Faculty of Computing and Information Technology

University of the Punjab,
Lahore

Artificial Intelligence Lab 11

Instructor: Qamar U Zaman

Implementing Simple Linear Regression with Weight Adjustment and Model Evaluation

Objective:
The objective of this lab is to implement **Linear Regression** model from scratch, calculate the slope and intercept manually, adjust the model's weights using basic gradient-based updates, and evaluate the model using **Mean Squared Error (MSE)**.

Key Concepts:

1. **Linear Regression:**
   - **Linear Regression** models the relationship between a single independent variable X and a dependent variable Y.
2. **Mean Squared Error (MSE):**
   - **MSE** is a common loss function used to evaluate the model's performance. It measures the average squared difference between the predicted and actual values.
3. **Weight Adjustment (Gradient Descent):**
   - In this lab, we introduce a simple **gradient descent** method to adjust the model's weights iteratively and minimize the error between predicted and actual values.

Steps to Implement Linear Regression:

1. Calculate the Mean of X and Y:

- First, calculate the mean of the input data X and the target data Y to use in the following calculations.

2. Calculate the Slope:

- Use the covariance between X and Y and the variance of X to compute the slope.

3. Calculate the Intercept:

- The intercept is calculated using the mean values of X, Y, and the slope

4. Make Predictions:

- Using the computed slope and intercept, we can make predictions for Y based on new values of X.

5. Evaluate the Model Using MSE:

- **MSE** is calculated by comparing the predicted values to the actual target values. The lower the MSE, the better the model.

6. Weight Adjustment Using Gradient Descent:

- After fitting the model, apply gradient descent to refine the weights. This involves adjusting the weights iteratively to minimize the cost (MSE).

Experiment with the Provided Dataset:

- **Dataset:**
  - Use the following dataset.

Experience	**Salary**
0	30000
1	35000
2	40000
3	45000
4	50000
5	55000
6	60000
7	65000
8	70000
9	75000
10	80000

Task Outline:

1. **Fit the Model:**
   - Start by calculating the slope and intercept manually using the basic formulas.
   - Once you have these values, use them to make predictions on the dataset.
2. **Implement Gradient Descent:**
   - Implement a simple gradient descent algorithm to iteratively adjust the model's weights based on the model's performance (MSE).

3. **Model Evaluation:**
    - Calculate the **MSE** on the training data to evaluate the performance of the model.
    - Compare the predicted values to the actual values to assess the model's accuracy.
4. **Experiment with Different Learning Rates and Iterations:**
    - Try different learning rates for gradient descent and see how the model’s accuracy improves or worsens.
    - Experiment with a different number of iterations and evaluate how it impacts the convergence of the model.

Code Template:

```
def calculate_mean(values):
    """
    This function takes a list (or numpy array) of values and returns their 
mean.
    Used for calculating the mean of X and Y.
    """
    pass  # Implement mean calculation logic here

def calculate_slope(X, Y, mean_X, mean_Y):
    """
    This function calculates the slope (theta_1) of the regression line.
    The slope is computed using the formula that relates covariance of X 
and Y to variance of X.
    """
    pass  # Implement slope calculation logic here

def calculate_intercept(mean_X, mean_Y, slope):
    """
    This function calculates the intercept (theta_0) of the regression 
line.
    The intercept is the value of Y when X = 0.
    """
    pass  # Implement intercept calculation logic here

def predict(X, theta_0, theta_1):
    """
    This function predicts the target values (Y) based on the learned 
model.
    It uses the formula: Y = theta_0 + theta_1 * X
    """
    pass  # Implement prediction logic here

def calculate_mse(Y, Y_pred):
    """
    This function calculates the Mean Squared Error (MSE) between the true 
target values Y and the predicted values Y_pred.
    MSE is used to evaluate the performance of the regression model.
    """
    pass  # Implement MSE calculation logic here
 def gradient_descent(X, Y, theta_0, theta_1, learning_rate,
     iterations):"""
     This function adjusts the weights (theta_0 and theta_1) using
 gradientdescent to minimize the Mean Squared Error.
     The function iteratively updates the weights to reduce the
 predictionerror.
```

```
    """
    pass  # Implement gradient descent logic here



def fit_linear_regression(X, Y, learning_rate=0.01,
    iterations=1000):"""
    This function fits the linear regression model by first calculating
theslope and intercept,
    then applying gradient descent to adjust the weights (theta_0
andtheta_1).
    It returns the optimal values for theta_0 and
    theta_1."""
    pass  # Implement fitting logic here, including gradient descent call


def
    test_model(
    ):"""
    This function tests the linear regression model using a given dataset.
    It calculates the model parameters, makes predictions, and
evaluatesperformance using MSE.
    """
```