山东大学 计算机科学与技术 学院

机器学习（双语） 课程实验报告

学号：201705130113	姓名：黄瑞哲	班级：计科 17.3
实验题目：线性回归		
实验学时：2	实验日期： 2019.9.13	

实验目的：

一、掌握梯度下降算法

二、理解损失函数与梯度下降算法之间的关系

三、能够利用梯度下降算法对多元线性回归进行拟合。

硬件环境：

Intel Core i5-8300H @ 2.3GHz

软件环境：

Windows10 Pro 1903

Python 3.7

Visual Studio Code 1.38.1

实验步骤与内容：

一、 二维线性回归

1. 从文件 ex1_1x.dat 和 文件 ex1_1y.dat 中读取 xy 的值，并绘制散点图。
2. 利用梯度下降算法求出最优的参数 theta

$$h_\theta(x) = \theta^T x = \sum_{i=0}^{1} \theta_i x_i = \theta_1 x_1 + \theta_2,$$

3. 绘制拟合后的直线。并且预测年龄为 3.5 和 7 时的身高。

二、理解损失函数

1. 在[-3, 3][-1, 1]之间绘制与 theta 相关的二元函数图像。

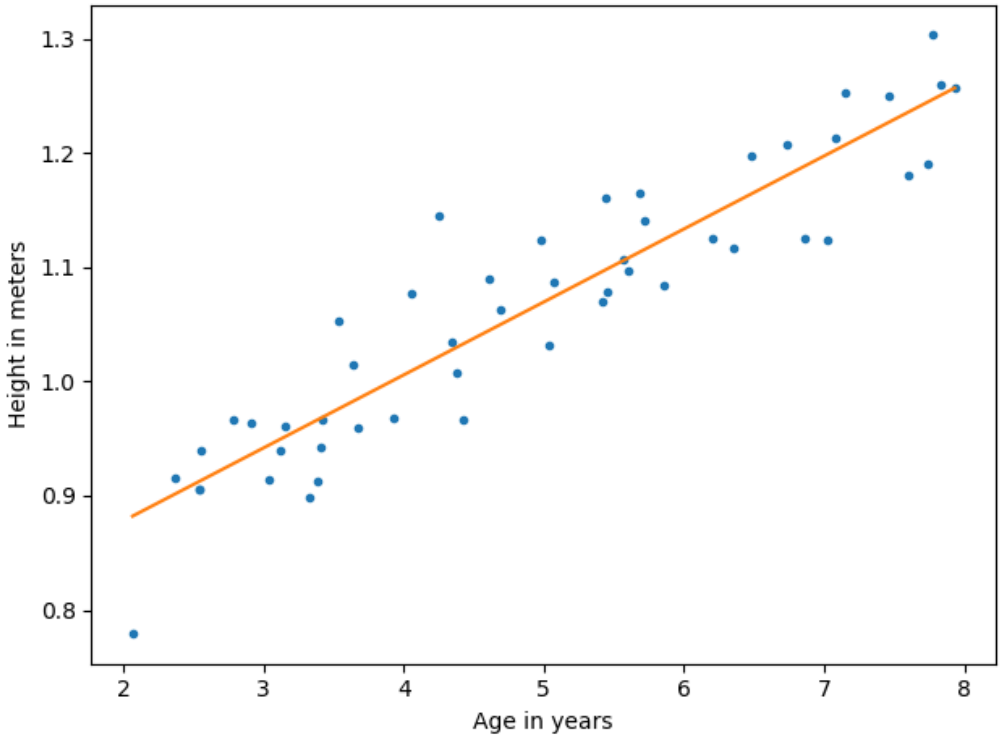
$$J(\theta) = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y})$$

