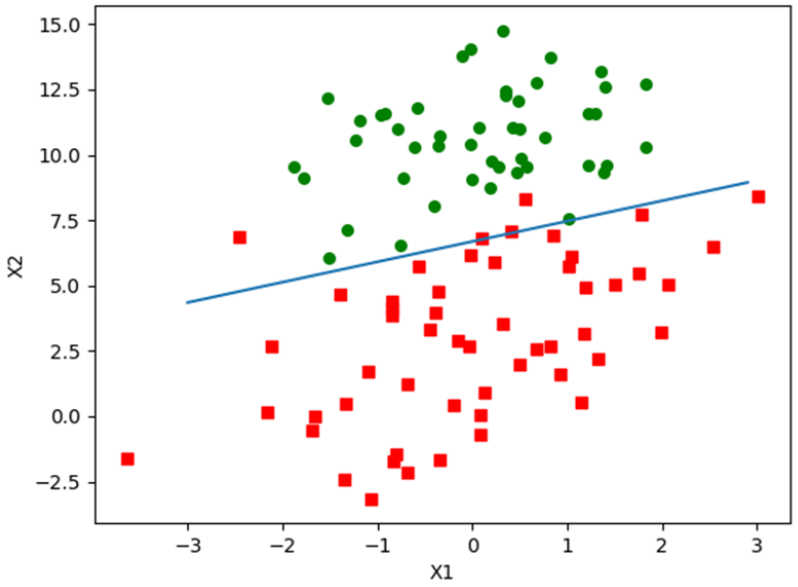
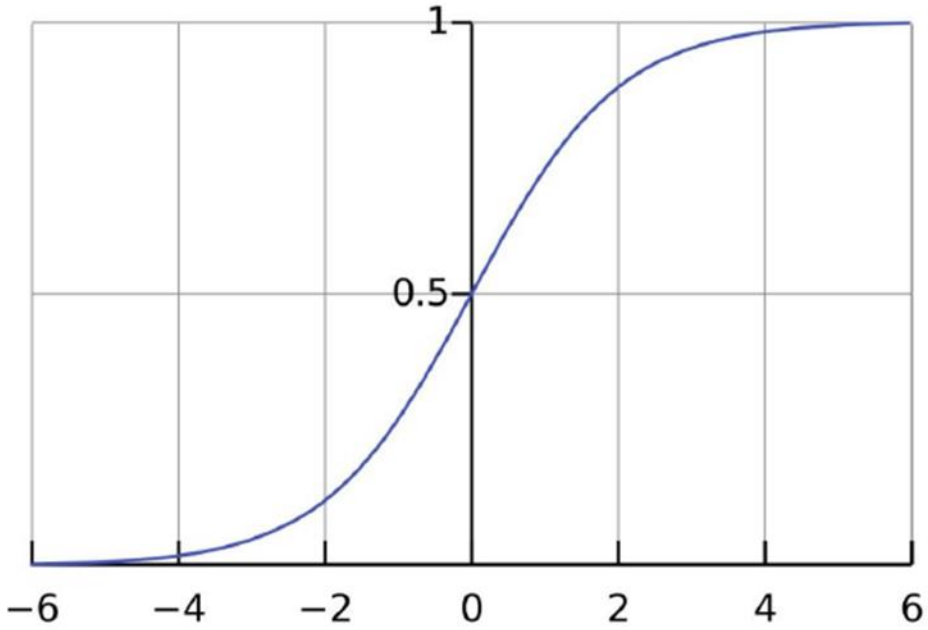
# 梯度下降

# 复习




$$g(z)=\frac{1}{1+e^{-z}}$$

$$h_\theta(\mathbf{x})=g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}\right)=\frac{1}{1+e^{-\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}}}$$

# 思考

$$-\frac{1}{m}\sum_{i=1}^{m}\left(\left(y^{(i)}-h_{\theta}\left(\mathbf{x}^{(\mathbf{i})}\right)\right)\mathbf{x}^{(\mathbf{i})}\right)$$

