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. Let us define the following system of linear equations which use coefficients w_0, w_1:

$$y_1 = w_0 + w_1 x_1$$

$$y_2 = w_0 + w_1 x_2$$

Solving for w_0 first, we derive an equality for w_1 for each linear equation:

$$w_1 = \frac{y_1 - w_0}{x_1}$$

$$w_1 = \frac{y_2 - w_0}{x_2}$$

Substituting, we have:

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

Then, we will single out w_0 to arrive at an expression in terms of x_1, x_2, y_1, y_2. Cross multiplying and rearranging terms, we get:

$$y_1 x_2 - w_0 x_2 = y_2 x_1 - w_0 x_1$$

$$w_0(x_2 - x_1) = y_1 x_2 - y_2 x_1$$

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

Next, we solve for w_1. First, we begin by subtracting the first equation by the second to arrive at:

$$y_1 - y_2 = w_1 x_1 - w_1 x_2$$

Then, we simply factor out w_1 to get:

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

2. We wish to put this system

$$y_1 = w_0 + w_1 x_1$$

$$y_2 = w_0 + w_1 x_2$$

into the form $y = Xw$ First, we note that we can simply express the left side of the system as:

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Then, we must find a way to express

$$w_0 + w_1 x_1$$

$$w_0 + w_1 x_2$$

in the form Xw. Note that if we have

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

Then this is satisfied. Therefore, we can express the above system of equations as the following matrix equation:

$$\boxed{\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}}$$

3. Using the hint, we know that matrix X must be invertible if there is a unique solution for w. Thus, to check if X is invertible, we recall that if the determinant of X is not 0, then it is invertible. Therefore, we have:

$$\det(X) = x_2 - x_1$$

$$x_2 - x_1 \neq 0$$

Therefore, when

$$\boxed{x_2 \neq x_1}$$

Then there exists a unique solution for w. Writing the solution for w (when it exists) out as a matrix expression, we have:

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

Then, through basic linear algebra and part (2), we have

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

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Therefore, we have:

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

4. From part (3), we have

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

After multiplying, we have:

$$w = \begin{bmatrix} \frac{y_1x_2 - y_2x_1}{x_2 - x_1} \\ \frac{y_1 - y_2}{x_1 - x_2} \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

Therefore, we get:

$$w_0 = \frac{y_1x_2 - y_2x_1}{x_2 - x_1} \quad w_1 = \frac{y_1 - y_2}{x_1 - x_2}$$

Which matches the answer we got in part (1)! One advantage of phrasing the problem of fitting the model in terms of matrix notation is that it is easily representable in programming languages– a n-D array is a relatively simple data structure compared to a data structure that keeps track of individual equations.

Moreover, while the system of equations is easily readable to the human eye, as the number of linear equations grows with the dataset, the complexity drastically increases and the matrix notation simplifies potentially many equations into a singular matrix equation to be solved.

5. In the real world, where datasets are much much larger than 2 points of data, this expression for w would not apply immediately. This is because it is extraordinarily unlikely that all of these millions of points are exactly colinear and can be modeled perfectly with a single line. In the example we just ran through, we only had two points so we are guaranteed that they are colinear unless $x_1 = x_2$.

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.

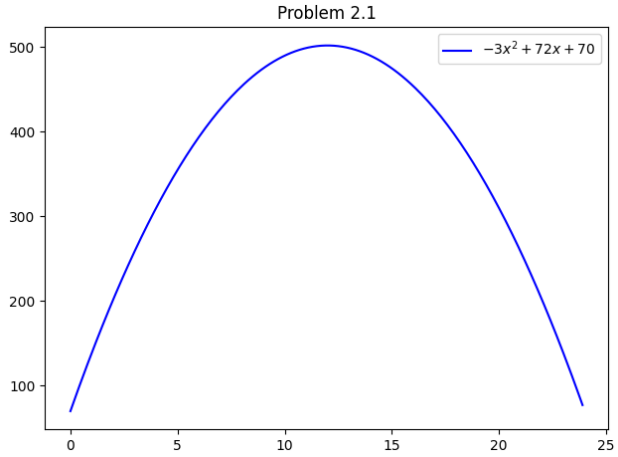
2. Use calculus to find how many ounces of water per week Nari should give to her plant in order to maximize its height. With this much water, how tall will her plant grow?