三、多元线性回归

1. 从文件 ex1_2x.dat 和 文件 ex1_2y.dat 中读取 xy 的值
2. 对 x 做标准化
3. 手动选择学习率满足梯度下降算法迭代 50 次后结果收敛并且绘制每次迭代损失函数的图像。学习率的选择从 0.01 开始，每次增加 3 倍进行尝试。
4. 求出对应的 theta 值并对 x1=1650 和 x2=3 时做出预测。

结论分析与体会：

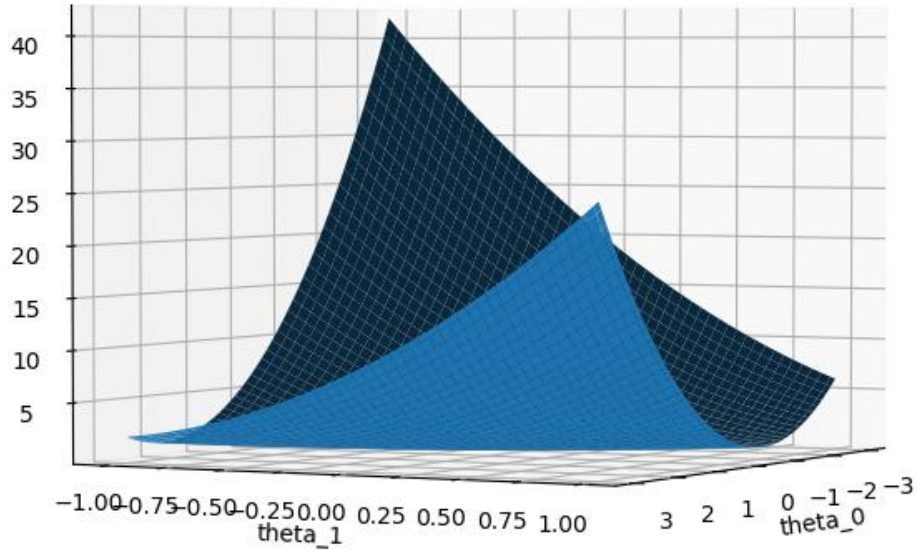
第一问的散点图与拟合曲线



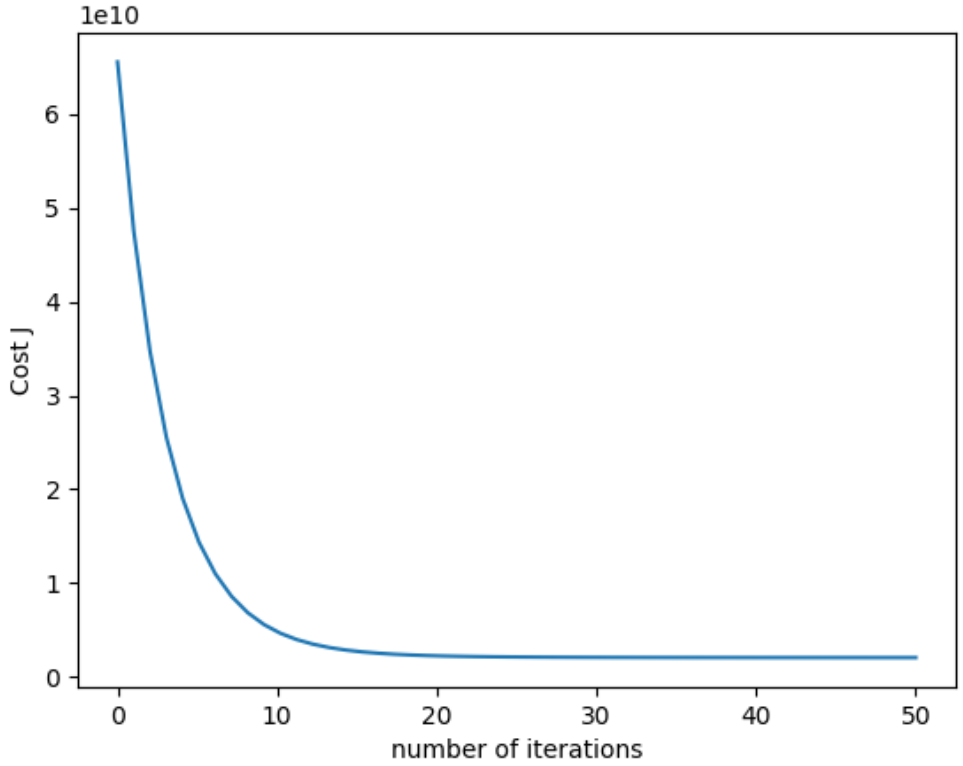
预测的身高为

```
Theta = [[0.75015039 0.06388338]]
prediction[3.5] = [0.97374221]
prediction[7.0] = [1.19733402]
```

