Homework #0

Due: January 26, 2024 at 11:59 PM

Welcome to CS181! 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. During the term, the staff will be prioritizing support for new material taught in CS181 over teaching prerequisites.**

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) Given our two points and the general equation of a line in 2-dimensions, we arrive at the following system of linear equations:

$$y_1 = w_0 + w_1x_1 \Rightarrow y_1 - w_1x_1 = w_0$$

$$y_2 = w_0 + w_1x_2$$

Substituting our obtained value for w_0 into the second equation, we solve for w_1:

$$y_2 = (y_1 - w_1x_1) + w_1x_2 \Rightarrow y_2 - y_1 = w_1(x_2 - x_1) \Rightarrow w_1 = \frac{y_2 - y_1}{x_2 - x_1}$$

We substitute the above value for w_1 to solve for w_0:

$$y_1 - w_1x_1 = w_0 \Rightarrow w_0 = y_1 - x_1 \cdot \frac{y_2 - y_1}{x_2 - x_1}$$

$$\Rightarrow w_0 = \frac{y_1x_2 - y_1x_1 - x_1y_2 + x_1y_1}{x_2 - x_1} \Rightarrow w_0 = \frac{y_1x_2 - x_1y_2}{x_2 - x_1}$$

2) In matrix notation, for $\mathbf{y}, \mathbf{w} \in \mathbb{R}^2$ and $\mathbf{X} \in \mathbb{R}^{2\times 2}$, we have $\mathbf{y} = \mathbf{Xw}$, where $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}$, and $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$. Here, $\mathbf{y}$ is the output vector which contains the y-coordinates of the two points. $\mathbf{X}$ is the feature matrix, which has a first column of 1s to represent the coefficients for w_0 (the y-intercept) and a second column of x_1 and x_2 to be multiplied by the slopes, w_0 and w_1 from $\mathbf{w}$, the weight vector.

3) We claim that above matrix equation of the form $\mathbf{y} = \mathbf{Xw}$ has a unique solution if $\mathbf{X}$ is invertible. In such case, we would have $\mathbf{X}^{-1}\mathbf{y} = \mathbf{w}$. Since the inverse of a matrix is unique, $\mathbf{w}$ is unique.

Now, a matrix is invertible iff its determinant is not equal to 0. Thus, we desire det $\mathbf{X} \neq 0 \Rightarrow x_2 - x_1 \neq 0 \Rightarrow x_1 \neq x_2$.

In such case, the solution for $\mathbf{w}$ exists and is unique. We have

$$\mathbf{w} = \mathbf{X}^{-1}\mathbf{y} \Rightarrow \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \frac{x_2}{x_2 - x_1} & \frac{-x_1}{x_2 - x_1} \\ \frac{-1}{x_2 - x_1} & \frac{1}{x_2 - x_1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

4) Performing the matrix multiplication from **(3)**, we arrive at the following system of linear equations. Observe that the derived results for w_0 and w_1 match exactly the solution obtained in **(1)**.

$$w_0 = \frac{x_2}{x_2 - x_1}y_1 + \frac{-x_1}{x_2 - x_1}y_2 \Rightarrow w_0 = \frac{x_2y_1 - x_1y_2}{x_2 - x_1}$$

$$w_1 = \frac{-1}{x_2 - x_1}y_1 + \frac{1}{x_2 - x_1}y_2 \Rightarrow w_1 = \frac{y_2 - y_1}{x_2 - x_1}$$

One advantage for phrasing the problem of fitting the model in terms of matrix notation is using linear algebra techniques to efficiently compute solutions. Especially when dealing with large datasets or higher-dimensional data, solving systems of linear equations by substitution is time-consuming. Matrices and vectors can be easily manipulated with computers to perform complex operations quickly and accurately. Additionally, using matrix notation makes it easier to understand qne generalize the problem to more variables and observations, enabling a concise representation of the problem and solution.

5) In such cases, our analytical expression for $\mathbf{w}$ that we derived in **(3)** *does not* apply immediately to the case when $\mathbf{D}$ consists of more than two points. From a linear algebra perspective, with more than two data points, the matrix $\mathbf{X}$ would no longer be square (no longer 2×2). Rather, it would be rectangular with dimensions $m \times n$ for integers m and n with $m > n$.