- 这个式子是关于谁的偏导数？
- 这个式子什么时候等于0？
- 这个式子再求偏导怎么求？

# 再抢救一下

- 逻辑回归的代价函数

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log h_{\boldsymbol{\theta}}\left(x^{(i)}\right)+\left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}\left(x^{(i)}\right)\right)\right)$$

- 逻辑回归的代价函数需要求最小值，但是以求导函数何时为0的方式求最小值代价太大。

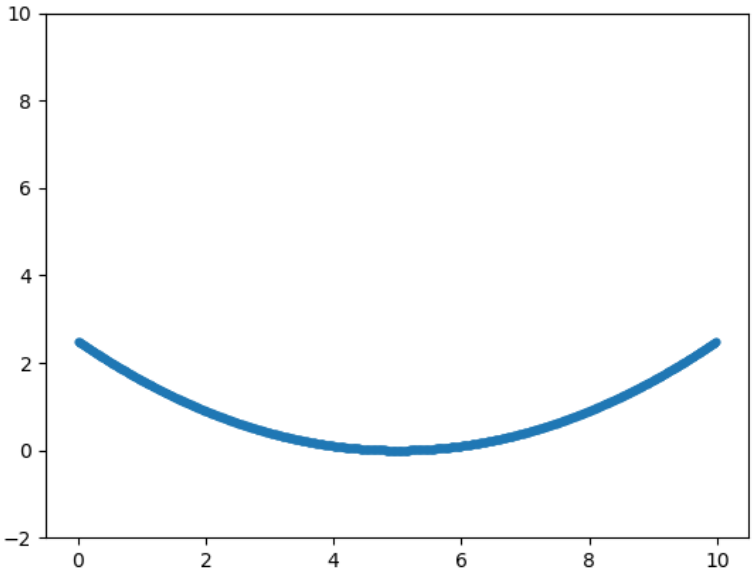
# 梯度下降

# 例

- 不使用导数为零处取得极值的结论计算 $f\left(x\right)=\frac{1}{10}\left(x-5\right)^{2}$ 什么时候取得最小值。

# 例

- 首先给定随机数$x_0$=7，由于f’(7)=0.4>0,说明在x=7处函数处于递增状态，那么要找x为多少时可以取得最小值就应该找一个小于7值赋给x，即令$x_0$减去一个数作为$x_1$，并考虑$x_1$处是否为最小值点。



# 例

- 可以令$x_0 - f'(x_0) = x_1$,那么$x_1 = 6.6$，由于$f'(6.6)=0.32$依然大于0，说明应该继续减小。

```
第0次迭代
6.6
第1次迭代
6.279999999999999
第2次迭代
6.023999999999999
第3次迭代
5.8191999999999995
第4次迭代
5.65536
```

```
第20次迭代
5.018446744073709
第21次迭代
5.014757395258967
第22次迭代
5.011805916207174
第23次迭代
5.009444732965739
第24次迭代
5.007555786372591
第25次迭代
```

# 思考

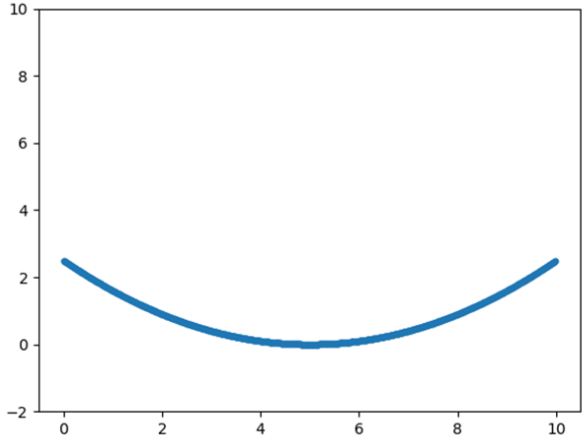
- 上面的例子中，一开始选择的$x_0$的导数大于0，所以我们可以令$x_0$减去它的导数。那么如果一开始选择的$x_0$的导数为负呢？

