第 1 章作业

1. 作业题目内容

使用以下商品房销售记录表数据，用梯度下降法，编程实现一个房价预测系统。

商品房销售记录

序号	面积（平方米）	房间数	销售价格（万元）	序号	面积（平方米）	房间数	销售价格（万元）
1	137.97	3	145.00	9	106.69	2	62.00
2	104.50	2	110.00	10	138.05	3	133.00
3	100.00	2	93.00	11	53.75	1	51.00
4	124.32	3	116.00	12	46.91	1	45.00
5	79.20	1	65.32	13	68.00	1	78.50
6	99.00	2	104.00	14	63.02	1	69.65
7	124.00	3	118.00	15	81.26	2	75.69
8	114.00	2	91.00	16	86.21	2	95.30
	x_1	x_2	y		x_1	x_2	y

2. 求解原理

数据：

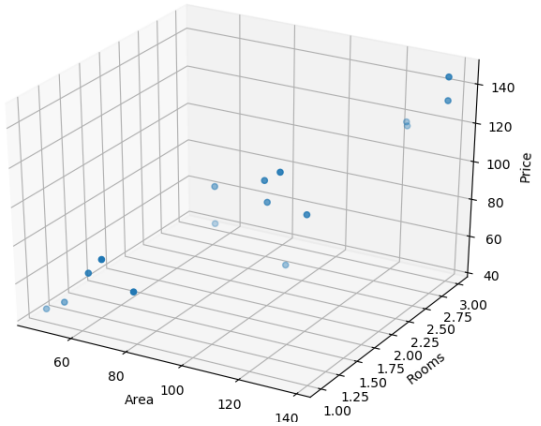
$x_1 = [137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00, 106.69, 138.05, 53.75, 46.91, 68.00, 63.02, 81.26, 81.21]^T$

$x_2 = [3.00, 2.00, 2.00, 3.00, 1.00, 2.00, 3.00, 2.00, 2.00, 3.00, 1.00, 1.00, 1.00, 1.00, 2.00, 2.00]^T$

$y = [145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00, 62.00, 133.00, 51.00, 45.00, 78.50, 69.65, 75.69, 95.30]^T$

维度:（16，3）

数据可视化：



[图 1.商品房面积、房间数和价格的关系]

目的：

根据已有的数据，对数据进行拟合，实现对房价的预测

求解方法：

对于二元线性回归问题，可以设置假设函数，并通过损失函数进行参数的优化，达到对样本数据的拟合

1、假设函数设置为：

$\hat{y}(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$ (w_1, w_2,为超参数，b为偏置,$\hat{y}$为预测房价)

2、损失函数设置为：

$\mathrm{L}(w_1, w_2, \mathrm{b}) = \frac{1}{n}\sum_{i=1}^{n}(w_1 x_1 + w_2 x_2 + b - y)^2$ (n 为样本总量，y 为样本的价格)

3、采用梯度下降法最小化损失函数，对参数w_1, w_2, b进行优化

梯度下降法各参数的更新表达式：

$$w_1 = w_1 - \eta\frac{\partial L}{\partial w_1} = \frac{2}{n}\sum_{i=1}^{n}(w_1 x_1 + w_2 x_2 + b - y) * x_1$$

$$w_2 = w_2 - \eta\frac{\partial L}{\partial w_2} = \frac{2}{n}\sum_{i=1}^{n}(w_1 x_1 + w_2 x_2 + b - y) * x_2$$

$$\mathrm{b} = w_2 - \eta\frac{\partial L}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(w_1 x_1 + w_2 x_2 + b - y)$$

