# 梯度下降

method:

1. First, two parameters are given：

称为学习率（learning rate）也叫步长，

这个值越大每次修正的就越多，不过这个不是越高越好，如果太高了可能会一直在最低点“摆动”而无法收敛。也有的使用可变的学习速率，一开始设置较高，接近最低点的过程中逐渐降低。当太小时，可能会需要多次迭代才能收敛，当太大时，可能导致无法收敛或者会发散，只有选择合适的时，才能更好的收敛

固定步长梯度下降

通过测量参数向量相关的误差函数的局部梯度，并不断沿着降低梯度的方向调整，直到梯度降为0或者小于，到达最小值。

1. Calculate the derivative

Iteration 0:

1. Calculate the current derivative value

1. Modify the current parameters

Iteration 1:

1. Calculate the current derivative value
2. Modify the current parameters

Iteration 2:

1. Calculate the current parameters
2. Modify the current parameters

Iteration 3:

1. Calculate the current parameters
2. Modify the current parameters

Iteration 4:

1. Calculate the current parameters
2. Modify the current parameters

Iteration 5:

1. Calculate the current parameters
2. Modify the current parameters

Iteration 6:

1. Calculate the current parameters
2. Modify the current parameters

**Iteration 7:**

**…**

Iteration 29:

Calculate the current derivative value

Modify the current parameters

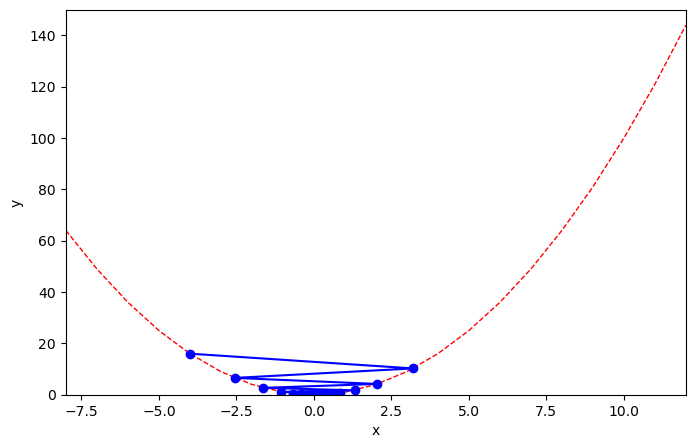
Iteration 30 and output image:

Calculate the current derivative value

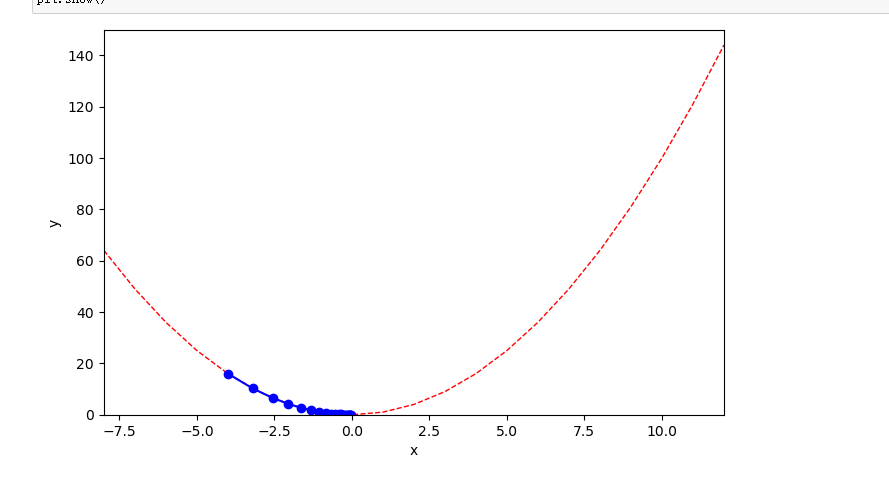
Modify the current parameters

当接近局部最小值时，梯度下降将自动采取较小的步骤，因为越接近最小值，导数会自动变小.

