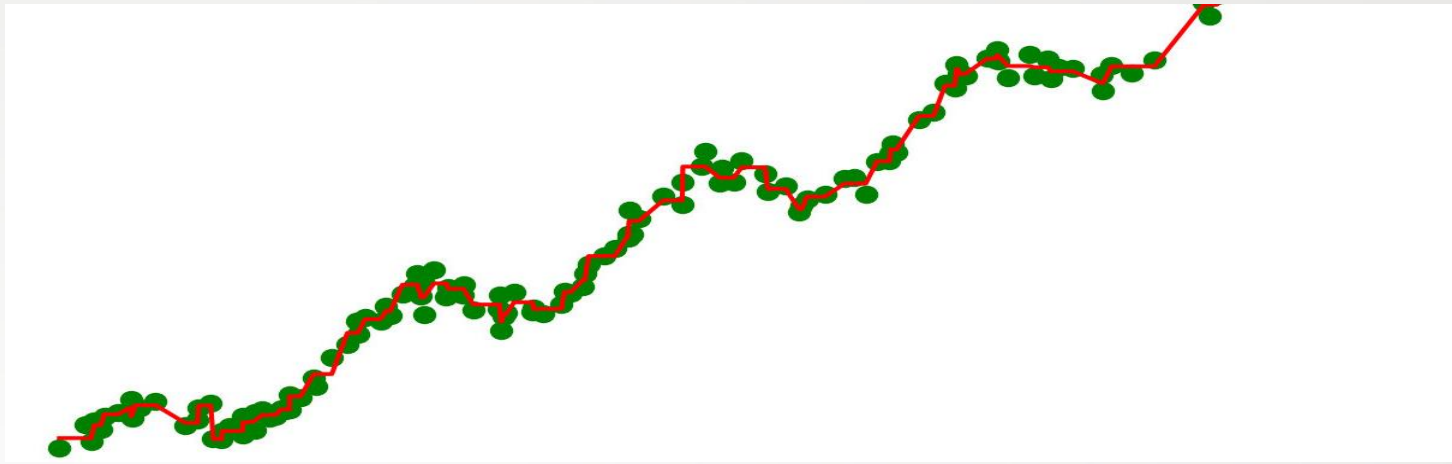
数值技术及管理应用

张晓洁 博士
zhangxiaojie@ouc.edu.cn

目录

- You can find and download the materials, including data and codes, on GitHub:

  https://github.com/JessicaXZ/BA2025.git
- Run the codes with Python.
- Important: You can NOT change the data and codes in this public repository!

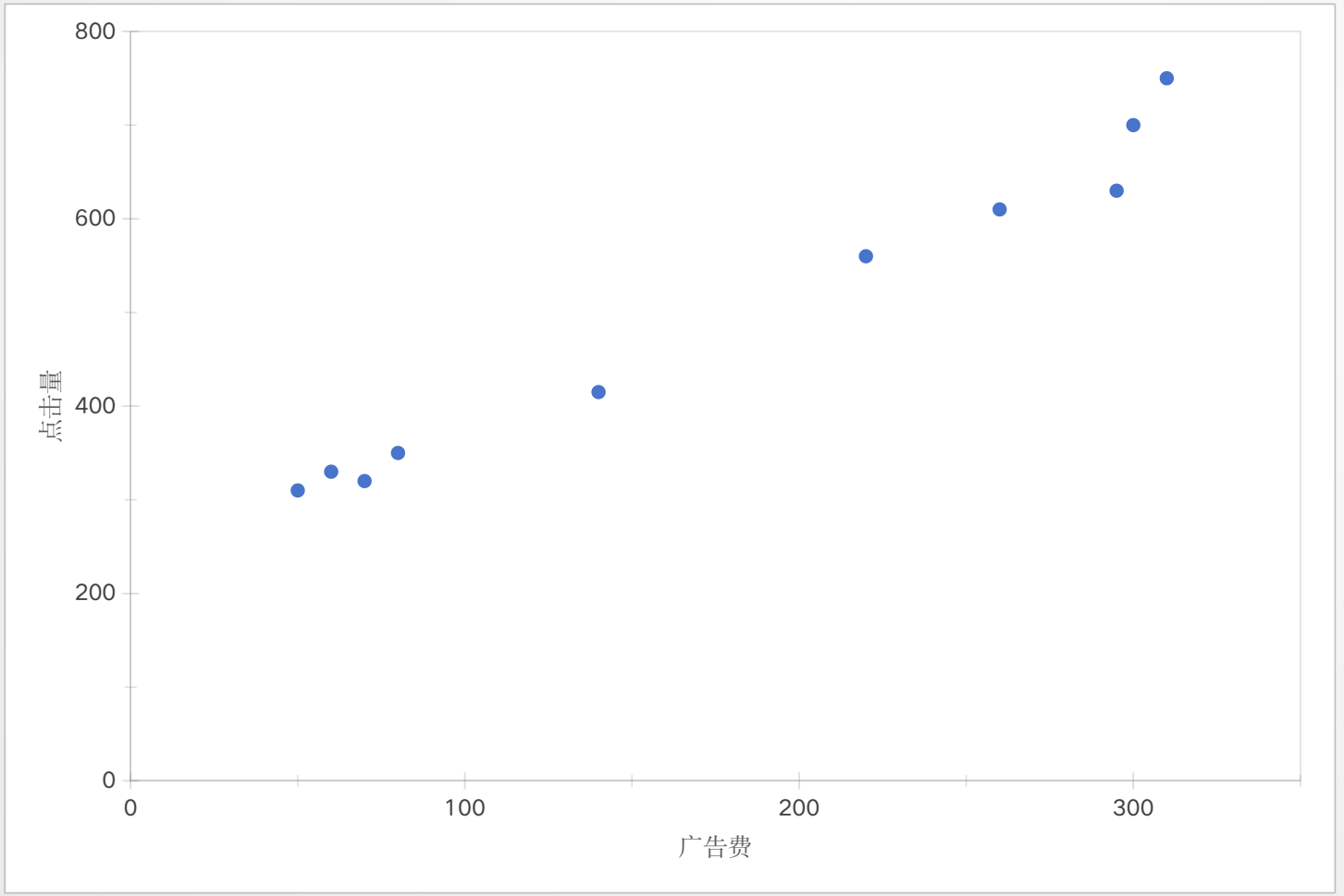


第一讲 回归
Regression Analysis

目录

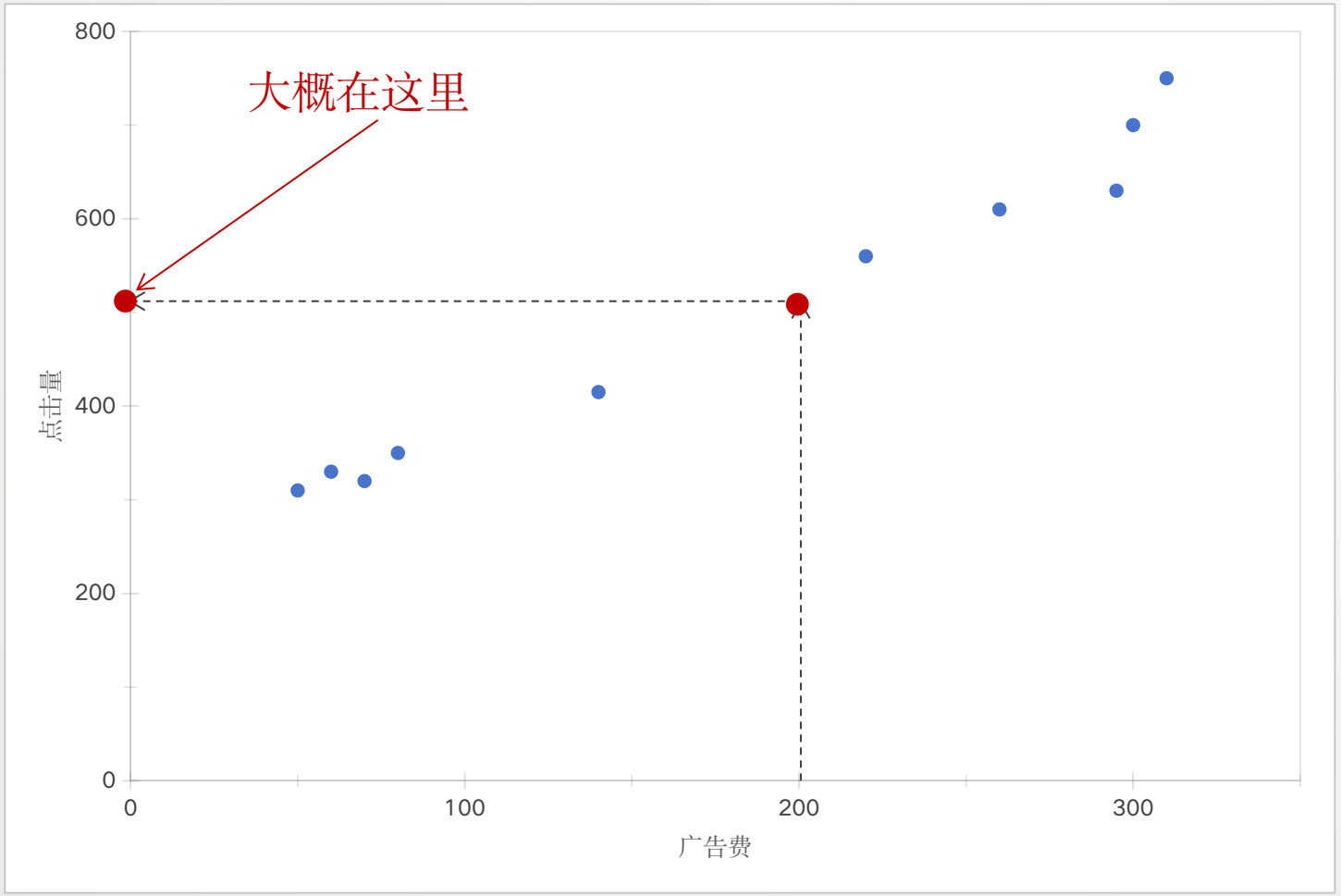
1.1 问题

- 投入网站的广告费与用户实际点击量的关系：



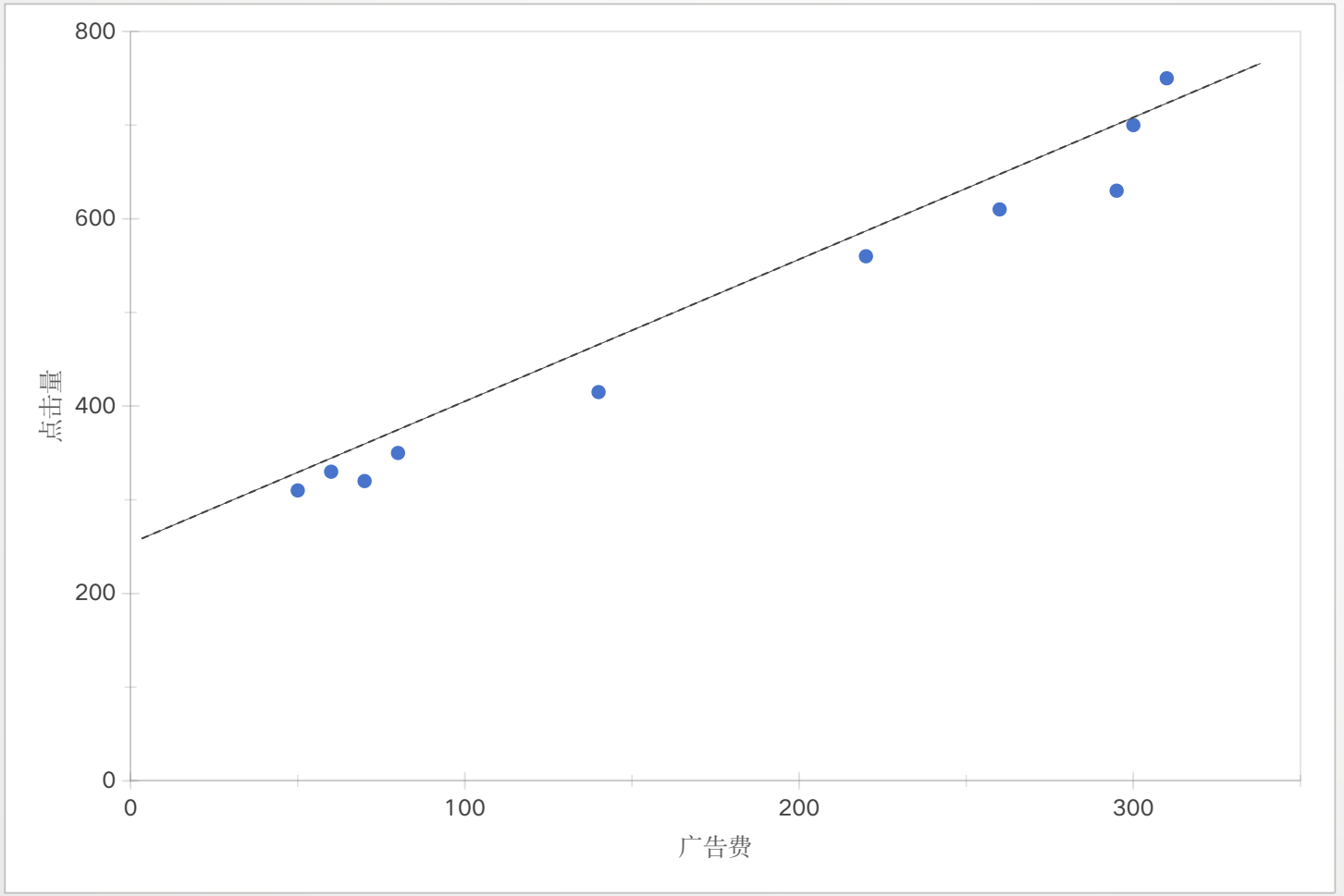
1.1 问题

- 问题：如果花了200的广告费，广告的点击量会是多少呢?



1.2 定义模型与最小二乘法

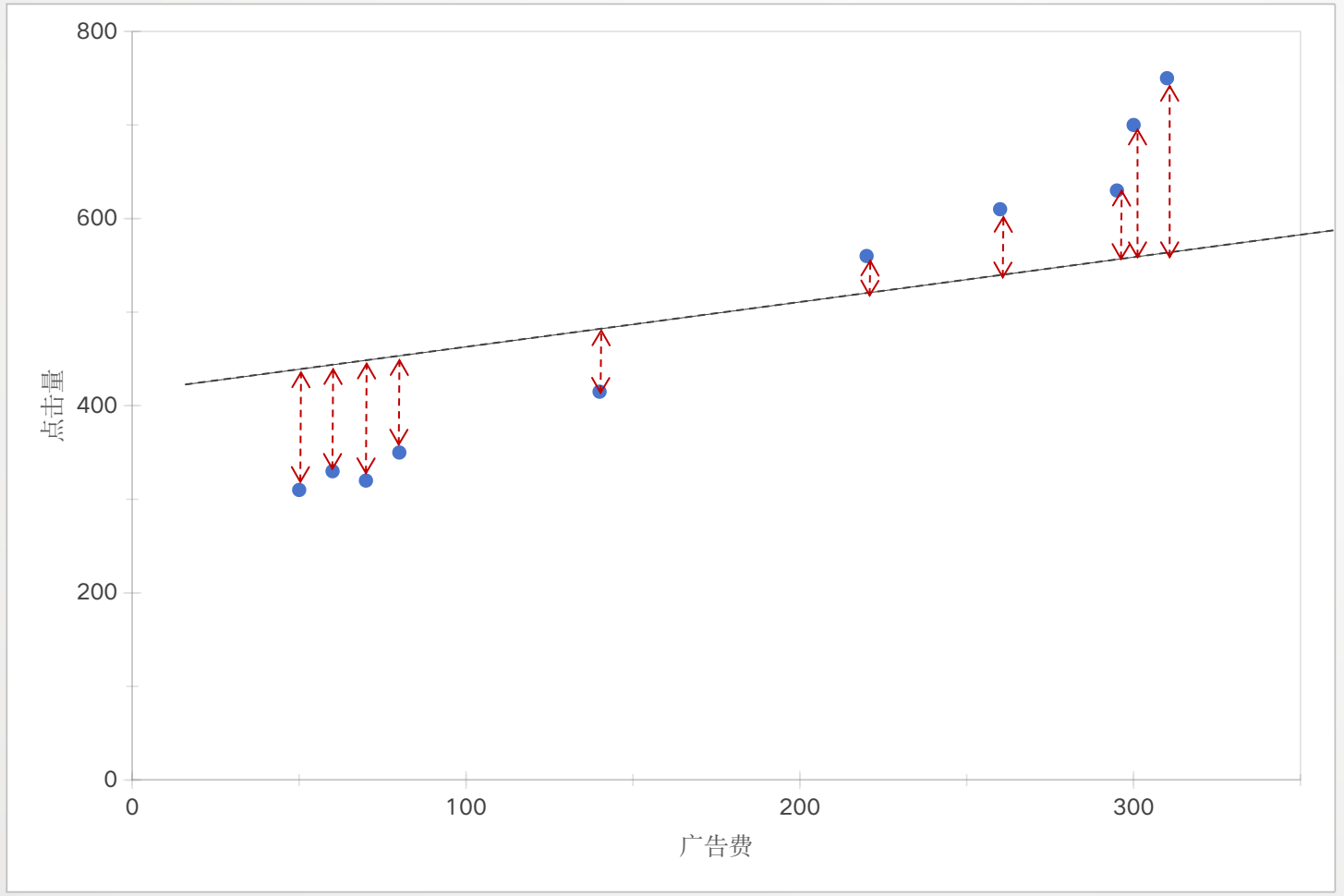
- 如果我们用机器学习的方法，应该怎么做呢?