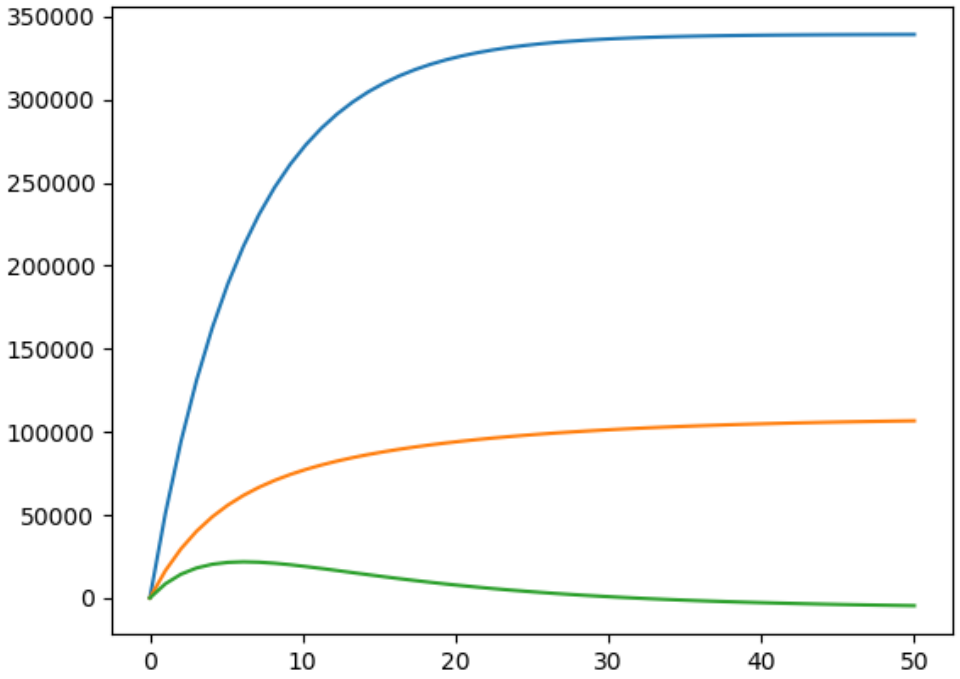
第二问绘制的三维图



当学习率为 0.15 时第三问绘制的损失函数图像



以及相应的 theta 值



对 x1=1650 和 x2=3 的预测

```
Theta = [[339265.63439555 106598.63547516  -4587.40048299]]
prediction[1650, 3] = [293338.84268821]
```

对于第二问的图像，可以看出损失函数是个凸函数，梯度下降求出的 theta 对应图像中的凹坑位置，调整学习率可以求得最小值。但是当学习率过小时，虽然求得的解是精确的，但是需要的迭代次数过多，耗时大，当最大迭代次数较小时，求出的结果是不正确的。当学习率过大时会导致结果不收敛。

附录：程序源代码

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# 标准化 dt
def scale(dt):
    mu = np.mean(dt, axis=0)
    sigma = np.std(dt, axis=0)
    return (dt - mu) / sigma


# 计算损失函数
def j_val(X, Y, theta):
    hx = np.dot(X, theta)
    return np.dot((hx - Y).transpose(), hx - Y) / (2 * X.shape[0])