即：

多次迭代更新参数后的函数

$$\hat{y}(x_1, x_2) = w_1 x_1 + w_2 x_2 + b(x_1, x_2 \in R)$$

为预测房价的函数

3. 编程求解

3.1. 编程实验平台说明

开发平台：

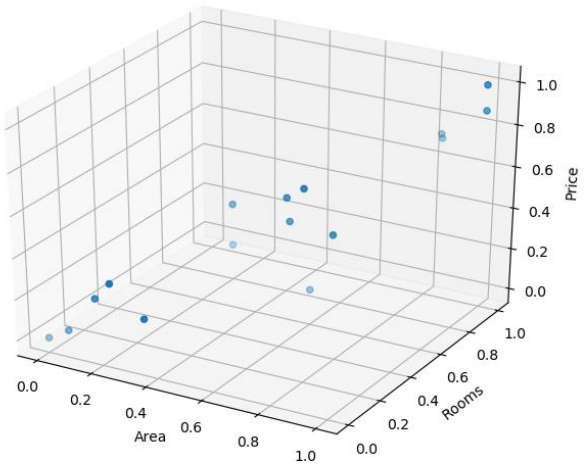
Linux version 5.4.0-40-generic (buildd@lcy01-amd64-011) (gcc version 9.3.0 (Ubuntu 9.3.0-10ubuntu2))

Python 版本：3.7.6

开发工具：Pycharm+Xshell+Xming

3.2. 实验方案

1、对数据进行归一化处理



[图 2.归一化数据后的函数图像（分布不改变）]

