Author: Zara Irfan

GitHub: Zara-Irfan

Gradient Descent for Linear Regression

Overview

This Python script demonstrates the implementation of **Gradient Descent** to perform simple linear regression on a small dataset. The goal is to fit a line that best approximates the relationship between the independent variable x and the dependent variable y by iteratively minimizing the mean squared error.

The code visualizes the learning process by plotting the regression line at each iteration, showing how the line converges towards the best fit.

Features

- Implements gradient descent from scratch without using external ML libraries.
- Visualizes each step of the gradient descent process.
- Final plot shows the original data points with the optimized regression line.
- Adjustable hyperparameters: learning rate (alpha) and number of iterations.

Code Explanation

- **Data**: The dataset consists of five points with input values x and corresponding target values y.
- **Initialization**: The slope (m) and intercept (b) are initialized to zero.
- **Gradient Descent Loop**: For a fixed number of iterations, the script:
    - Predicts the outputs (y_pred) based on current parameters.
    - Calculates the gradients of the loss function with respect to m and b.
    - Updates m and b by moving against the gradient scaled by the learning rate.
    - Plots the current regression line (with transparency) to visualize convergence.
- **Final Visualization**: Displays the original data points in red and the converged regression line.

Usage

1. Clone or download this repository.
2. Ensure you have the required libraries installed: numpy and matplotlib.
3. Run the script in a Jupyter notebook or Python environment.
4. Observe the stepwise improvement of the regression line toward fitting the data.

bash

CopyEdit

```
pip install numpy matplotlib
```

python

CopyEdit

```
# Run the provided Python code to see the gradient descent in action
python gradient_descent_linear_regression.py
```

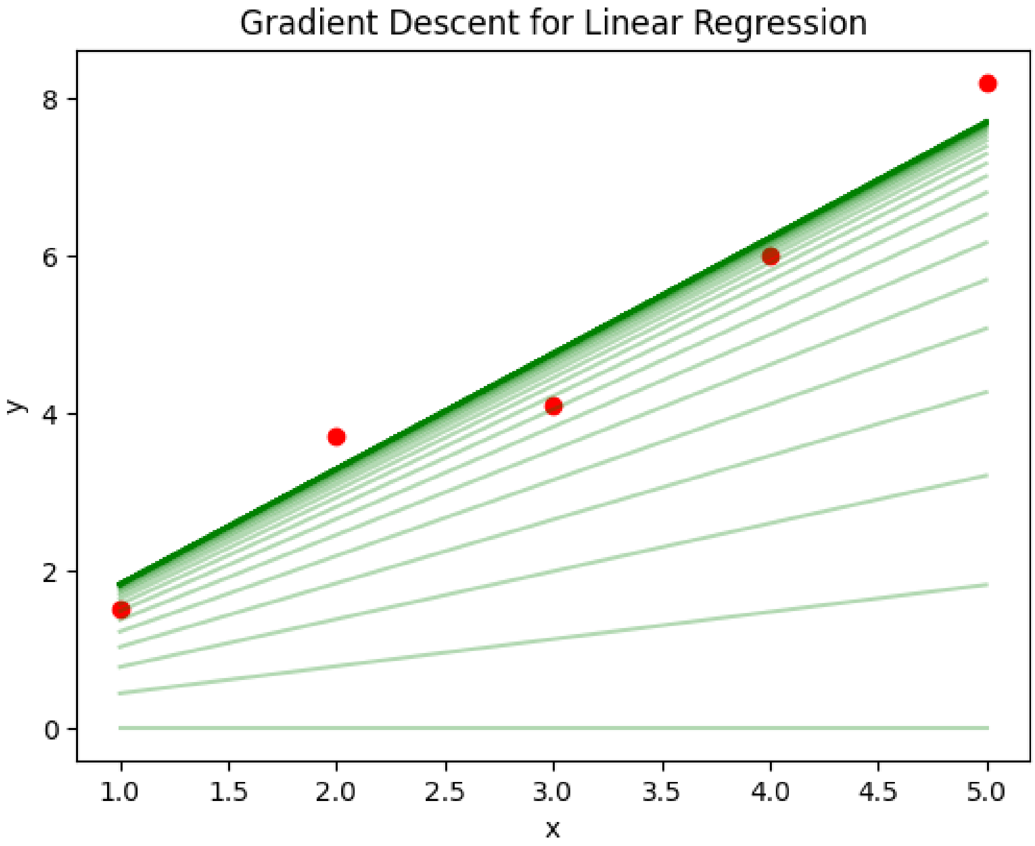
Code

python

CopyEdit

```
import numpy as np
import matplotlib.pyplot as plt


# Sample data
x = np.array([1, 2, 3, 4, 5])
y = np.array([1.5, 3.7, 4.1, 6.0, 8.2])


# Hyperparameters
alpha = 0.01  # learning rate
iterations = 50
m = 0  # initial slope
b = 0  # initial intercept
```

```
plt.figure()

# Gradient Descent Loop
for _ in range(iterations):
    y_pred = m * x + b
    error = y - y_pred

    # Plot current line (in green)
    plt.plot(x, y_pred, color='green', alpha=0.3)

    # Gradient calculation
    m_grad = -(2 / len(x)) * sum(x * error)
    b_grad = -(2 / len(x)) * sum(error)

    # Update parameters
    m -= alpha * m_grad
    b -= alpha * b_grad

# Final plot with data points
plt.scatter(x, y, color='red')  # actual data
plt.title("Gradient Descent for Linear Regression")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```


Gradient Descent for Linear Regression
y
x
0
2
4
6
8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0