4 - Refresher on linear regression

In this practice lab, you will fit the linear regression parameters (w, b) to your dataset.

- The model function for linear regression, which is a function that maps from `x` (city population) to `y` (your restaurant's monthly profit for that city) is represented as

$$f_{w,b}(x) = wx + b$$

- To train a linear regression model, you want to find the best (w, b) parameters that fit your dataset.
  - To compare how one choice of (w, b) is better or worse than another choice, you can evaluate it with a cost function $J(w, b)$
    - J is a function of (w, b). That is, the value of the cost $J(w, b)$ depends on the value of (w, b).
  - The choice of (w, b) that fits your data the best is the one that has the smallest cost $J(w, b)$.

- To find the values (w, b) that gets the smallest possible cost $J(w, b)$, you can use a method called **gradient descent**.
  - With each step of gradient descent, your parameters (w, b) come closer to the optimal values that will achieve the lowest cost $J(w, b)$.

- The trained linear regression model can then take the input feature x (city population) and output a prediction $f_{w,b}(x)$ (predicted monthly profit for a restaurant in that city).

5 - Compute Cost

Gradient descent involves repeated steps to adjust the value of your parameter (w, b) to gradually get a smaller and smaller cost $J(w, b)$.

- At each step of gradient descent, it will be helpful for you to monitor your progress by computing the cost $J(w, b)$ as (w, b) gets updated.
- In this section, you will implement a function to calculate $J(w, b)$ so that you can check the progress of your gradient descent implementation.

Cost function

As you may recall from the lecture, for one variable, the cost function for linear regression $J(w, b)$ is defined as

$$J(w, b) = \frac{1}{2m} \sum_{i=0}^{m-1} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

- You can think of $f_{w,b}(x^{(i)})$ as the model's prediction of your restaurant's profit, as opposed to $y^{(i)}$, which is the actual profit that is recorded in the data.
- m is the number of training examples in the dataset

Model prediction

- For linear regression with one variable, the prediction of the model $f_{w,b}$ for an example $x^{(i)}$ is representented as:

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

This is the equation for a line, with an intercept b and a slope w

Implementation

Please complete the `compute_cost()` function below to compute the cost $J(w, b)$.

Exercise 1

Complete the `compute_cost` below to:

- Iterate over the training examples, and for each example, compute:
  - The prediction of the model for that example

$$f_{wb}(x^{(i)}) = wx^{(i)} + b$$

  - The cost for that example

$$cost^{(i)} = (f_{wb} - y^{(i)})^2$$

- Return the total cost over all examples

$$J(\mathbf{w}, b) = \frac{1}{2m} \sum_{i=0}^{m-1} cost^{(i)}$$

  - Here, m is the number of training examples and $\sum$ is the summation operator

If you get stuck, you can check out the hints presented after the cell below to help you with the implementation.

6 - Gradient descent

In this section, you will implement the gradient for parameters w, b for linear regression.

As described in the lecture videos, the gradient descent algorithm is:

$$\begin{align*}& \text{repeat until convergence:} \; \lbrace \newline \; & b := b - \alpha \frac{\partial J(w,b)}{\partial b} \newline \; & w := w - \alpha \frac{\partial J(w,b)}{\partial w} \tag{1} \; & \newline & \rbrace\end{align*}$$

where, parameters w, b are both updated simultaniously and where

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m} \sum\limits_{i = 0}^{m-1} (f_{w,b}(x^{(i)}) - y^{(i)}) \tag{2}$$

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m} \sum\limits_{i = 0}^{m-1} (f_{w,b}(x^{(i)}) -y^{(i)})x^{(i)} \tag{3}$$

- m is the number of training examples in the dataset
- $f_{w,b}(x^{(i)})$ is the model's prediction, while $y^{(i)}$, is the target value

You will implement a function called `compute_gradient` which calculates $\frac{\partial J(w)}{\partial w}$, $\frac{\partial J(w)}{\partial b}$

Exercise 2

Please complete the `compute_gradient` function to:

- Iterate over the training examples, and for each example, compute:
  - The prediction of the model for that example
    $$f_{wb}(x^{(i)}) = wx^{(i)} + b$$
  - The gradient for the parameters w, b from that example
    $$\frac{\partial J(w,b)}{\partial b}^{(i)} = (f_{w,b}(x^{(i)}) - y^{(i)})$$
    $$\frac{\partial J(w,b)}{\partial w}^{(i)} = (f_{w,b}(x^{(i)}) -y^{(i)})x^{(i)}$$
- Return the total gradient update from all the examples
  $$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m} \sum\limits_{i = 0}^{m-1} \frac{\partial J(w,b)}{\partial b}^{(i)}$$
  $$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m} \sum\limits_{i = 0}^{m-1} \frac{\partial J(w,b)}{\partial w}^{(i)}$$
  - Here, m is the number of training examples and $\sum$ is the summation operator

If you get stuck, you can check out the hints presented after the cell below to help you with the implementation.