# 梯度下降
def gradient_descent(X, Y, theta, learning_rate=0.07, max_iteration=1500):
    # g = ((X0 - Y)' * (X))'
    cost = np.zeros((max_iteration, 1))
    _theta = np.zeros((max_iteration, theta.shape[0]))
    for i in range(max_iteration):
        cost[i] = j_val(X, Y, theta)
        _theta[i] = theta.transpose()
        hx = np.dot(X, theta)
        theta = theta - (learning_rate / X.shape[0]) * (np.dot((hx -
Y).transpose(), X)).transpose()
    return theta, cost, _theta


def task5():
    x = np.loadtxt('exp1/data/ex1_2x.dat')
    y = np.loadtxt('exp1/data/ex1_2y.dat').reshape(-1, 1)
```

```
    # 将x标准化
    x = np.vstack((x, [[1650, 3]]))
    x = scale(x)
    x = np.hstack((np.ones((x.shape[0], 1)), x))

    n = x.shape[1]
    max_iteration = 50
    Theta, cost, _theta = gradient_descent(x[0: x.shape[0] - 1], y, np.zeros((n,
1)),
                                           learning_rate=0.15,
max_iteration=max_iteration)
    print("Theta =", Theta.transpose())

    print("prediction[1650, 3] =", np.dot(Theta.transpose(), x[x.shape[0] -
1].transpose()))

    # 每次迭代损失函数的值
    plt.figure('cost')
    plt.plot(np.linspace(0, max_iteration, max_iteration), cost)
    plt.xlabel('number of iterations')
    plt.ylabel('Cost J')
    plt.show()

    # 每次迭代的theta
    plt.figure('theta')
    plt.plot(np.linspace(0, max_iteration, max_iteration), _theta[:, 0])     #
theta_0
    plt.plot(np.linspace(0, max_iteration, max_iteration), _theta[:, 1])     #
theta_1
    plt.plot(np.linspace(0, max_iteration, max_iteration), _theta[:, 2])     #
theta_2
    plt.show()


if __name__ == "__main__":
    x = np.loadtxt('exp1/data/ex1_1x.dat').reshape(-1, 1)
    y = np.loadtxt('exp1/data/ex1_1y.dat').reshape(-1, 1)

    x = np.hstack((np.ones((x.shape[0], 1)), x))

    m, n = x.shape
    Theta, cost, _theta = gradient_descent(x, y, np.zeros((n, 1)))
    print("Theta =", Theta.transpose())
```

```
    # 对 x=3.5 和 x=7 做预测
    y1 = np.dot([1, 3.5], Theta)
    y2 = np.dot([1, 7], Theta)
    print("prediction[3.5] =", y1)
    print("prediction[7.0] =", y2)

    # 散点图与拟合直线
    plt.figure('Linear Regression')
    plt.plot(x[:, 1], y, '.')   # 散点
    plt.plot(x[:, 1], np.dot(x, Theta))  # 直线
    plt.xlabel('Age in years')
    plt.ylabel('Height in meters')
    plt.show()

    # 损失函数可视化
    fig2 = plt.figure('J value')
    theta_0 = np.linspace(-3, 3, 100)
    theta_1 = np.linspace(-1, 1, 100)
    j_vals = np.zeros((100, 100))
    for i in range(100):
        for j in range(100):
            j_vals[i][j]      =      j_val(x,      y,      np.array([theta_0[i],
theta_1[j]]).reshape(-1, 1))
    theta_0, theta_1 = np.meshgrid(theta_0, theta_1)

    # 三维图
    Axes3D(fig2).plot_surface(theta_0, theta_1, j_vals)
    # 等高线
    # plt.contour(theta_0, theta_1, j_vals)
    plt.xlabel('theta_0')
    plt.ylabel('theta_1')
    plt.show()

    task5()
```