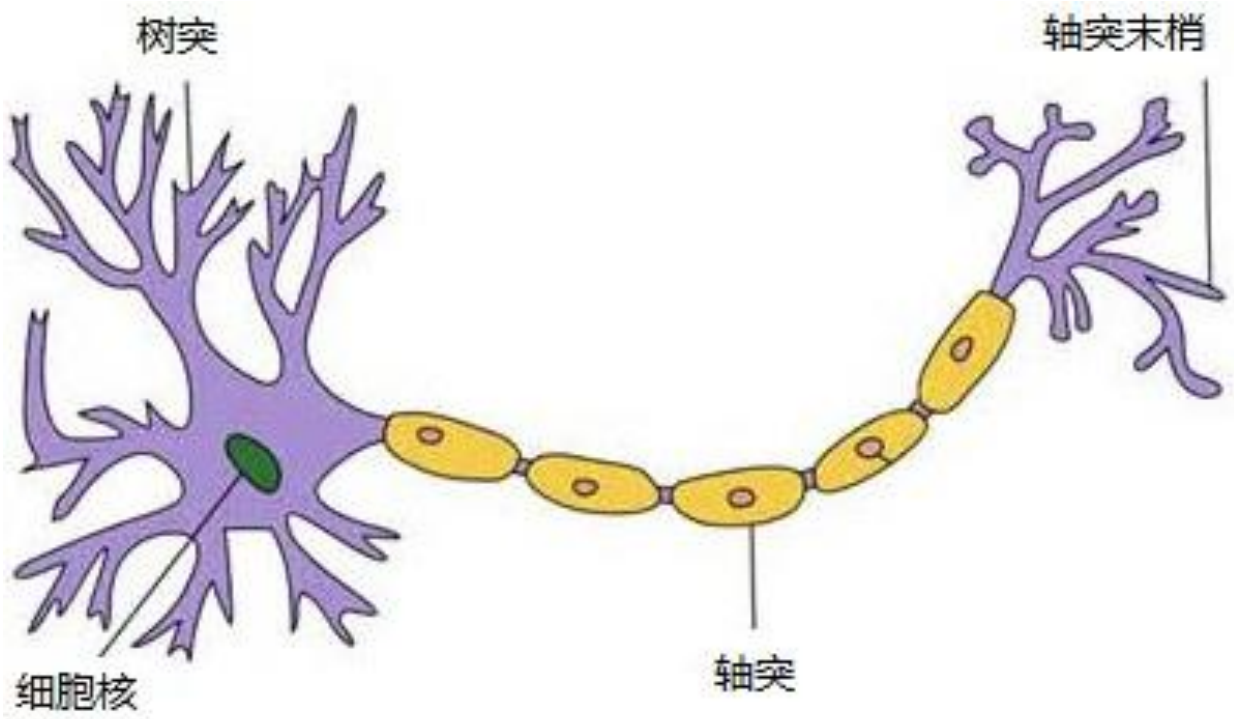
数据挖掘

神经网络分类器

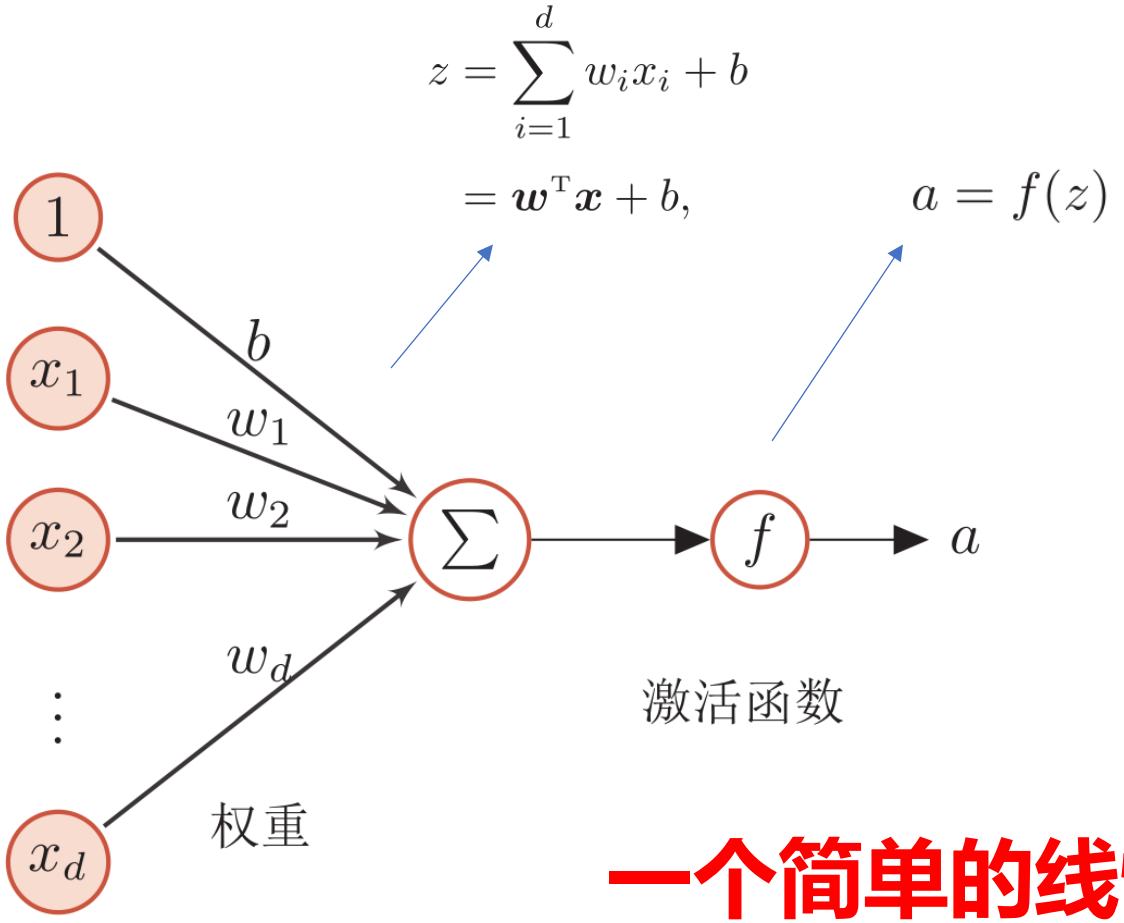
2025 年 11 月

生物神经元

- **大脑是如何学习的?**
  - 树突接收信息
  - 轴突传递信息
- **为什么要模拟人脑?**
  - 快速、智能
  - 稳定
- **我们如何模拟人脑的生物神经元?**