While non-square matrices have psuedoinverses, there does not exist a well-defined inverse – in other words, in such case, $\mathbf{X}$ would no longer be invertible and the system would be overdetermined.

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. Based on the above formula, is Nari's goal achievable: does the plant have a maximum height? Why or why not? Does her goal have a unique solution - i.e. is there one special watering schedule that would acheive the maximum height (if it exists)?

   Hint: plot this function. In your solution, words like "convex" and "concave" may be helpful.

2. Using calculus, find how many oz per week should Nari water her plant in order to maximize its height. With this much water, how tall will her plant grow?

   Hint: solve analytically for the critical points of the height function (i.e., where the derivative of the function is zero). For each critical point, use the second-derivative test to identify if each point is a max or min point, and use arguments about the global structure (e.g., concavity or convexity) of the function to argue whether this is a local or global optimum.

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:

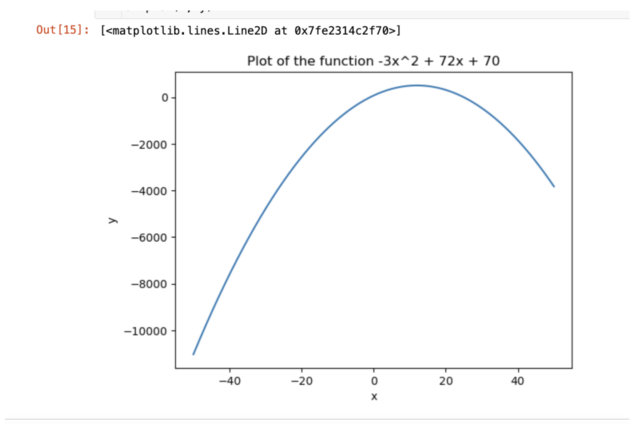
$$y = f(x_1, x_2) = \exp\left(-(x_1 - 2)^2 - (x_2 - 1)^2\right)$$

3. Using `matplotlib`, visualize in 3D the height function as a function of x_1 and x_2 using the `plot_surface` utility for $(x_1, x_2) \in (0, 6) \times (0, 6)$. Use this visualization to argue why there exists a unique solution to Nari's optimization problem on the specified intervals for x_1 and x_2.

   Remark: in this class, we will learn about under what conditions do *multivariate* optimization problems have unique global optima (and no, the second derivative test doesn't exactly generalize directly). Looking at the visualization you produced and the expression for $f(x_1, x_2)$, do you have any ideas for why this problem is guaranteed to have a global maxima? You do not need to write anything responding to this – this is simply food for thought and a preview for the semester.

Solution

1) Yes, the plant *does* have a maximum height. Consider the matplotlib plot of the the height, y, in mm of her Weeping Fig plant as a function of oz of water, x below. This given function is concave down and appears to have a unique maximum:



```
Out[15]: [<matplotlib.lines.Line2D at 0x7fe2314c2f70>]
```

Figure 1: Plot of Height (mm) as a function of Water (oz)

2) We solve analytically for the critical points of the height function by taking the derivative and setting it equal to 0:

$$y = -3x^2 + 72x + 70 \Rightarrow y' = -6x + 72 = 0 \Rightarrow x = 12$$

We confirm that $x = 6$ is a local max via the second derivative test, as $f''(12) = -6$, which means that the function is concave down, so $x = 12$ is a maximum. Note that this is a global maximum because the function is a downward-opening parabola (concave down everywhere), which means there can be no other higher points than the top of the parabola.

With $x = 12$ oz of water, the height of the plant would be $-3(12)^2 + 72(12) + 70 = 502$ mm.

3) Consider the matplotlib plot below. We argue that there exists a unique solution to Nari's problem on because on the boundaries of $[0, 6] \times [0, 6]$, the function value decreases as we move away from the center. In addition, the Gaussian function has a natural bell shape, so it has one global maximum.

