Lab1(一元线性回归-多元线性回归)

小组成员：计科 1802 张继伟（201808010829），

计科 1802 谢正宇（201808010824），

计科 1801 樊佳婷（201808010816），

计科 1801 刘怡聪（201808010813），

计科 1801 孙晶铭（201808010808）。

实验完成日期：2020 年 11 月 12 日

1. 实验描述

简要描述实验中所用算法的基本思想（包括调参参数的选择等），以及如何处理数据集。

原理及公式：

（1） 凸函数

什么是凸函数

对于一元函数 f(x)，如果对于任意 tϵ[0,1]均满足：f(tx1+(1−t)x2)≤tf(x1)+(1−t)f(x2)，则称 f(x)为凸函数(convex function)

如果对于任意 tϵ(0,1)均满足：f(tx1+(1−t)x2)<tf(x1)+(1−t)f(x2)，则称 f(x)为严格凸函数(convex function) 梯度下降法---求函数极值的数值解 一元凸函数 x^(k+1)=x^(k)- η df(x)/d(x) 二元凸函数 x^(k+1)=x^(k)- η ∂f(x,y)/∂x y^(k+1)=y^(k)- η ∂f(x,y)/∂y

（2） 线性回归

线性回归模型是

$h\theta(x) = \theta T x = j = 0 \sum n \theta j x j$

其中θ是我们需要优化的参数，x 是 n + 1 - 维特征向量。给定一个训练集,

$x(i) i = 1,\ldots,m,$

我们的目标是找出θ最佳值，使得目标函数 J (θ) 如图等式可以最小化

$J(\theta) = 2m1i = 1\sum m(h\theta - y(i))2$

优化方法之一是梯度下降算法。算法迭代执行，并在每次迭代中，我们更新θ遵循以下规则

$$\theta j := \theta j - \alpha m 1 i = 1 \sum m (h\theta(x(i)) - y(i)) x j(i)$$

其中α是所谓的学习率，基于我们可以调整收敛梯度下降。 x j ^(i)表示对应θ j 的系数。

(3) 梯度下降方法

梯度下降分析

a.先确定向下一步的步伐大小，我们称为 Learning rate (alpha)；

b.任意给定一个初始值：用 theta0 和 theta1 表示；

c.确定一个向下的方向，并向下走预先规定的步伐，并更新 theta0 和 theta1

d.当下降的高度小于某个定义的值，则停止下降；

特点

a.初始点不同，获得的最小值也不同，因此梯度下降求得的只是局部最小值；

b.越接近最小值时，下降速度越慢；

梯度下降能够求出一个函数的最小值；

线性回归需要使得 cost function 的最小

均值归一化处理(Mean normalization)

$$\frac{x - \bar{x}}{\max - \min}$$

2. 实验及结果分析

（1） 开发语言及运行环境；

开发语言：python

运行环境：python 可以使用 IDE 来编程，本次实验对 python 使用的 IDE 是 VScode

（2） 实验的具体步骤；

一元线性回归

过程分析：

1、加载样本数据 x,y 2、设置超参数学习率，迭代次数 3、设置模型参数初值 w0， b0 4、训练模型 w, b 5、结果可视化

1、加载样本数据 x,y

```
#设置字体
plt.rcParams['font.sans-serif'] =['SimHei']


def LoadFile(filename):
    data = np.loadtxt(filename, delimiter=',', unpack=True, usecols=(0,
 1))
    x = np.transpose(np.array(data[0]))
    y = np.transpose(np.array(data[1]))
    return x, y

#加载样本数据

if __name__ == '__main__':
    x, y = LoadFile('ex1data1.txt')
```

2、设置超参数学习率，迭代次数

```
learn_rate=0.01   #设置超参数,学习率
    iter=1500     #迭代次数
    display_step=50  #每 50 次迭代显示一下效果
```