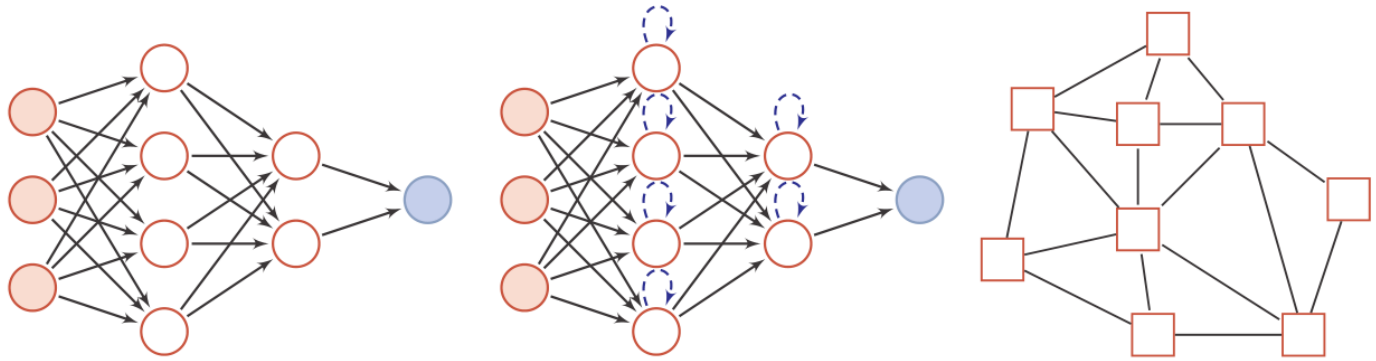
神经元模型



一个简单的线性模型！

人工神经网络

- **人工神经网络主要由大量的神经元以及它们之间的有向连接构成。包含三个方面：**
  - **神经元的激活规则**
    - 主要是指神经元输入到输出之间的映射关系，一般为非线性函数
  - **网络的拓扑结构**
    - 不同神经元之间的连接关系。
  - **学习算法**
    - 通过训练数据来学习神经网络的参数。