2、定义期望函数（Loss Function）求 Loss 值

3、定义梯度下降函数，对参数进行优化（求最小值）

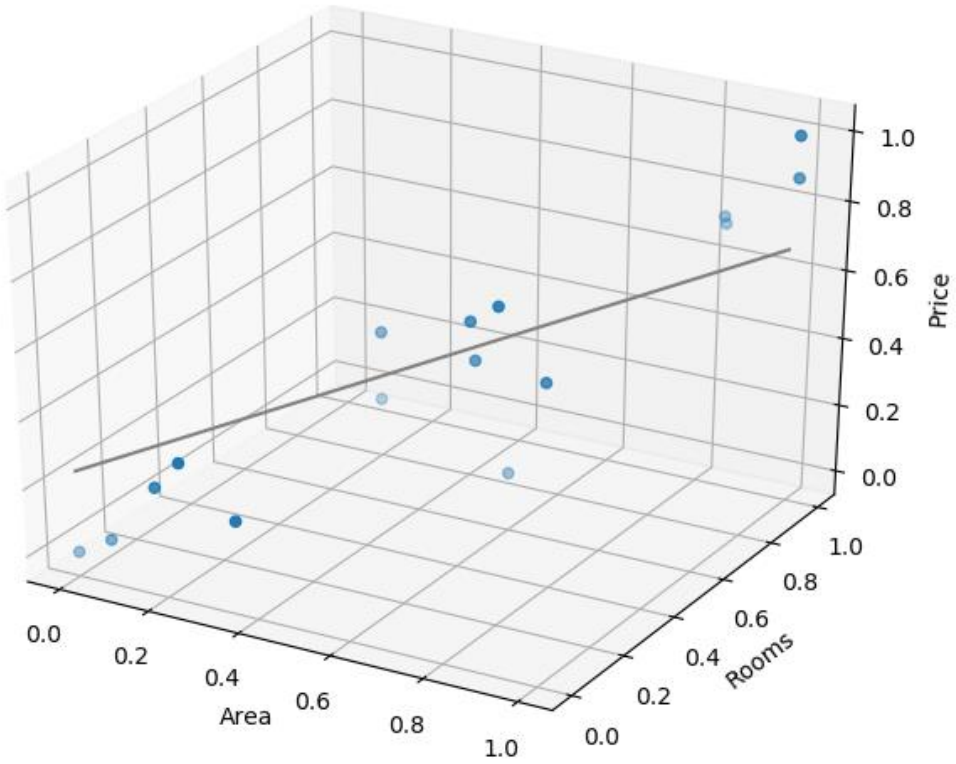
3.3. 程序及运行结果说明

```
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import MinMaxScaler

#归一化处理数据
mm = MinMaxScaler()
x_1 =
mm.fit_transform(np.array([137.97,104.50,100.00,124.32,79.20,99.00,12
4.00,114.00,106.69,138.05,53.75,46.91,68.00,63.02,81.26,81.21]).resha
pe(16,-1))
x_2 =
mm.fit_transform(np.array([3.00,2.00,2.00,3.00,1.00,2.00,3.00,2.00,2.
00,3.00,1.00,1.00,1.00,1.00,2.00,2.00]).reshape(16,-1))
y =
mm.fit_transform(np.array([145.00,110.00,93.00,116.00,65.32,104.00,11
8.00,91.00,62.00,133.00,51.00,45.00,78.50,69.65,75.69,95.30]).reshape
(16,-1))

#未归一化的数据
# x_1 =
```

```
np.array([137.97,104.50,100.00,124.32,79.20,99.00,124.00,114.00,106.6
9,138.05,53.75,46.91,68.00,63.02,81.26,81.21])
# x_2 =
np.array([3.00,2.00,2.00,3.00,1.00,2.00,3.00,2.00,2.00,3.00,1.00,1.00
,1.00,1.00,2.00,2.00])
# y =
np.array([145.00,110.00,93.00,116.00,65.32,104.00,118.00,91.00,62.00,
133.00,51.00,45.00,78.50,69.65,75.69,95.30])

#对数据进行可视化
fig = plt.figure()
ax = Axes3D(fig)
ax.set_xlabel("Area")
ax.set_ylabel("Rooms")
ax.set_zlabel("Price")
# plt.show()


#Loss calculate
data = np.stack((x_1,x_2,y),axis=1) #转为3D数组
# print("data=",data)
# print(data.shape)
# print(data.size)
# print(len(data))

#print function
def Print_Feature(data):
    for i in range(0,len(data)):
        x_1 = data[i,0]
        x_2 = data[i,1]
        y = data[i,2]
        print(x_1,x_2,y)

#Loss Function
def mse(b,w1,w2,data):
    totalError = 0
    for i in range(0,len(data)):
        x_1 = data[i,0]
        x_2 = data[i,1]
        y = data[i,2]
        totalError +=((w1*x_1+w2*x_2+b)-y)**2
    return totalError/float((len(data))*2)

#定义一步梯度下降运算
```

```
def step_gradient(b_current,w1_current,w2_current,data,lr):
    b_gradient = 0
    w1_gradient = 0
    w2_gradient = 0
    M = 16

    for i in range(0,len(data)):
        x_1 = data[i,0]
        x_2 = data[i,1]
        y = data[i,2]

        w1_gradient +=
(2/M)*x_1*((w1_current*x_1+w2_current*x_2+b_current)-y)
        w2_gradient +=
(2/M)*x_2*((w1_current*x_1+w2_current*x_2+b_current)-y)
        b_gradient +=
(2/M)*((w1_current*x_1+w2_current*x_2+b_current)-y)

    new_b = b_current-(lr*b_gradient)
    new_w1 = w1_current-(lr*w1_gradient)
    new_w2 = w2_current-(lr*w2_gradient)

    return [new_b,new_w1,new_w2]

#迭代num_iterations次梯度下降运算，优化参数w1，w2，b
def gradient_descent(data,star_b,star_w1,star_w2,lr,num_iterations):
    b = star_b
    w1 = star_w1
    w2 = star_w2
    for step in range(num_iterations):
        b,w1,w2 = step_gradient(b,w1,w2,data,lr)
        loss = mse(b,w1,w2,data)
        if step%5 == 0:

print(f"iterations:{step},loss:{loss},w1:{w1},w2:{w2},b:{b}")
    return [b,w1,w2]


def main():
    lr = 0.0001
    init_b = 0
    init_w1 = 0
    init_w2 = 0
    num_iterations = 10000
```

```
    Print_Feature(data)
    [b,w1,w2] =
gradient_descent(data,init_b,init_w1,init_w2,lr,num_iterations)
    loss = mse(b,w1,w2,data)
    print(f"Final Loss:{loss},w1:{w1},w2:{w2},b:{b}")
    xx = np.arange(0,1,0.01)
    yy = np.arange(0,1,0.01)
    Z = w1*xx+w2*yy+b
    ax.scatter(x_1, x_2, y)  # 绘制散点图
    ax.plot3D(xx,yy,Z,'gray') #绘制最后更新参数后的预测房价函数
    plt.show()
    # plt.savefig('result.png',bbox_inches='tight')
if __name__ == '__main__':
    main()
```



[图 3.拟合后的函数图像，灰色直线为预测房价的函数模型]

4. 讨论及收获

1、深刻理解线性回归问题

2、深刻理解梯度下降，并推到梯度下降算法

3、Python 函数库的运用，matplotlib