Homework #0

Due: January 31, 2025 at 11:59 PM

Welcome to CS1810! The purpose of this assignment is to help assess your readiness for this course. It will be graded for completeness and effort. **Areas of this assignment that are difficult are an indication of areas in which *you* need to self-study. If you find you are struggling with many of these questions, it might be prudent to postpone taking this course until after you have mastered the necessary prerequisites.** ***During the term, the staff will be prioritizing support for new material taught in CS1810 over teaching prerequisites.*** **If you are unsure about your readiness, please contact the head TFs for advice.**

1. Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page.

2. Please submit the **writeup PDF to the Gradescope assignment 'HW0'**. Remember to assign pages for each question.

3. Please submit your **LaTeX file and code files (i.e., anything ending in `.py`, `.ipynb`, or `.tex`) to the Gradescope assignment 'HW0 - Supplemental'**.

Problem 1 (Modeling Linear Trends - Linear Algebra Review)

In this class, we will be exploring the question of "how do we model the trend in a dataset" under different guises. In this problem, we will explore the algebra of modeling a linear trend in data. We call the process of finding a model that capture the trend in the data, "fitting the model."

Learning Goals: In this problem, you will practice translating machine learning goals ("modeling trends in data") into mathematical formalism using linear algebra. You will explore how the right mathematical formalization can help us express our modeling ideas unambiguously and provide ways for us to analyze different pathways to meeting our machine learning goals.

Let's consider a dataset consisting of two points $\mathcal{D} = \{(x_1, y_1), (x_2, y_2)\}$, where x_n, y_n are scalars for $n = 1, 2$. Recall that the equation of a line in 2-dimensions can be written: $y = w_0 + w_1 x$.

1. Write a system of linear equations determining the coefficients w_0, w_1 of the line passing through the points in our dataset $\mathcal{D}$ and analytically solve for w_0, w_1 by solving this system of linear equations (i.e., using substitution). Please show your work.

2. Write the above system of linear equations in matrix notation, so that you have a matrix equation of the form $\mathbf{y} = \mathbf{X}\mathbf{w}$, where $\mathbf{y}, \mathbf{w} \in \mathbb{R}^2$ and $\mathbf{X} \in \mathbb{R}^{2\times 2}$. For full credit, it suffices to write out what $\mathbf{X}$, $\mathbf{y}$, and $\mathbf{w}$ should look like in terms of x_1, x_2, y_1, y_2, w_0, w_1, and any other necessary constants. Please show your reasoning and supporting intermediate steps.

3. Using properties of matrices, characterize exactly when an unique solution for $\mathbf{w} = (w_0\ w_1)^T$ exists. In other words, what must be true about your dataset in order for there to be a unique solution for $\mathbf{w}$? When the solution for $\mathbf{w}$ exists (and is unique), write out, as a matrix expression, its analytical form (i.e., write $\mathbf{w}$ in terms of $\mathbf{X}$ and $\mathbf{y}$).

   Hint: What special property must our $\mathbf{X}$ matrix possess? What must be true about our data points in $\mathcal{D}$ for this special property to hold?

4. Compute $\mathbf{w}$ by hand via your matrix expression in (3) and compare it with your solution in (1). Do your final answers match? What is one advantage for phrasing the problem of fitting the model in terms of matrix notation?

5. In real-life, we often work with datasets that consist of hundreds, if not millions, of points. In such cases, does our analytical expression for $\mathbf{w}$ that we derived in (3) apply immediately to the case when $\mathcal{D}$ consists of more than two points? Why or why not?

Solution

1. We can being by creating two systems of linear equations since the two points lie on the same line:

$$y_1 = w_0 + w_1 x_1$$
$$y_2 = w_0 + w_1 x_2$$

We can use substitution:

$$w_0 = y_1 - w_1 x_1$$

and substitute this into the second expression:

$$y_2 = y_1 - w_1 x_1 + w_1 x_2$$
$$y_2 = y_1 - w_1(x_1 - x_2)$$
$$w_1 = \frac{y_1 - y_2}{x_1 - x_2}$$

We can find w_0 as well by the following

$$w_0 = y_1 - \frac{y_1 - y_2}{x_1 - x_2} x_1 = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2}$$