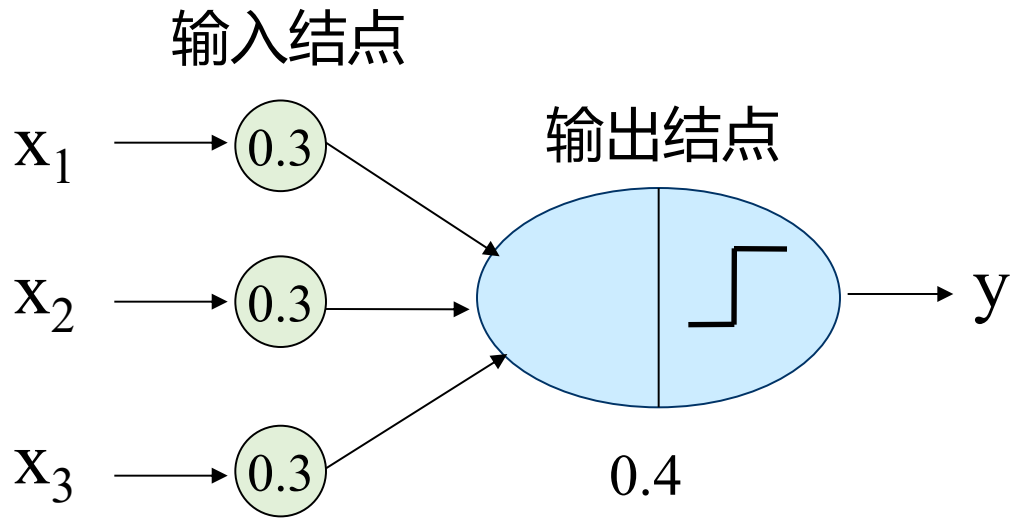
(a) 前馈网络　(b) 记忆网络　(c) 图网络

南京大学
NANJING UNIVERSITY

目录

感知机

x_1	x_2	x_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



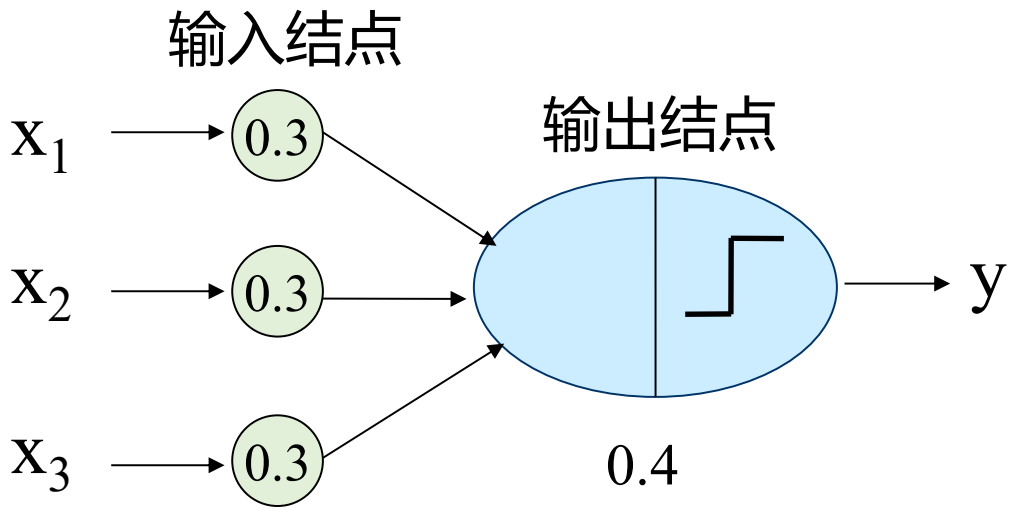
$$\hat{y} = \begin{cases} 1 & w_1x_1 + w_2x_2 + w_3x_3 - \theta \geq 0 \\ -1 & w_1x_1 + w_2x_2 + w_3x_3 - \theta < 0 \end{cases}$$

感知机

上一个例子的模型：

$$\hat{y}=\begin{cases}1 & \\ -1 & \end{cases}$$

化为一般的形式：



感知机

初始：用随机值初始化权值向量$w^{(0)}$

循环：直到输出和实例一致

{

对每个训练样例(x_i,y_i)，按照$w^{(k)}$计算输出值$y_i^{(k)}$;

for每个权值$w^{(k)}$

更新权值$w^{(k+1)}= w^{(k)}+\lambda(y_i-y_i^{(k)})x_i$;

}

感知机

更新权值: $w^{(k+1)}= w^{(k)}+\lambda(y_i-y_i^{(k)})x_i$

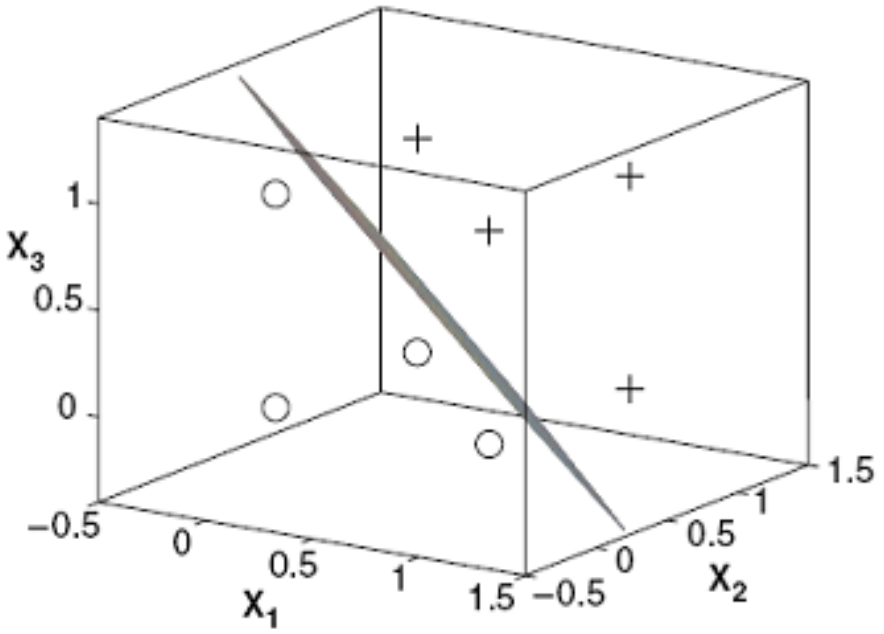
- 如果 $y_i=y_i^{(k)}$， 权值w不变；
- 如果 $y_i=1$(实际类)， $y_i^{(k)}=-1$,预测误差为2；
- 为了补偿这个误差，需要提高正输入链的权值，降低负输入链的；
- 如果 $y_i=-1$(实际类)， $y_i^{(k)}=1$,预测误差为-2；
- 为了补偿这个误差，需要降低正输入链的权值，提高负输入链的；

