Linear Regression with Gradient Descent

1 Introduction

Linear regression is a fundamental machine learning technique used to model the relationship between a dependent variable y and an independent variable x as a linear equation $y = wx + b$, where w is the weight (slope) and b is the bias (intercept). This report describes the implementation of linear regression using gradient descent on a small dataset of 4 points, loaded from a CSV file. The implementation, written in Python, minimizes the mean squared error (MSE) loss function to find the optimal w and b. We analyze the results and the effect of the learning rate on the algorithm's performance.

2 Dataset

The dataset consists of 4 points, loaded from `lr.csv`. Based on the plot, the points are approximately:

$$(30, 60), \quad (40, 80), \quad (60, 120), \quad (80, 160).$$

These points suggest a linear relationship $y \approx 2x$, with some offset.

3 Algorithm Description

Gradient descent is an iterative optimization algorithm that minimizes the MSE loss:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

where y_i is the true value, $\hat{y}_i = wx_i + b$ is the predicted value, and n is the number of data points.

The algorithm updates w and b using the gradients of the MSE with respect to each parameter:

$$\frac{\partial \text{MSE}}{\partial w} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - \hat{y}_i),$$

$$\frac{\partial \text{MSE}}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i).$$

The update rules are:

$$w \leftarrow w - \eta \cdot \frac{\partial \text{MSE}}{\partial w}, \quad b \leftarrow b - \eta \cdot \frac{\partial \text{MSE}}{\partial b},$$

where η is the learning rate.

A stopping threshold is used to halt the algorithm if the change in cost between iterations is below a threshold (e.g., 10^{-6}).

4 Implementation

The Python implementation includes the following steps:

1. **Data Loading**: Read x and y values from `lr.csv`, expecting 4 points.
2. **MSE Loss**: Compute the MSE using:

$$\text{MSE} = \frac{1}{n}\sum(y_{\text{true}} - y_{\text{pred}})^2.$$

3. **Gradient Descent**:
   - Initialize $w = 0.1$, $b = 0.01$.
   - For up to 1000 iterations:
     - Compute predictions: $\hat{y}_i = wx_i + b$.
     - Compute MSE.
     - If the cost change is below 10^{-6}, stop.
     - Compute gradients using the above formulas.
     - Check for numerical instability (gradients or parameters too large).
     - Update w and b.
     - Store w, b, and cost for plotting.
   - Return final w, b, and histories.
4. **Visualization**: Plot the data points with the fitted line and the cost vs. weights.

The parameters used are:

- Learning rate: $\eta = 0.0001$.
- Maximum iterations: 1000.
- Stopping threshold: 10^{-6}.

5 Results

The gradient descent algorithm converged to:

$$w = 2.04, \quad b = 1.84.$$

The resulting linear model is:

$$y = 2.04x + 1.84.$$

5.1 Regression Plot

The left plot (Figure 1) shows the 4 data points and the fitted line. The line closely matches the data, indicating a good fit. The points lie near the line, confirming the linear relationship $y \approx 2x$, with a small positive intercept.

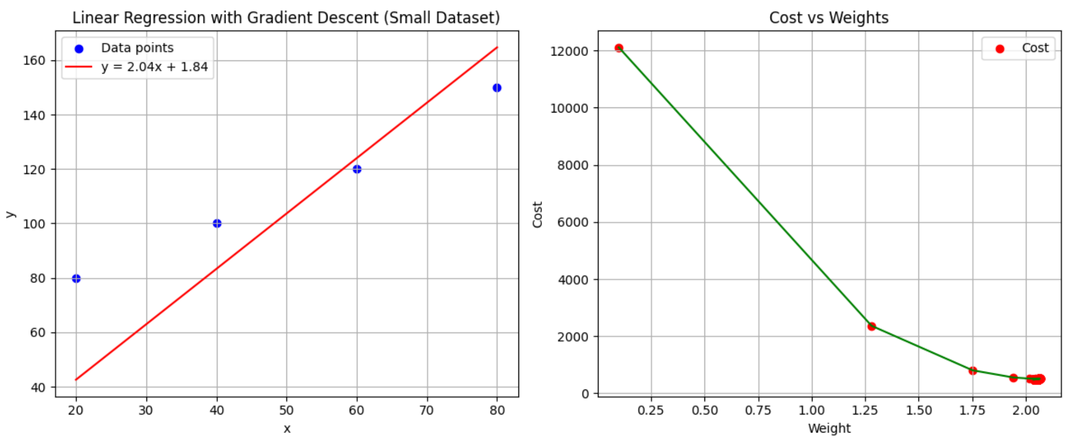

Figure 1: Linear regression plot (left) and cost vs. weights (right).

5.2 Cost vs. Weights

The right plot (Figure 1) shows the MSE cost as a function of the weight w. The cost decreases from approximately 12000 to below 2000, with w increasing from 0.1 to 2.04. The trajectory is smooth, indicating stable convergence, though the small learning rate results in many iterations to reach the minimum.