2. We can write this in the form of a matrix equation where we ultimately have two vectors

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_1 \\ w_0 + w_1 x_2 \end{bmatrix}$$

The right side can be observed to be the result of a multiplication of a 2x2 matrix and a 2x1 vector, resulting in a 2x1 vector. Thus, we get:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

We can verify that this is correct since

$$\mathbf{Xw} = \begin{bmatrix} w_0 + w_1 x_1 \\ w_0 + w_1 x_2 \end{bmatrix}$$

3. The matrix $\boldsymbol{X}$ must be full-rank and invertible for there to be one unique solution for $\boldsymbol{w}$. This means that the data points cannot lie on a vertical line (since then the columns of $\boldsymbol{X}$ would be linearly dependent). In other words, $x_1 \neq x_2$. Under this condition, we can write the solution for $\boldsymbol{w}$ as

$$\boldsymbol{w} = \boldsymbol{X}^{-1}\boldsymbol{y}$$

4. We can find the inverse of $\boldsymbol{X}$ as it is a 2×2 matrix:

$$\boldsymbol{X}^{-1} = \frac{1}{x_2 - x_1} \begin{bmatrix} x_2 & -x_1 \\ -1 & 1 \end{bmatrix}$$

   Thus, we can calculate

$$\begin{aligned} \boldsymbol{w} &= \frac{1}{x_2 - x_1} \begin{bmatrix} x_2 & -x_1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \\ &= \frac{1}{x_2 - x_1} \begin{bmatrix} x_2 y_1 - x_1 y_2 \\ y_2 - y_1 \end{bmatrix} \end{aligned}$$

   Therefore, we see that

$$w_1 = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_1 - y_2}{x_1 - x_2}$$

$$w_0 = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2}$$

   We can see that the final answers for $\boldsymbol{w}$ match. One advantage for phrasing the problem of fitting the model into matrix notation can be that it could make it easier to scale the problem to higher dimensions. Instead of creating large systems of linear equations and solving them one by one, we can use the matrix notation to write the problem in a simple/clean manner and solve it in the same way.

5. Our expression from (3) would not work when the data set consists of more than two points since for $\boldsymbol{X}$ to be invertible, it must be a square matrix. Having more than two points would result in $\boldsymbol{X}^{-1}$ not existing.

Problem 2 (Optimizing Objectives - Calculus Review)

In this class, we will write real-life goals we want our model to achieve into a mathematical expression and then find the optimal settings of the model that achieves these goals. The formal framework we will employ is that of mathematical optimization. Although the mathematics of optimization can be quite complex and deep, we have all encountered basic optimization problems in our first calculus class!

Learning Goals: In this problem, we will explore how to formalize real-life goals as mathematical optimization problems. We will also investigate under what conditions these optimization problems have solutions.

In her most recent work-from-home shopping spree, Nari decided to buy several house plants. *Her goal is to make them to grow as tall as possible.* After perusing the internet, Nari learns that the height y in mm of her Weeping Fig plant can be directly modeled as a function of the oz of water x she gives it each week:

$$y = -3x^2 + 72x + 70.$$

1. First, plot the height function. What does the plot tell you about the existence and uniqueness of a maximum plant height? Next, support your claim solely based on the form of the function.
2. Use calculus to find how many ounces of water per week Nari should give to her plant in order to maximize its height. With this much water, how tall will her plant grow?

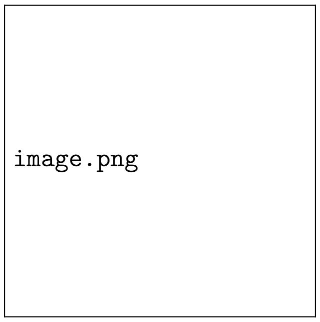
Now suppose that Nari want to optimize both the amount of water x_1 (in oz) *and* the amount of direct sunlight x_2 (in hours) to provide for her plants. After extensive research, she decided that the height y (in mm) of her plants can be modeled as a two variable function:

$$y = f(x_1, x_2) = \exp\left(-(x_1 - 2)^2 - (x_2 - 1)^2\right)$$

3. Using `matplotlib`, visualize in 3D the height function as a function of x_1 and x_2 using the `plot_surface` utility for $(x_1, x_2) \in (0, 6) \times (0, 6)$. Then, determine the values of x_1 and x_2 that maximize plant height. Do these yield a global maximum?