感知机

- **F. Rosenblatt 证明了对“线性可分”的问题**

感知机通过有限次训练就能学会正确的行为

x_1	x_2	x_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

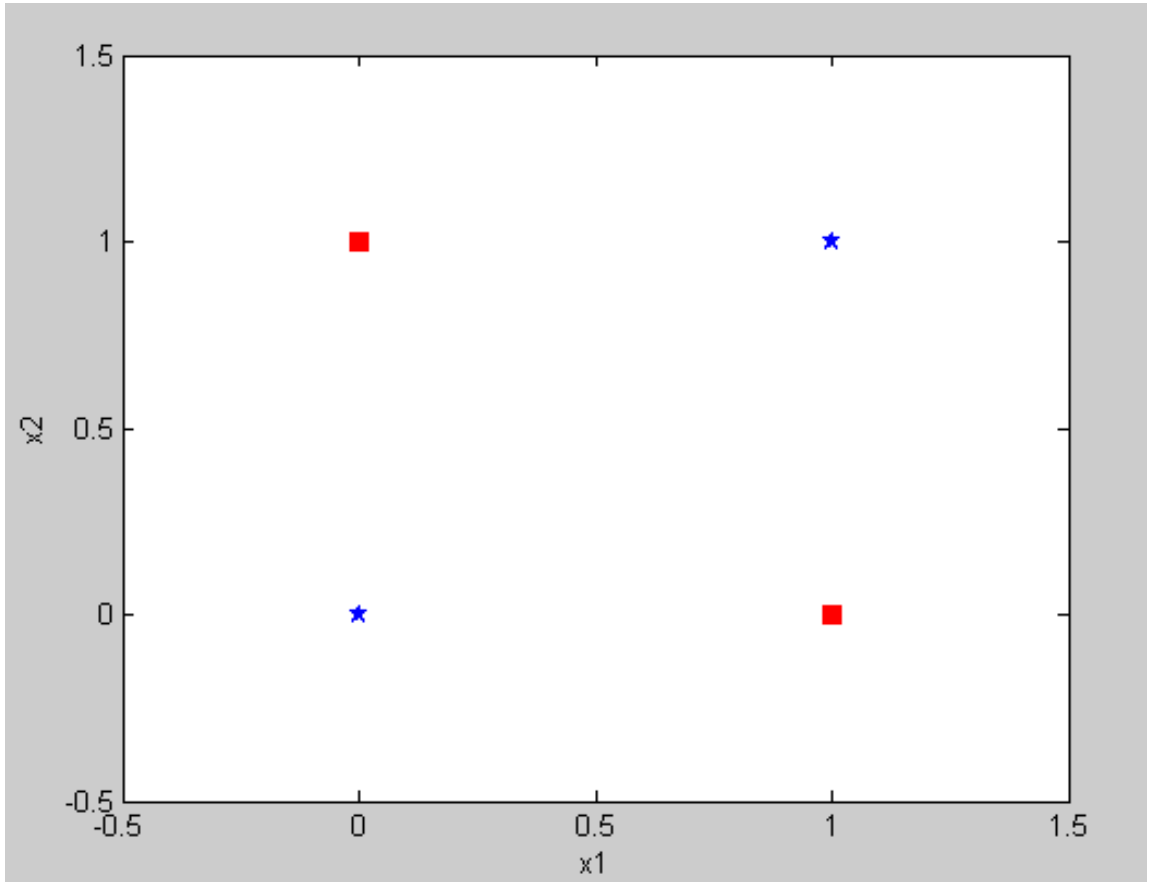


感知机

- Marvin Minsky, *Perceptrons*

证明了感知器无法执行“异或问题”