1.2 定义模型与最小二乘法

- $y=ax+b$
- $f(x)=\theta_0+\theta_1 x$
- 参数求解：最小二乘法
- 目标函数：$E(\theta)=\frac{1}{2}\sum_{i=1}^{n}(y^i-f(x^i))^2$
  - 其中：y表示观测值，f(x)为预测值

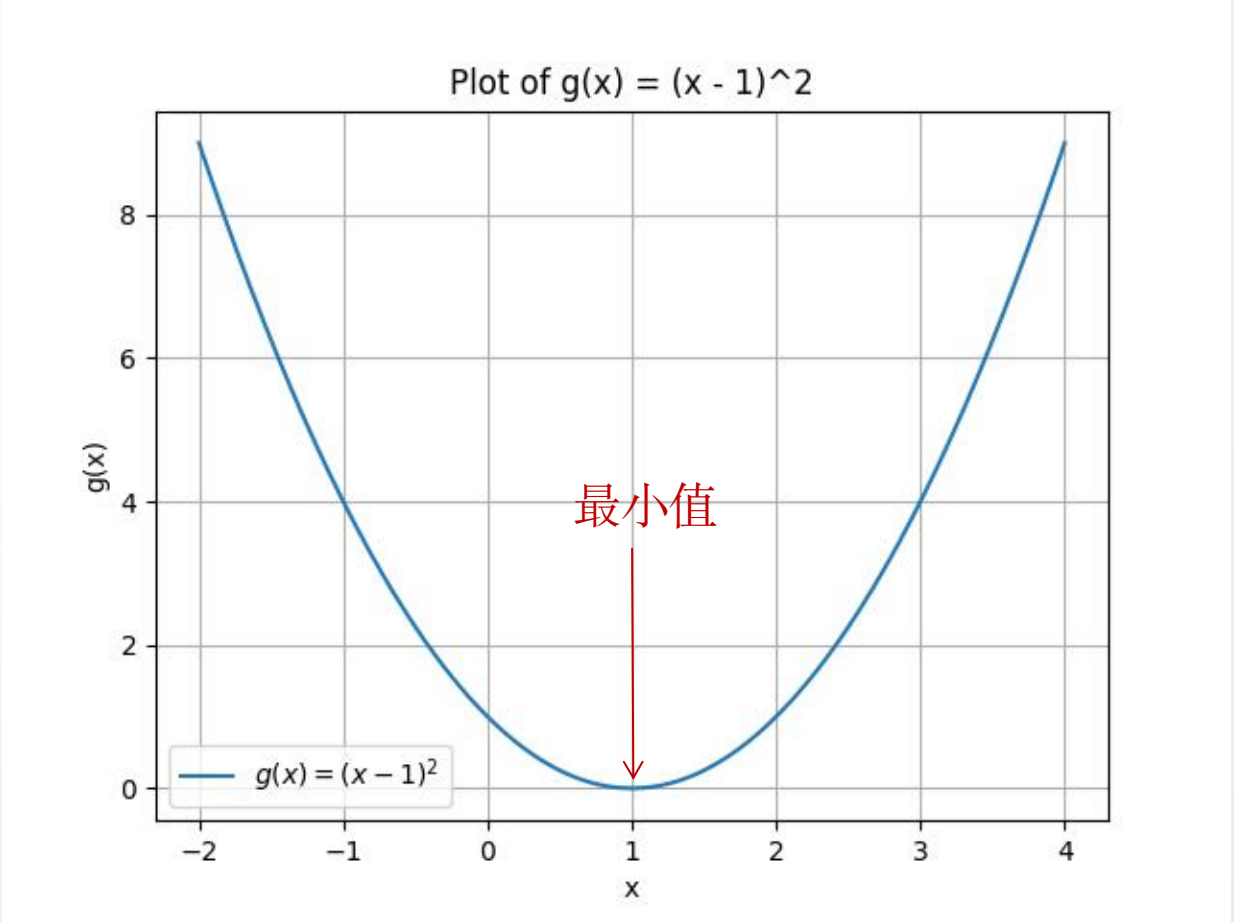
1.2 定义模型与最小二乘法



1.3 梯度下降法

- $E(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y^i - f(x^i))^2$

- 例子：$g(x) = (x-1)^2$



1.3 梯度下降法

- $\frac{d}{dx}g(x) = 2x - 2$

x	$\frac{d}{dx}g(x)$	$g(x)$的增减
$x < 1$	-	↘
$x = 1$	0	
$x > 1$	+	↗

1.3 梯度下降法

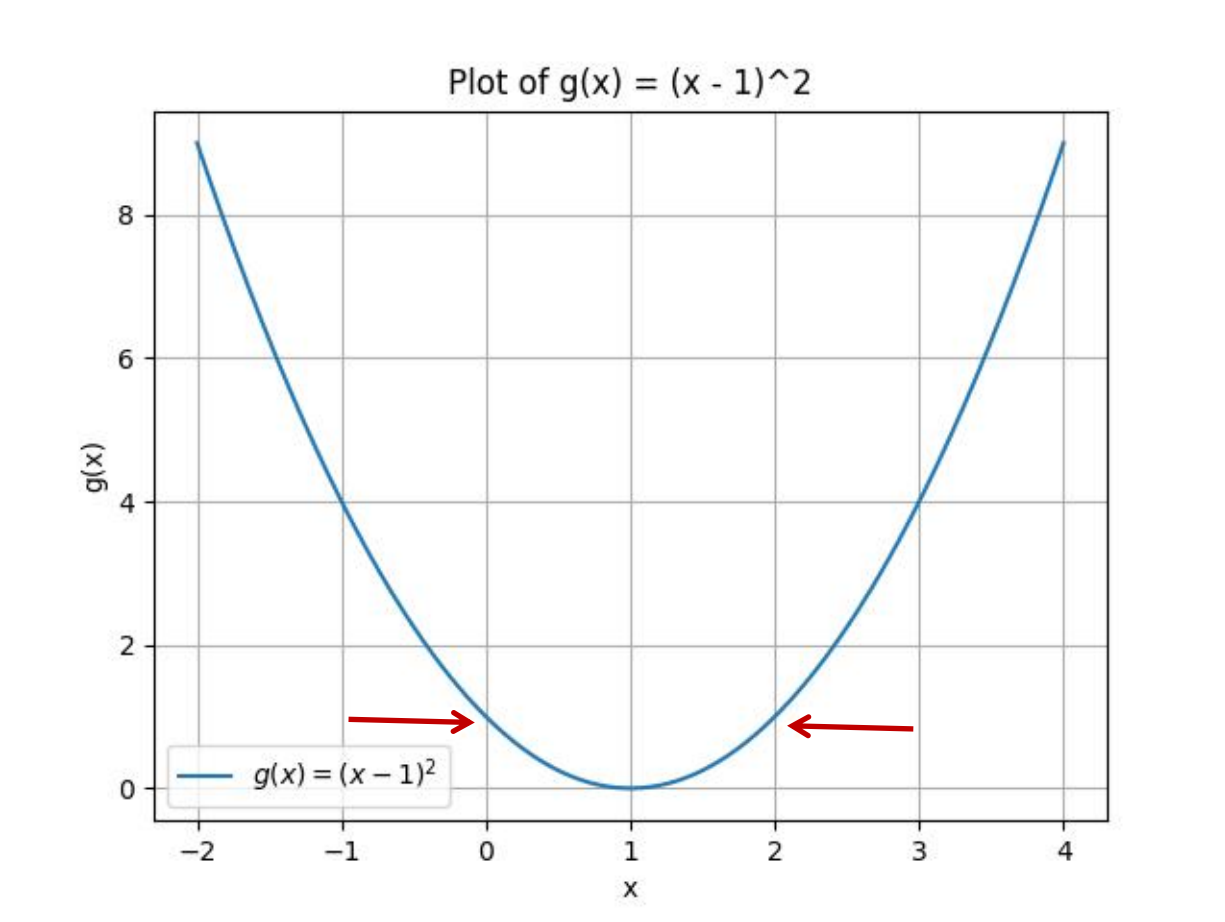
- 如果初始值 $x = 3$...
- 如果初始值 $x = -1$...

- $x := x - \eta \frac{d}{dx} g(x)$

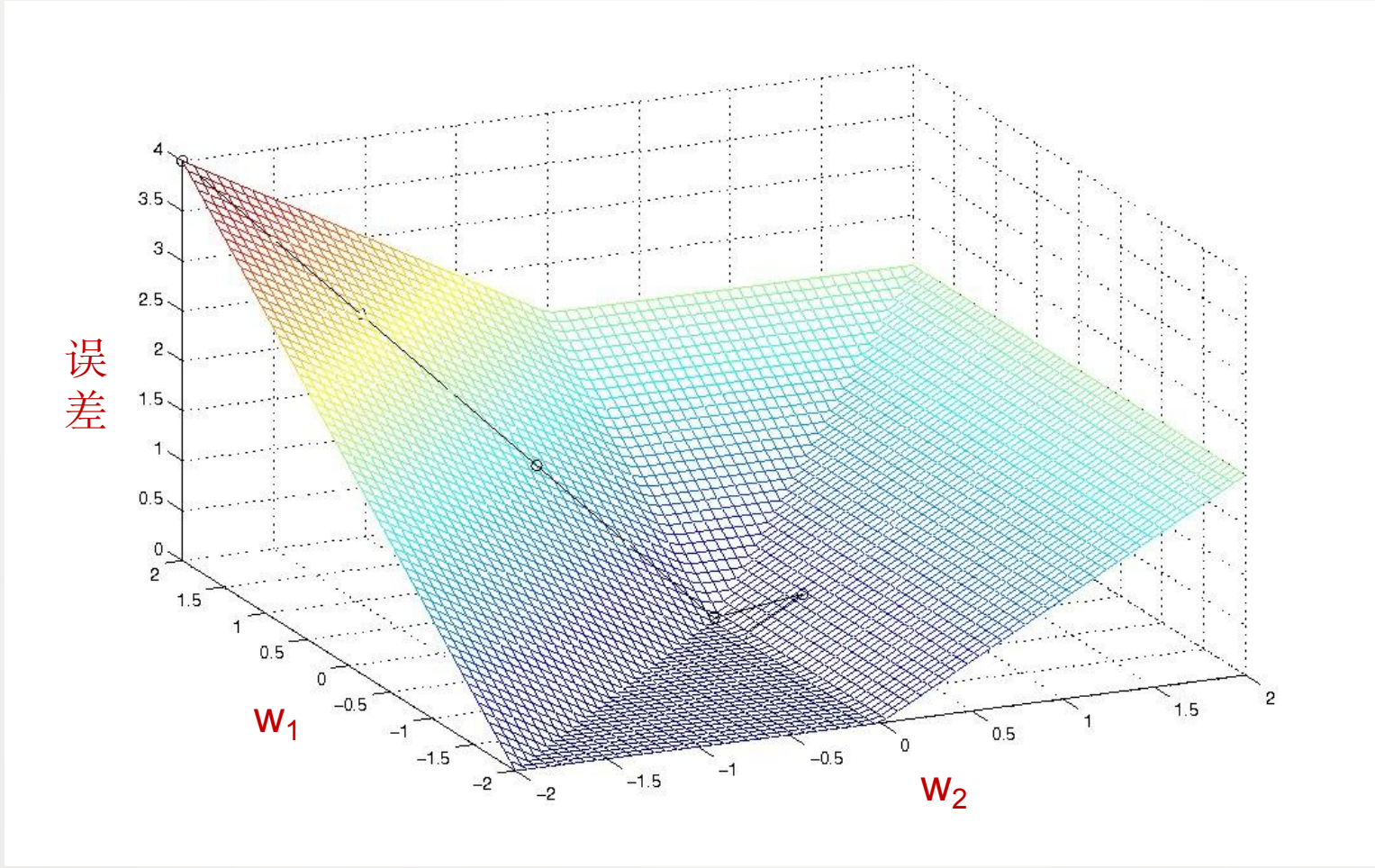
学习率
Learning Rate

- 最速下降法或梯度下降法

Steepest Descent Method/
Gradient Descent Method



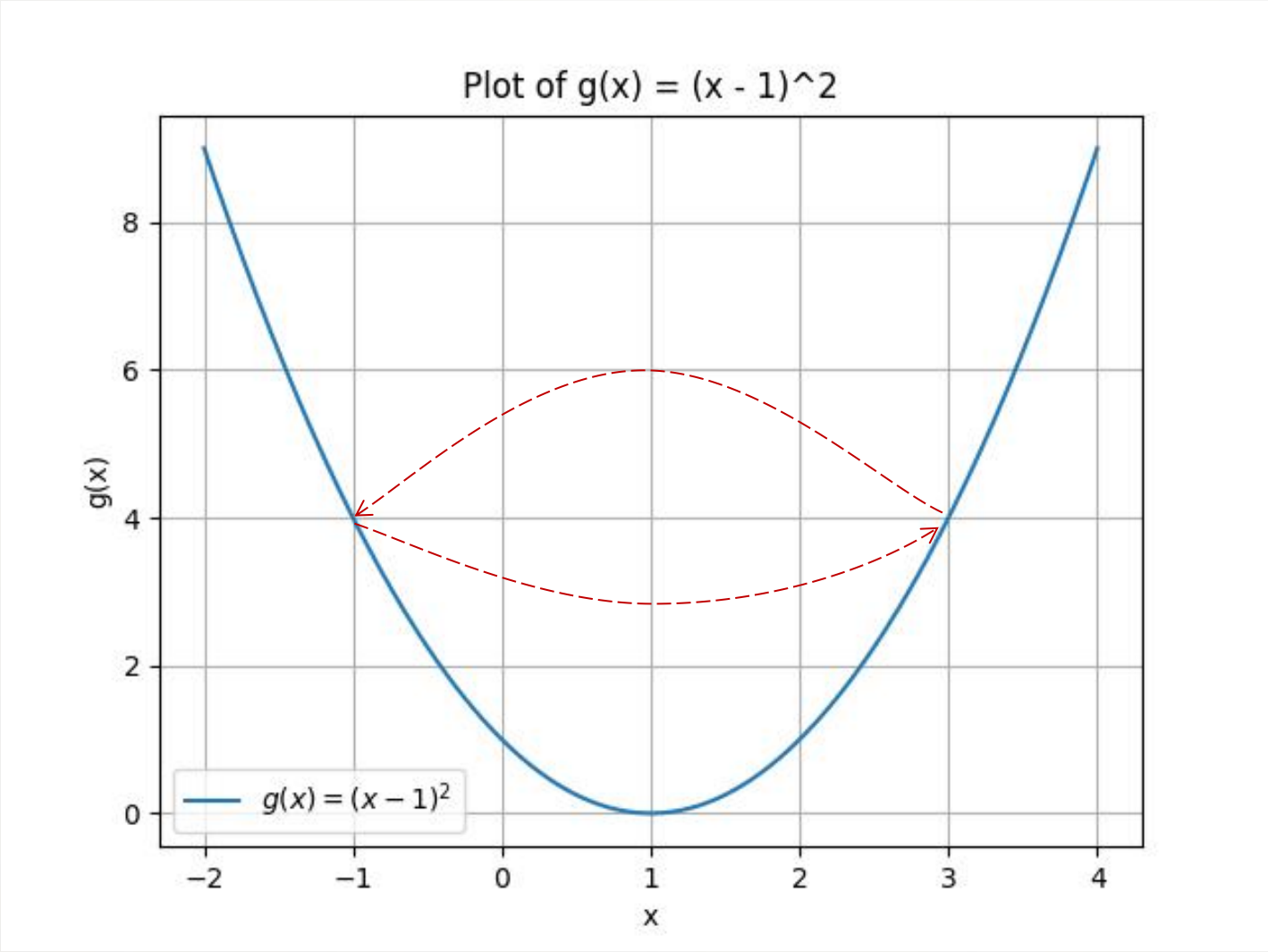
误差曲面



1.3 梯度下降法

- $x := x - \eta \frac{d}{dx} g(x)$
- 学习率η该如何取值？
- 对于函数g(x)，如果$\eta = 1$，从$x = 3$开始，x会如何变化？
- $x := x - \eta(2x - 2)$
- $x := 3 - (2 * 3 - 2) = 3 - 4 = -1$
- $x := -1 - (2 * (-1) - 2) = -1 + 4 = 3$
- $x := 3 - (2 * 3 - 2) = 3 - 4 = -1$