3、设置模型参数初值 w0， b0

```
#初始化为0
    w=0
    b=0
```

4、训练模型 w, b

```
#训练模型
    mse=[] #存放每次迭代的损失值
    for i in range(0,iter+1):
        #求偏导
        dL_dw=np.mean(x*(w*x+b-y))
        dL_db=np.mean(w*x+b-y)
        #更新模型参数
        w=w-learn_rate*dL_dw
        b=b-learn_rate*dL_db
```

```
        #得到估计值
        pred=w*x+b
        #计算损失(均方误差)
        Loss=np.mean(np.square(y-pred))/2
        mse.append(Loss)
        #显示模型
        #plt.plot(x,pred)
        if i%display_step==0:
            print("i:%i,Loss:%f,w:%f,b:%f"%(i,mse[i],w,b))
        print("城市人口为 35000 时的预测餐车利润:%f"%(3.5*w+b))
    print("城市人口为 70000 时的预测餐车利润:%f"%(7*w+b))
```

5、结果可视化

```
#模型和数据可视化
    plt.figure(figsize=(20,4))
    plt.subplot(1,3,1)
    #绘制散点图
    #张量和数组都可以作为散点函数的输入提供点坐标
    plt.scatter(x,y,color="red",label="数据集")
    plt.scatter(x,pred,color="green",label="梯度下降法")
    plt.plot(x,pred,color="blue")

    #设置坐标轴的标签文字和字号
    plt.xlabel("城市人口（万人）",fontsize=14)
    plt.ylabel("餐车利润（万美元）",fontsize=14)

    #在左上方显示图例
    plt.legend(loc="upper left")

    #损失变化可视化
    plt.subplot(1,3,2)
    plt.plot(mse)
    plt.xlabel("迭代次数",fontsize=14)
    plt.ylabel("损失值",fontsize=14)
    #估计值与标签值比较可视化
    plt.subplot(1,3,3)
    plt.plot(y,color="red",marker="o",label="数据集")
    plt.plot(pred,color="blue",marker="o",label="预测利润")
    plt.legend()
    plt.xlabel("sample",fontsize=14)
    plt.ylabel("price",fontsize=14)
    #显示整个绘图
    plt.show()
```

多元线性回归

过程分析：

1、加载样本数据 area,room,price 以及数据处理归一化，X，Y 2、设置超参数学习率，迭代次数 3、设置模型参数初值 W0(w0,w1,w2) 4、训练模型 W 5、结果可视化

线性归一化：适用于样本分布均匀且集中的情况，如果最大值(或者最小值)不稳定，和绝大数样本数据相差较大，使用这种方法得到的结果也不稳定.为了抑制这个问题，在实际问题中可以用经验值来代替最大值和最小值 标准差归一化适用于样本近似正态分布，或者最大最小值未知的情况，有时当最大最小值处于孤立点时也可以使用标准差归一化 非线性映射归一化，通常用于数据分化较大的情况（有的很大有的很小） 总结：样本属性归一化需要根据属性样本分布规律定制

1、加载样本数据 area,room,price 以及数据处理归一化，X，Y

```
def LoadFile(filename):
    data = np.loadtxt(filename, delimiter=',', unpack=True, usecols=(0, 1, 2))
    x = np.transpose(np.array(data[0]))
    y = np.transpose(np.array(data[1]))
    z = np.transpose(np.array(data[2]))
    return x, y, z
#=======================【1】加载样本数据以及数据处理====================
==============
```

```
if __name__ == '__main__':
    area, room, price = LoadFile('ex1data2.txt')
    num=len(area) #样本数量
    x0=np.ones(num)
    #归一化处理，这里使用线性归一化
    x1=(area-np.average(area))/(area.max()-area.min())
    x2=(room-np.average(room))/(room.max()-room.min())
    #堆叠属性数组，构造属性矩阵
    #从(16,)到(16,3),因为新出现的轴是第二个轴所以axis为1
    X=np.stack((x0,x1,x2),axis=1)
    #print(X)
    #得到形状为一列的数组
    Y=price.reshape(-1,1)
    #print(Y)
```

2、设置超参数学习率，迭代次数

```
#========================【2】设置超参数====================================
============
learn_rate=0.001      #设置超参数
iter=1500     #迭代次数
display_step=50      #每50次迭代显示一下效果
```

3、设置模型参数初值 W0(w0,w1,w2)

```
#========================【3】设置模型参数初始值===========================
============
#设置模型参数初始值
W=[[0],
   [0],
   [0]]
```

