Assignment-2

Part-1: answer with examples the following questions:

1. What is Linear Regression? Can you share the intuition behind linear regression? Why regression why not classification?
2. Write about Linear regression algorithm as a pseudocode and explain about linear regression.
3. Suppose you use Batch Gradient Descent and you plot the validation error at every epoch. If you notice that the validation error consistently goes up, what is likely going on? How can you fix this?
4. The table below gives the amount of time students in a class studied for a test and their test scores. Graph the data on a scatter plot, find the line of best fit, and write the equation for the line you draw (show all the values θs at each iteration).

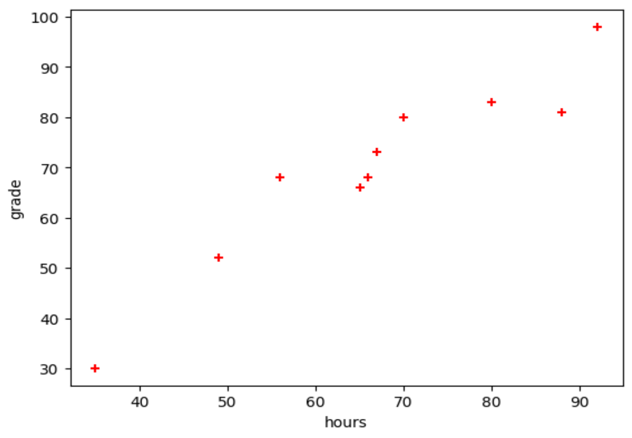
Hours Studied	1	0	3	1.5	2.75	1	0.5	2
Test Score	78	75	90	89	97	85	81	80

- Using the linear regression equation, predict a student's test score if they studied for 4 hours.

Part-2: Programming Exercise: To get started with the exercise, you need to download Python and VScode

In this part of this exercise, you will implement linear regression with one variable to predict the final grades of the students based on the number of hours they studied. You have data for hours of study and grades from the stduents. The file hours_grades.csv contains the dataset for our linear regression problem. The first column is the name of the student, the second column is the number of the hours studied, and the third column is the final grade.

When you plot the dataset, it looks like the following



2.1 Gradient Descent

In this part, you will fit the linear regression parameters **θ** to our dataset using gradient descent. The objective of linear regression is to minimize the cost function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

where the hypothesis ***hθ(x)*** is given by the linear model.

Recall that the parameters of your model are the ***θj*** values. These are the values you will adjust to minimize cost ***J(θ)***. One way to do this is to use the batch gradient descent algorithm. In batch gradient descent, each iteration performs the update.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J\left(\theta_0, \theta_1\right)$$

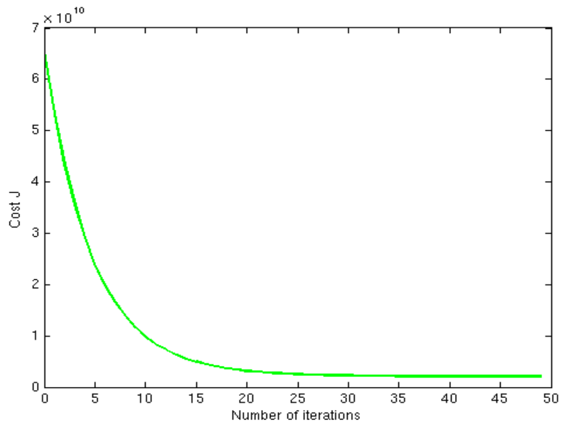
With each step of gradient descent, your parameters ***θj*** come closer to the optimal values that will achieve the lowest cost ***J(θ)***.

2.2 Implementation

As you perform gradient descent to minimize the cost function ***J(θ)***, it is helpful to monitor the convergence by computing the cost. In this section, you will implement a function to calculate ***J(θ)*** so you can check the convergence of your gradient descent implementation.
You also need to initialize the initial parameters to 0 and the learning rate alpha to 0.002, and the iterations = 2000. A good way to verify that gradient descent is working correctly is to look at the value of ***J(θ)*** and check that it is decreasing with each step.

2.3 Plotting the *θs and the cost function J(θ).*

To understand the cost function ***J(θ)*** better, you need to plot the cost over the ***J*** values against the number of the iterations. If you picked a learning rate within a good range, your plot looks similar to the below figure.



If your graph looks very different, especially if your value of the cost function ***J(θ)*** increases or even blows up, adjust your learning rate and try again. You may also want to adjust the number of iterations you are running if that will help you see the overall trend in the curve.

2.4 Profit Prediction

Finally, use your final ***θ*** values to make predictions on

What to Submit: Canvas Classroom.

- Assignment2YourName.pdf (e.g., Assignment-2AliAburas.pdf). That contains all the answers to the above questions + screen-shots of your code+results
- Python code

Note: Dont zip the pdf file!