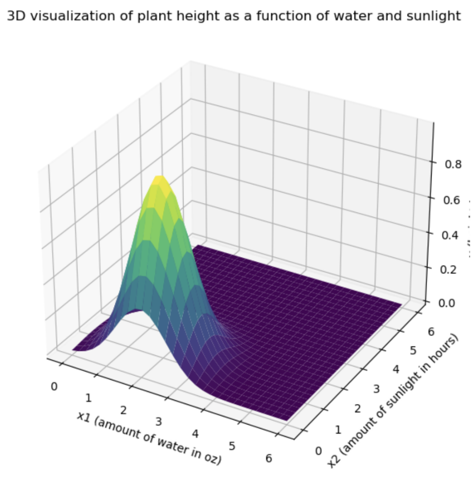


Figure 2: 3D Plot height (mm) as a function of water (oz) and sunlight (hours)

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 packages arriving at the US Postal Service (USPS):

- Packages arrive randomly in any given hour according to a Poisson distribution. That is, the number of packages in a given hour N is distributed $Pois(\lambda)$, with $\lambda = 3$.
- Each package has a random size S (measured in in^3) and weight W (measured in pounds), with joint distribution

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

- Processing time T (in seconds) for each package is given by $T = 60 + 0.6W + 0.2S + \epsilon$, where ϵ is a random noise variable with Gaussian distribution $\epsilon \sim \mathcal{N}(0, 5)$.

For this problem, you may find the `multivariate_normal` module from `scipy.stats` especially helpful. You may also find the `seaborn.histplot` function quite helpful.

1. Perform the following tasks:
   (a) Visualize the Bivariate Gaussian distribution for the size S and weight W of the packages by sampling 500 times from the joint distribution of S and W and generating a bivariate histogram of your S and W samples.
   (b) Empirically estimate the most likely combination of size and weight of a package by finding the bin of your bivariate histogram (i.e., specify both a value of S and a value of W) with the highest frequency. 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?
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. Give one reason for why the Gaussian distribution is an appropriate model for the size and weight of packages. Give one reason for why it may not be appropriate.
4. 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.
5. Let us treat the *total* amount of time it takes to process *all* packages received at the USPS office within *an entire day* (assuming a single day is 24 hours long) as a random variable T^*.
   (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

1a) Below, we visualize the Bivariate Gaussian distribution for the size S and weight W of the packages by sampling 500 times from the joint distribution of S and W and generating a bivariate histogram of your S and W samples (see code in .ipynb file):

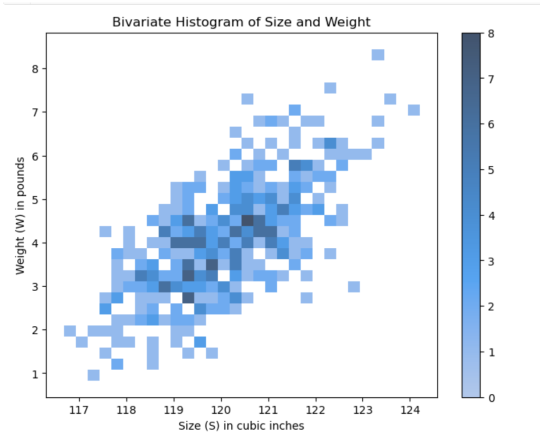


Figure 3: Visualization of Bivariate Gaussian Distribution

1b) We empirically estimate the most likely combination of size and weight of a package by finding the bin of our bivariate histogram with the highest frequency. Visually, this yields about 120.4 cubic inches for size and 3.8 pounds for weight. This estimate is relatively close to the theoretical expected size (120 cubic inches) and expected weight of a package (4 pounds).

2) From the below plot, we can conclude that there is a positive correlation between S and W. When S increases from $S = 118$ to $S = 122$, the peak of the PDF for W also shifts to the right, from about 2.2 to 5.5. This is consistent with the positive value in the covariance matrix and with our intuition, which indicates that as the size of the package increases, the weight tends to increase as well.

3) The Gaussian distribution is a good model for natural variation in measurements when no single factor dominates the variation, which is often the case with the size and weight of packages. When many such small, independent factors contribute to a measurement, via CLT, that the distribution of those measurements will approach a Gaussian distribution.

