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：   1. 掌握梯度下降算法 2. 理解损失函数与梯度下降算法之间的关系 3. 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：   1. 二维线性回归 2. 从文件ex1\_1x.dat 和 文件ex1\_1y.dat中读取xy的值，并绘制散点图。 3. 利用梯度下降算法求出最优的参数theta      1. 绘制拟合后的直线。并且预测年龄为3.5和7时的身高。 2. 理解损失函数 3. 在[-3, 3][-1, 1]之间绘制与theta相关的二元函数图像。      1. 多元线性回归 2. 从文件ex1\_2x.dat 和 文件ex1\_2y.dat中读取xy的值 3. 对x做标准化 4. 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。 5. 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    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    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 = ((XO - 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()*