电子科技大学

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

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。

四、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X到输出特征空间Y的最优的线性映射函数简单来说给定d个属性描述的示例$x=(x_1,x_2,\ldots,x_d)$，其中x_i表示x在第i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：

$$f(x)=w_1x_1+w_2x_2+\cdots+w_dx_d+b$$

   写成向量形式：

$$f(x)=w^Tx+b=(1,x^T)\begin{pmatrix}b\\w\end{pmatrix}$$

   其中，$w=(w_1,w_2,\ldots,w_d)$，w^T表示w的转置。

   我们可以使用均方误差确定w和b，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w和b，即：

$$(w^*, b^*) = \underset{(w,b)}{\arg\min} \frac{1}{2m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 = \underset{(w,b)}{\arg\min} \frac{1}{2m}\sum_{i=1}^{m}(wx_i + b - y_i)^2$$

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。

2. 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w，b使得代价函数最小，即使用梯度下降法更新权重即可求出w和b。

梯度就是多元函数的偏导数，我们求出代价函数对w，b的偏导数，即：

$$\frac{\partial}{\partial w} = \frac{1}{m}\sum_{i=1}^{m}(wx_i + b - y_i)x_i$$

$$\frac{\partial}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(wx_i + b - y_i)$$

沿着负梯度方向搜寻w，b使得代价函数最小，即使用梯度下降法更新权重：

$$w^* = w - \eta\frac{\partial}{\partial w} = w - \eta\frac{1}{m}\sum_{i=1}^{m}(wx_i + b - y_i)x_i$$

$$b^* = b - \eta\frac{\partial}{\partial b} = b - \eta\frac{1}{m}\sum_{i=1}^{m}(wx_i + b - y_i)$$

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

3. 最小二乘法

使用最小二乘法求解w，b，即令代价函数对w，b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。

w，b的计算公式为：

$$w = \frac{\sum_{i=1}^{m} x_i y_i - \frac{1}{m}\sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}(\sum_{i=1}^{m} x_i)^2}$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$$

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。

五、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y，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。

六、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

七、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$(1,w^T)$，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：

```python
# 构造训练数据
x = np.arange(0., 10., 0.2)
m = len(x)
x0 = np.full(m, 1.0)
input_data = np.vstack([x0, x]).T
w = 2
b = 5
target_data = w * x + b + np.random.randn(m)
```

2. 梯度下降法

   设学习率 η=0.001，随机初始化 w 和 b，设$\binom{b}{w}$向量为 theta。沿着负梯度方向搜寻w，b

使得代价函数最小。在每次循环中，更新w，b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：

```
# 终止条件
loop_max = 1e4  # 最大迭代次数
epsilon = 1e-3  # 收敛条件最小值

# 初始化权值
np.random.seed(0)
theta = np.random.randn(2)
alpha = 1e-3  # 步长，也叫学习率
diff = 0.
error = np.zeros(2)
count = 0  # 循环次数
finish = 0  # 终止标志

# 迭代
while count < loop_max:
    count += 1
    # 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
    sum_m = np.zeros(2)
    for i in range(m):
        dif = (np.dot(theta, input_data[i]) - target_data[i]) * input_data[i]
        # 当 alpha 取值过大时,sum_m 会在迭代过程中会溢出
        sum_m = sum_m + dif
    # 注意步长 alpha 的取值,过大会导致振荡
    theta = theta - alpha * sum_m
    # 判断是否已收敛
    if np.linalg.norm(theta - error) < epsilon:
        finish = 1
        break
    else:
        error = theta
    # 打印迭代次数、更新后的 w 和 b
    print('迭代次数 = %d' % count, '\t w:', theta[1], '\t b:', theta[0])

print('迭代次数 = %d' % count, '\t w:', theta[1], '\t b:', theta[0])
```

3. 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：

```
# 用 scipy 线性最小二乘回归进行检查
slope, intercept, r_value, p_value, slope_std_error = stats.linregress(x, target_data)
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：

```
# 用 plot 进行展示
plt.scatter(x, target_data, color='b', marker='*')
# 梯度下降法
plt.plot(x, theta[1] * x + theta[0], label='gradient descent', color='red')
# 最小二乘法
plt.plot(x, slope * x + intercept, label='least square', color='green')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('Experiment 1: Linear regression')
plt.savefig('result.png')
plt.show()
```

5. 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。

```
迭代次数 = 284     w: 1.9945895981781232    b: 4.8650384657096675
迭代次数 = 285     w: 1.994438439942564     b: 4.866028343893336
迭代次数 = 286     w: 1.9942891844189052    b: 4.867005761935533
使用最小二乘法计算，斜率 = 1.9825810475525898 截距 = 4.943677927448151
```

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。

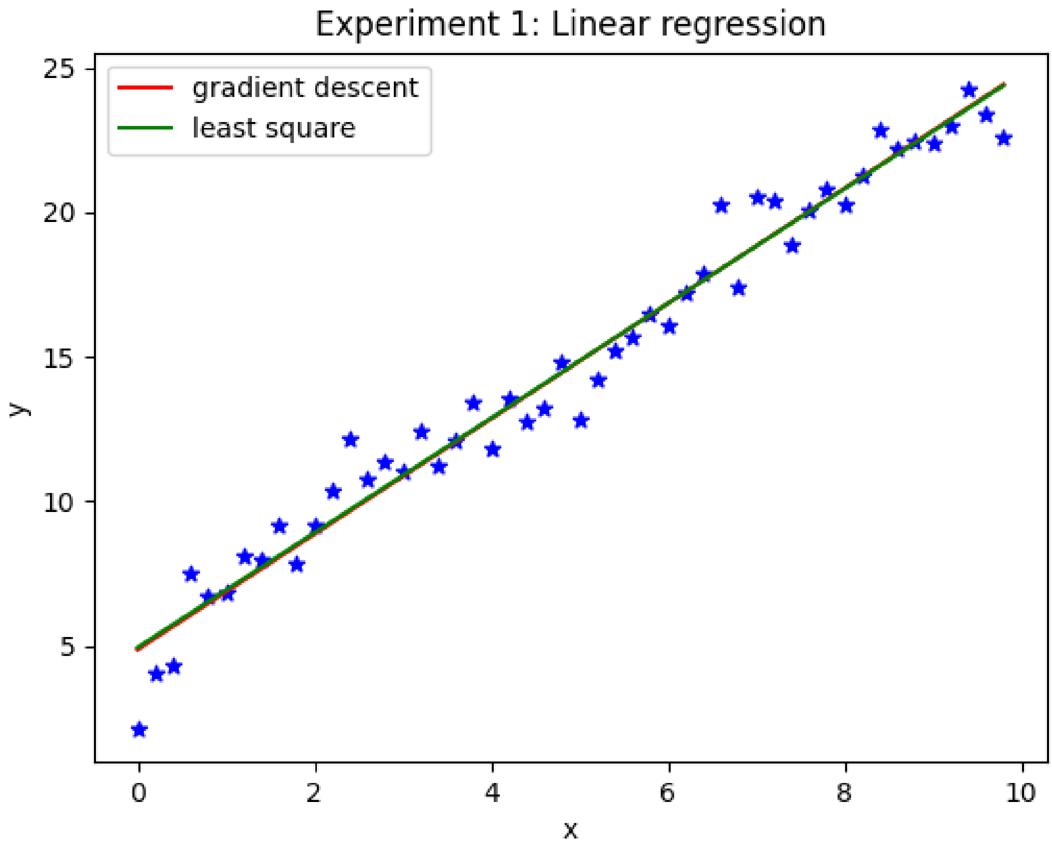

图 2 result.png

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。

八、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。