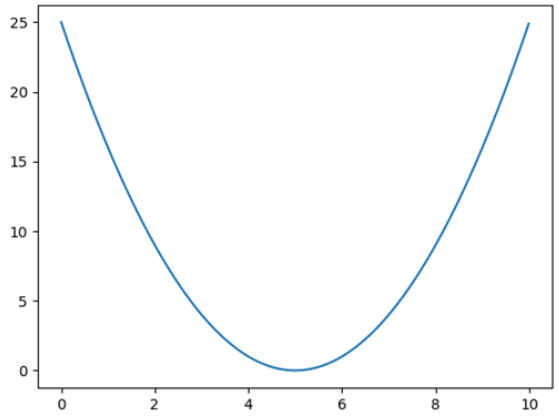
# 梯度下降

- 首先随机选择$x_0$，计算函数在$x_0$处的导数，然后用$x_0$减去它自己的导数，并将得到的结果作为$x_1$，并计算$x_1$处的导数，然后得到$x_2$……，直到收敛。

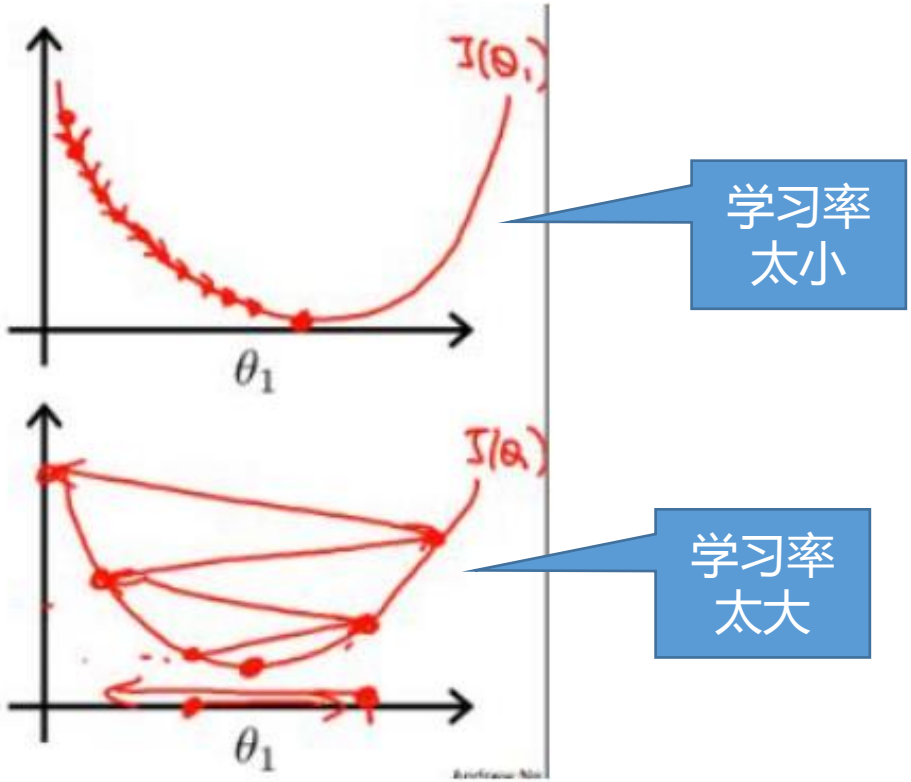
# 学习率

- 如果去掉上面的例子中的1/10，初始值还选7，后果会怎么样？
- f'(7)=4, 用7-f'(4)=3取代7, f'(3)=-4, 3-f'(3) = 7
- 为了控制x的变化速度，需要设定一个常数α，用$x_0-\alpha f(x_0)$取代$x_0$。α称为学习率。