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. After looking at the plot of the height function, it is clear that there is a maximum plant height since the parabola points downwards. That is, we can tell from the plot of the height function that there exists a unique maximum plant height. This is supported based on the form of the function itself. We know that the degree of the polynomial height function is 2, so it must be parabolic shaped. Then, since the a term in the form $y = ax^2 + 72x + 70$ is negative ($a < 0$), then we conclude that it is a downward pointing parabola. Therefore, the vertex of the parabola will be the unique global maximum.

2. Note that we can find the critical points of

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

by simply taking the first derivative and setting $y' = 0$. This yields:

$$-6x + 72 = 0$$

Therefore, we have a critical point at $x = 12$. Since we have $-6x + 72 > 0$ when $x < 12$ and $-6x + 72 < 0$ when $x > 12$, we know that the slope of the function y is increasing to the left of the critical point and decreasing to the right of the critical point, indicating a maximum. Since this is the only critical point in the range $x \in \mathrm{R}$, then we have found the optimal amount of water to maximize height.

Therefore, Nari should give her plant 12 oz. of water in order to maximize its height, which will yield a plant height of

$$y = -2(12)^2 + 72(12) + 70 = 512\text{mm}$$

3. To determine the values of x_1 and x_2 which yield maximum height y, we can simply examine the form of $f(x_1, x_2)$.

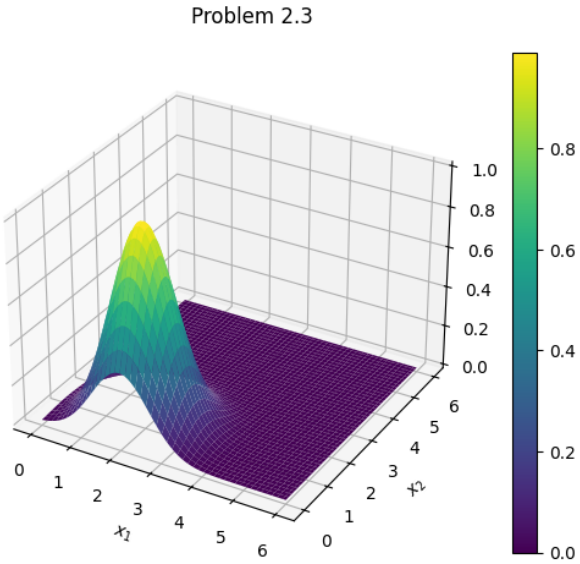
$$y = f(x_1, x_2) = \exp\left(-(x_1 - 2)^2 - (x_2 - 1)^2\right)$$

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

Since taking the square of any number is non-negative, then we have

$$(x_1 - 2)^2 + (x_2 - 1)^2 \geq 0$$

$$-((x_1 - 2)^2 + (x_2 - 1)^2) \leq 0$$



Therefore, we can reason that $f(x_1, x_2) = e^{-((x_1-2)^2+(x_2-1)^2)}$ takes on a maximum value when its exponent is 0. This happens only when $x_1 = 2$ and $x_2 = 1$, which is derived through simple algebra. Therefore, the values of x_1 and x_2 that achieve a maximum plant height are:

$$x_1 = 2 \quad x_2 = 1 \quad f(2, 1) = 1$$

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.