4、训练模型 W

```
#========================【4】训练模型======================================
==========
mse=[]
for i in range(0,iter+1):
    #求偏导
    dL_dW=np.matmul(np.transpose(X),np.matmul(X,W)-Y)     #XT(XW-Y)
    #更新模型参数
    W=W-learn_rate*dL_dW
    #得到估计值
    PRED=np.matmul(X,W)
    #计算损失(均方误差)
    Loss=np.mean(np.square(Y-PRED))/2
    mse.append(Loss)
    if i % display_step==0:
        print("i:%i,Loss:%f"%(i,mse[i]))
xx0=np.ones(1)
xx1=(1650.0-np.average(area))/(area.max()-area.min())
xx2=(3.0-np.average(room))/(room.max()-room.min())
XX=[xx0,xx1,xx2]
print("房屋面积为1650平方英尺房间数量为3时预测房屋的价格:%f"%(np.matmul(XX,W)))
```

5、结果可视化

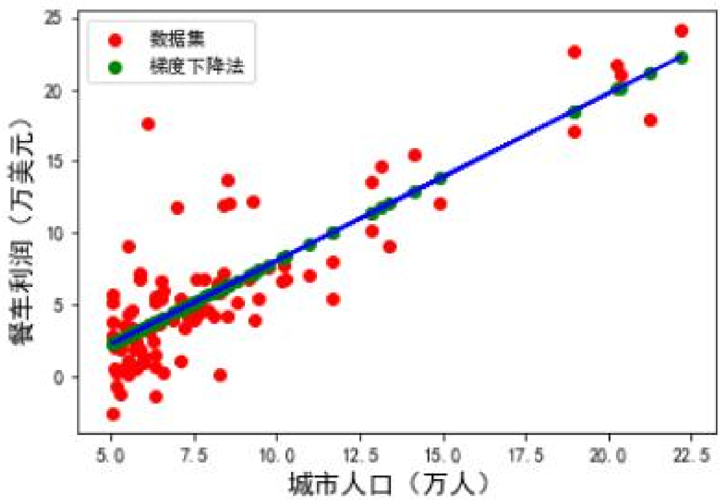
#======================= 【5 】结果可视化==

```
#结果可视化
plt.rcParams['font.sans-serif'] =['SimHei']
plt.figure(figsize=(12,4))
#损失变化可视化
plt.subplot(1,2,1)
plt.plot(mse)
plt.xlabel("迭代次数",fontsize=14)
plt.ylabel("损失值",fontsize=14)
#估计值与标签值比较可视化
plt.subplot(1,2,2)
PRED=PRED.reshape(-1)
plt.plot(price,color="red",marker="o",label="数据集")
plt.plot(PRED,color="blue",marker="o",label="预测房价")
plt.xlabel("sample",fontsize=14)
plt.ylabel("price",fontsize=14)
plt.legend()
plt.show()


fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(area, room, price, color="red")
ax.set_zlabel('price', fontdict={'size': 15, 'color': 'red'})
ax.set_ylabel('room', fontdict={'size': 15, 'color': 'red'})
ax.set_xlabel('area', fontdict={'size': 15, 'color': 'red'})
ax.scatter(area, room, PRED,color="b")
XX, YY = np.meshgrid(area, room)
ax.plot_surface(XX,
                YY,
                Z=W[:,0][0]*x0+W[:,0][1]*((XX-np.average(area))/(area.max()-area.min()))
                   +W[:,0][2]*((YY-np.average(room))/(room.max()-room.min())),
                color='g',
                alpha=0.9
               )

plt.show()
```