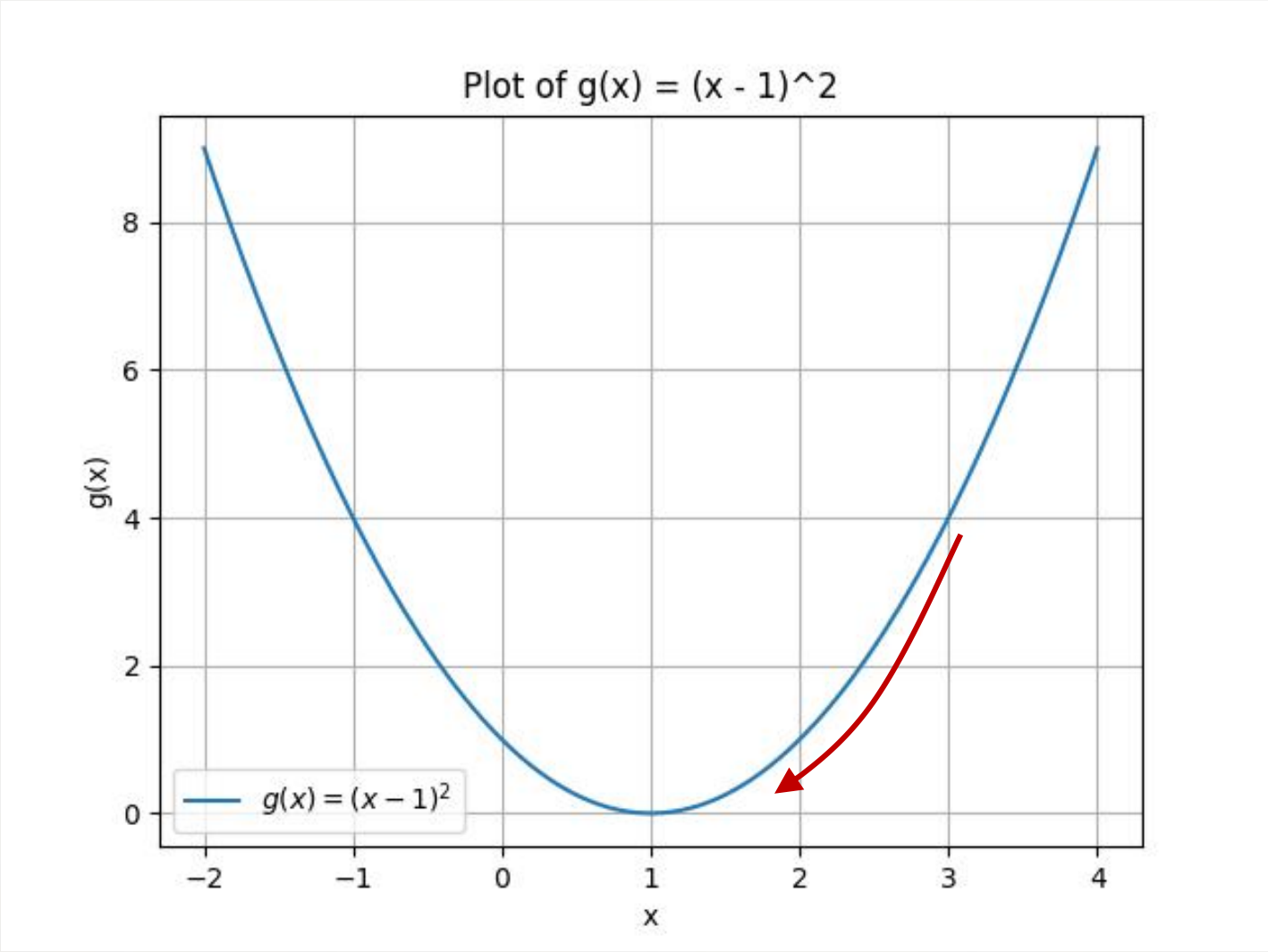
1.3 梯度下降法



1.3 梯度下降法

- $x := x - \eta \frac{d}{dx} g(x)$
- 学习率η该如何取值？
- 对于函数g(x)，如果$\eta = 0.1$，从$x = 3$开始，x会如何变化？
- $x := x - \eta(2x - 2)$
- $x := 3 - 0.1 * (2 * 3 - 2) = 3 - 0.4 = 2.6$
- $x := 2.6 - 0.1 * (2 * 2.6 - 2) = 2.6 - 0.3 = 2.3$
- $x := 2.3 - 0.1 * (2 * 2.3 - 2) = 2.3 - 0.2 = 2.1$
- $x := 2.1 - 0.1 * (2 * 2.1 - 2) = 2.1 - 0.2 = 1.9$

1.3 梯度下降法



1.3 梯度下降法

- $E(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y^i - f(x^i))^2$，其中$f(x) = \theta_0 + \theta_1 x$
- 如何更新参数θ_0和θ_1呢？
- 偏微分：
  - $\theta_0 := \theta_0 - \eta\frac{\partial E}{\partial \theta_0}$
  - $\theta_1 := \theta_1 - \eta\frac{\partial E}{\partial \theta_1}$

1.3 梯度下降法

- $E(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y^i - f(x^i))^2$，其中$f(x) = \theta_0 + \theta_1 x$
- $\theta_0 := \theta_0 - \eta\frac{\partial E}{\partial \theta_0}$
- $\theta_1 := \theta_1 - \eta\frac{\partial E}{\partial \theta_1}$
- 复合函数的偏微分问题
- $u = E(\theta)$
- $v = f(x)$
- $\frac{\partial u}{\partial \theta_0} = \frac{\partial u}{\partial v} * \frac{\partial v}{\partial \theta_0}$

1.3 梯度下降法

$$\frac{\partial u}{\partial \theta_0} = \frac{\partial u}{\partial v} * \frac{\partial v}{\partial \theta_0}$$

$$\frac{\partial u}{\partial v} = \frac{\partial}{\partial v}\left(\frac{1}{2}\sum_{i=1}^{n}(y^i - v)^2\right)$$
$$= \frac{1}{2}\sum_{i=1}^{n}\left(\frac{\partial}{\partial v}(y^i - v)^2\right)$$
$$= \frac{1}{2}\sum_{i=1}^{n}\left(\frac{\partial}{\partial v}(y^{i^2} - 2y^i v + v^2)\right)$$
$$= \frac{1}{2}\sum_{i=1}^{n}(-2y^i + 2v)$$
$$= \sum_{i=1}^{n}(v - y^i)$$

$$\frac{\partial v}{\partial \theta_0} = \frac{\partial}{\partial \theta_0}(\theta_0 + \theta_1 x)$$
$$= 1$$

1.3 梯度下降法

$$\frac{\partial u}{\partial \theta_0} = \frac{\partial u}{\partial v} * \frac{\partial v}{\partial \theta_0}$$
$$= \sum_{i=1}^{n} (v - y^i) * 1$$
$$= \sum_{i=1}^{n} (f(x^i) - y^i)$$

类似地，

$$\frac{\partial u}{\partial \theta_1} = \frac{\partial u}{\partial v} * \frac{\partial v}{\partial \theta_1}$$
$$= \sum_{i=1}^{n} (f(x^i) - y^i) x^i$$

1.3 梯度下降法

- $E(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y^i - f(x^i))^2$，其中$f(x) = \theta_0 + \theta_1 x$

- $\theta_0 := \theta_0 - \eta\sum_{i=1}^{n}(f(x^i) - y^i)$
- $\theta_1 := \theta_1 - \eta\sum_{i=1}^{n}(f(x^i) - y^i)x^i$

1.3 梯度下降法

- 如果拓展到一元二次回归：

- $f(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

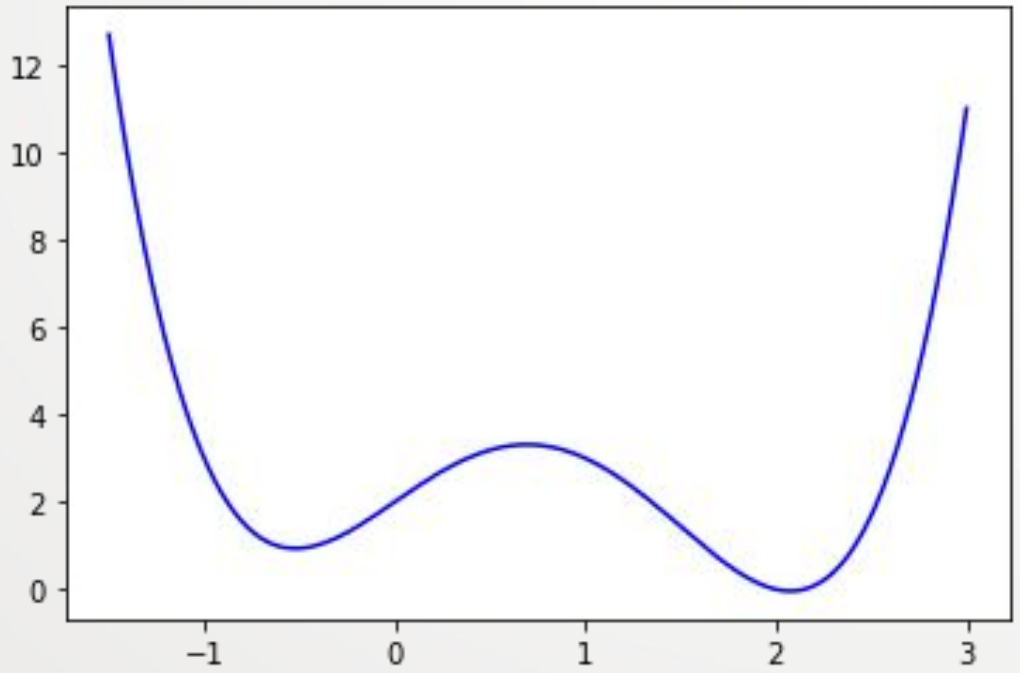
- $\theta_0 := \theta_0 - \eta \sum_{i=1}^{n} (f(x^i) - y^i)$
- $\theta_1 := \theta_1 - \eta \sum_{i=1}^{n} (f(x^i) - y^i) x^i$
- $\theta_2 := \theta_2 - \eta \sum_{i=1}^{n} (f(x^i) - y^i) (x^i)^2$

1.3 梯度下降法

- 如果拓展到多元回归:
- $f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$
- $\theta_j := \theta_j - \eta \sum_{i=1}^{n} (f(x^i) - y^i) x_j^i$

1.3 梯度下降法

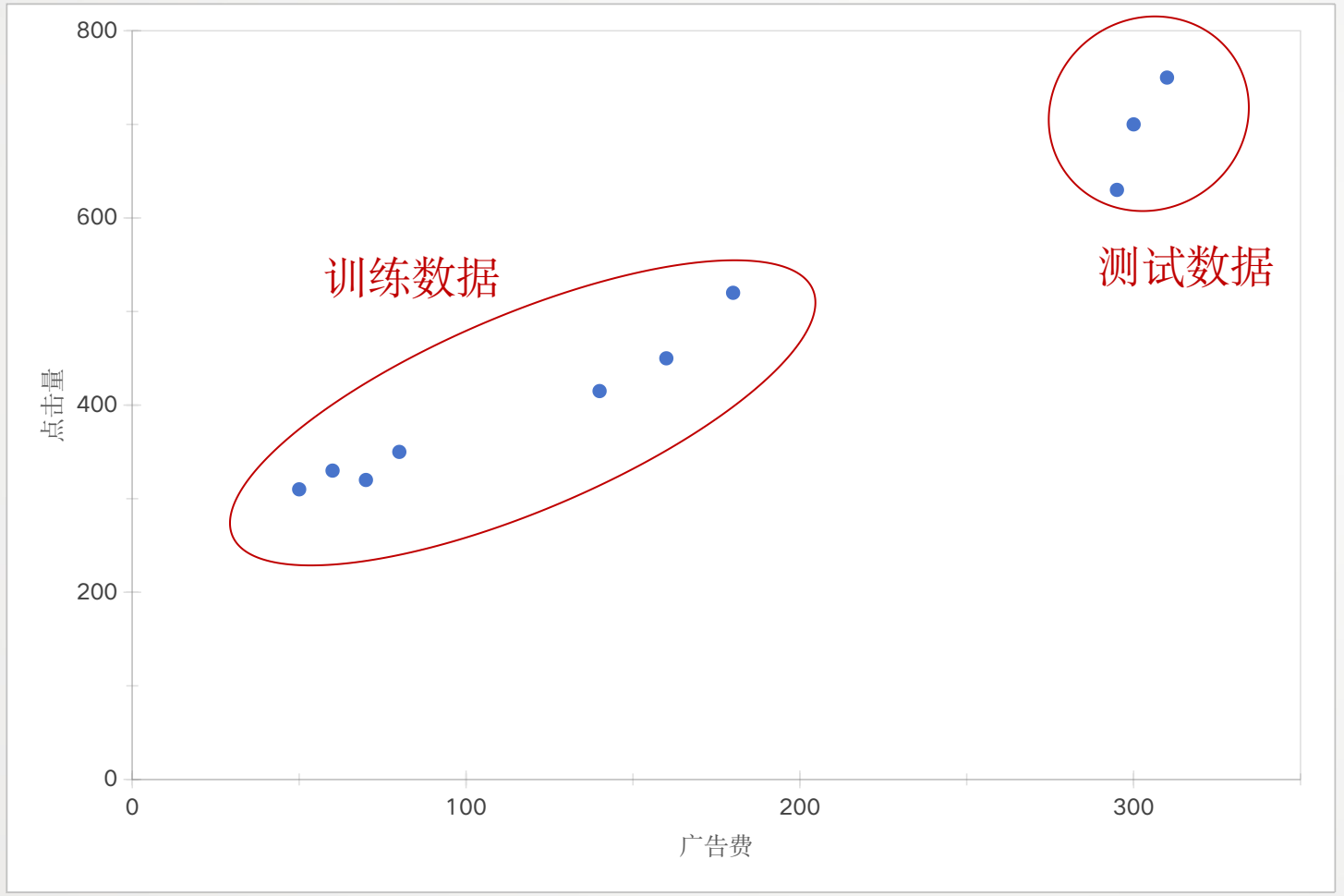
- 最速下降法的问题：
  - 训练数据越多，循环的次数越多，计算效率下降；
  - 容易陷入局部最优解。



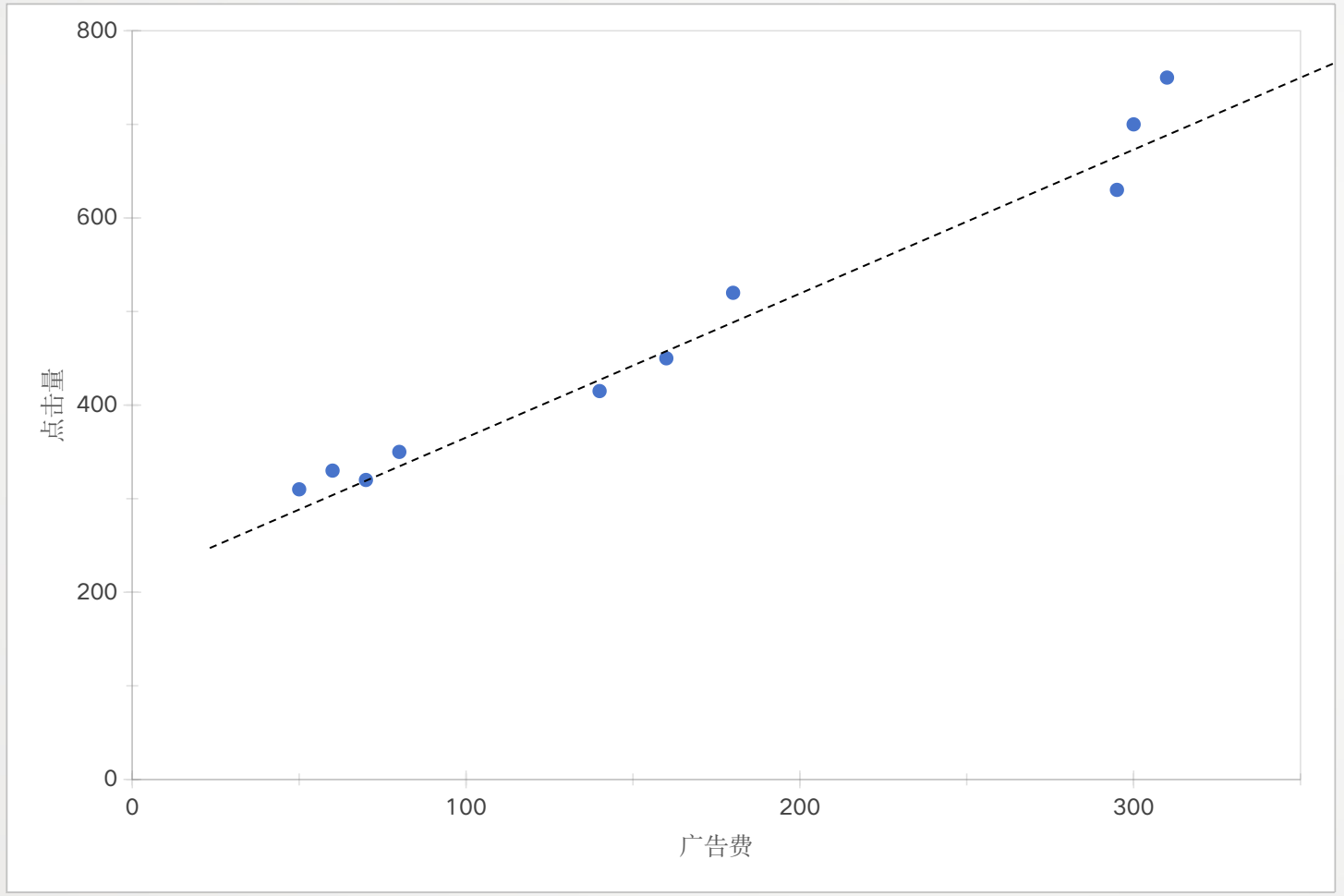
1.3 梯度下降法

- 梯度下降法
  - $\theta_j := \theta_j - \eta \sum_{i=1}^{n} (f(x^i) - y^i) x_j^i$
- 随机梯度下降法：随机选择一个数据更新参数
  - $\theta_j := \theta_j - \eta (f(x^k) - y^k) x_j^k$
- 小批量(*mini-batch*)梯度下降法：随机选择m个训练数据的集合K更新参数
  - $\theta_j := \theta_j - \eta \sum_{k \in K} (f(x^k) - y^k) x_j^k$
  - 假设训练数据有100个，那么在$m = 10$时，创建一个有10个随机数的集合，例如$K = \{61, 53, 59, 16, 30, 21, 85, 31, 51, 10\}$

