|  |
| --- |
|  |
| **实验报告** |
| **课程名称： 机器学习**  **学　　院： 电子科技大学（深圳）高等研究院**  **专　　业：电子信息**  **指导教师：张栗粽**  **学生姓名： 刘文晨**  **学　　号：202222280328** |
|  |
|  |

**电 子 科 技 大 学**

**实 验 报 告**

# 实验一

## 实验项目名称

线性回归

## 二、实验学时：4学时

## 三、实验目的

1. 掌握线性回归的基本原理；
2. 掌握线性回归的求解方法；
3. 掌握梯度下降法原理；
4. 掌握最小二乘法。

## 实验原理

1. 线性回归

线性回归的任务是找到一个从输入特征空间到输出特征空间的最优的线性映射函数简单来说给定个属性描述的示例，其中表示在第个属性上的取值。线性模型试图学到通过属性的线性组合来进行预测的函数，即：

写成向量形式：

其中，，表示的转置。

我们可以使用均方误差确定和，均方误差是回归任务中最常用的性能度量，我们试图通过均方误差最小来求解和，即：

我们称上式为代价函数或损失函数，我们只需要使代价函数最小即可。

1. 梯度下降法

函数沿着导数方向是变化最快的，为了更快的达到优化目标，沿着负梯度方向搜寻，使得代价函数最小，即使用梯度下降法更新权重即可求出和。

梯度就是多元函数的偏导数，我们求出代价函数对，的偏导数，即：

沿着负梯度方向搜寻，使得代价函数最小，即使用梯度下降法更新权重：

其中，为学习率，随机初始化，，通过不断地迭代上述2个更新公式计算，的最优值。

1. 最小二乘法

使用最小二乘法求解，，即令代价函数对，的偏导数为0，这种基于均方误差进行线性模型求解的方法称为最小二乘法。

，的计算公式为：

可以看出最小二乘法是梯度下降法的一种特殊情况，很多函数解析不出导数等于零的点，梯度下降法是求解损失函数参数更常用的方法。

## 实验内容与要求

1. 准备数据，np.random,rand()产生一组随机数据x，根据y=wx+b，产生数据，并用np.random.rand()添加随机噪声，y=wx+b+噪声，得到数据集(x,y)；
2. 建立线性模型，y\_pre=wx+b；
3. 采用均方误差，构建损失函数；
4. 训练模型，梯度下降法进行优化权重求解w和b；
5. 直接使用最小二乘法求解w和b，并与梯度下降法求解的w和b进行比较；
6. 绘制样本点，预测直线。

## 实验器材（设备、元器件）

处理器：Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz

Python 3.9.0

matplotlib 3.6.1

numpy 1.23.4

scipy 1.9.3

## 实验步骤

1. 随机生成数据集

使用np.arange()产生0到10、步长为0.2的数据x，再生成与x相同长度的全1向量，两者行叠加再转置得到，记为input\_data。设置w和b，令y=wx+b+random噪声，记为target\_data。input\_data和target\_data组成数据集。代码如下：

1. # 构造训练数据
2. x = np.arange(0., 10., 0.2)
3. m = len(x)
4. x0 = np.full(m, 1.0)
5. input\_data = np.vstack([x0, x]).T
6. w = 2
7. b = 5
8. target\_data = w \* x + b + np.random.randn(m)
9. 梯度下降法

设学习率η=0.001，随机初始化w和b，设向量为theta。沿着负梯度方向搜寻，使得代价函数最小。在每次循环中，更新，的值，并打印。当循环次数超过设定最大次数或w和b达到收敛条件时，退出循环。代码如下：