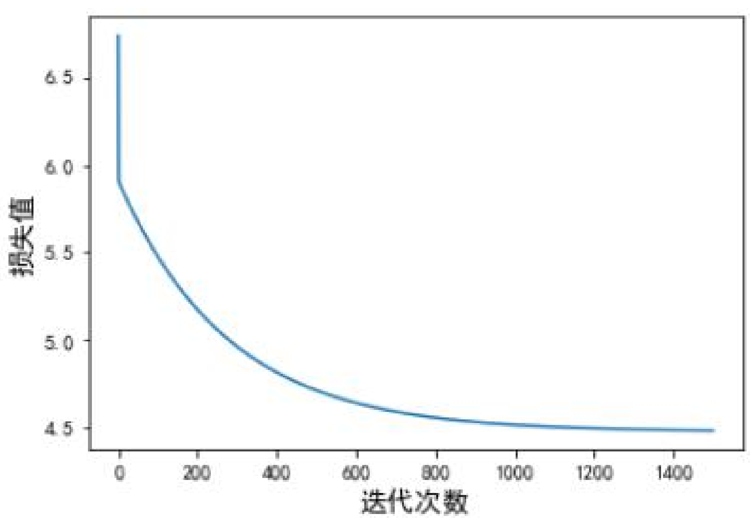
输出结果

一元线性回归输出结果

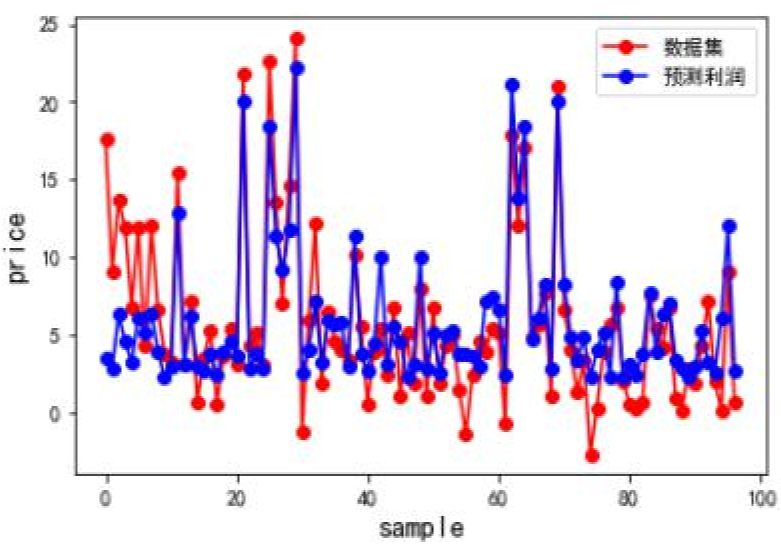
```
i:0, Loss:6. 737190, w:0. 653288, b:0. 058391
i:50, Loss:5. 673965, w:0. 828760, b:-0. 269752
i:100, Loss:5. 476363, w:0. 860183, b:-0. 582539
i:150, Loss:5. 311381, w:0. 888895, b:-0. 868345
i:200, Loss:5. 173635, w:0. 915130, b:-1. 129497
i:250, Loss:5. 058628, w:0. 939103, b:-1. 368122
i:300, Loss:4. 962606, w:0. 961007, b:-1. 586162
i:350, Loss:4. 882437, w:0. 981022, b:-1. 785394
i:400, Loss:4. 815501, w:0. 999311, b:-1. 967439
i:450, Loss:4. 759616, w:1. 016022, b:-2. 133782
i:500, Loss:4. 712956, w:1. 031291, b:-2. 285775
i:550, Loss:4. 673999, w:1. 045243, b:-2. 424657
i:600, Loss:4. 641474, w:1. 057992, b:-2. 551558
i:650, Loss:4. 614317, w:1. 069641, b:-2. 667513
i:700, Loss:4. 591644, w:1. 080285, b:-2. 773466
i:750, Loss:4. 572713, w:1. 090011, b:-2. 870279
i:800, Loss:4. 556908, w:1. 098898, b:-2. 958740
i:850, Loss:4. 543712, w:1. 107018, b:-3. 039571
i:900, Loss:4. 532694, w:1. 114438, b:-3. 113429
i:950, Loss:4. 523495, w:1. 121218, b:-3. 180916
i:1000, Loss:4. 515815, w:1. 127413, b:-3. 242582
i:1050, Loss:4. 509403, w:1. 133073, b:-3. 298928
i:1100, Loss:4. 504049, w:1. 138246, b:-3. 350413
i:1150, Loss:4. 499579, w:1. 142972, b:-3. 397458
i:1200, Loss:4. 495847, w:1. 147290, b:-3. 440444
i:1250, Loss:4. 492731, w:1. 151236, b:-3. 479722
i:1300, Loss:4. 490129, w:1. 154842, b:-3. 515612
i:1350, Loss:4. 487957, w:1. 158136, b:-3. 548406
i:1400, Loss:4. 486143, w:1. 161146, b:-3. 578371
i:1450, Loss:4. 484629, w:1. 163897, b:-3. 605752
i:1500, Loss:4. 483365, w:1. 166410, b:-3. 630770
城市人口为35000时的预测餐车利润:0. 451666
城市人口为70000时的预测餐车利润:4. 534103
```