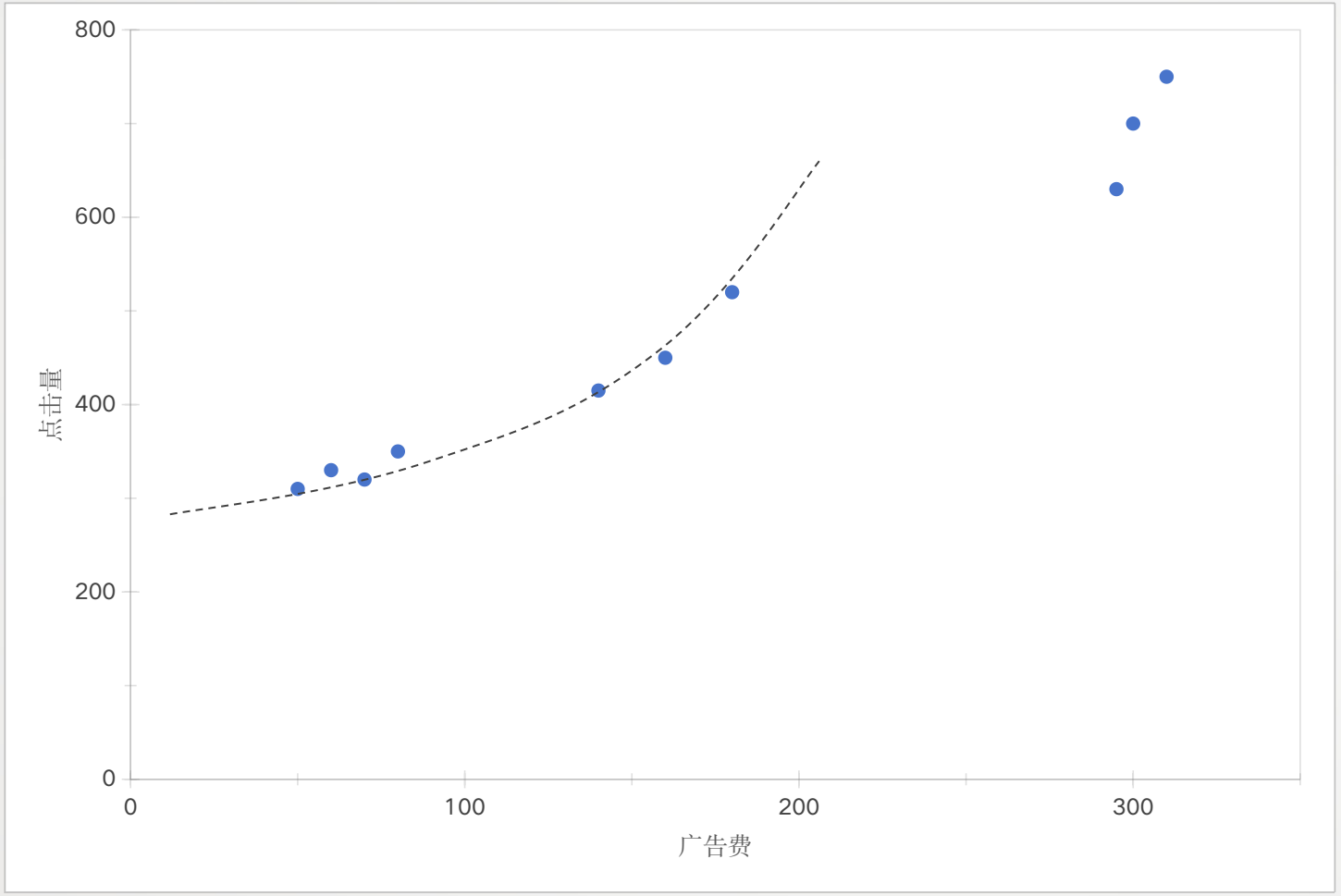
1.4 模型评估



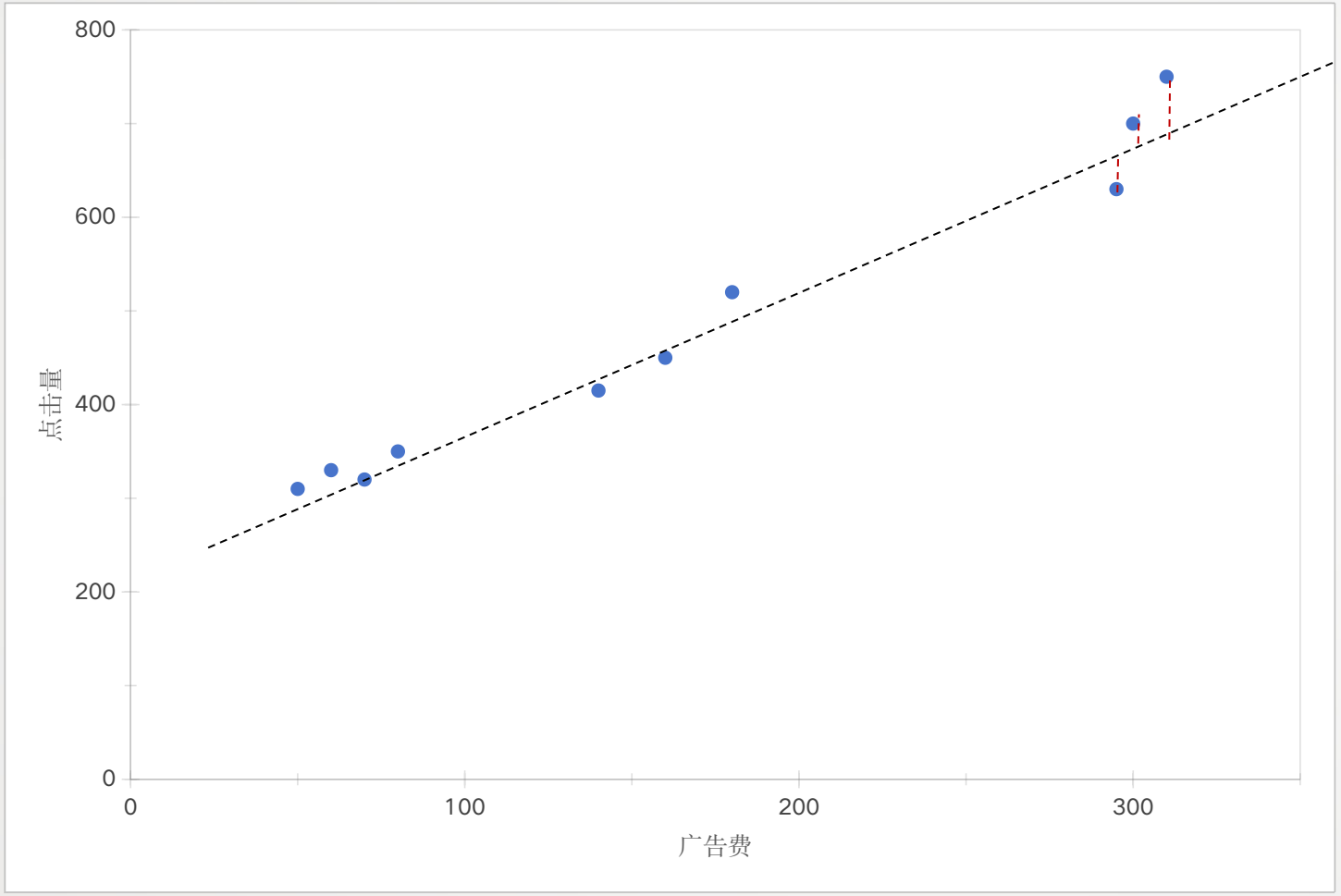
1.4 模型评估



1.4 模型评估



1.4 模型评估

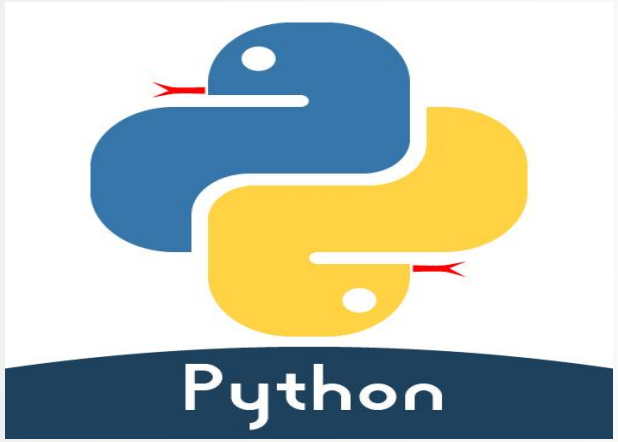


1.4 模型评估

- $MSE = \frac{1}{n}\sum_{i=1}^{n}(y^i - f(x^i))^2$
  - where MSE represents for Mean Squared Error

- 误差越小，模型适配度越高

1.5 Python实现

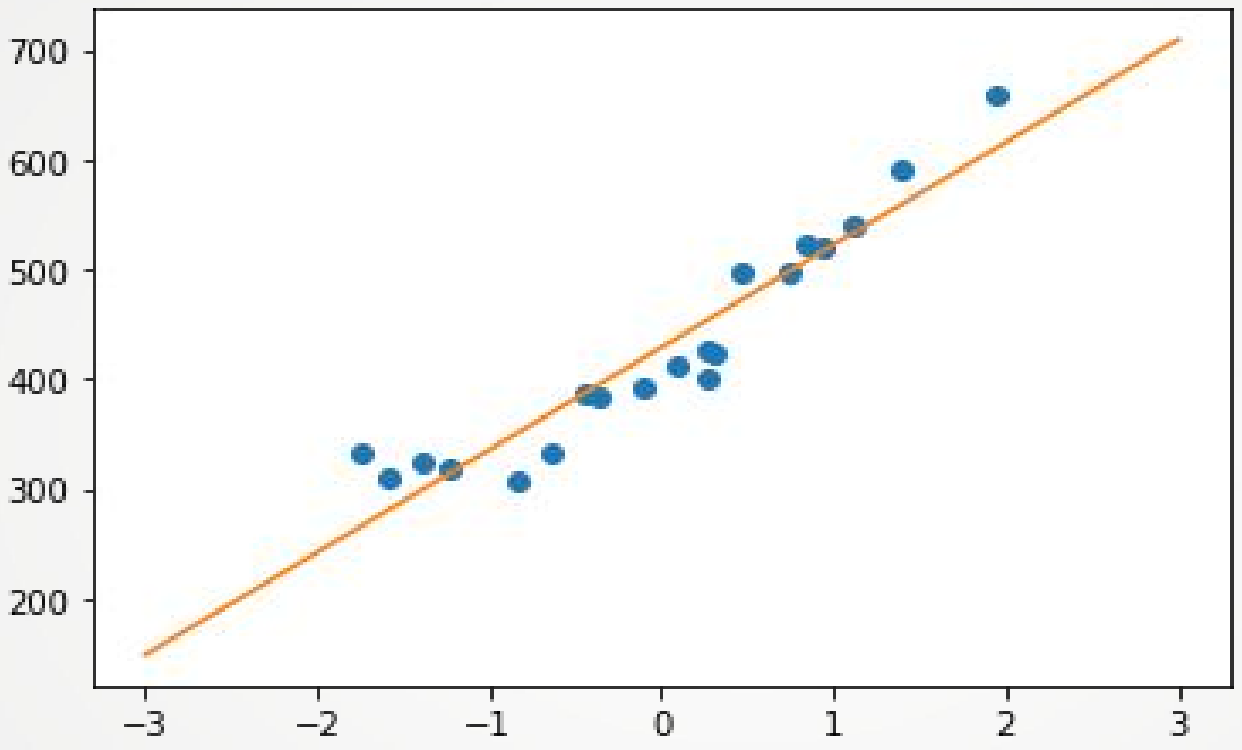


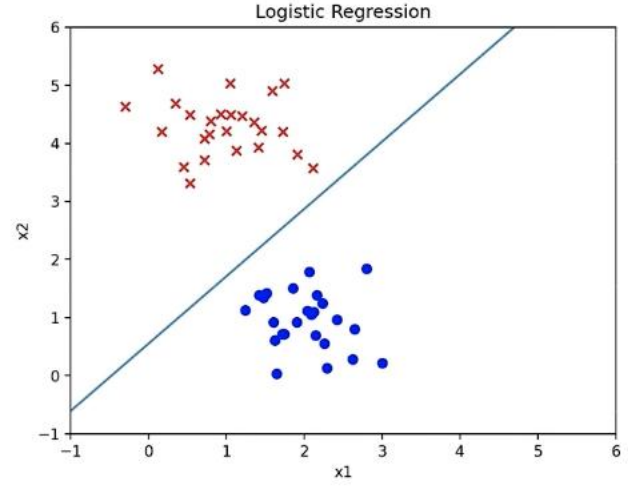
Python 3.7.4 Shell

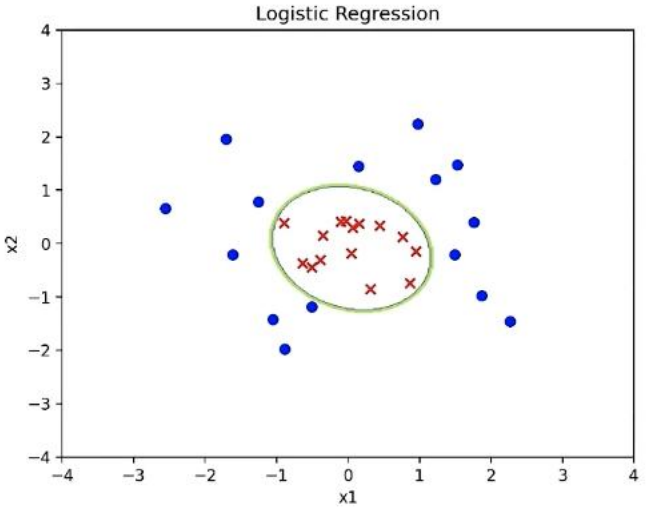
```
Python 3.7.4 (v3.7.4:e09359112e, Jul  8 2019, 14:54:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> #读入训练数据，根据你的文件存储路径做出相应的修改
>>> train = np.loadtxt('/Users/xiaokeai/Desktop/click.csv',delimiter=',',skiprows=1)
>>> train_x = train[:,0]
>>> train_y = train[:,1]
>>> #参数初始化
>>> theta0=np.random.rand()
>>> theta1=np.random.rand()
>>> #预测函数
>>> def f(x):
        return theta0+theta1*x

>>> #目标函数
>>> def E(x,y):
        return 0.5*np.sum(y-f(x)**2)

>>>  #标准化
>>> mu=train_x.mean()
>>> sigma=train_x.std()
>>> def standardize(x):
        return (x-mu)/sigma

>>> train_z=standardize(train_x)
>>> plt.plot(train_z,train_y,'o')
[<matplotlib.lines.Line2D object at 0x7fc231e00cd0>]
>>> plt.show()
>>> #学习率
>>> ETA=1e-3
>>> #误差的差值，初始化为1
>>> diff=1
>>> #更新的次数，初始化为0
>>> count=0
>>> #重复学习
>>> error=E(train_z,train_y)
>>> while diff>1e-2:
        #更新结果保存到临时变量
        tmp0=theta0-ETA*np.sum((f(train_z)-train_y))
        tmp1=theta1-ETA*np.sum((f(train_z)-train_y)*train_z)
        #更新参数
        theta0=tmp0
        theta1=tmp1
        #计算与上一次误差的差值
        current_error=E(train_z,train_y)
        diff=error-current_error
        error=current_error
        #输出日志
        count += 1
        log='第{}次: theta0={:.3f},theta1={:.3f},差值={:.4f}'
        print(log.format(count,theta0,theta1,diff))
```

Ln: 26 Col: 0

```
第765次: theta0=429.150,theta1=93.479,差值=0.0153
第766次: theta0=429.150,theta1=93.479,差值=0.0149
第767次: theta0=429.150,theta1=93.479,差值=0.0146
第768次: theta0=429.150,theta1=93.479,差值=0.0144
第769次: theta0=429.150,theta1=93.479,差值=0.0141
第770次: theta0=429.150,theta1=93.479,差值=0.0138
第771次: theta0=429.150,theta1=93.479,差值=0.0135
第772次: theta0=429.150,theta1=93.479,差值=0.0132
第773次: theta0=429.150,theta1=93.479,差值=0.0130
第774次: theta0=429.150,theta1=93.479,差值=0.0127
第775次: theta0=429.150,theta1=93.479,差值=0.0125
第776次: theta0=429.150,theta1=93.479,差值=0.0122
第777次: theta0=429.150,theta1=93.479,差值=0.0120
第778次: theta0=429.150,theta1=93.479,差值=0.0117
第779次: theta0=429.150,theta1=93.479,差值=0.0115
第780次: theta0=429.150,theta1=93.479,差值=0.0113
第781次: theta0=429.150,theta1=93.479,差值=0.0110
第782次: theta0=429.150,theta1=93.479,差值=0.0108
第783次: theta0=429.150,theta1=93.479,差值=0.0106
第784次: theta0=429.150,theta1=93.479,差值=0.0104
第785次: theta0=429.150,theta1=93.479,差值=0.0102
第786次: theta0=429.150,theta1=93.479,差值=0.0100
>>>
```