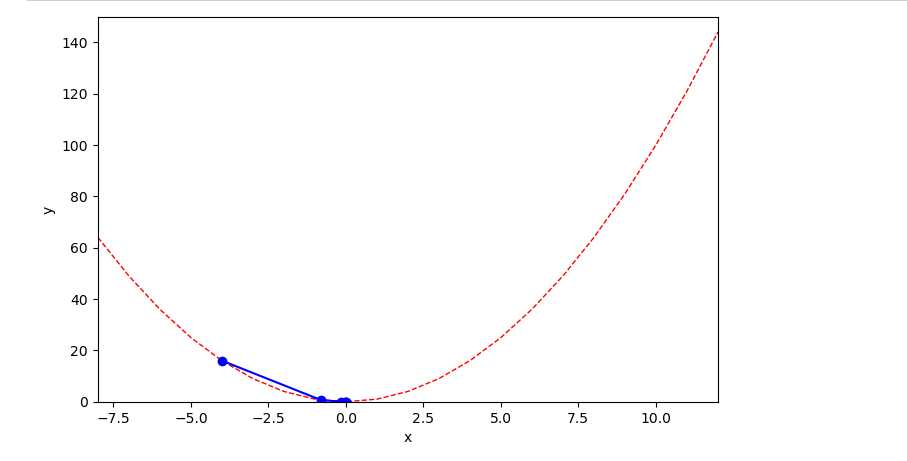
Output Image：



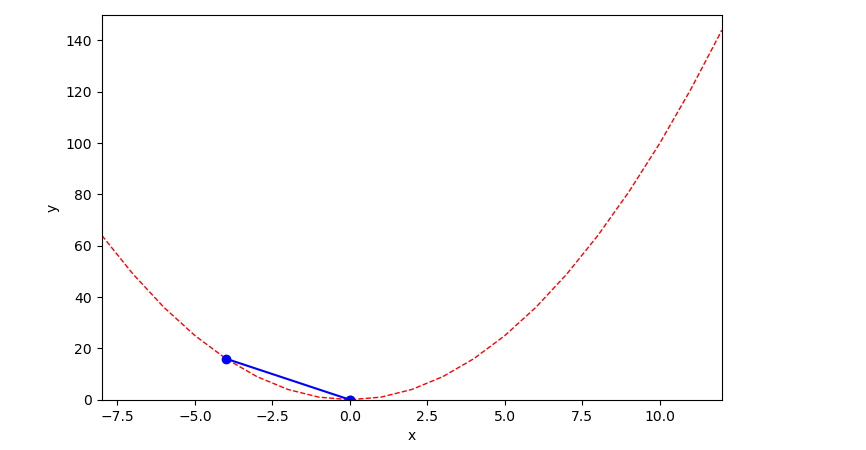
Output Image：



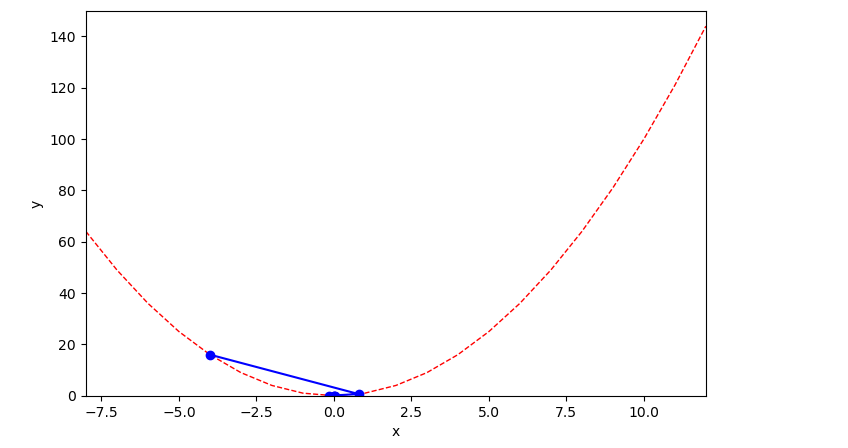
Output Image：



Output Image：



Output Image：



# 一元线性回归

2.1线性回归

自变量(x)和因变量(y)之间的关系可以用一条直线近似表示，这种关系称为线性回归

2.2最小二乘法

假设特征的数据只有一个

在线性回归中，最小二乘法就是试图找到一条直线，使所有样本到直线上的欧式距离最小

由最小二乘法导出损失函数E(w,b)