数据集
梯度下降法
餐车利润（万美元）
城市人口（万人）



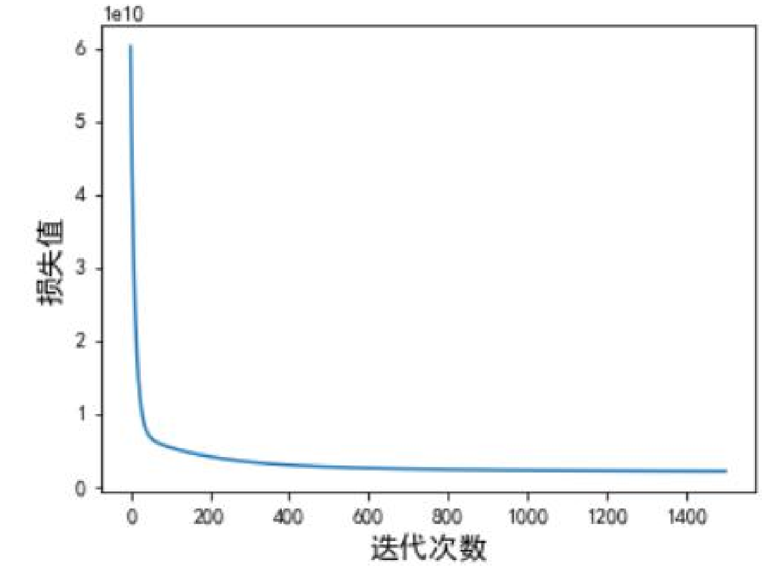
损失值
迭代次数

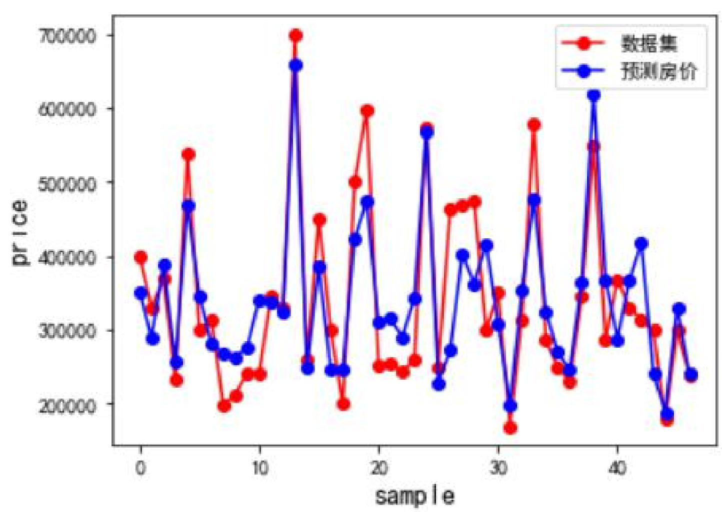


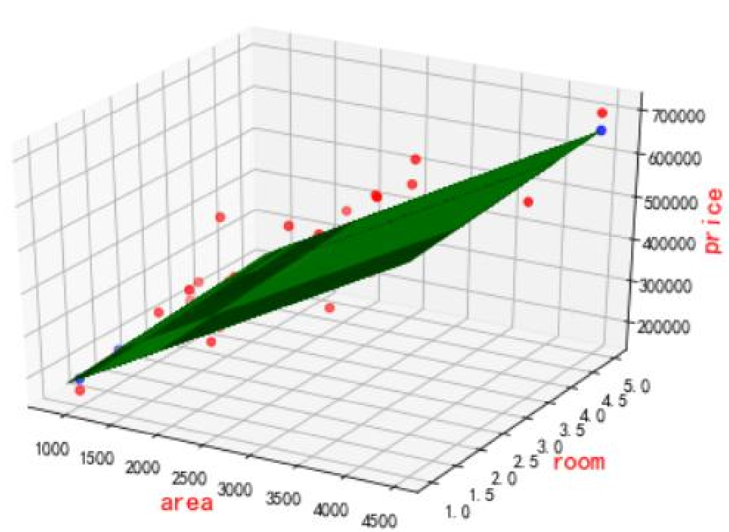
数据集
预测利润
price
sample

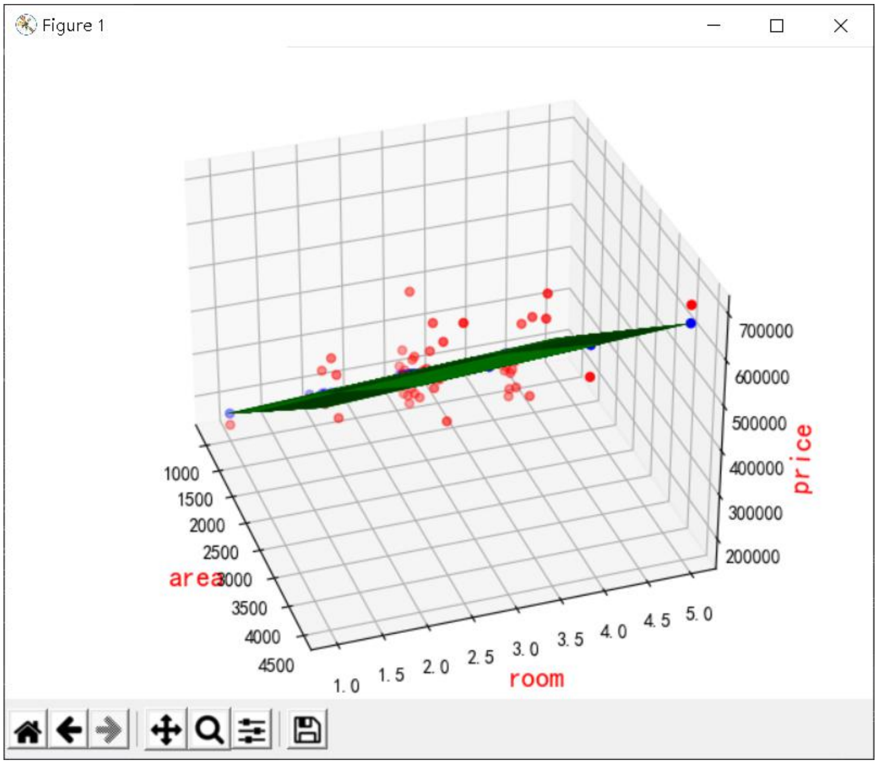
多元线性回归输出结果

```
i:0, Loss:60243485516.962494
i:50, Loss:6766097390.159694
i:100, Loss:5382571239.527689
i:150, Loss:4658040560.158987
i:200, Loss:4114069824.121022
i:250, Loss:3701697312.108251
i:300, Loss:3387218513.321174
i:350, Loss:3145736449.576025
i:400, Loss:2958832363.462618
i:450, Loss:2812867934.514849
i:500, Loss:2697733527.919854
i:550, Loss:2605925102.784432
i:600, Loss:2531863415.169103
i:650, Loss:2471391852.543878
i:700, Loss:2421405969.187702
i:750, Loss:2379580126.538430
i:800, Loss:2344165734.796679
i:850, Loss:2313842294.452954
i:900, Loss:2287607377.069402
i:950, Loss:2264695326.691985
i:1000, Loss:2244517148.051528
i:1050, Loss:2226616026.858234
i:1100, Loss:2210634386.485046
i:1150, Loss:2196289460.895329
i:1200, Loss:2183355156.584526
i:1250, Loss:2171648560.865690
i:1300, Loss:2161019884.798658
i:1350, Loss:2151344946.810517
i:1400, Loss:2142519537.338157
i:1450, Loss:2134455177.577557
i:1500, Loss:2127075912.818542
房屋面积为1650平方英尺房间数量为3时预测房屋的价格:295986.089862
```