# 学习率的选择



# 学习率的选择

- 学习率的选择没有太好的方法，一般都是根据实际情况首先尝试一个常用的值，比如0.1，0.01，0.001等，根据代价的变化情况决定继续使用当前学习率还是对学习率进行调整。

# 例

```
learning_rate = 0.01
a = 3
for i in range(1000):
    a = a - learning_rate*(2 * a - 10)
    print(a)
```

4.999999996423753
4.999999996495278
4.999999996565372
4.999999996634065

# 问题

- 用梯度下降一定能找到最小值吗？

- 未必，找到的可能是局部最小值，即极小值。

# 例

- 利用上面的方法计算$\sqrt{73}$。

# 例

- 根据题意 $x=\sqrt{73} \Rightarrow x^2=73 \Rightarrow x^2-73=0$
- 假设有 $f(x)=x^2-73, F'(x)=f(x)$，此题目转换为求F(x)的极值(根据二阶导数可以知道是极小值)问题。
- 所以，可以利用上一题的方法

```
learning_rate = 0.01
a = 3
for i in range(1000):
    a = a - learning_rate*(a* a - 73)
    print(a)
```

```
8.544003745317527
8.544003745317527
```

# 梯度下降

- 如果一个函数f(x)至少有一个极小值，那么就可以利用下面的方法找到这个极小值：
  - 首先随机确定一个点$x_0$以及一个称为学习率的正数；
  - 计算f(x)在$x_0$处的导数，并用$x_0$减去f(x)在$x_0$处的倒数与学习率的乘积作为$x_1$；
  - 重复第二步，直到x不再变化或者达到指定的迭代步数。
- 如果求极大值，则加上导数与学习率的乘积，称为梯度上升。

# 多元函数的梯度下降

- 用梯度下降的方法求函数 $z=(x-5)^2+(y-3)^2$的极小值。

# 多元函数的梯度下降

首先，随机给定x和y的初始值，比如x=10, y=10

计算偏导数，$\frac{\partial z}{\partial x} = 2x - 10 = 10, \frac{\partial z}{\partial y} = 2y - 6 = 14$

指定学习率$\alpha$=0.1

$x = 10 - 0.1*10 = 9, y = 10 - 0.1*14 = 8.6$

$x = 9 - 0.1*8 = 8.2, y = 8.6 - 0.1*11.2 = 7.48$

…

# 多元函数的梯度下降

```
x = 10
y = 10
learning_rate = 0.1
for i in range(100):
    x = x - learning_rate * (2 * x - 10)
    y = y - learning_rate * (2 * y - 6)
    print("第" + str(i) + "次迭代")
    print("x = " + str(x))
    print("y = " + str(y))
```

# 多元函数的梯度下降

第0次迭代
x = 9.0
y = 8.6
第1次迭代
x = 8.2
y = 7.4799999999999995
第2次迭代
x = 7.56
y = 6.584
第3次迭代
x = 7.048
y = 5.8671999999999995
第4次迭代

y = 3.013539969179684
第28次迭代
x = 5.007737125245534
y = 3.010831975343747
第29次迭代
x = 5.006189700196427
y = 3.008665802749975
第30次迭代
x = 5.004951760157142
y = 3.006932464219998
第31次迭代
x = 5.003961408125713
y = 3.005545971375984

# 多元函数的梯度下降

16.将长为2m的铁丝分成三段，依次围成圆、正方形与正三角形，三个图形的面积之和是否存在最小值？若存在，求出最小值.

