - m|.$$

For simplicity, assume that n is odd. Show that the optimal value m^* is the median.
Hint: Try drawing the loss on top of a plot of the points on the number line.

Problem 2: Leave One Out Linear Regression

Generally, we have access to a limited amount of data. In machine learning, there's an inherent tension in how we use this data. On one hand, we want to use as much data as possible when training the model, so that the trained model will be more accurate. On the other hand, we're also interested in evaluating the trained model to see how well it performs. If we evaluate the model on data that it was trained on, the model may perform well but only because it has seen the data before.

One solution is to separate the dataset into a training set and an evaluation set. Most commonly, 80% of the data is used to train the model, while the remaining 20% is reserved for evaluating the model. However, this isn't ideal because we're only using a fraction of our data for training, and only evaluating its performance on a fraction of the data.

In a perfect world, we would use (almost) all the data for both training and evaluation. This would involve training a model on all but one data point, and evaluating how well its prediction matches the true label. Most of the time, this leave-one-out approach is quite expensive because we need to retrain a model from scratch for each of the n points in the dataset.

In this problem, we'll see how we can more efficiently compute the leave-one-out prediction for linear models with some clever linear algebra.

Part A: LOO Weights

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the data matrix where the ith row is the ith input $\mathbf{x}^{(i)} \in \mathbb{R}^d$. Let $\mathbf{y} \in \mathbb{R}^n$ be the target vector where the ith entry is the ith label $y^{(i)} \in \mathbb{R}$.

Recall that the optimal weights when using all n points are:

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Show that the leave-one-out weights when the ith labeled data point is removed are:

$$\mathbf{w}^*_{-i} = \left(\mathbf{X}^\top \mathbf{X} - \mathbf{x}^{(i)} {\mathbf{x}^{(i)}}^\top\right)^{-1} \left(\mathbf{X}^\top \mathbf{y} - \mathbf{x}^{(i)} y^{(i)}\right). \tag{1}$$

Hint: Use the outer product definition of matrix multiplication (twice).

Part B: Sherman-Morrison

Computing the inverse of a $d \times d$ matrix is expensive, taking roughly $O(d^3)$ time. We would like to compute $(\mathbf{X}^\top \mathbf{X})^{-1}$ only once, and then reuse our results for multiple left-out points $i \in \{1, \ldots, n\}$. Luckily, the Sherman-Morrison formula gives us just the tool.

Apply the Sherman-Morrison formula to show that:

$$\mathbf{w}^*_{-i} = \left(\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} + \frac{\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{x}^{(i)} {\mathbf{x}^{(i)}}^\top \left(\mathbf{X}^\top \mathbf{X}\right)^{-1}}{1 - {\mathbf{x}^{(i)}}^\top \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{x}^{(i)}}\right) \left(\mathbf{X}^\top \mathbf{y} - \mathbf{x}^{(i)} y^{(i)}\right). \tag{2}$$

Part C: LOO Prediction

Define the *leverage* of the ith point as $\ell_i = \mathbf{x}^{(i)\top} \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{x}^{(i)}$. Now show that the leave-one-out prediction for the ith point is:

$$\hat{y}_{-i}^{(i)} = \mathbf{x}^{(i)\top} \mathbf{w}_{-i}^* = \frac{\hat{y}^{(i)} - \ell_i y^{(i)}}{1 - \ell_i} \tag{3}$$

where $\hat{y}^{(i)} = \mathbf{x}^{(i)\top} \mathbf{w}^*$ is the prediction when all n points are used in training.

Part D: Time Complexity

After the initial $O(d^3)$ cost of computing $\left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$, what is the time complexity of computing all n leave-one-out predictions? What would the cost have been if we naively retrained the model for each prediction?

Part E: In Practice

Load a labeled dataset of your choice (e.g., the first 100 points in the California housing dataset available on `scikitlearn`.) Compute the regular predictions, and *efficiently* compute the leave-one-out predictions. Plot the leave-one-out predictions and the regular predictions against the true labels, with an identity line to mark the *ideal* performance. What do you notice?