700
600
500
400
300
200
−3
−2
−1
0
1
2
3





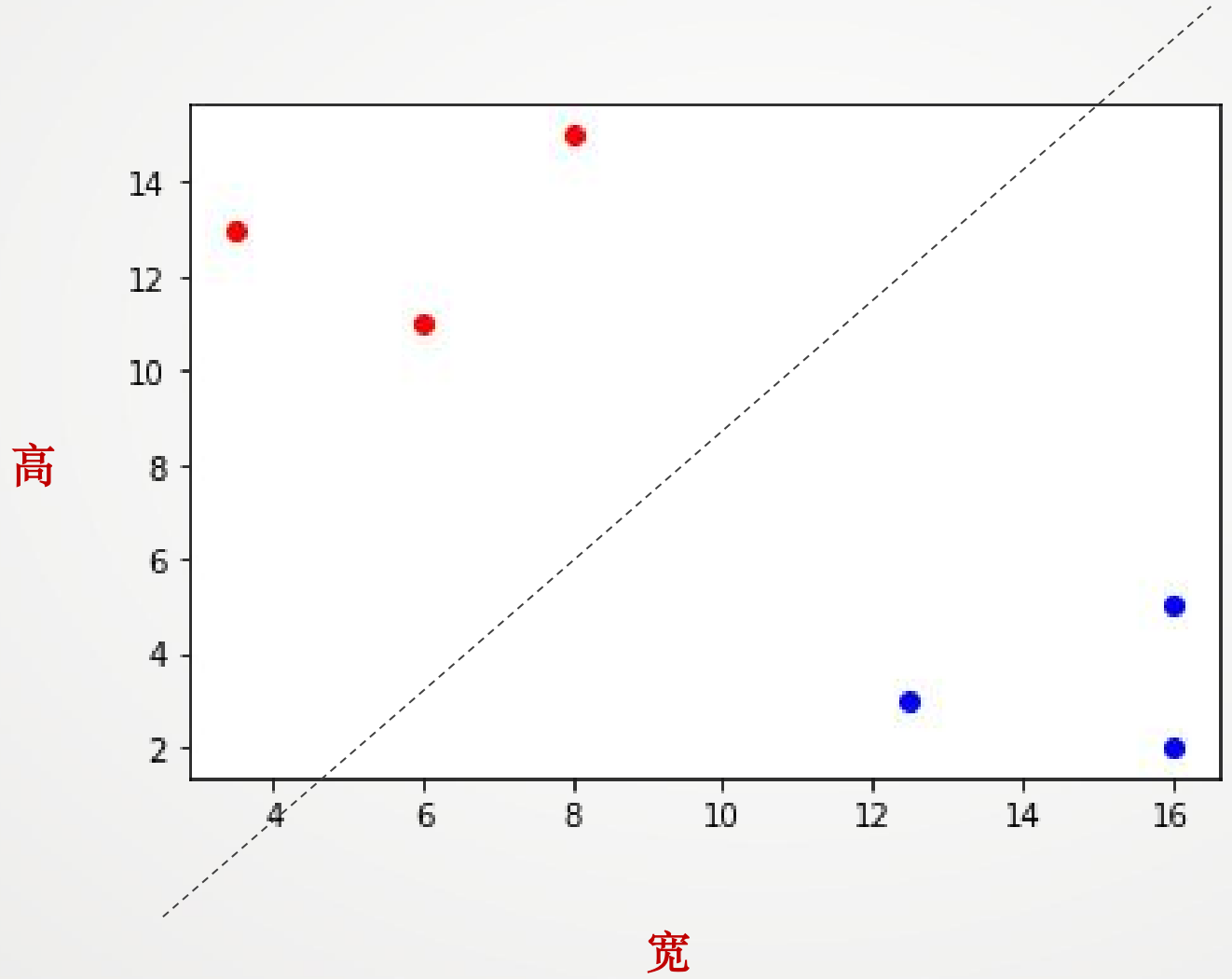
第二讲 逻辑回归
Logit Regression

目录

3.1 问题：二分类问题

图像大小	形状	宽(x1)	高(x2)	y
8*15	纵向	8	15	0
6*11	纵向	6	11	0
3.5*13	纵向	3.5	13	0
16*5	横向	16	5	1
16*2	横向	16	2	1
12.5*3	横向	12.5	3	1

3.1 问题：二分类问题



3.2 定义模型——逻辑回归

逻辑回归是解决分类问题的一种模型。根据数据特征或属性，计算其归属于某一类别的概率P(x)，根据概率数值判断其所属类别。

3.2 定义模型

- 线性回归：$g(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$
  - ✓ 在本案例中：x_1表示图片的宽，x_2表示图片的高。

- Sigmoid函数
  - ✓ 表达式：$P(z) = \frac{1}{1+e^{-z}}$
  - ✓ $\frac{dP(z)}{dz} = P(z) * (1 - \mathrm{P(z)})$

3.2 定义模型

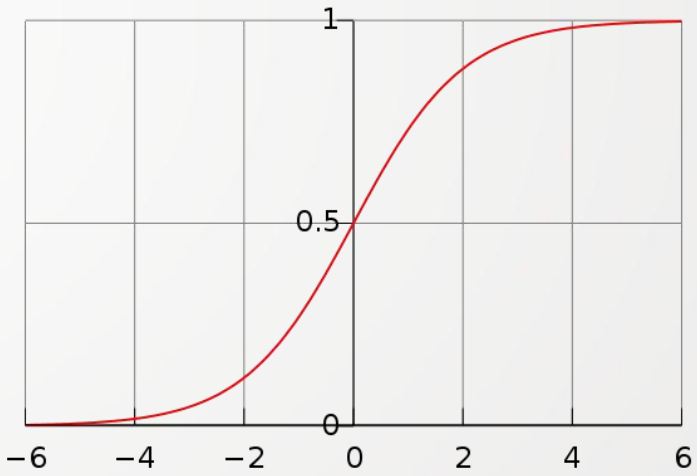
- 将线性回归$z = g(x)$带入Sigmoid函数:

$$f(x) = \frac{1}{1 + exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

3.2 定义模型——决策边界

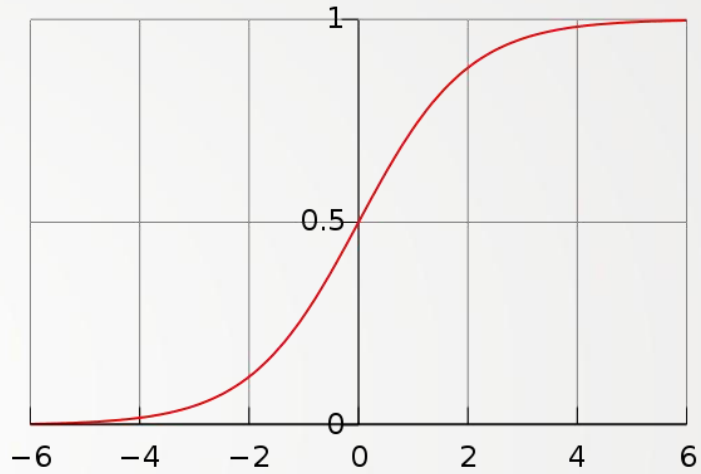
- 假设 $P(y=1|x)=f_\theta(x)$
  - 给定x时，y=1（图像为横向）的概率。
- 如果$f_\theta(x)$=0.7，图像为横向or纵向？
- 如果$f_\theta(x)$=0.2，图像为横向or纵向？

- $y=\begin{cases}1 & f(x)\geq 0.5\\ 0 & f(x)<0.5\end{cases}$



3.2 定义模型——决策边界

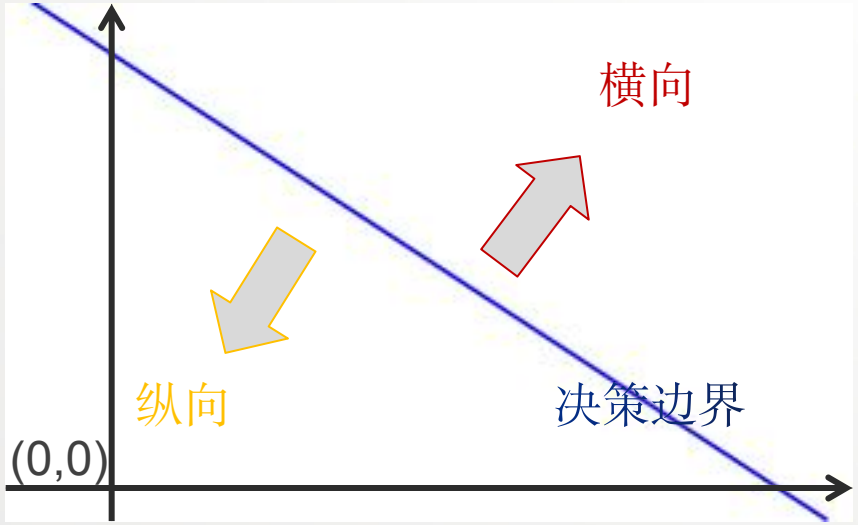
- $y = \begin{cases} 1 & f(x) \geq 0.5 \\ 0 & f(x) < 0.5 \end{cases}$

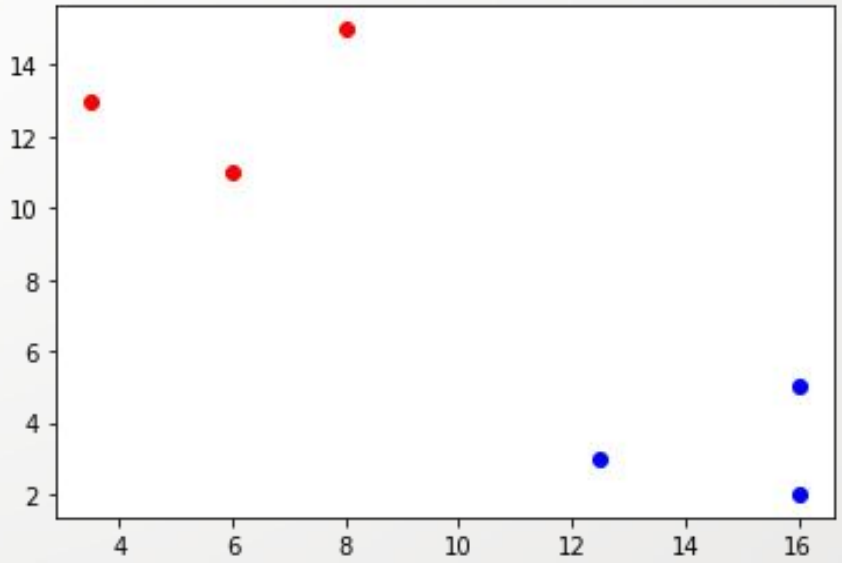


- $y = \begin{cases} 1 & \boldsymbol{\theta}^T \boldsymbol{x} \geq 0 \\ 0 & \boldsymbol{\theta}^T \boldsymbol{x} < 0 \end{cases}$

3.2 定义模型——决策边界

- $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -100 \\ 2 \\ 1 \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$
- $\boldsymbol{\theta}^T \boldsymbol{x} = -100 + 2x_1 + x_2 \geq 0$





参数如何更新？

3.3 模型求解：似然函数

图像大小	形状	y	概率
8*15	纵向	0	期待P(y=0|x)最大
6*11	纵向	0	期待P(y=0|x)最大
3.5*13	纵向	0	期待P(y=0|x)最大
16*5	横向	1	期待P(y=1|x)最大
16*2	横向	1	期待P(y=1|x)最大
12.5*3	横向	1	期待P(y=1|x)最大

3.3 模型求解：似然函数

- 假定所有的训练数据都互不影响、相互独立，整体概率为联合概率
- $L(\theta) = P(y^{(1)} = 0|x^{(1)})P(y^{(2)} = 0|x^{(2)})\ldots P(y^{(6)} = 1|x^{(6)})$

- 似然函数
- $L(\theta) = \prod_{i=1}^{n} P(y^{(i)} = 1|x^{(i)})^{y^{(i)}} P(y^{(i)} = 0|x^{(i)})^{1-y^{(i)}}$

3.3 模型求解：对数似然函数

- 似然函数$L(\theta)$等式右侧中单项的概率都是1以下的数值，联合概率的值会越来越小，编程时容易出现精度问题。

- 解决办法：对数似然函数
- $logL(\theta) = log\prod_{i=1}^{n} P(y^{(i)} = 1|x^{(i)})^{y^{(i)}} P(y^{(i)} = 0|x^{(i)})^{1-y^{(i)}}$

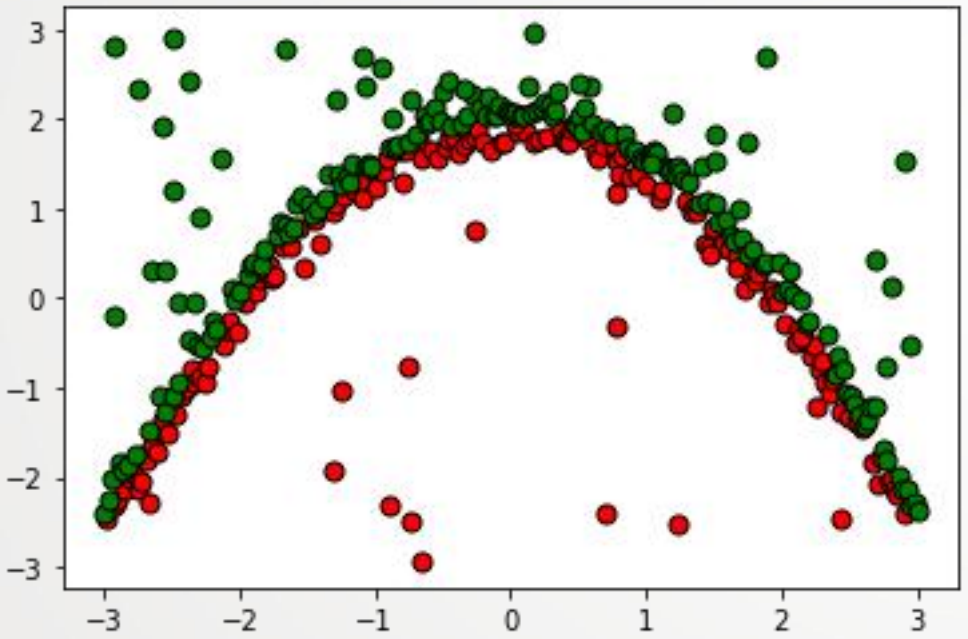
3.3 模型求解：对数似然函数

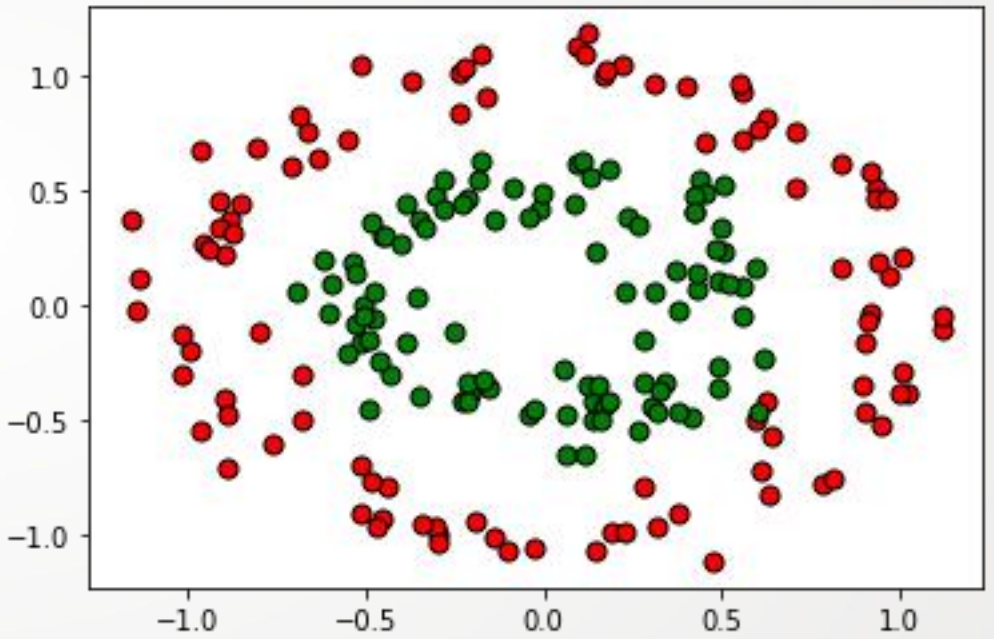
- $logL(\theta) = log\prod_{i=1}^{n} P(y^{(i)} = 1|x^{(i)})^{y^{(i)}} P(y^{(i)} = 0|x^{(i)})^{1-y^{(i)}}$
- $= \sum_{i=1}^{n} (\log P(y^{(i)} = 1|x^{(i)})^{y^{(i)}} + \log P(y^{(i)} = 0|x^{(i)})^{1-y^{(i)}})$
- $= \sum_{i=1}^{n} (y^{(i)} \log P(y^{(i)} = 1|x^{(i)}) + (1 - y^{(i)}) \log P(y^{(i)} = 0|x^{(i)}))$
- $= \sum_{i=1}^{n} (y^{(i)} \log P(y^{(i)} = 1|x^{(i)}) + (1 - y^{(i)}) log(1 - P(y^{(i)} = 1|x^{(i)})))$
- $= \sum_{i=1}^{n} (y^{(i)} \log f_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - f_{\theta}(x^{(i)})))$

3.3 模型求解：对数似然函数

- $\frac{\partial logL(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} (y^{(i)} \log f_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - f_\theta(x^{(i)})))$
- $= \sum_{i=1}^{n} \left(y^{(i)} - f_\theta(x^i) \right) x_j^{(i)}$
- θ_j的更新表达式：
- $\theta_j = \theta_j + \eta * \frac{\partial logL(\theta)}{\partial \theta_j}$
- $= \theta_j + \eta * \sum_{i=1}^{n} \left(y^{(i)} - f_\theta(x^i) \right) x_j^{(i)}$