x_1	x_2	Y
0	0	-1
1	0	1
0	1	1
1	1	-1

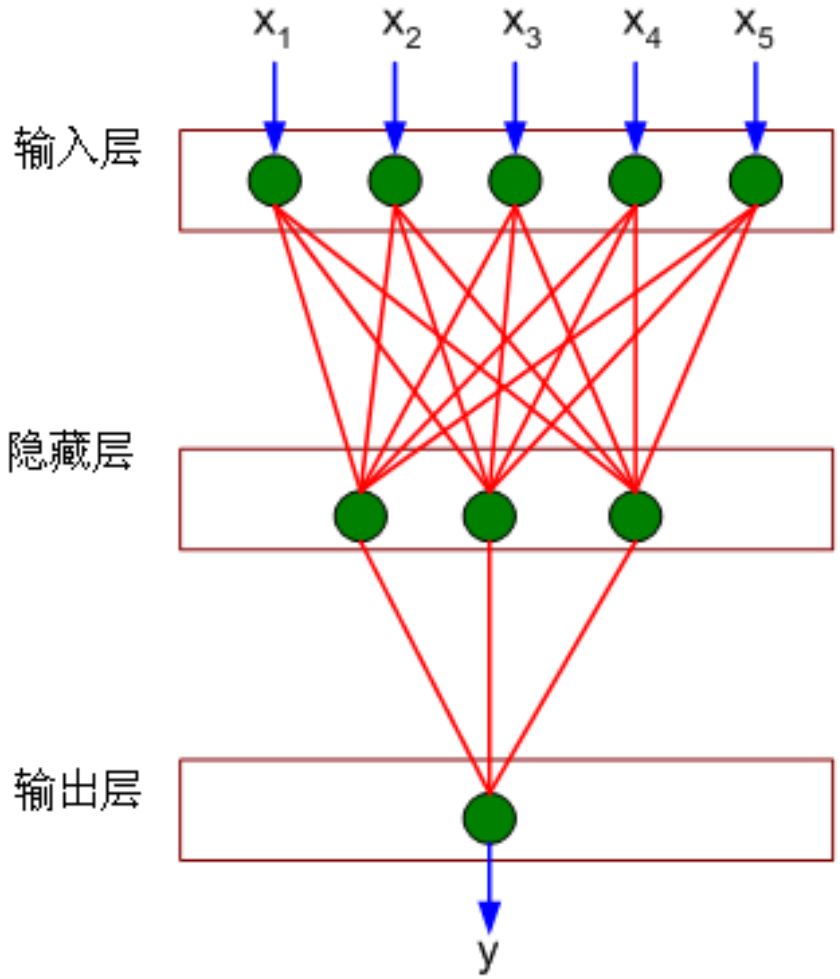


南京大学
NANJING UNIVERSITY

目录

多层人工神经网络

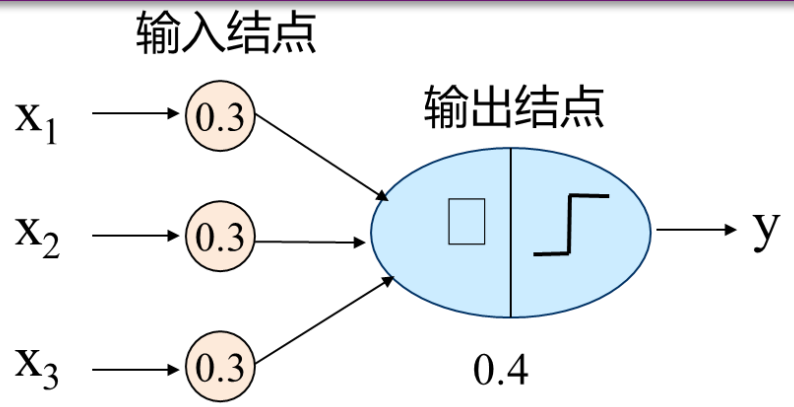
- 人工神经网络比感知机模型复杂
  - 输入层和输出层之间包含隐藏层



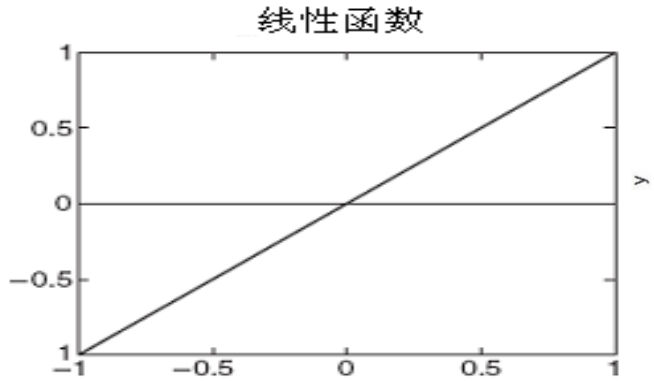
多层人工神经网络

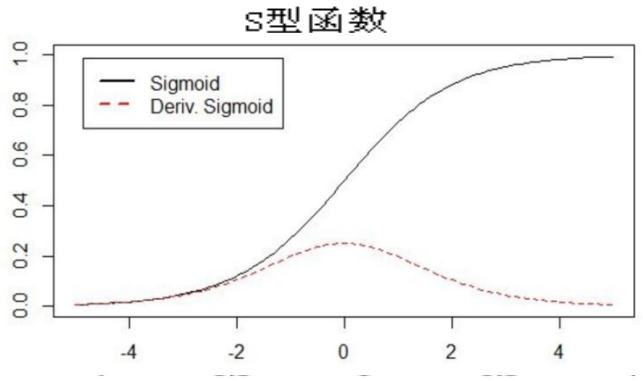
- **人工神经网络比感知机模型复杂**
  - 激活函数可以是多种函数

$$\theta_0 \leftarrow \theta_0 - \eta \frac{\partial f}{\partial \theta_0}$$



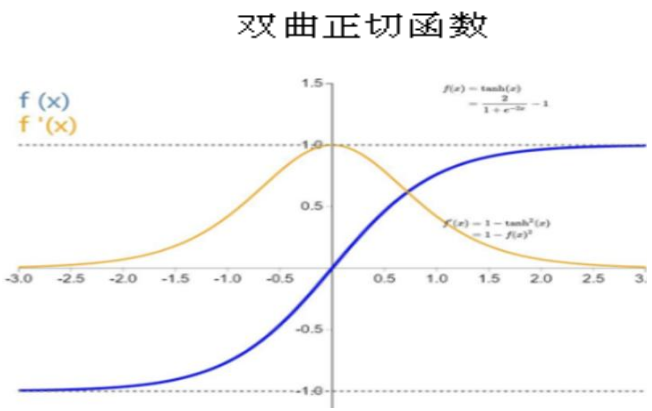
线性可分

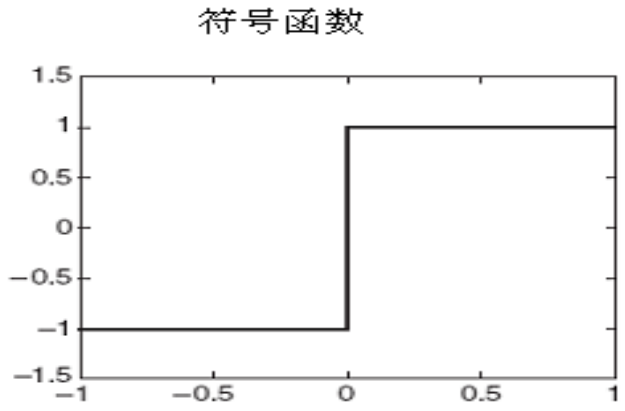




梯度消失