### 2. 定义损失函数

#损失函数是系数的函数，另外还要传入数据的x,y

def compute\_cost(w,b,points):

total\_cost = 0

# 逐点计算平方损失误差，然后求平均数

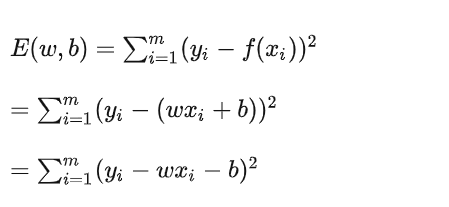
for i in range(M):

x = points[i,0]

y = points[i,1]

total\_cost+= (y - w\*x - b)\*\*2 ## 这个对应最小二乘法函数

return total\_cost/M



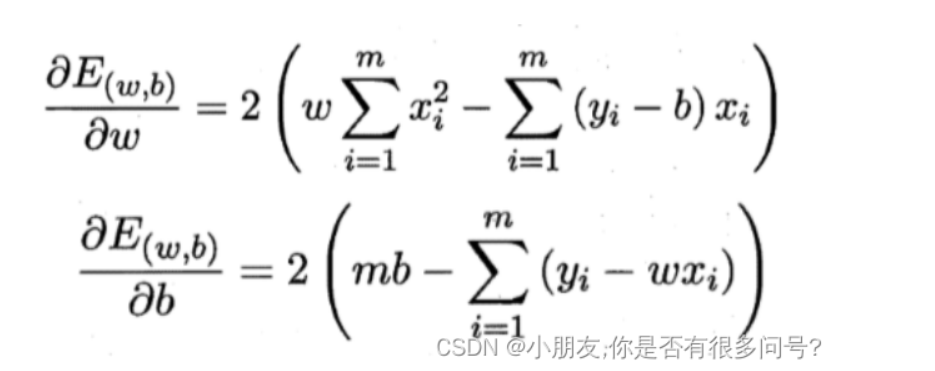
我们的目标是求出使损失函数最小化的参数

这就转换成了二元函数求最小值的问题了

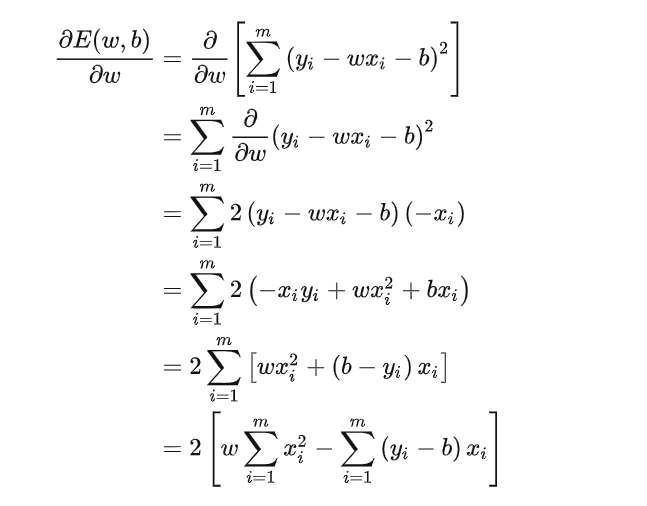
3. 求解线性回归

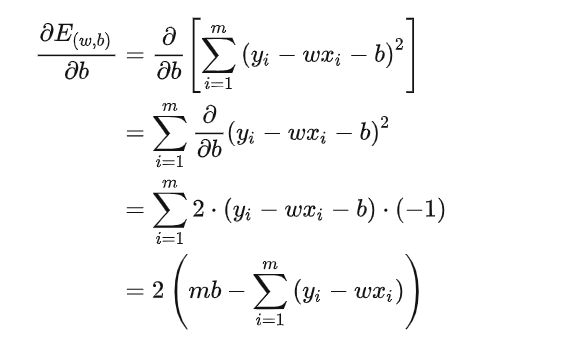
求解和 ，使得欧式距离最小的过程，称为线性回归模型的"最小二乘参数估计"