3.4 线性不可分问题



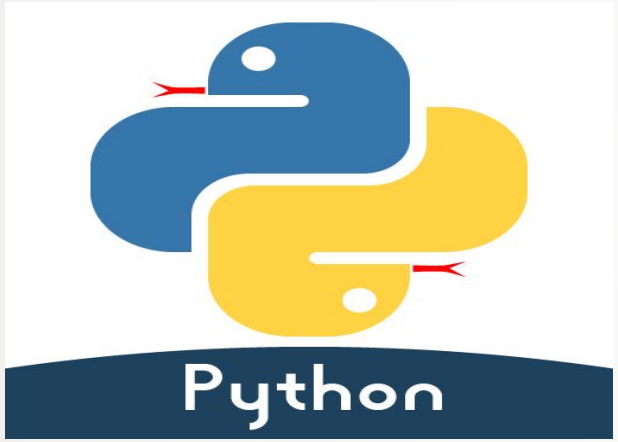


3.5 模型评估

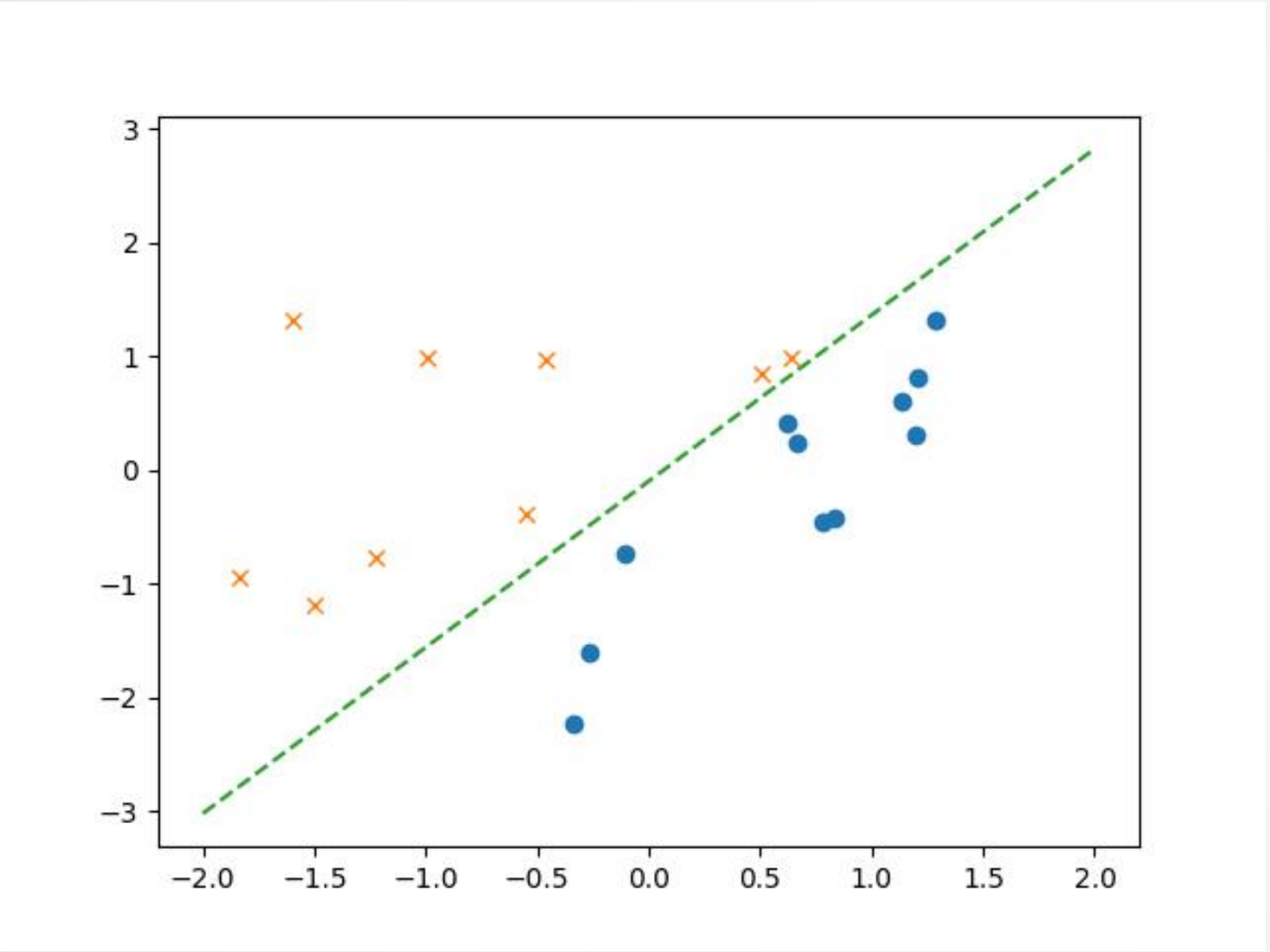
分类	正确结果标签	
	正	负
正	True Positive (TP)	False Positive (FP)
负	False Negative (FN)	True Negative (TN)

Accuracy、Precision、Recall、F measure、Weighted F measure

3.5 Python实现



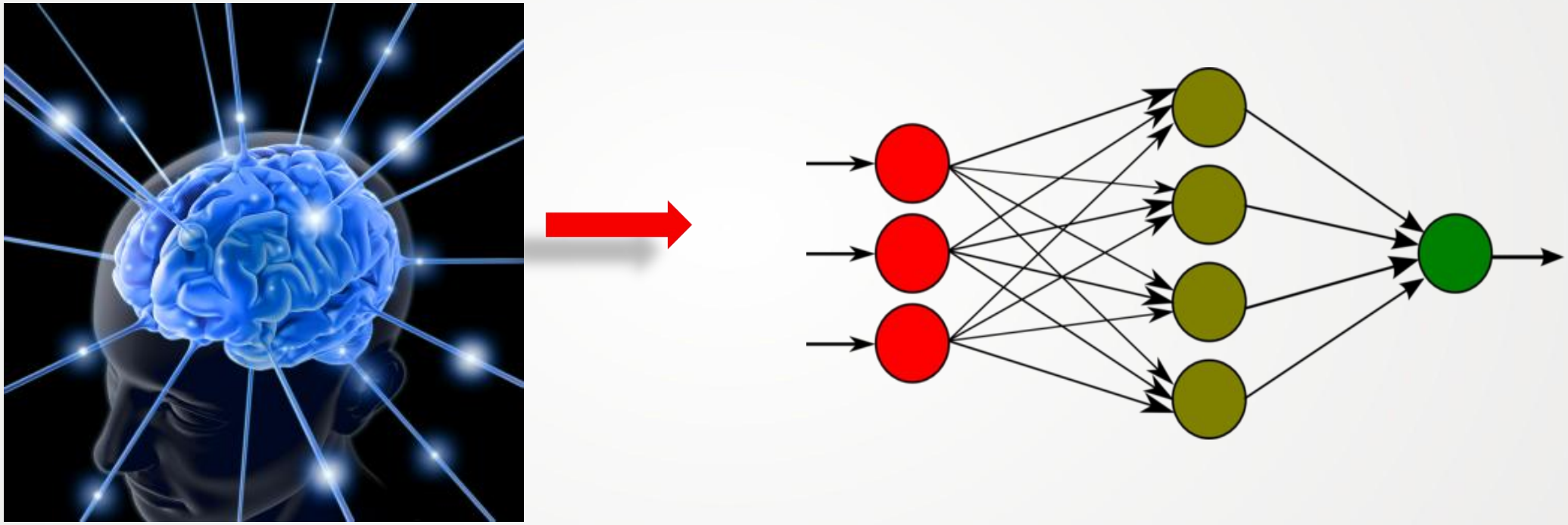

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> #读入训练数据
>>> train = np.loadtxt('/Users/xiaokeai/Desktop/images_for_logit.csv',delimiter=
',',skiprows=1)
>>> train_x = train[:,0:2]
>>> train_y = train[:,2]
>>> #初始化参数
>>> theta = np.random.rand(3)
>>> #标准化
>>> mu = train_x.mean(axis=0)
>>> sigma = train_x.std(axis=0)
>>> def standardize(x):
        return (x-mu)/sigma

>>> train_z = standardize(train_x)
>>> #增加x0
>>> def to_matrix(x):
        x0 = np.ones([x.shape[0],1])
        return np.hstack([x0,x])

>>> X = to_matrix(train_z)
>>> #将标准化后的训练数据画成图
>>> plt.plot(train_z[train_y==1,0],train_z[train_y==1,1],'o')
[<matplotlib.lines.Line2D object at 0x7fbaee04b2d0>]
>>> plt.plot(train_z[train_y==0,0],train_z[train_y==0,1],'x')
[<matplotlib.lines.Line2D object at 0x7fbaee52add0>]
>>> plt.show()
>>> #sigmoid函数
>>> def f(x):
        return 1/(1+np.exp(-np.dot(x,theta)))

>>> #学习率
>>> ETA = 1e-3
>>> #重复次数
>>> epoch = 5000
>>> #重复学习
>>> for _ in range(epoch):
        theta=theta-ETA*np.dot(f(X)-train_y, X)