However, the Gaussian distribution may not be entirely appropriate because it assumes that size and weight can take on any value from negative infinity to positive infinity. In reality, sizes and weights are physically constrained; they cannot be negative, and there may be practical upper limits as well, given something like shipping laws.

This bounded nature of the data could make other distributions, like the log-normal distribution or truncated Gaussian distribution, more appropriate in some cases.

4) Using properties of expectations and variance (ie. linearity of expectations), we compute $E(T)$ analytically:

$$T = 60 + 0.6W + 0.2S + \epsilon \Rightarrow E(T) = E(60 + 0.6W + 0.2S + \epsilon) = 60 + 0.6E(W) + 0.2E(S) + E(\epsilon)$$

$$= 60 + 2.4 + 24 + 0 = 86.4$$

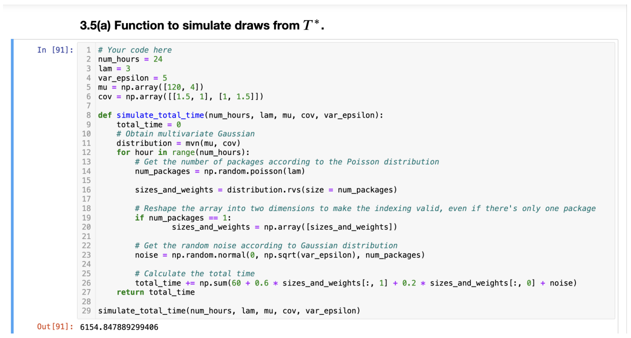
Additionally, we compute $\text{Var}(T)$ analytically:

$$T = 60 + 0.6W + 0.2S + \epsilon \Rightarrow Var(T) = Var(60 + 0.6W + 0.2S + \epsilon)$$

$$= Var(0.6W) + Var(0.2S) + 2(0.6)(0.2)Cov(W, S) + Var(\epsilon)$$

$$= 0.36(1.5) + 0.04(1.5) + .24(1) + 5 = 5.84$$

5a) Below is the function to simulate draws from the distribution of T* (can also see code in the .ipynb file):

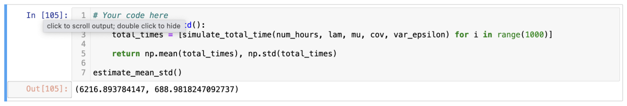


3.5(a) Function to simulate draws from T^*.

```
In [91]:
# Your code here
num_hours = 24
lam = 3
var_epsilon = 5
mu = np.array([120, 4])
cov = np.array([[1.5, 1], [1, 1.5]])

def simulate_total_time(num_hours, lam, mu, cov, var_epsilon):
    total_time = 0
    # Obtain multivariate Gaussian
    distribution = mvn(mu, cov)
    for hour in range(num_hours):
        # Get the number of packages according to the Poisson distribution
        num_packages = np.random.poisson(lam)

        sizes_and_weights = distribution.rvs(size = num_packages)

        # Reshape the array into two dimensions to make the indexing valid, even if there's only one package
        if num_packages == 1:
                sizes_and_weights = np.array([sizes_and_weights])

        # Get the random noise according to Gaussian distribution
        noise = np.random.normal(0, np.sqrt(var_epsilon), num_packages)

        # Calculate the total time
        total_time += np.sum(60 + 0.6 * sizes_and_weights[:, 1] + 0.2 * sizes_and_weights[:, 0] + noise)
    return total_time

simulate_total_time(num_hours, lam, mu, cov, var_epsilon)

Out[91]: 6154.847889299406
```

Figure 4: Function to Simulate Draws from T* Distribution

5b) Via the code below, we empirically estimate the mean and standard deviation of T* to be about 6216.893784147 and 688.9818247092737, respectively.



```
In [105]:
# Your code here
[illegible]d():
    total_times = [simulate_total_time(num_hours, lam, mu, cov, var_epsilon) for i in range(1000)]

    return np.mean(total_times), np.std(total_times)

estimate_mean_std()

Out[105]: (6216.893784147, 688.9818247092737)
```

Figure 5: Simulating 1000 Samples From T* and Estimating Mean/SD