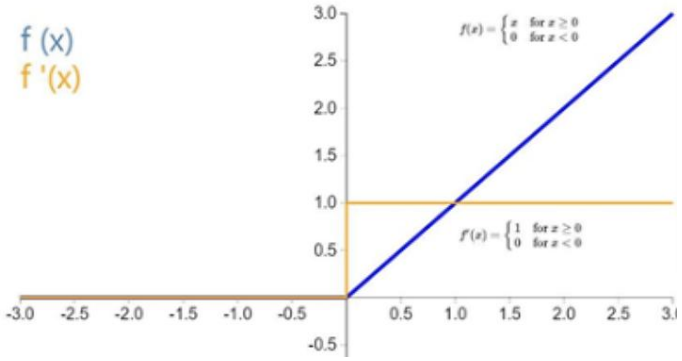
梯度消失 收敛比较快





多层人工神经网络

- **人工神经网络比感知机模型复杂**
  - 激活函数可以是多种函数



解决了部分梯度消失问题

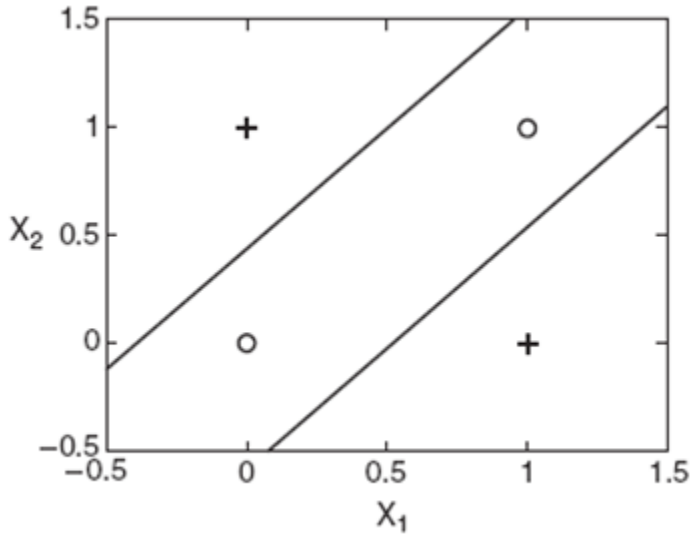
激活函数	函数	导数
Logistic 函数	$f(x)=\frac{1}{1+\exp(-x)}$	$f'(x)=f(x)\big(1-f(x)\big)$
Tanh 函数	$f(x)=\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$f'(x)=1-f(x)^2$
ReLU	$f(x)=\max(0,x)$	$f'(x)=I(x>0)$
ELU	$f(x)=\max(0,x)+\min\big(0,\gamma(\exp(x)-1)\big)$	$f'(x)=I(x>0)+I(x\leq 0)\cdot\gamma\exp(x)$
SoftPlus 函数	$f(x)=\log\big(1+\exp(x)\big)$	$f'(x)=\frac{1}{1+\exp(-x)}$