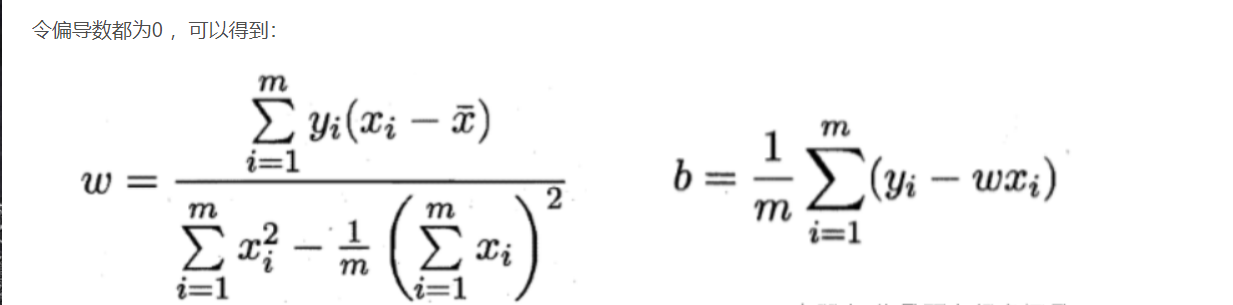
将E(w,b)分别对w和b求导，可得到：



推导过程：







# 定义核心拟合函数

def fit(points):

M = len(points)

x\_bar = average(points[:,0]) ##传入所有的第一列 并求均值

sum\_yx = 0

sum\_x2 = 0

sum\_delta=0

for i in range(M):

x = points[i,0]

y = points[i,1]

sum\_yx += y \* ( x - x\_bar ) ## 对应求偏导W的分子部分

sum\_x2 += x \*\* 2

# 根据公式计算W

w=sum\_yx/(sum\_x2- M \*(x\_bar\*\*2)) ## 对应求偏导W公式

for i in range(M):

x = points[i,0]

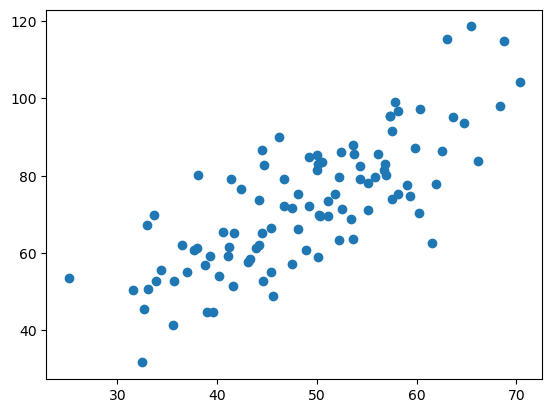
y = points[i,1]

sum\_delta+= (y - w\*x ) ##对应求偏导b公式

b = sum\_delta / M

return w,b

一元线性回归-最小二乘法 代码示意图

++

### 4. 测试

w,b=fit(points) ## 计算出 最小二乘法系数

print("w is :",w)

print("b is :",b)

cost=compute\_cost(w,b,points)

print("cost is :",cost)