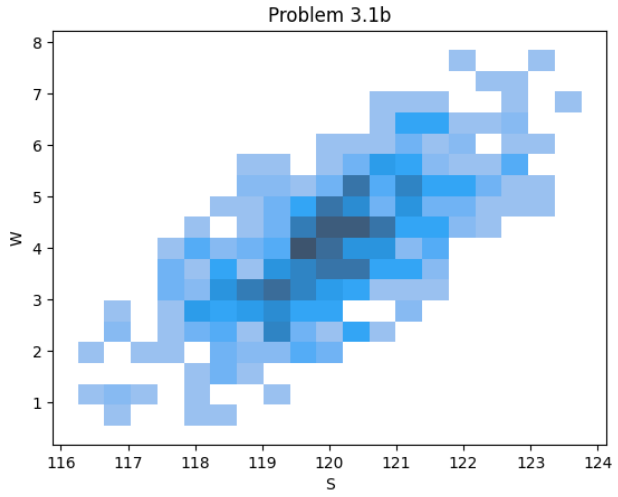
Consider the following model for each package that arrives at the US Postal Service (USPS):

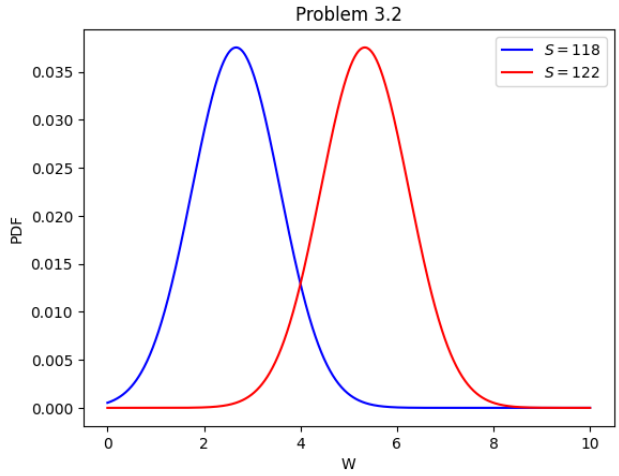
- Every package has a random size S (measured in in^3) and weight W (measured in pounds), with joint distribution

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

- 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

1a. The Gaussian distribution allows random variables to take on negative values. However, in the real world, size and weight can never be negative, so the Gaussian distribution would not be appropriate here.

1b. The theoretical expected size and expected weight is given in the μ vector, with the $E(S) = 120$ and $E(W) = 4$. After sampling and plotting as described, I get a plot that is centered around the point $(120, 4)$, the theoretical expected value of (S, W). After calculating the mean with numpy, I get empirical means of $\bar{S} = 120.026$ and $\bar{W} = 4.019$. These are extremely close to the theoretical expected values.

2. The two PDF plots appear to be identically shaped and indicate a positive correlation between S and W. When we increase S from 118 to 122, we see a similar rightward shift in the distribution of W towards larger values.

3. Using the given definition of T, we have by linearity that:

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

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

$$\boxed{E(T) + 60 + 2.4 + 24 + 0 = 86.4}$$

Moving onto the variance, we have:

$$\text{Var}(T) = \text{Var}(60) + 0.36\text{Var}(W) + 0.04\text{Var}(S) + 2(0.6)(0.2)\text{Cov}(S, W) + \text{Var}(\epsilon)$$

$$\text{Var}(T) = 0 + 0.36 * 1.5 + 0.04 * 1.5 + 0.24 + 5$$

$$\boxed{\text{Var}(T) = 5.84}$$

4. If we have fixed $N = n$, then the distribution of the number of fragile packages is Binomial, with success probability $P(W < 4)$. Since we have that $E(W) = 4$, then we know from the symmetry of the normal that $P(W < 4) = 0.5$. Therefore, if we define F as the number of fragile packages, we have

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

5a. See code.

5b. After generating 1000 samples using the function from part 5.1 and calculating the sample mean and standard deviation using numpy, I get the following empirical estimates:

$$\bar{T*} = 6261.86 \quad \sigma = 749.27$$

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?