   Hint: You don't need to take any derivatives here; reasoning about the form of $f(x_1, x_2)$ suffices.

Solution



1. From the plot of the height function, we can see that there exists a unique maximum plant height (at the top of the parabola) since every other point has a smaller value than it. Since the function is a downward facing parabola, we know from the form of the function that the vertex point will have the greatest height for all possible inputs x.

2. Since the equation $y = -3x^2 + 72x + 70$ is a negative-facing parabola, the height y will be maximized when $y' = 0$. Thus,

$$y' = -6x + 72$$

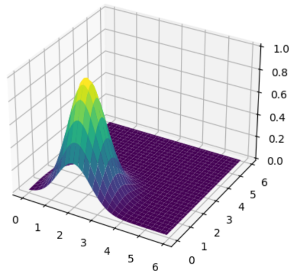
Setting y' to 0, we can solve for the amount of ounces of water per week:

$$0 = -6x + 72$$
$$x = 12$$

The height would be

$$y = -3(12)^2 + 72(12) + 70 = 502$$

Nari should give her plants 12 ounces of water per week to maximize its height. With this much water, her plant will grow 502 mm tall.



3. The values of x_1 and x_2 that maximize plant height are $(x_1, x_2) = (2, 1)$. From the form of the plotted function, these yield a global maximum. There is a peak in the function located at $(2, 1)$ and all other height values are smaller.

Problem 3 (Reasoning about Randomness - Probability and Statistics Review)
In this class, one of our main focuses is to model the unexpected variations in real-life phenomena using the formalism of random variables. In this problem, we will use random variables to model how much time it takes an USPS package processing system to process packages that arrive in a day.

Learning Goals: In this problem, you will analyze random variables and their distributions both analytically and computationally. You will also practice drawing connections between said analytical and computational conclusions.

Consider the following model for each package that arrives at the US Postal Service (USPS):

- Every package has a random size S (measured in in^3) and weight W (measured in pounds), with joint distribution

$$(S, W)^T \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \text{ with } \boldsymbol{\mu} = \begin{bmatrix} 120 \\ 4 \end{bmatrix} \text{ and } \boldsymbol{\Sigma} = \begin{bmatrix} 1.5 & 1 \\ 1 & 1.5 \end{bmatrix}.$$

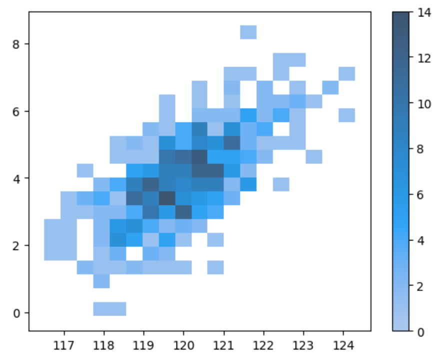
- The size and weight of each package is independent of those of all the other packages.
- Processing time T (in seconds) for each package is given by $T = 60 + 0.6W + 0.2S + \epsilon$, where ϵ is an independent random noise variable with Gaussian distribution $\epsilon \sim \mathcal{N}(0, 5)$.