## 5.拟合曲线

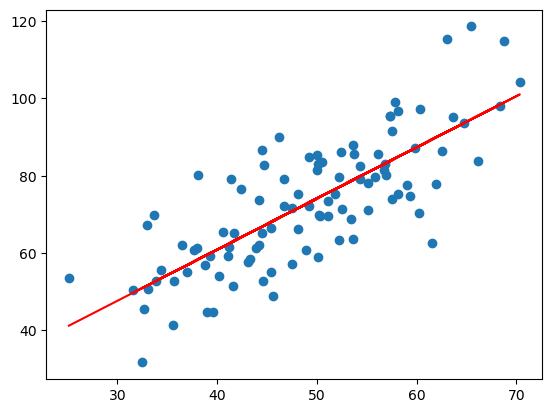
plt.scatter(x,y) ##散点图

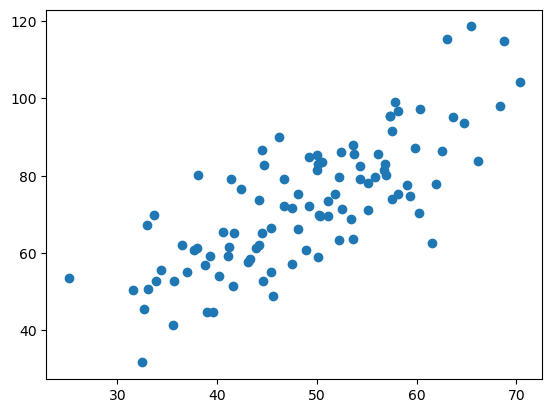
# 针对每一个x,计算出预测的y值

pred\_y = w \* x + b #ny 可以用一个常数与向量直接相乘，对应原特征公式

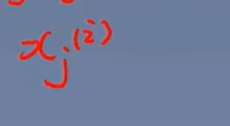
plt.plot(x,pred\_y,c='r') ##点图

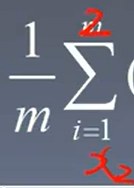
plt.show()





一元线性回归-梯度下降法

 表示第i个元素的第j个特征

m表示有m个数据

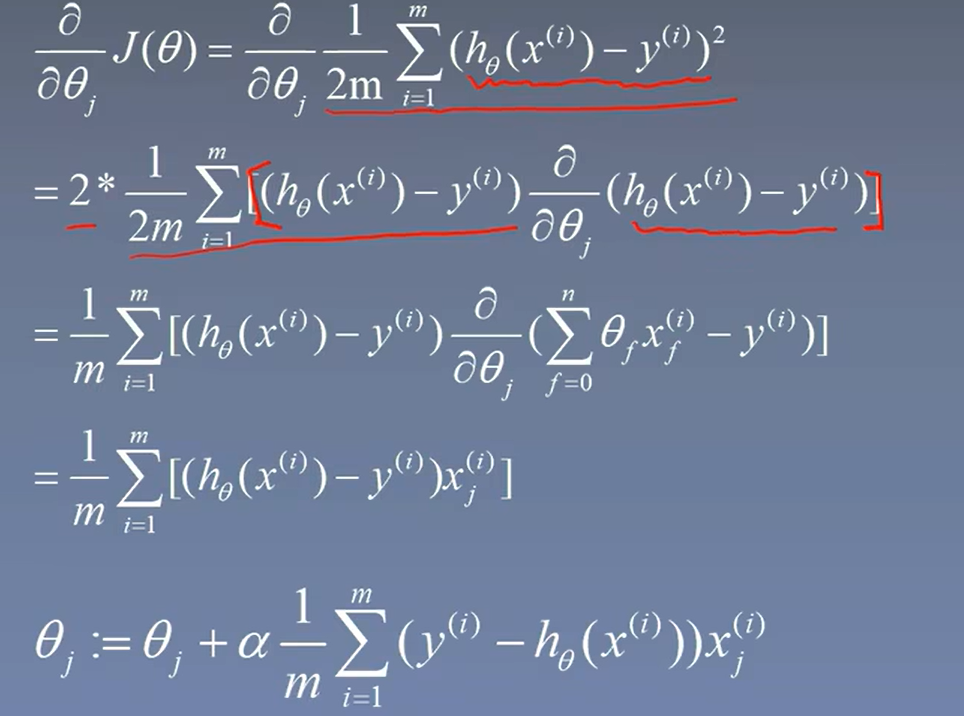
例如一个大房子x（1）是一个元素数据，具有房价x1（1）、朝向x2（1）两个特征，根据一个房子的两个特征有一个价格y（1）