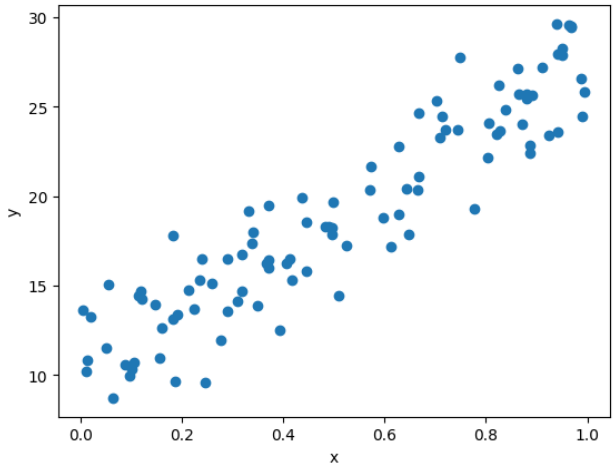
4. Having walked through the idea of linear regression, Steve then attempts to implement a

   `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. The issue with Steve's current code is that the data is formatted such that the first column is the x vector and the second column is the y vector. However, he indexes the first two rows and assigns them to x and y, not columns, which basically gives him the first two data points. After fixing his code, there appears to be a positive linear correlation between x and y.

2. The problem was that the intercept vector was not properly shaped as a column vector. We also need x to be a column vector for np.hstack to concatenate properly. Therefore, the problem lied in the shaping of the inputs.

The shape of $X.shape$ is (100,2) and the shape of $y.shape$ is (100,). The reason there is no second term in the y.shape is because of how numpy works. Y is a vector that is a column slice from a numpy array, which means that it is 1 dimensional and therefore does not need a second dimension specified.

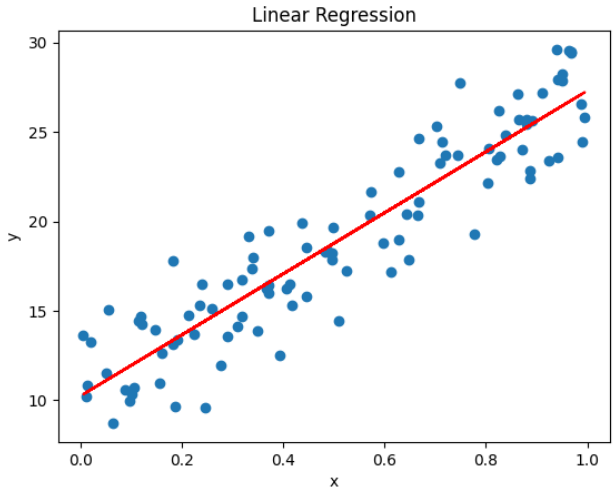
3. If X^T has shape (2,100), then naturally $X^T X$ has shape (2,2). Then, the inverse of this also has shape $(2, 2)$ and it is then multiplied by X^T, giving us shape (2,100). Finally, it is multiplied by y which has shape $(100,)$. Therefore, $\hat{w}$ has shape of (2,).

4. In the predict method, Steven incorrectly accesses w. Since w is an internal data structure within the Regression class, he must call $self.w$ in order to access w. In the fit function, there are a lot of issues. First, we can't use "*" to denote matrix multiplication (we need @). Then, the third X has to be transposed and y has to be not transposed for dimensions to match, Then, we can't use $**$ to denote inverse, we must use the np.linalg.inv method. Finally, we need to store the result of w in fit to self.w instead of returning it.

5. The formula is as follows: mse = np.mean(np.square($\mathbf{y}$ - $\hat{y}$))

This can be written as:

$$\overline{(\mathbf{y} - \hat{\boldsymbol{y}})^2}$$


Linear Regression
30
25
20
15
10
y
0.0
0.2
0.4
0.6
0.8
1.0
x