THE UNIVERSITY OF LAHORE

Faculty of Engineering and Technology
Department of Technology
Spring Semester 2024

Instructor's Name:	**Asim Anwar**	**Subject:**	**Artificial Intelligence and Optimization**
Subject Code:	**CS13315**	**Assignment #:**	**04**
Total Marks:	**10**	**Assigning Date:**	**07-3-2024**
Due Date& Time:	**14-3-2024**		

Instructions for Candidates

i. Provide your original work reflecting your own "understanding".
ii. Copying or reproducing others work may lead to zero credits in that particular question(s).

Topic: Gradient Descent for Linear Regression

Assignment Goals:

In this assignment, you will:

- automate the process of optimizing *w* and *b* using gradient descent.

Tools

In this assignment, we will make use of:

1. NumPy, a popular library for scientific computing
2. Matplotlib, a popular library for plotting data
3. plotting routines in the lab_utils.py file in the local directory

Problem Statement

Let's use the same two data points as before - a house with 1000 square feet sold for $300,000 and a house with 2000 square feet sold for $500,000.

Size (1000 sqft)	Price (1000s of dollars)
1.0	300
2.0	500

Task 1: Implement gradient descent algorithm

You will implement gradient descent algorithm for one feature. You will need three functions.

1. compute_cost` implementing equation (2) given in Annexure (already developed the code in Lab 3)
2. compute_gradient` implementing equations (4) and (5) given in Annexure
3. gradient_descent`, utilizing compute_gradient and compute_cost

Implement compute_gradient Function

`compute_gradient` implements equations (4) and (5) given in Annexure, and returns $\frac{\partial J(w,b)}{\partial w}$, $\frac{\partial J(w,b)}{\partial b}$. The embedded comments describe the operations.

Please complete the following function in Python*:*

```
def compute_gradient():
    """
    Computes the gradient for linear regression
    Args:
      x (ndarray (m,)): Data, m examples
      y (ndarray (m,)): target values
      w,b (scalar)    : model parameters
    Returns
```

```
      dj_dw (scalar): The gradient of the cost w.r.t. the parameters w
      dj_db (scalar): The gradient of the cost w.r.t. the parameter b
     """

    # Number of training examples
```

Implement Gradient Descent Function in Python

Now that gradients can be computed, gradient descent, described in equation (3) of Annexure can be implemented by completing the function `gradient_descent`. The details of the implementation are described in the comments. Below, you will complete this function to find optimal values of *w* and *b* on the training data.

```
def gradient_descent( ):
    """
    Performs gradient descent to fit w,b. Updates w,b by taking
    num_iters gradient steps with learning rate alpha

    Args:
      x (ndarray (m,))  : Data, m examples
      y (ndarray (m,))  : target values
      w_in,b_in (scalar): initial values of model parameters
      alpha (float):     Learning rate
      num_iters (int):   number of iterations to run gradient descent
      cost_function:     function to call to produce cost
      gradient_function: function to call to produce gradient

    Returns:
      w (scalar): Updated value of parameter after running gradient
descent
      b (scalar): Updated value of parameter after running gradient
descent
      J_history (List): History of cost values
```

```
    p_history (list): History of parameters [w,b]
    """
```

After completing the function, please call this function to compute the optimal values of the parameters by running the following code

```
# initialize parameters
w_init = 0
b_init = 0
# some gradient descent settings
iterations = 10000
tmp_alpha = 1.0e-2
# run gradient descent
w_final, b_final, J_hist, p_hist = gradient_descent(x_train ,y_train, w_init, b_init, tmp_alpha,
                                                    iterations, compute_cost, compute_gradient)
print(f"(w,b) found by gradient descent: ({w_final:8.4f},{b_final:8.4f})")
```

Task 2: Plotting and Prediction

1. Write a Python code to obtain a plot between cost versus iterations of gradient descent. Also, properly label the graphs i.e. assign labels to a and y axes, title etc to the graphs

2. Predict and print three housing price values that are not in the training data set.

Annexure

Gradient descent summary

So far in this course, you have developed a linear model that predicts $f_{w,b}(x^{(i)})$:

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b \tag{1}$$

In linear regression, you utilize input training data to fit the parameters w,b by minimizing a measure of the error between our predictions $f_{w,b}(x^{(i)})$ and the actual data $y^{(i)}$. The measure is called the $cost$, $J(w,b)$. In training you measure the cost over all of our training samples $x^{(i)}, y^{(i)}$

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

In lecture, *gradient descent* was described as:

$$\begin{aligned} &\text{repeat until convergence: } \{ \\ &\quad w = w - \alpha \frac{\partial J(w,b)}{\partial w} \\ &\quad b = b - \alpha \frac{\partial J(w,b)}{\partial b} \\ &\} \end{aligned} \tag{3}$$

where, parameters w, b are updated simultaneously.
The gradient is defined as:

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m}\sum_{i=0}^{m-1}(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)} \tag{4}$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m}\sum_{i=0}^{m-1}(f_{w,b}(x^{(i)}) - y^{(i)}) \tag{5}$$

Here *simultaniously* means that you calculate the partial derivatives for all the parameters before updating any of the parameters.