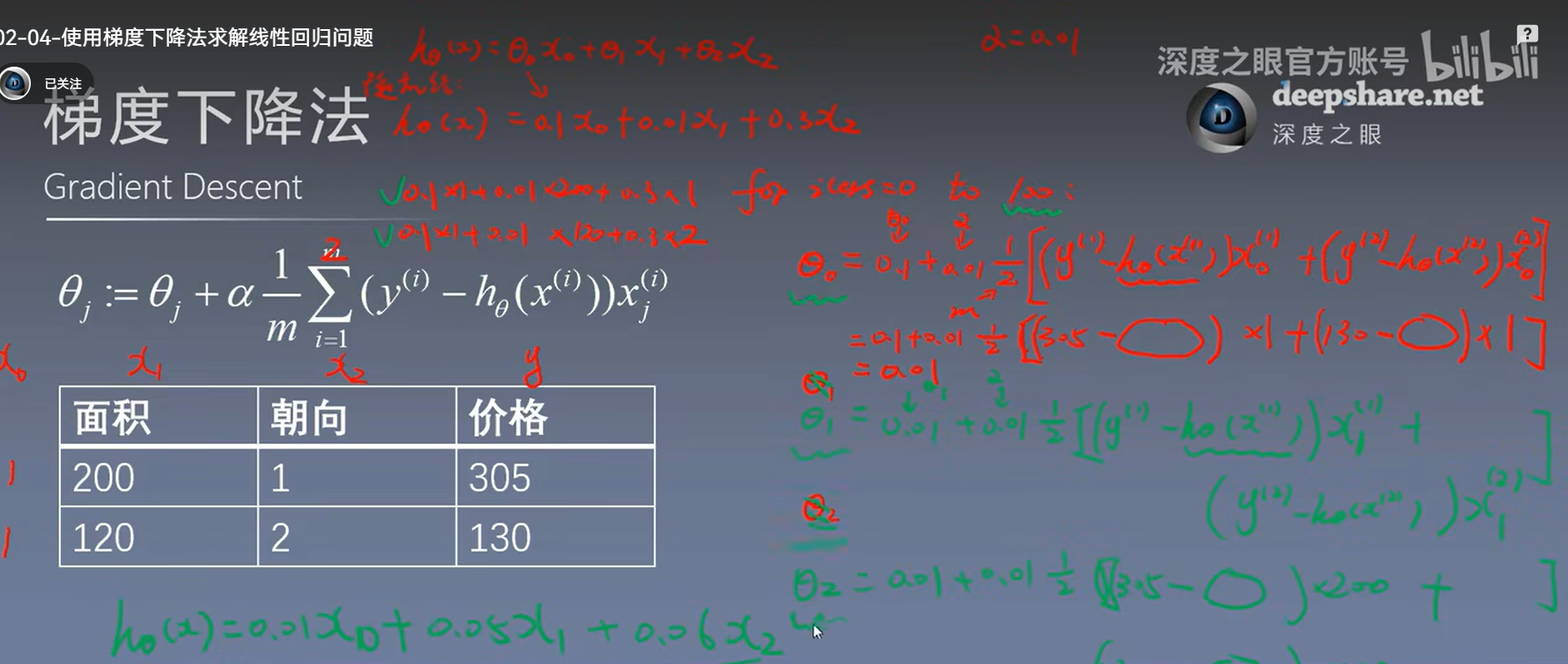
例如一个小房子x（2）是一个元素数据，具有房价x1（2）、朝向x2（2）两个特征，根据一个房子的两个特征有一个价格y（2）

