基于梯度下降法的线性拟合

2151094 宋正非

基于梯度下降法针对 data_x.txt 与 data_y.txt 中的数据实现线性拟合。其中，损失函数为预测值与真实值的均方误差；学习率设置为 0.3，最大迭代次数设置为 50，w 和 b 的初始值设置为随机生成的 0 到 1 之间的任意数值。

1. 前 30 次迭代中的 w、b 和 Loss 值

Iteration 1: w = 1.340774907996923, b = 2.0042583626939163, Loss: 29.035019645728234
Iteration 2: w = 1.8766595518586393, b = 2.9951391575481985, Loss: 11.268839534158097
Iteration 3: w = 2.2126110808408965, b = 3.609176844332731, Loss: 4.410551141648595
Iteration 4: w = 2.4242851079732537, b = 3.9891144230280386, Loss: 1.7628380895727747
Iteration 5: w = 2.5586874546266847, b = 4.223637135085599, Loss: 0.7404649735472415
Iteration 6: w = 2.6450238753824715, b = 4.367844285047856, Loss: 0.3455030978297147
Iteration 7: w = 2.7014409652423144, b = 4.455968331539203, Loss: 0.1927409416804739
Iteration 8: w = 2.7392142687615486, b = 4.509277226238958, Loss: 0.13348230510669448
Iteration 9: w = 2.765350342154707, b = 4.540984116956181, Loss: 0.11032841895853565
Iteration 10: w = 2.7842034083667646, b = 4.559297733559353, Loss: 0.10112230491238644
Iteration 11: w = 2.7984797812260327, b = 4.569317584849083, Loss: 0.09731070565769012
Iteration 12: w = 2.8098619120602737, b = 4.574211439382292, Loss: 0.09559156072347656
Iteration 13: w = 2.819396108698506, b = 4.575946891126654, Loss: 0.09468984326866975
Iteration 14: w = 2.8277334247128514, b = 4.57574587914693, Loss: 0.09411299839527479
Iteration 15: w = 2.835279323692193, b = 4.574367079332992, Loss: 0.09367052858253345
Iteration 16: w = 2.842286665664944, b = 4.572281353464804, Loss: 0.09328854800449206
Iteration 17: w = 2.8489134797080955, b = 4.5697807550741185, Loss: 0.0929381979749591
Iteration 18: w = 2.8552588582949427, b = 4.567046254315231, Loss: 0.09260801401229425
Iteration 19: w = 2.861385258726788, b = 4.5641898150638625, Loss: 0.0922932587148741
Iteration 20: w = 2.867332359312922, b = 4.561280537123776, Loss: 0.09199180363904617
Iteration 21: w = 2.873125668529922, b = 4.558360898128674, Loss: 0.0917025391807935
Iteration 22: w = 2.878781874213274, b = 4.555456844413379, Loss: 0.09142476031234298
Iteration 23: w = 2.8843121673325087, b = 4.5525840602686936, Loss: 0.09115792902716317
Iteration 24: w = 2.889724307373519, b = 4.549751862839208, Loss: 0.09090158221574386
Iteration 25: w = 2.895023905878068, b = 4.546965621842478, Loss: 0.09065529569587819
Iteration 26: w = 2.9002152242199406, b = 4.5442282627668416, Loss: 0.09041866993520552
Iteration 27: w = 2.9053016695712794, b = 4.541541200640128, Loss: 0.09019132416259773
Iteration 28: w = 2.9102861033491014, b = 4.538904920016755, Loss: 0.08997289373252537
Iteration 29: w = 2.91517103315021, b = 4.536319335164387, Loss: 0.08976302875932544
Iteration 30: w = 2.9199587322918372, b = 4.533784013692265, Loss: 0.08956139325532401