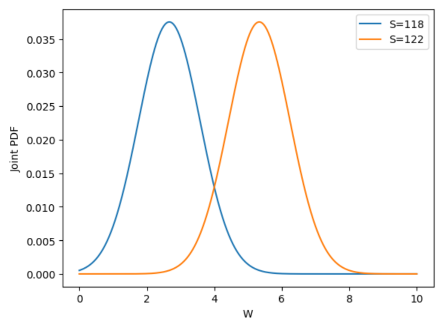
1. Perform the following tasks:
   (a) Give one reason for why the Gaussian distribution may not be appropriate for modeling the size and weight of packages.
   (b) Empirically estimate the most likely combination of size and weight of a package by sampling 500 times from the joint distribution of S and W and generating a bivariate histogram of your S and W samples. A visual inspection is sufficient – you do not need to be incredibly precise. How close are these empirical values to the theoretical expected size and expected weight of a package, according to the given Bivariate Gaussian distribution?
   Hint: For this part, you may find the `multivariate_normal` module from `scipy.stats` especially helpful. You may also find the `seaborn.histplot` function quite helpful.
2. For 1001 evenly-spaced values of W between 0 and 10, plot W versus the joint Bivariate Gaussian PDF $p(W, S)$ with S fixed at $S = 118$. Repeat this procedure for S fixed at $S = 122$. Comparing these two PDF plots, what can you say about the correlation of random variables S and W?
3. Because T is a linear combination of random variables, it itself is a random variable. Using properties of expectations and variance, please compute $\mathbb{E}(T)$ and $\text{Var}(T)$ analytically.
4. Define N to be the number of packages that arrive today, and suppose that packages that weigh less than 4 pounds are considered fragile. Conditional on $N = n$, what is the name and PMF of the distribution of the number of fragile packages that arrive today?
5. Now suppose that $N = \sum_{h=1}^{24} P_h$, where the P_h are independent and identically distributed as $\text{Pois}(\lambda = 3)$. Then define $T^* = \sum_{i=1}^{N} T_i$ as the *total* amount of time it takes to process *all* these packages, where T_i follows the distribution of T that we previously defined for each package.
   (a) Write a function to simulate draws from the distribution of T^*.
   (b) Using your function, empirically estimate the mean and standard deviation of T^* by generating 1000 samples from the distribution of T^*.

Solution

1. (a) A Guassian distribution might not be appropriate since size and weight cannot be negative values (a Guassian distribution can result in negative values).

   (b) Empirically, the most likely combination seems to be around a size of 120 and weight of 4 from the plot. These align with the expected theoretical size and weight of the package from the distribution.





2. Comparing these two PDF plots, it seems that there is a slight positive correlation between S and W. As values for S are set at a lower fixed number, we see the PDF curve of W shift left and vice versa.

3. By linearity, we see that

$$\begin{aligned} E(T) &= E(60 + 0.6W + 0.2S + \epsilon) \\ &= 60 + 0.6E(W) + 0.2E(S) + E(\epsilon) \\ &= 60 + 2.4 + 24 + 0 \\ &= 86.4 \end{aligned}$$

   We can find variance:

$$\begin{aligned} \text{Var}(T) &= \text{Var}(60 + 0.6W + 0.2S + \epsilon) \\ &= \text{Var}(0.6W + 0.2S) + \text{Var}(\epsilon) \\ &= \text{Var}(0.6W) + \text{Var}(0.2S) + 2\text{Cov}(0.6W, 0.2S) + 5 \\ &= 0.36(1.5) + 0.04(1.5) + 2(0.12) + 5 \\ &= 5.84 \end{aligned}$$

4. Since we know that $W \sim \mathcal{N}(4, 1.5)$, we know that the probability for any arbitrarily selected package being fragile is 0.5 since the mean weight is 4. Let Y be the number of packages that weight less than 4 pounds. Conditional on $N = n$, we see that

$$Y|N = n \sim \text{Bin}(n, 0.5)$$

In other words, conditional on the number of packages that arrive, the number of fragile packages follows a binomial distribution with the PMF

$$P(Y = k|N = n) = \binom{n}{k}(0.5)^k(0.5)^{n-k}$$

5. Both parts in code file. An empirical mean and standard deviation is mean of 6249.468, std of 731.672.

Problem 4 (Implementing a Linear Regression - Coding Review)

In this class, we will bridge theory and practice through implementing the methods that we cover from scratch. In this problem, we follow up on Problem 1 through exploring a more practical version of linear regression (fitting a linear model). Namely, we use ordinary least squares (OLS) to estimate a *line of best fit* rather than a perfect fit to our data. Note that the focus of this problem is on coding rather than math—we will cover the relevant theory in much more depth during the course.

Learning Goals: In this problem, you will gain experience with the procedure of modeling real-world data. You will also get useful practice with debugging and writing clean, efficient code in Python.

Steve is a fictional CS 1810 TF giving a live demo of how to fit a linear regression. However, he quickly realizes that coding live in front of an audience isn't for the faint of heart. As a star student, you will help him with his code. Just like Problem 1, the demo uses a 2-D dataset, so that the goal is to model the relationship between the x and y coordinates. The data are stored in the `data` variable, with the first column corresponding to the x-coordinate and the second corresponding to the y-coordinate.