【答案】设分成的三段分别为$x,y,z$，则有$x+y+z=2$及$x,y,z>0$，圆的面积为$S_1=\frac{1}{4\pi}x^2$，正方形的面积为$S_2=\frac{1}{16}y^2$，正三角形的面积为$S_3=\frac{\sqrt{3}}{36}z^2$，总面积$S=\frac{1}{4\pi}x^2+\frac{1}{16}y^2+\frac{\sqrt{3}}{36}z^2$，则问题转化为在条件$x+y+z=2,x,y,z>0$下，求函数$\frac{1}{4\pi}x^2+\frac{1}{16}y^2+\frac{\sqrt{3}}{36}z^2$的最小值。令$L=\frac{1}{4\pi}x^2+\frac{1}{16}y^2+\frac{\sqrt{3}}{36}z^2+\lambda(x+y+z-2)$，

解得唯一条件极值点为
$$\begin{cases} x=\dfrac{2\sqrt{3}\pi}{\sqrt{3}\pi+4\sqrt{3}+9} \\ y=\dfrac{8\sqrt{3}}{\sqrt{3}\pi+4\sqrt{3}+9} \\ z=\dfrac{18}{\sqrt{3}\pi+4\sqrt{3}+9} \end{cases}$$

# 多元函数的梯度下降

- 由于x+y+z=2，所以z =2-x-y，所以总面积公式可以写为：

$$S = \frac{1}{4\pi}x^2 + \frac{1}{16}y^2 + \frac{\sqrt{3}}{36}(2-x-y)^2$$

- 求偏导：

$$\frac{\partial S}{\partial x} = \frac{1}{2\pi}x - \frac{\sqrt{3}}{18}(2-x-y)$$

$$\frac{\partial S}{\partial y} = \frac{1}{8}y - \frac{\sqrt{3}}{18}(2-x-y)$$

# 多元函数的梯度下降

- 面积公式实现：

```
def f(x,y):
    return (x * x)/(4 * math.pi) + (y * y)/16 + \
           (math.sqrt(3)* (2 - x - y) * (2 - x - y)) /36
```

- 偏导数

```
def hx(x, y):
    return x/(2 * math.pi) - (math.sqrt(3) * (2 - x - y))/18

def hy(x, y):
    return y / 8 - (math.sqrt(3) * (2 - x - y))/18
```