1e10
损失值
迭代次数


数据集
预测房价
price
sample


price
room
area



（4） 对实验结果进行简要分析。

一元线性回归结果分析

(1-4)图见上

初始化 w = [0,0] learn_rate=0.01

while 迭代次数小于 1500：　更新 w 的值，记录代价 cost

得到最优参数，然后绘制回归直线

代价变化曲线（下降不明显是因为，将数据的值除以了 m，同时初始参数和最优参数的值比较接近）

(5)利润预测

人口数	利润
35000	0.451666
70000	4.534103

多元线性回归:结果分析

(1)注意点：选择归一化方式

学习速率的大小对于系统是否收敛有决定性的影响。如果学习速率太大，那么可能导致系统震荡发撒；如果学习速率太小，那么可能导致系统收敛速度变慢。

(a) 根据给定数据架设预测函数 h(x)

(b) 计算代价函数 J

(c) 计算各参数偏导

(d) 更新参数

(e) 重复 2~4 直到代价函数跟新的步长小于设定值或者是重复次数达到预设值。

为了在相同学习速率的前提下加快相同收敛，可采用训练数据归一化的方法来对样本数据进行预处理。采用均值均值归一化处理后，缩小了原来数据的变化幅度，从而可极大地提高学习速率，从而提高了梯度下降的执行速度。在第四部分的代码中，对输入值向量进行归一化处理之后，将学习速率从 0.01 提高到 1.9。

(2)一定范围内，当α增大时，算法收敛需要的次数就越少，但是最后得到的最小耗费不会改变，超过一定返回，算法不收敛

α=0.0001 时，收敛需要的迭代次数大于 1500

α=0.03 时，收敛需要的迭代次数小于 100

α=3 时，收敛需要的迭代次数小于 10

α=5 时，算法已经不收敛。

所以阿尔法可以取 0.01

得到的结果（选取的α=0.01）

⑶ **当房屋面积为 1650 时,房间数为 3 时,预测房价为:[[295986.089862]]**

3. 实验心得（每个人心得必须分开写，比如组员 1 张三心得：……；组员 2 李四心得：……）

组员 1 张继伟心得:

一元线性回归是分析只有一个自变量（自变量 x 和因变量 y）线性相关关系的方法，一个经济指标的数值往往受许多因素影响，若其中只有一个因素是主要的，起决定性作用，则可用一元线性回归进行预测分析；而多元线性回归是由多个因素影响，但是多个因素之间的关系是线性的。我在训练模型的时候发现，无论多元还是一元，训练模型的本质是一样的，都是两个矩阵的相乘，只不过矩阵元素的个数多少不同而已。线性模型的训练总体来说比较简单，通过递归下降的方式，限制迭代次数，和一定的学习速率，总能得带收敛效果。在本次实验中最有意思的不是训练模型，而是通过 Python 的 matplotlib 和 numpy 包绘制图形，特别是多元回归时

的 3D 图形，虽然 3D 图形绘制较为麻烦投入时间多，并且遇到了许多困难，比如训练时应用了归一化处理导致绘制图形时要特别注意处理数据，但是自己还是收获了许多，而且自己最后绘制出来的图形自己也比较满意。

组员 2 谢正宇心得:

通过本次实验我对线性回归有了更深层次的理解，用一句话来解释线性回归是什么的话，我的理解是这样子的：线性回归，是从大量的数据中找出最优的线性（y=wx+b）拟合函数,通过数据确定函数中的未知参数，进而进行后续操作（预测。回归的概念是从统计学的角度得出的，用抽样数据去预估整体（回归中，是通过数据去确定参数），然后再从确定的函数去预测样本。我们在实验中用到的方法是梯度下降法来求函数极值，梯度下降算法是求解最小值的一种方法，但并不是唯一的方法。梯度下降法的核心思想就是对损失函数求偏导，从随机值（任一初始值）开始，沿着梯度下降的方向对 w 和 b 的迭代，最终确定 w 和 b 的值，注意，这里要同时迭代 w 和 b（这一点在编程过程中很重要）。 总结和思考：本次实验有两个部分，第一部分是一元线性回归采用梯度下降法求解，模型比较好训练。第二部分是多元线性回归线性归一化：适用于样本分布均匀且集中的情况，如果最大值(或者最小值)不稳定，和绝大数样本数据相差较大，使用这种方法得到的结果也不稳定.为了抑制这个问题，在实际问题中可以用经验值来代替最大值和最小值。标准差归一化适用于样本近似正态分布，或者最大最小值未知的情况，有时当最大最小值处于孤立点时也可以使用标准差归一化。非线性映射归一化，通常用于数据分化较大的情况（有的很大有的很小）样本属性归一化需要根据属性样本分布规律定制。总的来说，这次试验收获还是挺大的，第一次切身体验了真正的机器学习。这让我对后续更加期待。