1. Using the provided data, Steve has defined variables `y` and `x` corresponding to the respective coordinates. What is wrong with his current code? Fix the code and then plot the data. Does there appear to be a linear trend?

2. Steve then defines a new variable `X`, which is meant to resemble $\boldsymbol{X}$ from Problem 1. Specifically, `X` is supposed to have one column of all ones (recall that this allows us to fit an intercept) and one column which is just `x`, the x-coordinates. However, he realizes that his code yields the wrong shape for `X`. What's going on here? Fix the code and then report what `y.shape` and `X.shape` are. Why is there no second coordinate in the output for `y.shape`?

   Hint: check the documentation for `np.hstack`.

3. Steve takes a much-needed break from coding to give the following high level overview of linear regression: given a target (response) $\boldsymbol{y}$ and features (predictors) $\boldsymbol{X}$, the goal of linear regression is to find weights $\boldsymbol{w}$ such that $\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w}$ closely approximates the true data $\boldsymbol{y}$. In OLS, we estimate $\boldsymbol{w}$ to be

   $$\hat{\boldsymbol{w}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$$

   Steve skips over the derivation of the result but assures you that you will learn it later in the course. What should the shape of $\hat{\boldsymbol{w}}$ be in Steve's demo?

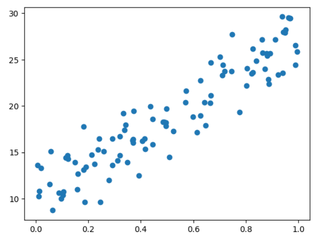
4. Having walked through the idea of linear regression, Steve then attempts to implement a `LinearRegression` class. He correctly identifies that we need 3 components: a constructor, a `fit` function for computing $\hat{\boldsymbol{w}}$ from the data, and a `predict` function for computing the estimate $\boldsymbol{X}\hat{\boldsymbol{w}}$. However, he realizes that there is something wrong (meaning logic or syntax) with at least one of these components. Please point out the issues, fix them, and include the plot of the fitted line.

5. As his final act for the day, Steve introduces the Mean Squared Error (MSE) loss function:

   $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

   This captures how well the outputs of our model, $\hat{\boldsymbol{y}}$, fit the actual data $\boldsymbol{y}$. Steve manages to correctly implement an MSE computation! However, you realize that he can vectorize his code to make it faster, meaning that he can directly compute the MSE from NumPy arrays without using any for loops. Implement the vectorized MSE and write down the corresponding mathematical expression, which should directly be in terms of the vectors $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ rather than their components.

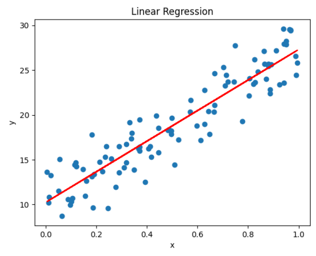
Solution



1. Steve was extracting rows into x and y instead of columns. The plotted data looks roughly linear.

2. The arrays were being horizontally appended to each other with hstack. To fix this, each element in the intercept and $\boldsymbol{x}$ array when used as a paramter for np.hstack needs to be reshaped to be its own vector so that the inputs become 2D vectors. This will result in y.shape being $(100,)$ and X.shape being $(100, 2)$ or 100×2. There is no second coordinate in the output for y.shape since $\boldsymbol{y}$ only has one dimension.

3. The dimensions of X is 100×2, so $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ will result in a 2×2 matrix (since $\boldsymbol{X}^\top$ is 2×100) The result of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top$ will be a 2×100 matrix. We can mathematically treat $\boldsymbol{y}$ as a 100×1 column vector, so our result shape of $\hat{\boldsymbol{w}}$ in the demo will be 2×1.



4. The issues include that $**-1$ does not properly invert a matrix, also we need @ instead of $*$ for matrix multiplication. Also, we do not transpose y and instead transpose X. Instead of returning, we modify self.w to the proper weight. (Actual locations in the code)

5. We can vectorize MSE so that we square the each element of the difference vector between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, then find the mean. We can write this mathematically as

$$\text{MSE} = \frac{1}{n}(\boldsymbol{y} - \hat{\boldsymbol{y}})^\top (\boldsymbol{y} - \hat{\boldsymbol{y}})$$