>>> x0 = np.linspace(-2,2,100)
>>> plt.plot(train_z[train_y==1,0],train_z[train_y==1,1],'o')
[<matplotlib.lines.Line2D object at 0x7fbaee70f490>]
>>> plt.plot(train_z[train_y==0,0],train_z[train_y==0,1],'x')
[<matplotlib.lines.Line2D object at 0x7fbaee70f810>]
>>> plt.plot(x0,-(theta[0]+theta[1]*x0)/theta[2],linestyle='dashed')
[<matplotlib.lines.Line2D object at 0x7fbaee70fb50>]
>>> plt.show()
```


3
2
1
0
−1
−2
−3
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

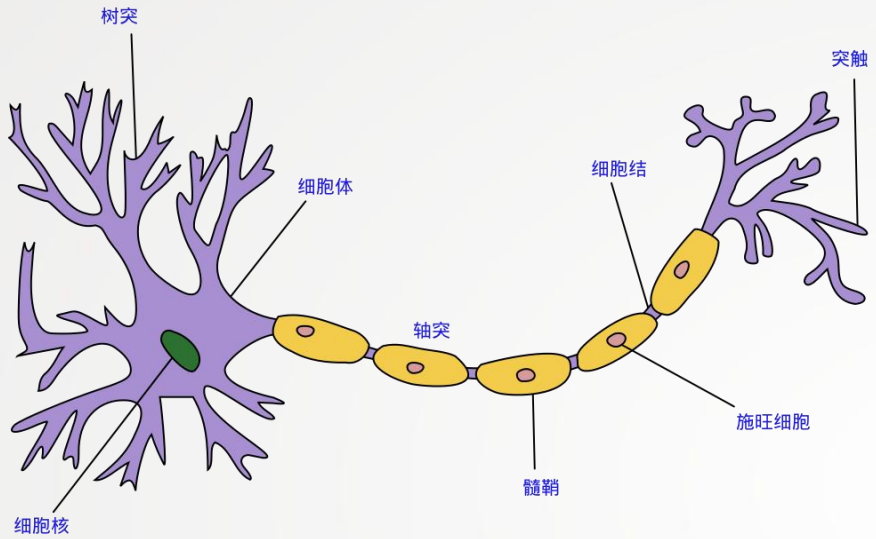
第三讲　神经网络
Artificial Neural Networks

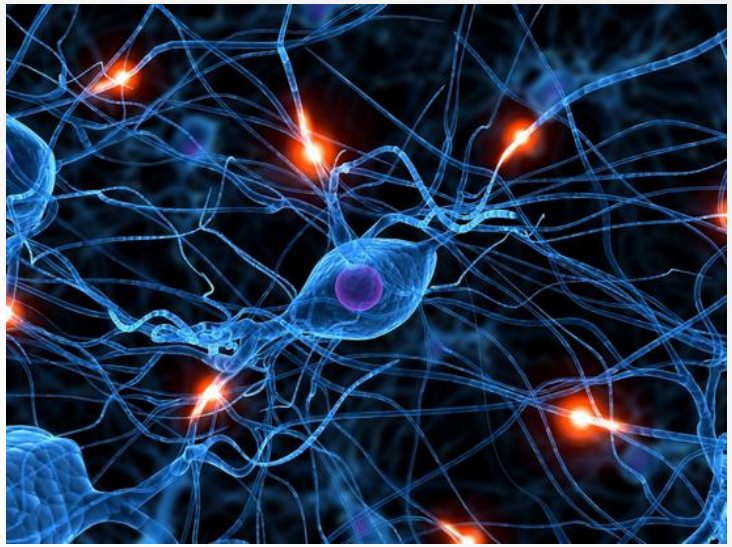
目录

2.0 概览



生物学动机





- 10^{11}: 大脑中神经元的数量
- 10^{4}: 每个神经元的连接数量
- 10^{-3}: 单个神经元的转换速度
- 10^{-10}: 计算机的转换速度

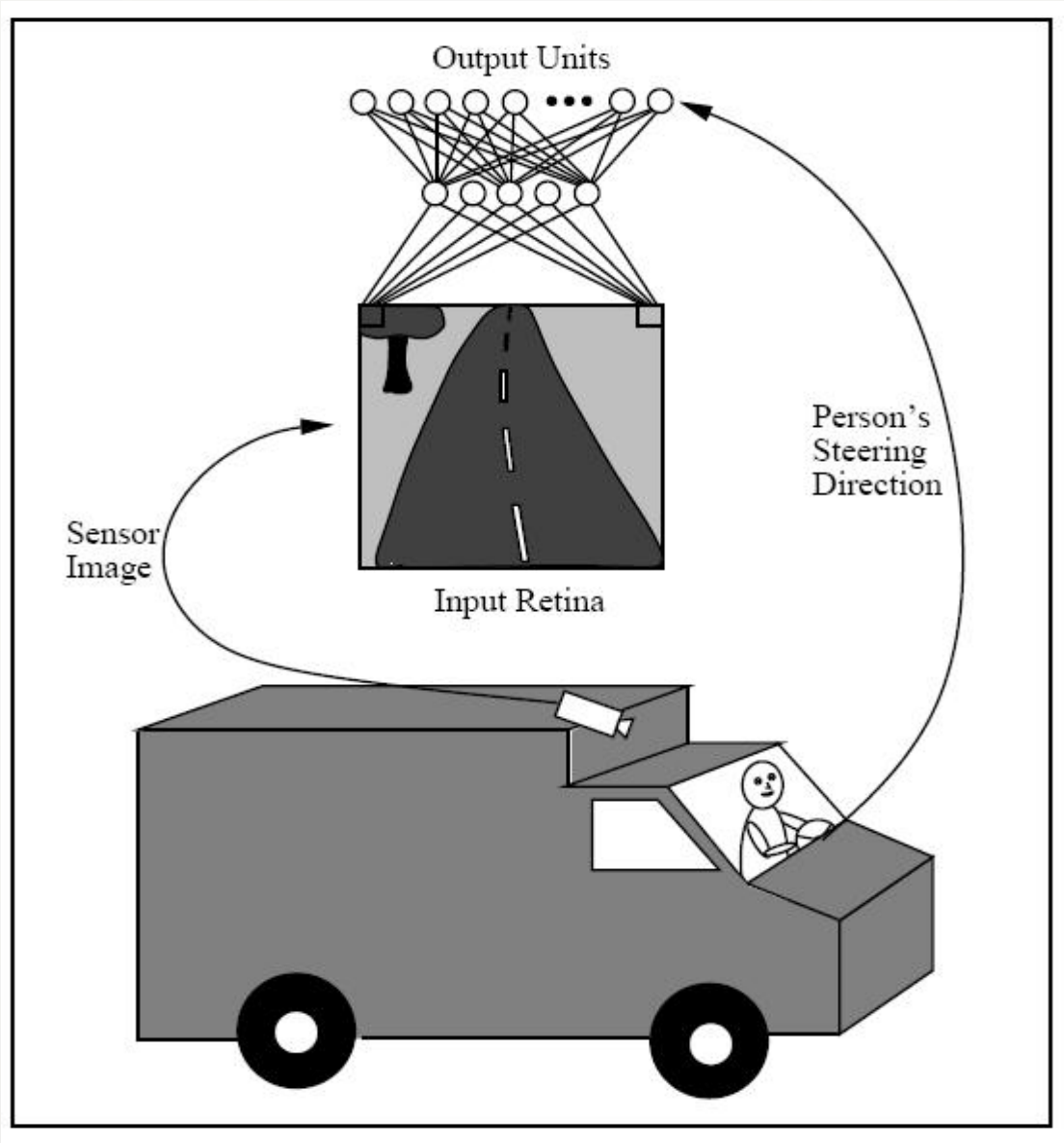
生物学动机

- 并行处理
  - 生物神经元系统的信息处理能力来自于其高度并行处理的过程。
  - 在这一过程中信息表征分布于多个神经元。

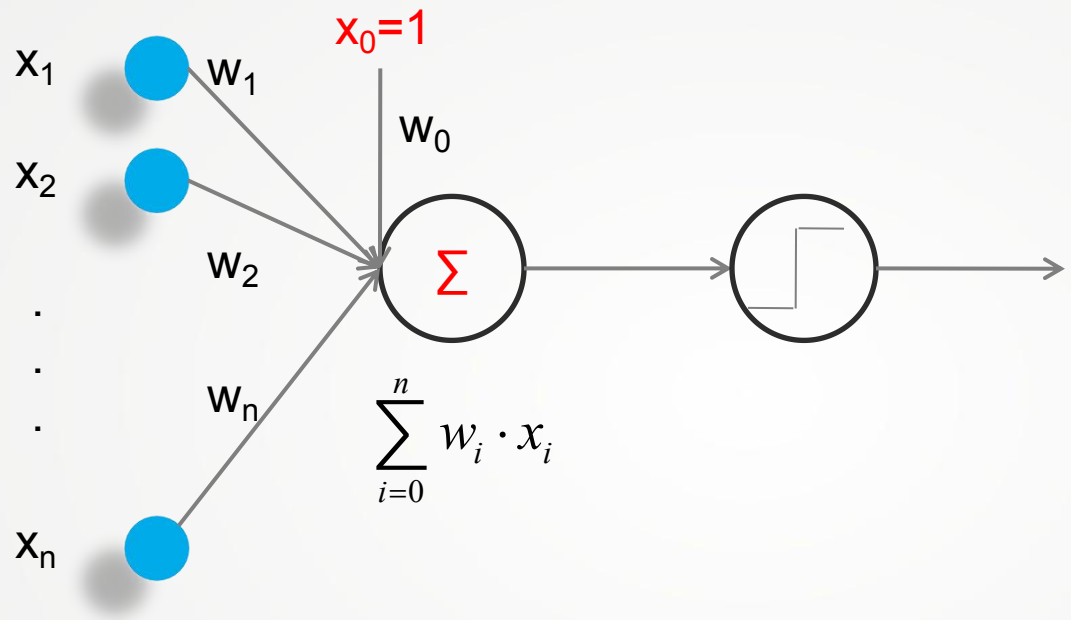
- ANN是模拟生物神经元的并行式信息处理方式而提出的一种计算模型。

- ANN的两种用途。
  - A：使用ANN学习和模拟生物学习过程（认知科学）。
  - B：算法。

一个案例


Output Units
Person's
Steering
Direction
Sensor
Image
Input Retina

2.1 感知机 Perceptrons



门限函数

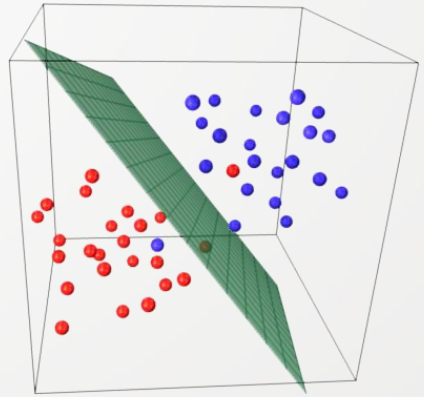
threshold function

$$o = \begin{cases} 1, & if \sum_{i=0}^{n} w_i \cdot x_i > 0 \\ 0, & otherwise \end{cases}$$

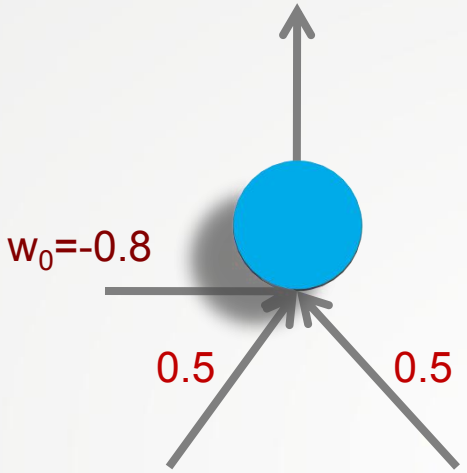
激活

非激活

$$o(x_1,\ldots,x_n) = \begin{cases} 1, & if\ w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n > 0 \\ 0, & otherwise \end{cases}$$

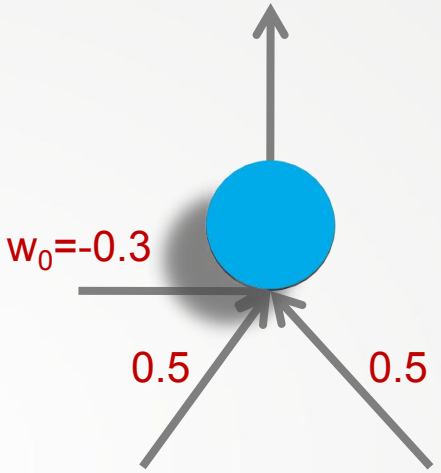


感知机---简单逻辑电路



AND

Input		∑	Output
0	0	-0.8	0
0	1	-0.3	0
1	0	-0.3	0
1	1	0.2	1



OR

Input		∑	Output
0	0	-0.3	0
0	1	0.2	1
1	0	0.2	1
1	1	0.7	1

简单的逻辑电路

AND gate “与”

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

OR gate “或”

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

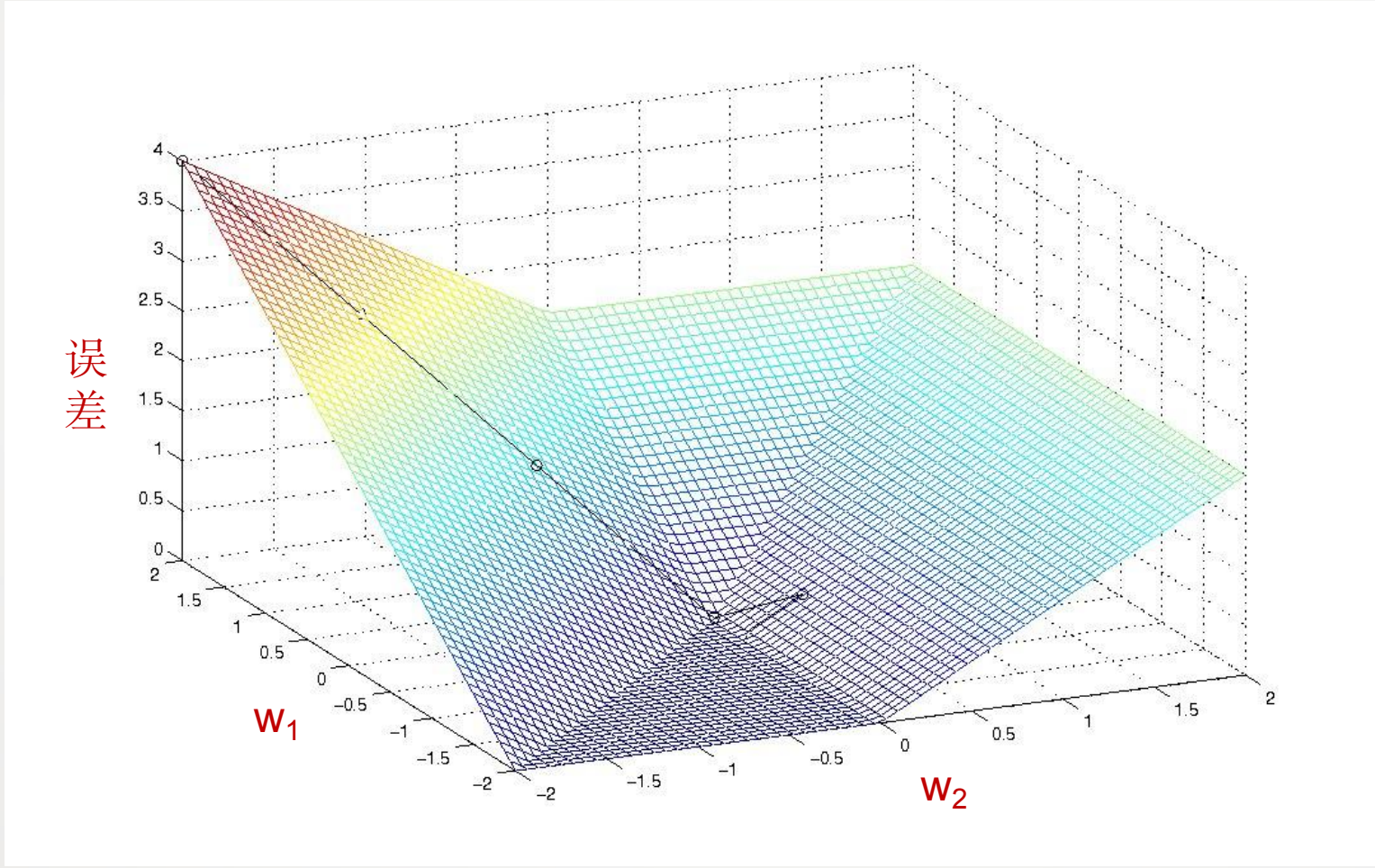
NAND gate “与非”

x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

XOR gate “异或”

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

误差曲面



梯度下降法 Gradient Descent

$$E(\vec{w}) \equiv \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$

批处理的学习
Batch Learning

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$$

$$w_i \leftarrow w_i + \Delta w_i \quad where \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

学习率
Learning Rate

Delta法则

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

$$= \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2$$

$$= \frac{1}{2}\sum_{d\in D}2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d)$$

$$= \sum_{d\in D}(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - w\cdot x_d)$$

$$= \sum_{d\in D}(t_d - o_d)(-x_{id})$$

$$\Delta w_i = \eta\sum_{d\in D}(t_d - o_d)x_{id}$$

Delta Rule

$$o(x) = w\cdot x$$

批处理的学习 Batch learning

GRADIENT_DESCENT (*training_examples*, η)

- Initialize each w_i to some small random value.
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero.
  - For each <x, t> in *training_examples*, Do
    - Input the instance x to the unit and compute the output o
    - For each linear unit weight w_i, Do
      - $\Delta w_i \leftarrow \Delta w_i + \eta(t-o)x_i$
  - For each linear unit weight w_i, Do
    - $w_i \leftarrow w_i + \Delta w_i$

随机学习 Stochastic learning

$$w_i \leftarrow w_i + \Delta w_i \qquad where \quad \Delta w_i = \eta(t-o)x_i$$

For example, if x_i=0.8, η=0.1, t=1 and o=0

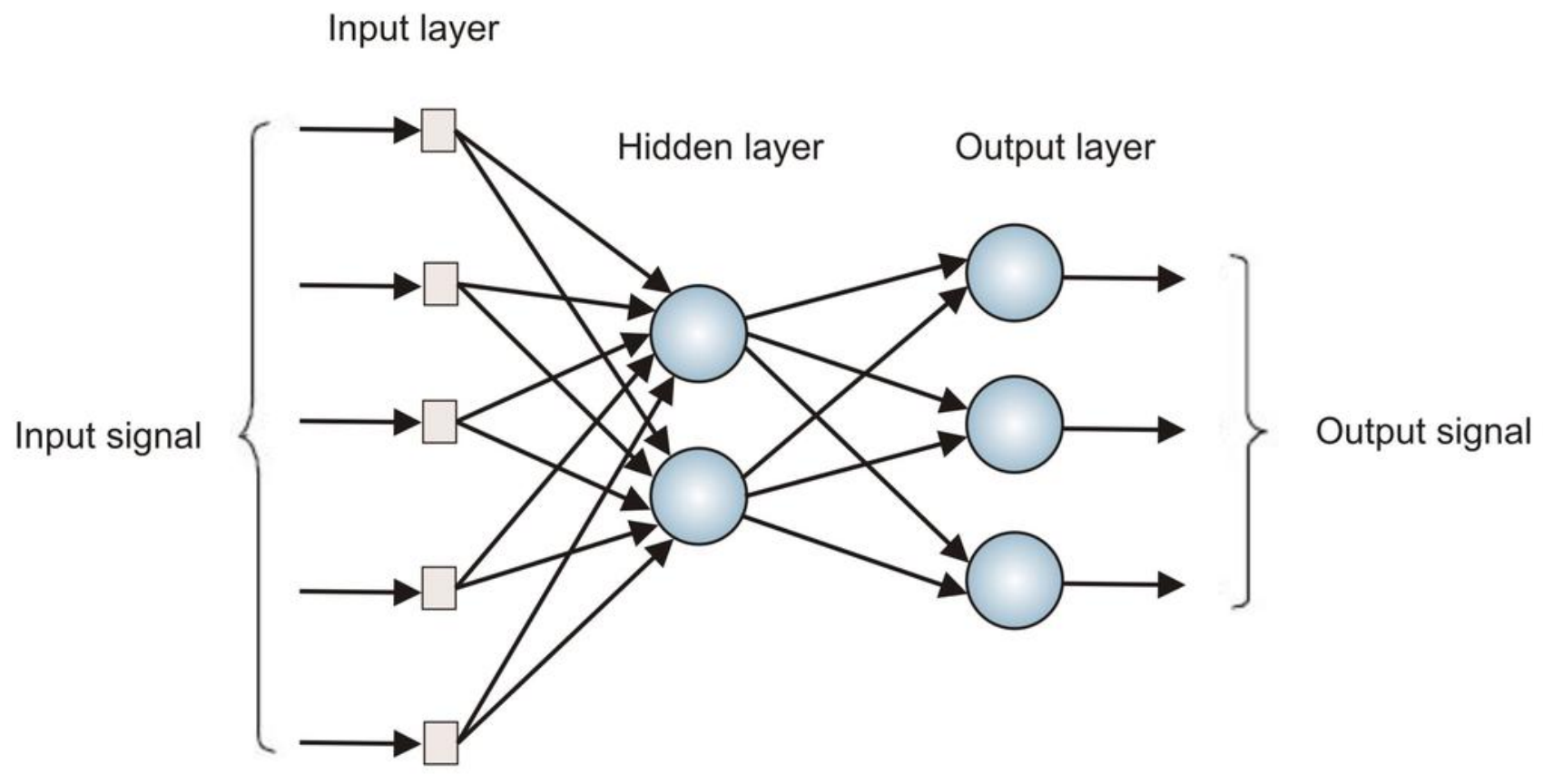
$\Delta w_i = \eta(t\text{-}o)x_i = 0.1 \times (1\text{-}0) \times 0.8 = 0.08$

随机学习的例子：与非门（NAND）

Input			Target	Initial Weights			Output					Error	Correction	Final Weights		
							Individual			Sum	Final Output					
x_0	x_1	x_2	t	w_0	w_1	w_2	C_0	C_1	C_2	S	o	E	R	w_0	w_1	w_2
							$x_0 \cdot w_0$	$x_1 \cdot w_1$	$x_2 \cdot w_2$	$C_0+C_1+C_2$		t-o	LR x E			
1	0	0	1	0	0	0	0	0	0	0	0	1	+0.1	0.1	0	0
1	0	1	1	0.1	0	0	0.1	0	0	0.1	0	1	+0.1	0.2	0	0.1
1	1	0	1	0.2	0	0.1	0.2	0	0	0.2	0	1	+0.1	0.3	0.1	0.1
1	1	1	0	0.3	0.1	0.1	0.3	0.1	0.1	0.5	0	0	0	0.3	0.1	0.1
1	0	0	1	0.3	0.1	0.1	0.3	0	0	0.3	0	1	+0.1	0.4	0.1	0.1
1	0	1	1	0.4	0.1	0.1	0.4	0	0.1	0.5	0	1	+0.1	0.5	0.1	0.2
1	1	0	1	0.5	0.1	0.2	0.5	0.1	0	0.6	1	0	0	0.5	0.1	0.2
1	1	1	0	0.5	0.1	0.2	0.5	0.1	0.2	0.8	1	-1	-0.1	0.4	0	0.1
1	0	0	1	0.4	0	0.1	0.4	0	0	0.4	0	1	+0.1	0.5	0	0.1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	1	0	1	0.8	-.2	-.1	0.8	-.2	0	0.6	1	0	0	0.8	-.2	-.1

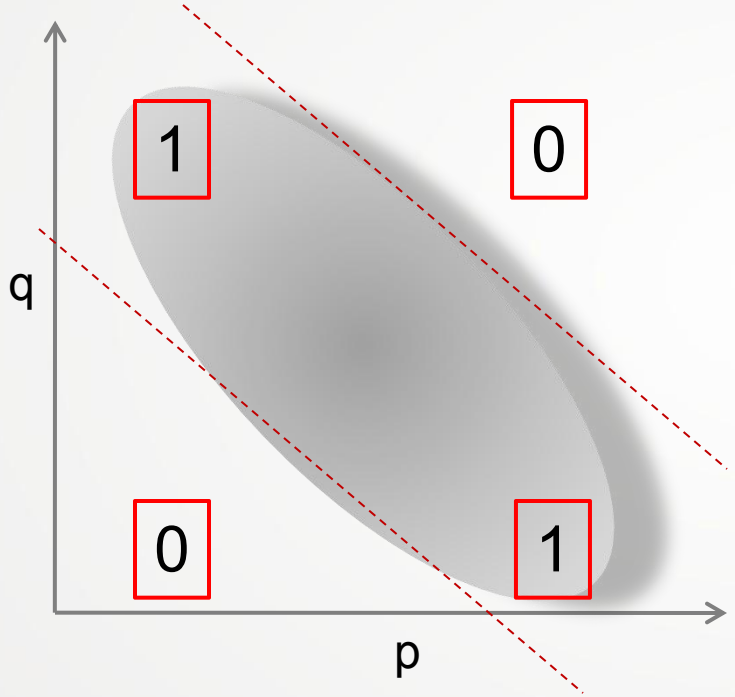
threshold=0.5 learning rate=0.1 70

2.2 多层感知机 Multilayer Perceptron



异或问题：XOR

$$p \oplus q = p\bar{q} + \bar{p}q = (p+q)(\overline{pq})$$

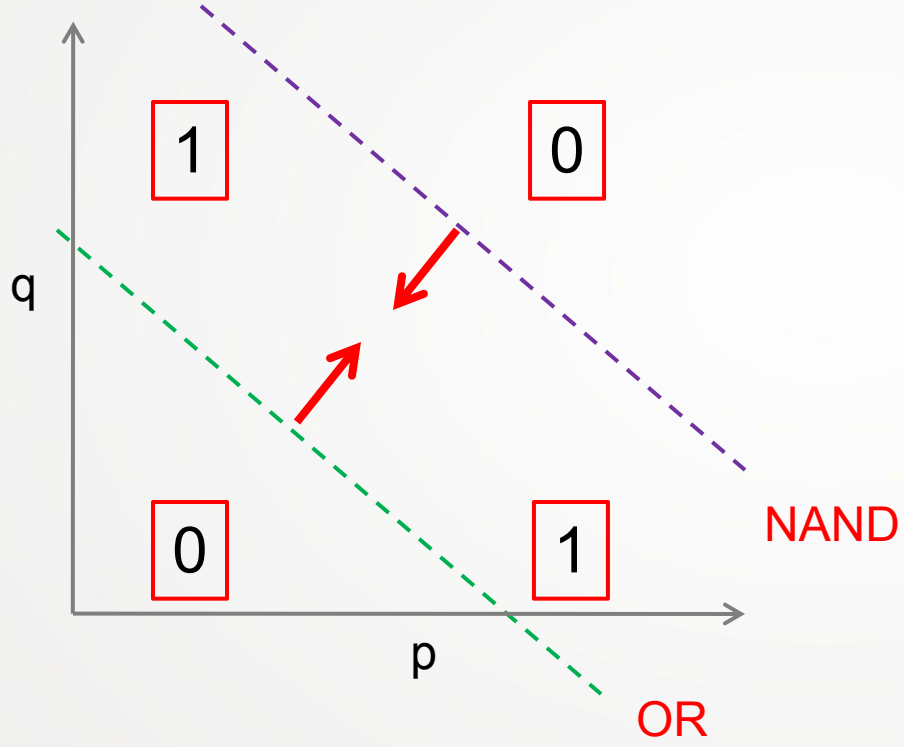


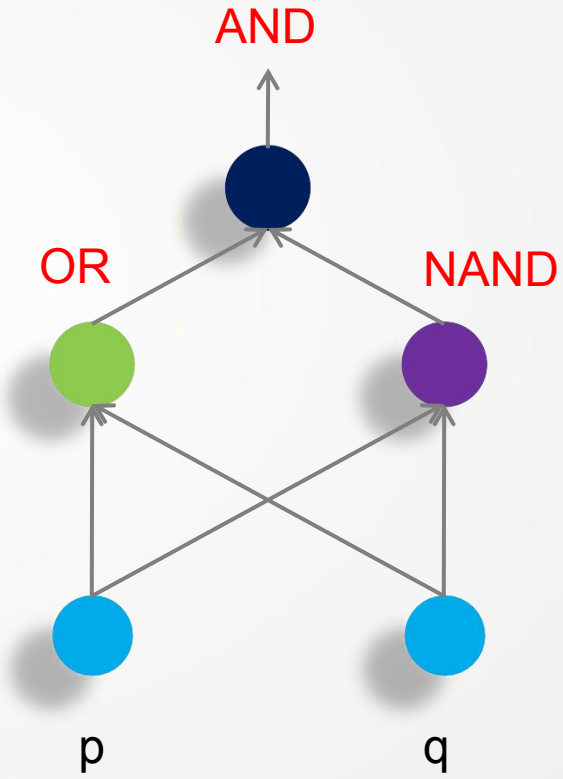
Input		Output
0	0	0
0	1	1
1	0	1
1	1	0

线性不可分：不能被一条直线分开

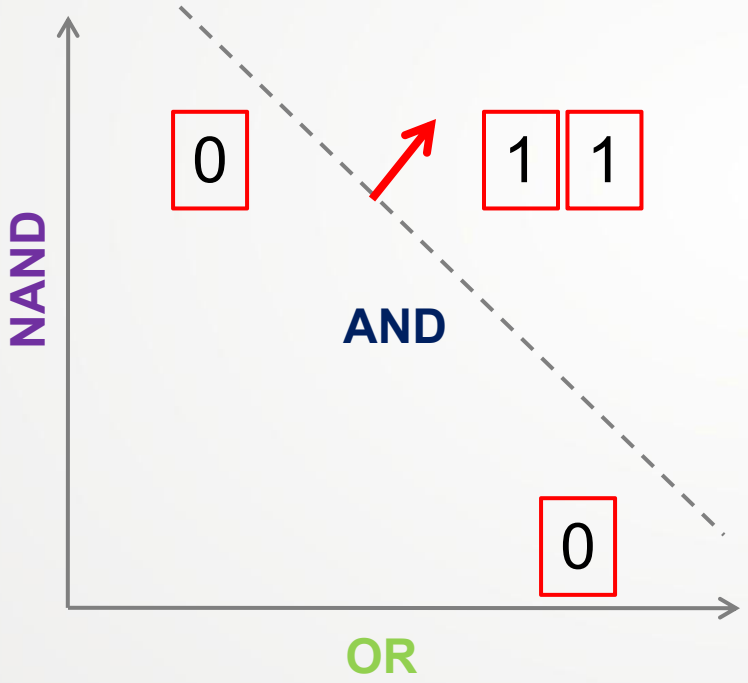