1. # 终止条件
2. loop\_max = 1e4  # 最大迭代次数
3. epsilon = 1e-3  # 收敛条件最小值
5. # 初始化权值
6. np.random.seed(0)
7. theta = np.random.randn(2)
8. alpha = 1e-3  # 步长，也叫学习率
9. diff = 0.
10. error = np.zeros(2)
11. count = 0  # 循环次数
12. finish = 0  # 终止标志
14. # 迭代
15. **while** count < loop\_max:
16. count += 1
17. # 在标准梯度下降中，权值更新的每一步对多个样例求和，需要更多的计算
18. sum\_m = np.zeros(2)
19. **for** i **in** range(m):
20. dif = (np.dot(theta, input\_data[i]) - target\_data[i]) \* input\_data[i]
21. # 当alpha取值过大时,sum\_m会在迭代过程中会溢出
22. sum\_m = sum\_m + dif
23. # 注意步长alpha的取值,过大会导致振荡
24. theta = theta - alpha \* sum\_m
25. # 判断是否已收敛
26. **if** np.linalg.norm(theta - error) < epsilon:
27. finish = 1
28. **break**
29. **else**:
30. error = theta
31. # 打印迭代次数、更新后的w和b
32. **print**('迭代次数 = %d' % count, '\t w:', theta[1], '\t b:', theta[0])
34. **print**('迭代次数 = %d' % count, '\t w:', theta[1], '\t b:', theta[0])
35. 最小二乘法

使用Python第三方库——scipy中的统计模块——stats中的linregress()计算两组测量值x和target\_data的线性最小二乘回归。代码如下：

1. # 用scipy线性最小二乘回归进行检查
2. slope, intercept, r\_value, p\_value, slope\_std\_error = stats.linregress(x, target\_data)
3. **print**('使用最小二乘法计算，斜率 = %s 截距 = %s' % (slope, intercept))
4. 绘图

使用Python第三方库——matplotlib中的plot()进行绘图。样本点用蓝色星形点表示，梯度下降法得到的预测直线用红色实线表示，而最小二乘法得到的预测直线用绿色实线表示。代码如下：

1. # 用plot进行展示
2. plt.scatter(x, target\_data, color='b', marker='\*')
3. # 梯度下降法
4. plt.plot(x, theta[1] \* x + theta[0], label='gradient descent', color='red')
5. # 最小二乘法
6. plt.plot(x, slope \* x + intercept, label='least square', color='green')
7. plt.xlabel('x')
8. plt.ylabel('y')
9. plt.legend()
10. plt.title('Experiment 1: Linear regression')
11. plt.savefig('result.png')
12. plt.show()
13. 实验结果对比

编写好代码后，运行代码。运行的部分结果如图1所示。

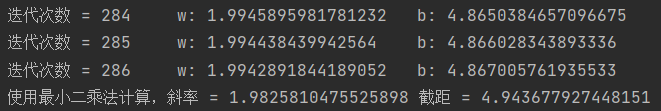


图1 程序运行部分结果

实验生成的result.png如图2所示。其中，蓝色的星形点是样本点，红色实线是由梯度下降法得到的预测直线，而绿色实线是由最小二乘法得到的预测直线。

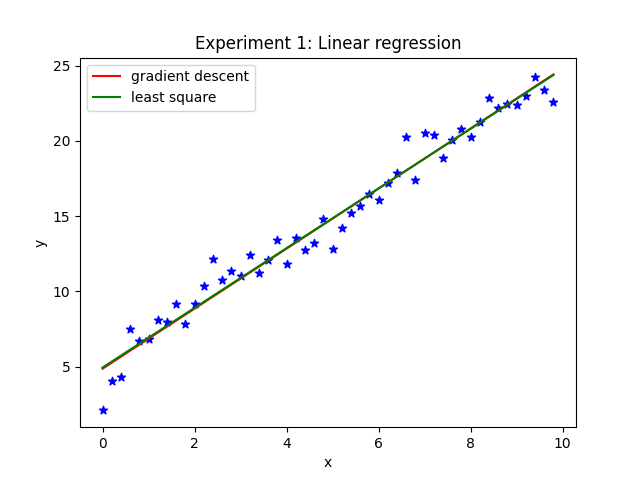


图2 result.png

在本次实验中，预设的斜率w=2，截距b=5。从图1中我们可以看出，由梯度下降法计算得到的w=1.9942891844189052，b=4.867005761935533；而由最小二乘法计算得到的w=1.9825810475525898，b=4.943677927448151。可以看出，无论是梯度下降法还是最小二乘法，都非常逼近w和b的预设值。从图2中我们也可以看出，梯度下降法和最小二乘法几乎重合，都很好地实现了对样本点的拟合。

## 心得体会

本实验实现了梯度下降法进行线性回归，并通过最小二乘法验证了结果的正确性。通过此次实验，很好地掌握了线性回归、梯度下降法和最小二乘法的原理，熟悉了Python第三方库——numpy、matplotlib和scipy的一些简单函数的使用。