组员 3 樊佳婷心得:

本次实验我负责报告的撰写和对部分资料进行收集整理和结果分析。

使用梯度下降算法是解决如何一步步调整参数，使代价函数达到最小值，从而确定参数 w 和 b，然后确定模型，梯度下降算法只适用凸函数。参数的结果与初始值的选取有极大关系，选不好容易陷入局部极小值，而不是全局最小值，一直求导求得代价函数的最小值。

然而梯度下降法是很依赖数据的归一性的，所以使用梯度下架法的时候，必须要对数据做数值归一化。如果使用公式法就没有这样的问题，因为公式法求解过程并不考虑数值的单位，所以也对最后结果没有太大影响。不过在数据的特征值很大的时候，梯度下降法的优势就出来了,运行速度快了很多,适合处理特征值很大的数值。

选择的学习率过小，收敛速度会比较慢。学习率过大，有可能导致代价函数错过最优解，从图像中我们会发现代价函数呈现出指数性增长，不断远离最优点。

组员 4 刘怡聪:

本次实验我负责一元线性回归部分，实验内容为用梯度下降法拟合回归曲线，得到回归方程（求出斜率和截距），并求出指定 x 下的 y。我们小组使用 python 编写，画出了散点图和曲线图等可视化结果，这些都很有助于我们理解数据间的关系，简单推测出数据间满足的方程特点。实验的难点在于梯度下降法，梯度下降法是按下面的流程进行的：1）首先对θ赋值，这个值可以是随机的，也可以让θ是一个全零的向量。2）改变θ的值，使得 J(θ)按梯度下降的方向进行减少。 我们需要反复尝试迭代次数，让该方法帮我们拟合出一个方差最小的直线方程，尽量满足数据点均匀分布在直线两侧。调整学习率：太大会造成无法到达全局最小点，太小会造成收敛速度过慢，所以需要随着 epochs 的增加而减小，给不同的参数不同的学习率。通过这次实验，将上课的内容转换为实际的代码操作，并且在修改和调试中完成结果分析，收获很大。

组员 5 孙晶铭:

对于一个多元线性回归模型，我们可以得到其训练集，同时为了选出最合适的线性回归模型需要找出损失函数最小的向量，需要使用到的算法为梯度下降算法。它是一种求局部最优解的方法，对于 F(x)，在 a 点的梯度是 F(x)增长最快的方向，那么它的相反方向则是该点下降最快的方向。但线性回归使用梯度下降算法最优化问题只有一个全局最优，没有其它局部最优。这是因为 J(θ) 是凸二次函数。所以这里的梯度下降会一直收敛到全局最小。同时我们需要注意梯度下降算法是通过每次迭代后，使得当前的向量θ \thetaθ代入 J(θ)损失函数后，使得其值逐渐减少，直到最后收敛。在实际的操作过程中，我们可能会遇到 J(θ)经过迭代后，其值不但没有减少反而增大的反常情况，那么这种情况通常是因为我们选取的学习率α太大，我们需要减小α。当然，我们又不能使α太小，从而使得收敛需要的迭代的次数太大。

4. 程序文件名的清单

|___code

|__*.ipynb*checkpoints

| |__*lab*1-checkpoint.ipynb

|__*lab*1.ipynb

|___ex1data1.txt

|___ex1data2.txt

|___计科 1802 班+张继伟+谢正宇+樊佳婷+刘怡聪+孙晶铭.pdf