异或问题：XOR

$$p \oplus q = \neg(p \wedge q) \wedge (p \vee q)$$



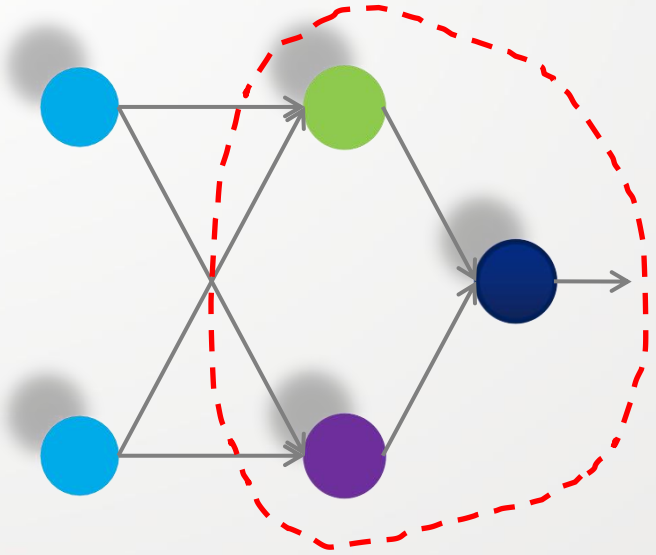


隐含层表征 Hidden Layer Representations

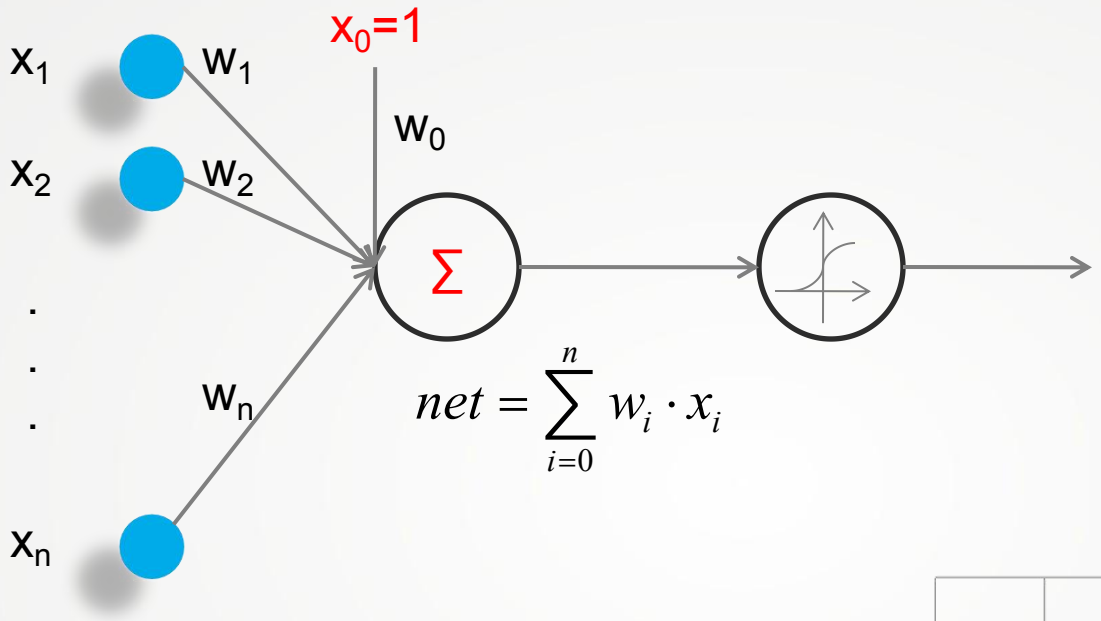


p	q	OR	NAND	AND
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Input Hidden Output



Sigmoid函数



$$net = \sum_{i=0}^{n} w_i \cdot x_i$$

$$o = \sigma(net) = \frac{1}{1+e^{-net}}$$

*Sigmoid*函数

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

$$\frac{d\sigma(y)}{dy} = \sigma(y)\cdot(1-\sigma(y))$$

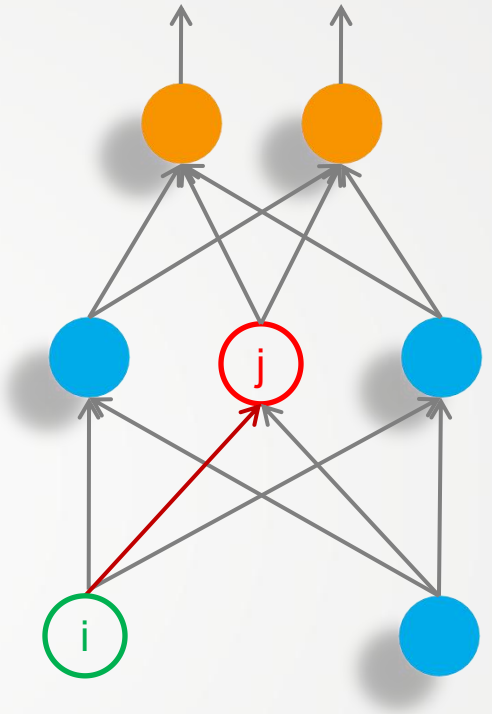
其他的激活函数

- tanh函数（双曲正切函数）：
  - $tanh(y) = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}}$
- ReLU函数（线性整流函数）：
  - $ReLU(y) = max(0, y)$

2.3 逆向传播算法 Backpropagation Rule

$$E_d(\bar{w}) \equiv \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 \qquad \Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}$$



- x_{ji} = the i^{th} input to unit j
- w_{ji} = the weight associated with the i^{th} input to unit j
- $net_j = \sum w_{ji} x_{ji}$ (the weighted sum of inputs for unit j)
- o_j= the output of unit j
- t_j= the target output of unit j
- σ = the sigmoid function
- *outputs* = the set of units in the final layer
- *Downstream* (j) = the set of units directly taking the output of unit j as inputs

输出单元的训练法则

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k \in outputs}(t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2$$

$$= \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j)$$

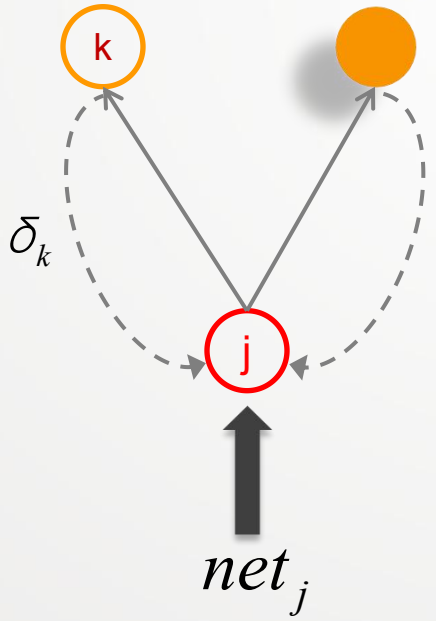
$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)o_j(1 - o_j)$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1 - o_j)x_{ji}$$

隐含层的训练法则

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j (1 - o_j)$$



$$\delta_k = -\frac{\partial E_d}{\partial net_k}$$

$$\delta_j = o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}$$

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

BP神经网络的基本框架

- BACKPROPAGATION (*training_examples*, η, n_{in}, n_{out}, n_{hidden})
- Create a network with n_{in} inputs, n_{hidden} hidden units and n_{out} output units.
- Initialize all network weights to small random numbers.
- Until the termination condition is met, Do
  - For each <*x*, *t*> in *training_examples*, Do
    - Input the instance *x* to the network and computer the output *o* of every unit.
    - For each output unit *k*, calculate its error term δ_{k}

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

    - For each hidden unit *h*, calculate its error term δ_{h}

$$\delta_h \leftarrow o_h(1-o_h)\sum_{k\in outputs} w_{kh}\delta_k$$

    - Update each network weight w_{ji} $\quad w_{ji} \leftarrow w_{ji} + \Delta w_{ji} = w_{ji} + \eta\delta_j x_{ji}$

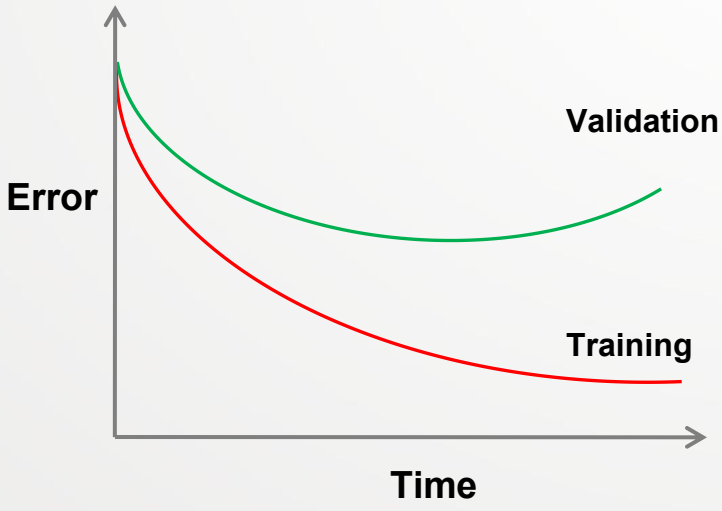
BP神经网络

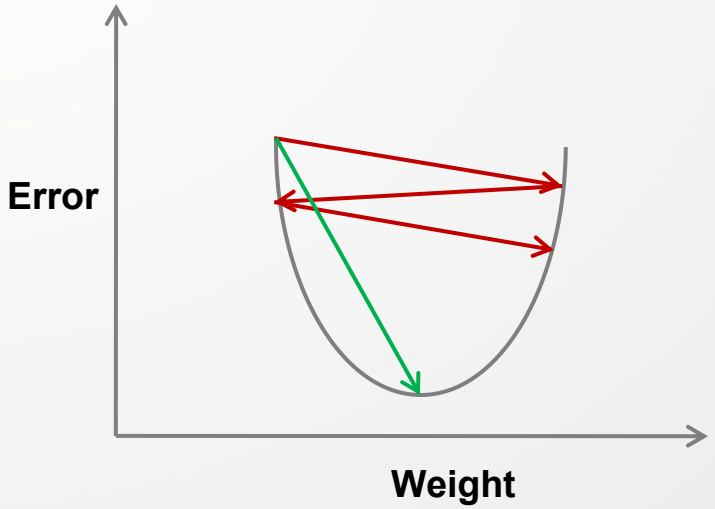
- 收敛和局部最优点
  - 搜索空间极有可能有多个局部最优点。
  - 收敛到局部最优点。
  - 使用不同的初始权重进行多次尝试训练。

- 进化神经网络
  - 黑箱优化算法 Black-box optimization techniques (e.g., Genetic Algorithms)
  - 更高的准确率
  - 做更高级的训练
  - Xin Yao (1999) “Evolving Artificial Neural Networks”, *Proceedings of the IEEE*, pp. 1423-1447.
- 表达能力 (Representational Power)
- 深度学习

BP神经网络

- 过学习
  - 在训练时经常发生。
  - 使用测试集中止训练。
- 实际操作
  - 冲量 (momentum)
  - 学习率的动态太调整
    - 小：收敛速度慢，容易遇到困难
    - 大：收敛速度快，不稳定





什么时候使用神经网络？

- 期望输出可以是离散值或实值.
- 训练集可能包含错误
- 允许较长的学习时间
  - 从几秒到几个小时不等
- 对学习方程的解释不重要
  - 权重是很难解释的

2.5 ANN在管理学中的应用

- 预测模型：
  - 推荐系统
    - 推荐系统的功能是帮助用户主动找到满足偏好的个性化物品并推荐给用户。本质上是一个个性化的搜因引擎，输入用户的行为信息、偏好信息等，输出为最符合查询条件的物品列表。
    - 图神经网络：Gao et al., 2021. Graph Neural Networks for Recommender Systems:Challenges, Methods, and Directions. ACM Transactions on Information Systems.
  - 银行风险控制
    - 输入用户年龄、职业、收入等信息，输出用户是否会借贷违约。

ANN在管理学中的应用

- 评价指标（体系）的构建
  - 创新能力评价体系
  - 高质量发展评价体系
  - 顾客满意度评价体系
  - 营销效果评价体系
- 企业大脑（Enterprise Agent Brain）
  - 数字化转型升级/中国智能制造
  - 数据、算法（ANN/DL）和算力（硬件）
- E-health
  - 临床诊断
  - 医疗影像分析和判读
  - 信号分析和判读
  - 药物研制

ANN在管理学中的研究

- Wang et al., 2021. Evaluating the Effectiveness of Marketing Campaigns for Malls Using a Novel Interpretable Machine Learning Model. Information Systems Research, 33(2), 659-677.
- Lee & Ram, 2024. Explainable Deep Learning for False Information Identification: An Argumentation Theory Approach. Information Systems Research, Forthcoming.
- 张诚,王富荣,郁培文等.基于深度增强学习的个性化动态促销[J].管理世界,2023,39(05):160-178.
- 赵蓉英,陈文欣.深度学习视角下的评价科学方法创新[J].情报科学,2022,40(11):3-11+19.
- 李伟卿,池毛毛,王伟军.基于感知价值的网络消费者偏好预测研究[J].管理学报,2021,18(06):912-918.
- 薛奕曦,张佳陈,张译等.中国制造业企业高质量发展的联动路径研究[J/OL].科研管理:1-11[2024-03-26].
- 朱茂然,朱艳鹏,高松等.基于深度哈希的相似图片推荐系统:以Airbnb为例[J].管理科学,2020,33(05):17-28.

- 逻辑回归：适合小规模数据、简单问题、需要解释性的场景。
- 神经网络：适合大规模数据、复杂问题、对可解释性要求不高的场景。