Lab No # 04

Name Maaz Ahamd
Reg No B23F0722AI170
Section Ai Yellow
Submitted to Sir Abdullah Sajid

## Task No 01 Batch Gradient Descent

```
import numpy as np

def batch_gradient_descent(X, y, w, b, learning_rate, epochs):
    m = len(y)
    for _ in range(epochs):
        y_pred = np.dot(X, w) + b
        error = y_pred - y
        dw = (1/m) * np.dot(X.T, error)
        db = (1/m) * np.sum(error)
        w -= learning_rate * dw
        b -= learning_rate * db
    return w, b
```

## Description

Uses the entire dataset to compute the gradient in each iteration.

Provides stable convergence but can be slow for large datasets.

Suitable for small to medium-sized data.

## Task No 02 Stochastic Gradient Descent

```
def stochastic_gradient_descent(X, y, w, b, learning_rate, epochs):
    m = len(y)
    for _ in range(epochs):
        for i in range(m):
            xi = X[i]
            yi = y[i]
            y_pred = np.dot(xi, w) + b
            error = y_pred - yi
            dw = error * xi
            db = error
            w -= learning_rate * dw
            b -= learning_rate * db
    return w, b
```

## Description Updates weights using one training example at a time.

Much faster and noisier, causing the cost function to fluctuate.

Helps escape local minima and is ideal for large datasets.

## Task No 3 Mini-Batch Gradient Descent

```
def mini_batch_gradient_descent(X, y, w, b, learning_rate, epochs, batch_size):
    m = len(y)
    for _ in range(epochs):
        indices = np.random.permutation(m)
        X_shuffled = X[indices]
        y_shuffled = y[indices]
        for i in range(0, m, batch_size):
            X_batch = X_shuffled[i:i+batch_size]
            y_batch = y_shuffled[i:i+batch_size]
            y_pred = np.dot(X_batch, w) + b
            error = y_pred - y_batch
            dw = (1/len(y_batch)) * np.dot(X_batch.T, error)
            db = (1/len(y_batch)) * np.sum(error)
            w -= learning_rate * dw
            b -= learning_rate * db
    return w, b
```

Description

bold text Divides data into small batches to balance speed and stability.

Offers faster convergence than batch and more stability than SGD.

Commonly used in modern machine learning and deep learning models.

## Summary

1. Gradient Descent is an optimization algorithm used to minimize the cost function by updating model parameters.
2. It works by moving step-by-step in the direction of the negative gradient to find the minimum point.
3. There are three main types — Batch, Stochastic, and Mini-Batch — each differing in how much data they use per update.
4. Batch provides accuracy but is slow, SGD is fast but noisy, and Mini-Batch gives the best balance between speed and stability.
5. Gradient Descent is a core technique in training machine learning and deep learning models efficiently.