# 多元函数的梯度下降

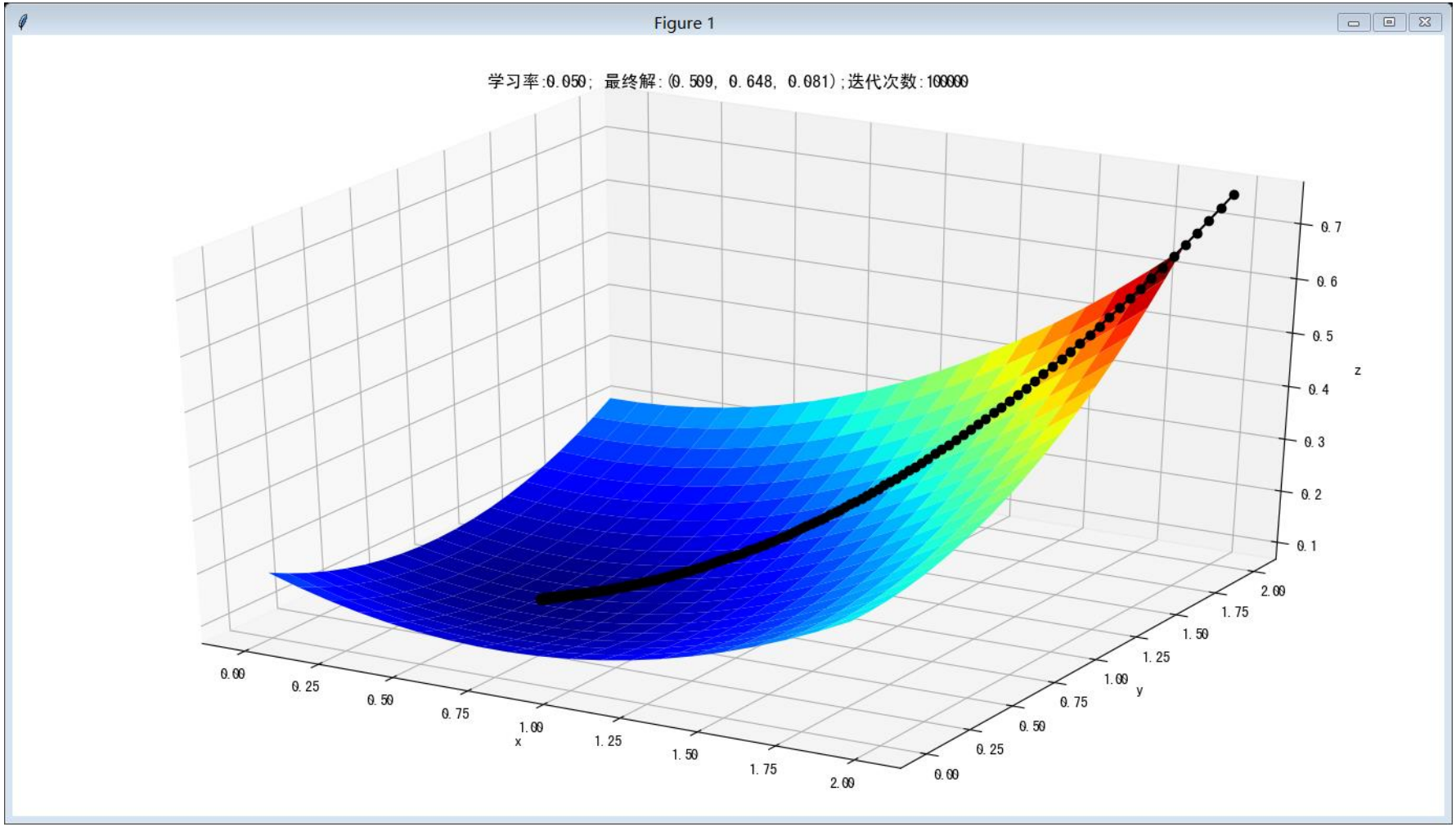


Figure 1
学习率:0.050; 最终解:(0.509, 0.648, 0.081);迭代次数:100000
0.7
0.6
0.5
z
0.4
0.3
0.2
0.1
0.00
0.25
0.50
0.75
1.00
x
1.25
1.50
1.75
2.00
2.00
1.75
1.50
1.25
1.00
y
0.75
0.50
0.25
0.00

# 多元函数梯度下降总结

- 首先生成一个n维随机向量$a_0$，其中n代表多元函数中有n个自变量。
- 分别求多元函数对这n个自变量在$a_0$处的偏导数这n个偏导数构成一个新的n维向量$b_0$。
- 设定学习率alpha。
- 令$a_1 = a_0 - aloha \times b_0$，并求多元函数在$a_1$处的n个偏导数，构成$b_1$。
- 重复以上过程，直到收敛或者达到指定次数。

# 逻辑回归中的梯度下降

- 逻辑回归的代价函数为：

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathbf{i})}\right)+\left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathbf{i})}\right)\right)\right)$$

# 代价函数求偏导

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left( y^{(i)} \log h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathbf{i})}\right) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathbf{i})}\right)\right)\right)$$

$$\frac{\partial}{\partial\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left( y^{(i)} \bullet \frac{1}{h_\theta\left(\mathbf{x}^{(\mathbf{i})}\right)} \bullet \frac{\partial}{\partial\theta_j} h_\theta\left(\mathbf{x}^{(\mathbf{i})}\right) - \left(1-y^{(i)}\right) \bullet \frac{1}{1-h_\theta\left(\mathbf{x}^{(\mathbf{i})}\right)} \bullet \frac{\partial}{\partial\theta_j} h_\theta\left(\mathbf{x}^{(\mathbf{i})}\right)\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(\left( y^{(i)} \bullet \frac{1}{h_\theta\left(\mathbf{x}^{(\mathbf{i})}\right)} - \left(1-y^{(i)}\right) \bullet \frac{1}{1-h_\theta\left(\mathbf{x}^{(\mathbf{i})}\right)}\right)\frac{\partial}{\partial\theta_j} h_\theta\left(\mathbf{x}^{(\mathbf{i})}\right)\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(\left( y^{(i)} \bullet \frac{1}{g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)} - \left(1-y^{(i)}\right) \bullet \frac{1}{1-g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)}\right)\frac{\partial}{\partial\theta_j} g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\right)$$