收敛速度比双面正切更快

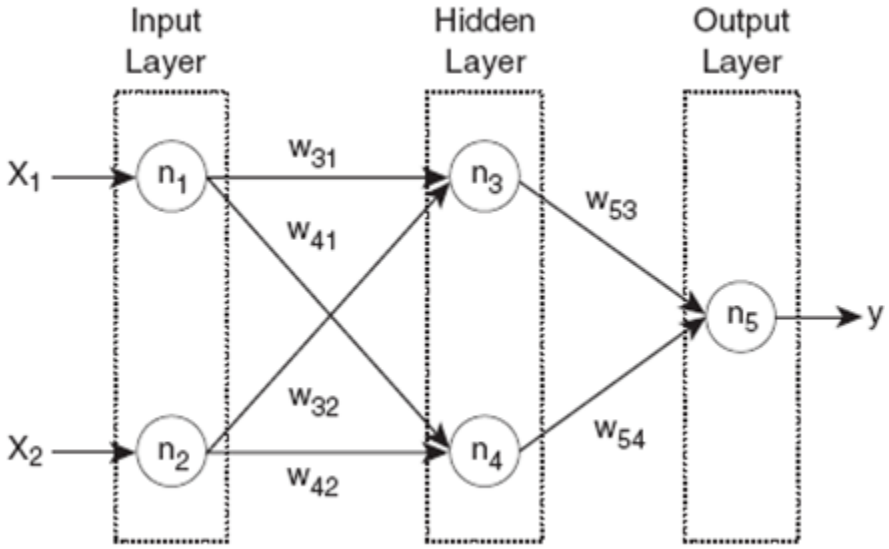
导致神经元懒惰　　**设置较小的学习率**

多层人工神经网络

- **例：用两层前馈神经网络解决异或问题**



决策边界



神经网络拓扑结构

南京大学
NANJING UNIVERSITY

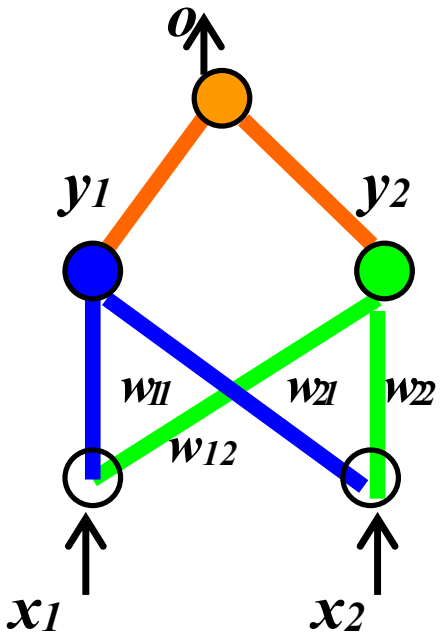
目录

多层感知机

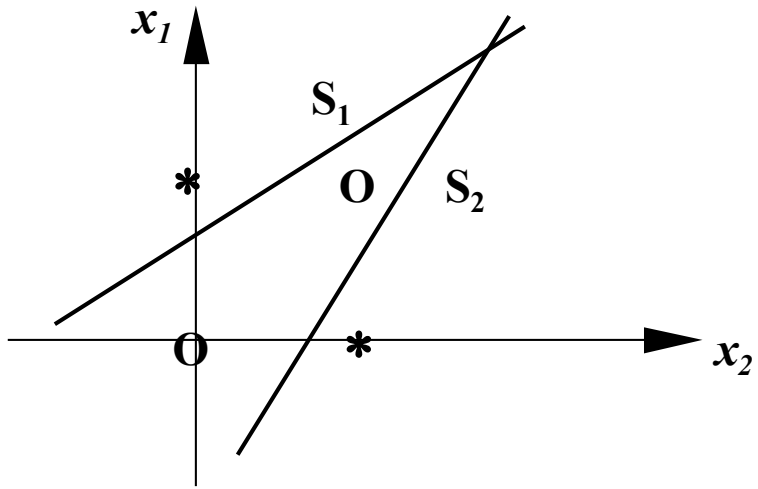
用两计算层感知器解决“异或”问题。

“异或”的真值表

x1	x2	y1	y2	o
0	0	1		
0	1	1		
1	0	0		
1	1	1		



双层感知器



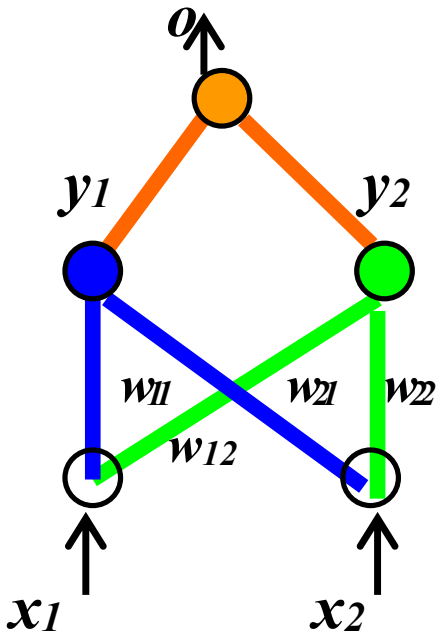
“异或”问题分类

多层感知机

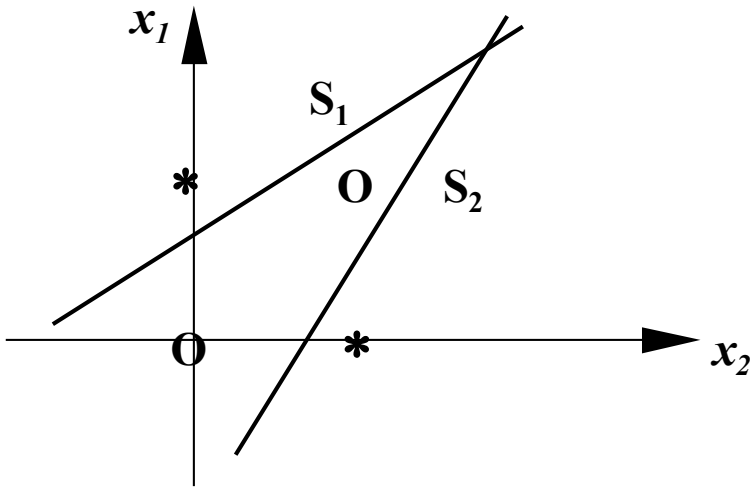
用两计算层感知器解决“异或”问题。

“异或”的真值表

x1	x2	y1	y2	o
0	0		1	
0	1		0	
1	0		1	
1	1		1	



双层感知器



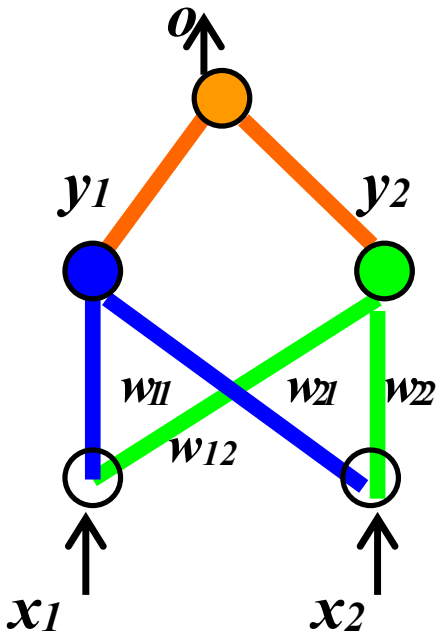
