Machine Learning Task

In this task, you'll be working on implementing the cost function and gradient descent function for a simple linear regression model using Python.

Cost function and gradient descent are the fundamental concepts behind the working of linear Regression models. So lets dive deep into then and implement them in either Google colab , Jupiter notebook or any editor of your choice.

To learn more about these terminologies you can follow these url's and also feel free to search over the internet to find something that excites you more about the concept.

https://www.youtube.com/watch?v=4b4MUYve_U8

https://www.youtube.com/watch?v=sDv4f4s2SB8

https://www.youtube.com/watch?v=erfeZg27B7A

You are provided with a dataset in CSV format containing 1000 rows and 2 columns:

experience: Experience in months.

salary: Salary in thousands.

Your objective is to implement the following functions in the provided code:

compute_cost: Implement the cost function for linear regression.

gradient_descent: Implement the gradient descent algorithm to minimize the cost function and find the optimal parameters (slope and intercept).

Once you've implemented these functions, you'll use them to predict output values on the given custom data.

Theory for Linear Regression

here w and b refers to parameter like weight's and bias

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 $J(w, b)$.

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 1

(cost function in linear regression)

- 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

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:

$$\begin{aligned} &\text{repeat until convergence: } \{ \\ &\quad b := b - \alpha \frac{\partial J(w,b)}{\partial b} \\ &\quad w := w - \alpha \frac{\partial J(w,b)}{\partial w} \\ &\} \end{aligned} \tag{1}$$

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

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

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

- 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

Implementation 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
  $$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m} \sum_{i=0}^{m-1} \frac{\partial J(w,b)}{\partial b}^{(i)}$$
  $$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m} \sum_{i=0}^{m-1} \frac{\partial J(w,b)}{\partial w}^{(i)}$$
  - Here, m is the number of training examples and $\sum$ is the summation operator

Submission Instructions

Complete the functions compute_cost , gradient_descent and some other functions in the given code.

Ensure your code runs without errors.

Upload your completed code file to the submission portal provided.

IMP :

1 provide a screenshot of the output of the linear fit graph

2 provide a screenshot of the final predicted values on the custom data

3 provide a screenshot of the minimum values found out for w,b and the cost

Evaluation Criteria

Your performance will be evaluated based on the following criteria:

Accuracy of implementation: How accurately you implement the cost function and gradient descent algorithm.

Efficiency: How efficiently your code runs and converges to the optimal solution.

Code Clarity: The clarity and readability of your code.

Correctness of Predictions: How well your model predicts output values on the given custom data.

Task Completion

Once you've completed the task, please submit your solution within the provided deadline. We'll review your submission and notify you of the results shortly thereafter.

Good luck, and happy coding!