# 代价函数求偏导

$$\frac{\partial}{\partial\theta_j}g\left(\mathbf{\theta^T x^{(i)}}\right)=\frac{\partial}{\partial\theta_j}\left(\frac{1}{1+e^{-\mathbf{\theta^T x^{(i)}}}}\right)$$

$$=-\frac{1}{\left(1+e^{-\mathbf{\theta^T x^{(i)}}}\right)^2}\bullet\frac{\partial}{\partial\theta_j}\left(1+e^{-\mathbf{\theta^T x^{(i)}}}\right)$$

$$=-\frac{e^{-\mathbf{\theta^T x^{(i)}}}}{\left(1+e^{-\mathbf{\theta^T x^{(i)}}}\right)^2}\bullet\frac{\partial}{\partial\theta_j}\left(-\mathbf{\theta^T x^{(i)}}\right)$$

$$=\frac{e^{-\mathbf{\theta^T x^{(i)}}}}{\left(1+e^{-\mathbf{\theta^T x^{(i)}}}\right)^2}\bullet\mathbf{x^{(i)}}$$

# 代价函数求偏导

$$g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)=\frac{1}{1+e^{-\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}}}\Rightarrow e^{-\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}}=\frac{1}{g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)}-1$$

$$\Rightarrow-\frac{e^{-\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}}}{\left(1+e^{-\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}}\right)^{2}}=g^{2}\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\left(\frac{1}{g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)}-1\right)$$

$$=g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\left(1-g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\right)$$

# 代价函数求偏导

$$-\frac{1}{m}\sum_{i=1}^{m}\left(\left(y^{(i)}\bullet\frac{1}{g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)}-\left(1-y^{(i)}\right)\bullet\frac{1}{1-g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)}\right)g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\left(1-g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\right)\mathbf{x}^{(\mathbf{i})}\right)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(\left(y^{(i)}\left(1-g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\right)-\left(1-y^{(i)}\right)\bullet g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\right)\mathbf{x}^{(\mathbf{i})}\right)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(\left(y^{(i)}-g\left(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})}\right)\right)\mathbf{x}^{(\mathbf{i})}\right)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(\left(y^{(i)}-h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathbf{i})}\right)\right)\mathbf{x}^{(\mathbf{i})}\right)$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left(\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathbf{i})}\right)-y^{(i)}\right)\mathbf{x}^{(\mathbf{i})}\right)$$

# 参数的迭代

- 根据梯度下降原理，要求代价函数的最小值，迭代过程如下

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left( y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

}

此处可以没有1/m，参见代码

# 逻辑回归实现

- 逻辑回归的实现主要分为三个部分：读取和处理数据，梯度下降，可视化。

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def init_data():
    data = np.loadtxt('data.csv')
    dataMatIn = data[:, 0:-1]
    classLabels = data[:, -1]
    dataMatIn = np.insert(dataMatIn, 0, 1, axis=1)
    return dataMatIn, classLabels
```

# 逻辑回归实现

```
def grad_descent(dataMatIn, classLabels):
    dataMatrix = np.mat(dataMatIn)
    labelMat = np.mat(classLabels).transpose()
    m, n = np.shape(dataMatrix)
    weights = np.ones((n, 1))
    alpha = 0.001
    maxCycle = 500

    for i in range(maxCycle):
        h = sigmoid(dataMatrix * weights)
        weights = weights - alpha * dataMatrix.transpose() * (h - labelMat)
    return weights
```

# 作业

- 保留鸢尾花数据集的前两个类别，用逻辑回归进行分类，只能调用numpy、pandas、matpltlib等基础库。
- 加注释。
- 下周二上课前通过qq发给我，文件以学号加姓名命名。