2. 画出 50 次迭代求得的直线

迭代完成后的参数值：

Optimal w: 2.9980790466890217

Optimal b: 4.492384963650698

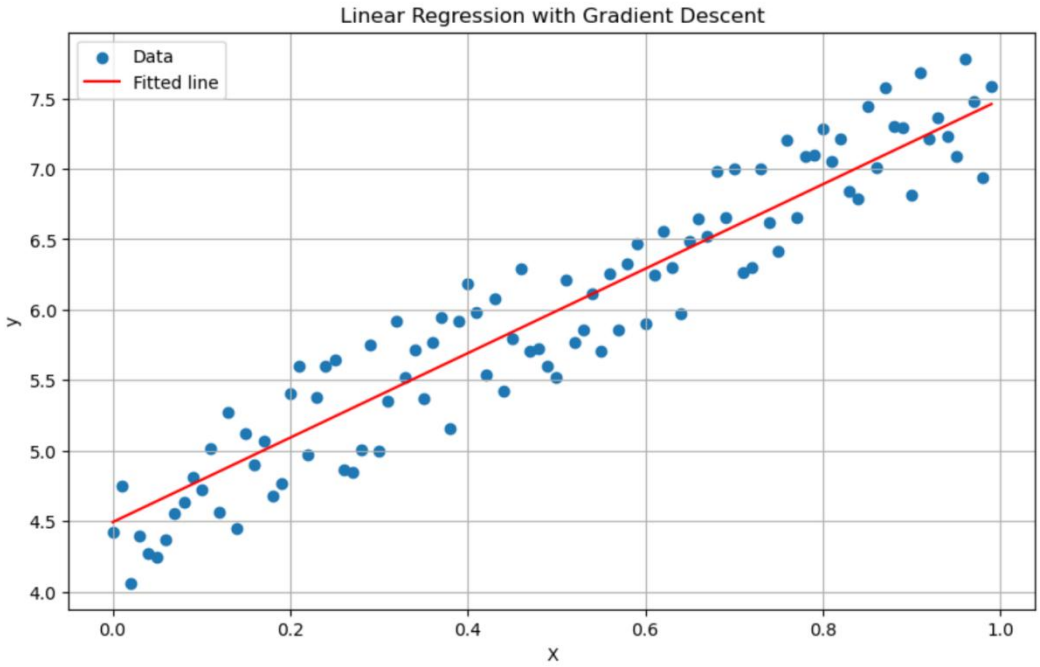

图 1 迭代所求得的直线

3. 画出 Loss 随迭代次数增加的变化曲线

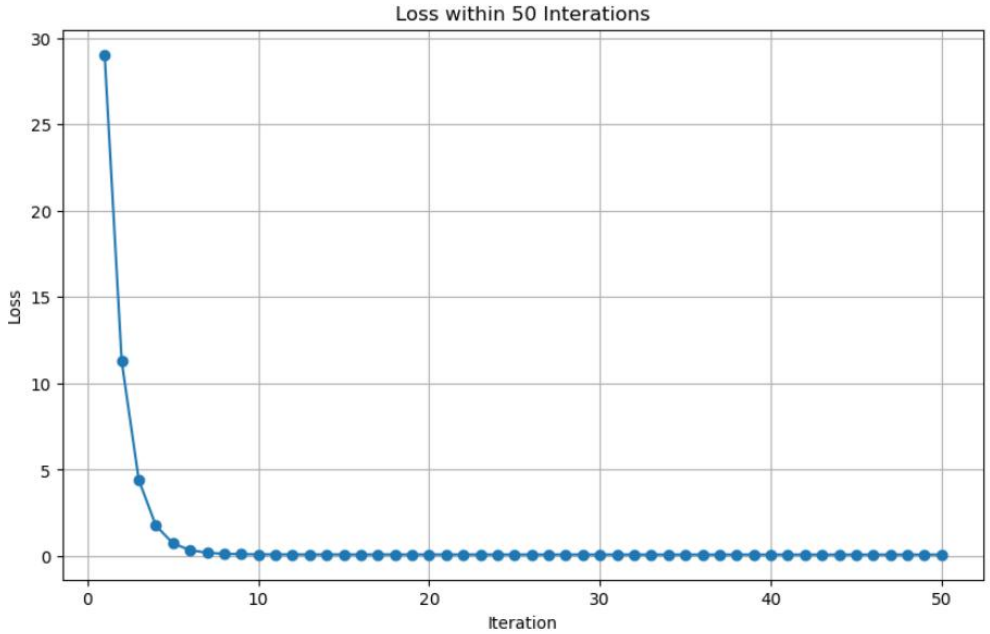

图 2 Loss 曲线

4. 代码

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

def compute_loss(y_pred, y_true):
    '''均方误差'''
    m = len(y_true)
    loss = np.sum(np.square(y_pred - y_true)) / m
    return loss

def gradient_descent(X, y, w, b, alpha, num_iters):
    '''梯度下降'''
    m = len(y)
    losses = []   # 存储每次迭代的损失值
    for i in range(num_iters):
        y_pred = np.dot(X, w) + b # 预测值

        dw = np.dot(X.T, (y_pred - y)) / m
        db = np.sum(y_pred - y) / m

        w -= alpha * dw
        b -= alpha * db

        loss = compute_loss(y_pred, y)
        losses.append(loss)

        # 输出前 30 次迭代的参数和损失
        if i < 30:
            print(f'Iteration {i+1}: w = {w.item()}, b = {b}, Loss: {loss}')

    return w, b, losses

x_data = np.loadtxt(r'C:\Users\JeffSong\Desktop\data_x.txt')
y_data = np.loadtxt(r'C:\Users\JeffSong\Desktop\data_y.txt')
X = x_data.reshape(-1, 1)
y = y_data.reshape(-1, 1)

# 初始化
w = np.random.rand(X.shape[1], 1)
b = np.random.rand()
# 定义超参数
alpha = 0.3
num_iters = 50
```

```
w_opt, b_opt, losses = gradient_descent(X, y, w, b, alpha, num_iters)

print("\nOptimal w:", w_opt.item())
print("Optimal b:", b_opt)

# 绘制数据可视化的视图
plt.figure(figsize=(10, 6))
plt.scatter(X, y, label='Data')
plt.plot(X, X*w_opt + b_opt, color='red', label='Fitted line')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression with Gradient Descent')
plt.legend()
plt.grid(True)
plt.show()

# 绘制损失随迭代次数增加的变化曲线
plt.figure(figsize=(10, 6))
plt.plot(range(1, num_iters + 1), losses, marker='o')
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.title('Loss within 50 Interations')
plt.grid(True)
plt.show()
```