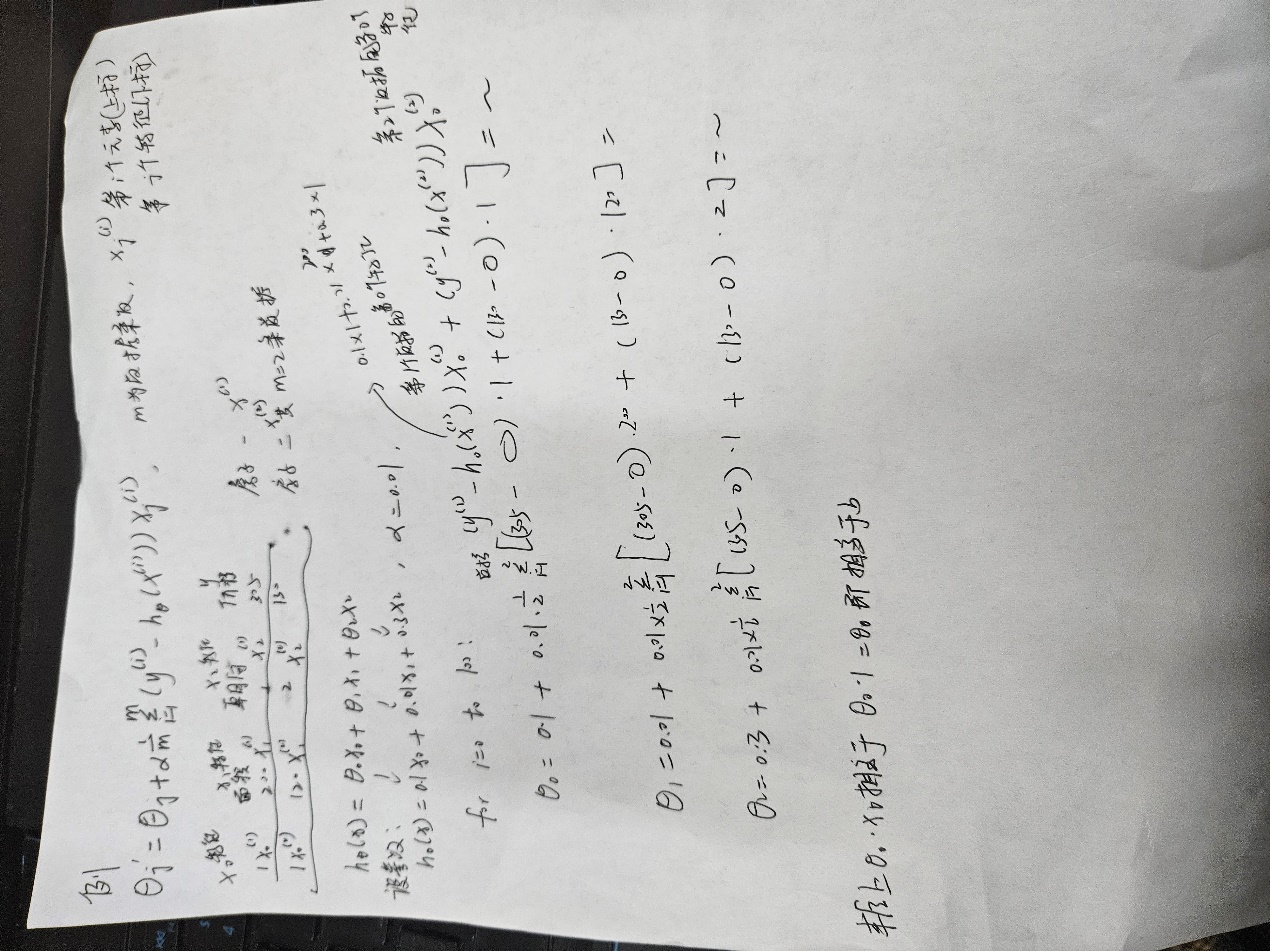
梯度下降求解f（x）线性回归问题

这里是假设函数，以三组为例，相当于和相乘并累加求和n次，n表示现有的特征数，特征数用上标（1）（2）表示

*表示代价函数，表示假设函数算出的预测值和实际值的差距. 代价函数越小越好，故用代价函数对求偏导，使用梯度下降更新参数。*





**

*了解jhs的三次梯度下降 代码实现看懂*

*三种梯度下降定义s*

*使用02博客中实现梯度下降代码02 python 实现线性回归-一元线性回归-梯度下降*

# 本文作者： hang shun @航 順  
# 本文链接： https://jiang-hs.gitee.io/posts/202f1f0f/  
# 版权声明： 本站所有文章除特别声明外，均采用 (CC)BY-NC-SA 许可协议。转载请注明出处！  
  
#求损失函数 J (θ)

for i in range(m):

diss = diss + (1/(2\*m))\*pow((theta0+theta1\*x[i]-y[i]),2)

loss.append(diss)

#求损失函数 J (θ)：

rand\_i = random.randrange(0,m)

diss = diss + (1/(2\*m))\*pow((theta0+theta1\*x[rand\_i]-y[rand\_i]),2)

loss.append(diss)

#求损失函数 J (θ)

rand\_ls = list(range(3))

for i in range(3): rand\_i = random.randrange(0,m) rand\_ls[i] = rand\_i for i in rand\_ls: diss = diss + (1/(2\*m))\*pow((theta0+theta1\*x[i]-y[i]),2) loss.append(diss)