“异或”问题分类

多层感知机

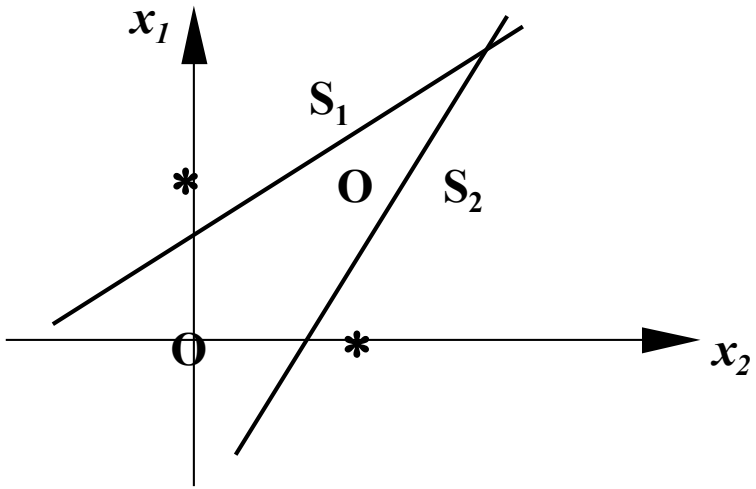
用两计算层感知器解决“异或”问题。

“异或”的真值表

x1	x2	y1	y2	o
0	0	1	1	
0	1	1	0	
1	0	0	1	
1	1	1	1	



双层感知器



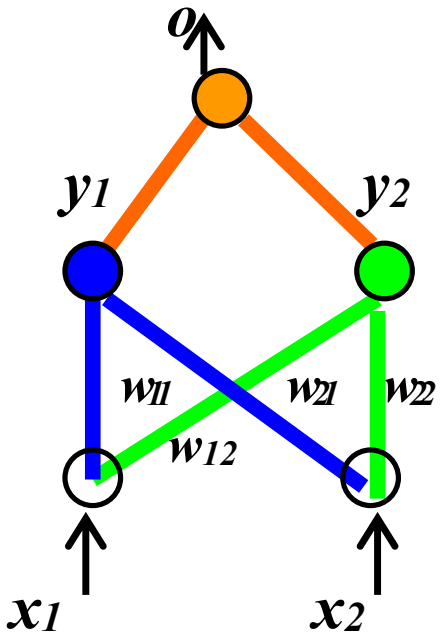
“异或”问题分类

多层感知机

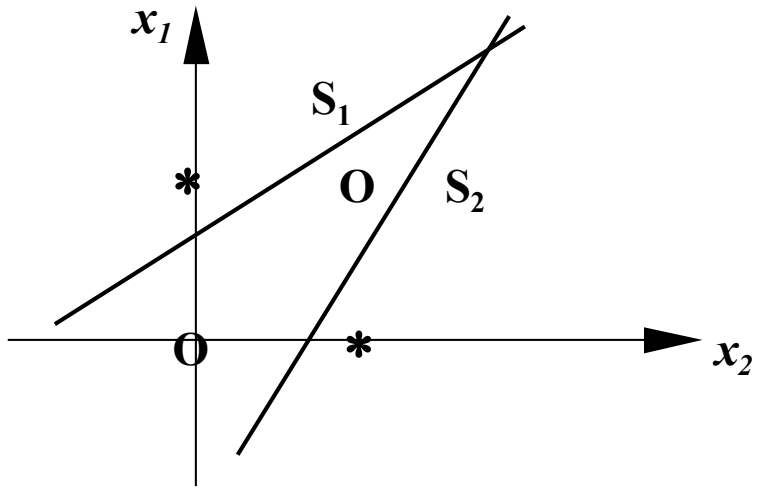
用两计算层感知器解决“异或”问题。

“异或”的真值表

x1	x2	y1	y2	o
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	1	1	0



双层感知器



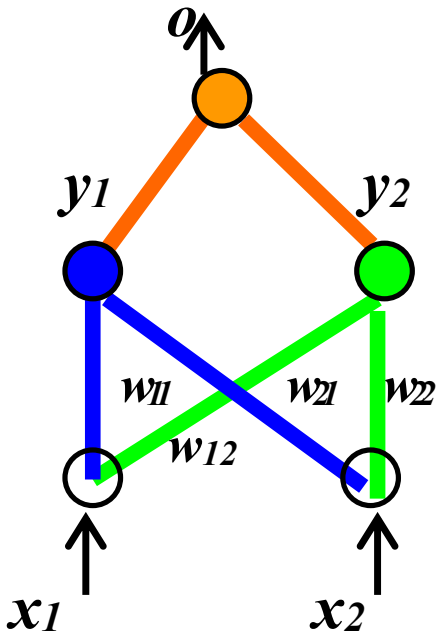
“异或”问题分类

多层感知机

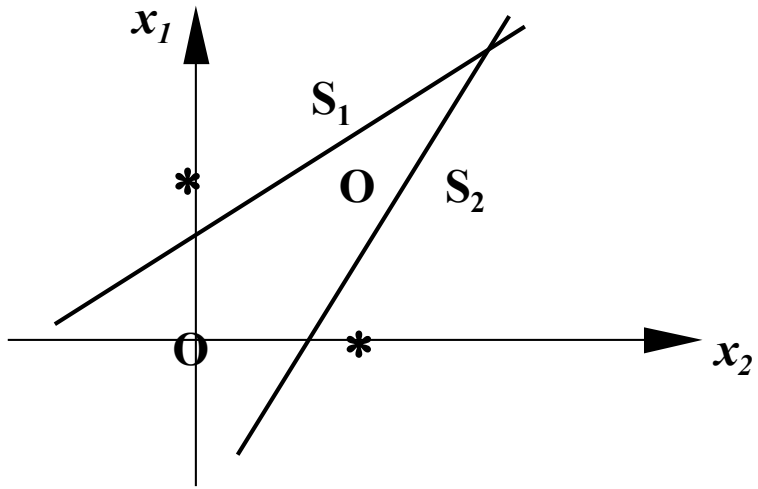
用两计算层感知器解决“异或”问题。

“异或”的真值表

x1	x2	o
0	0	0
0	1	1
1	0	1
1	1	0



双层感知器

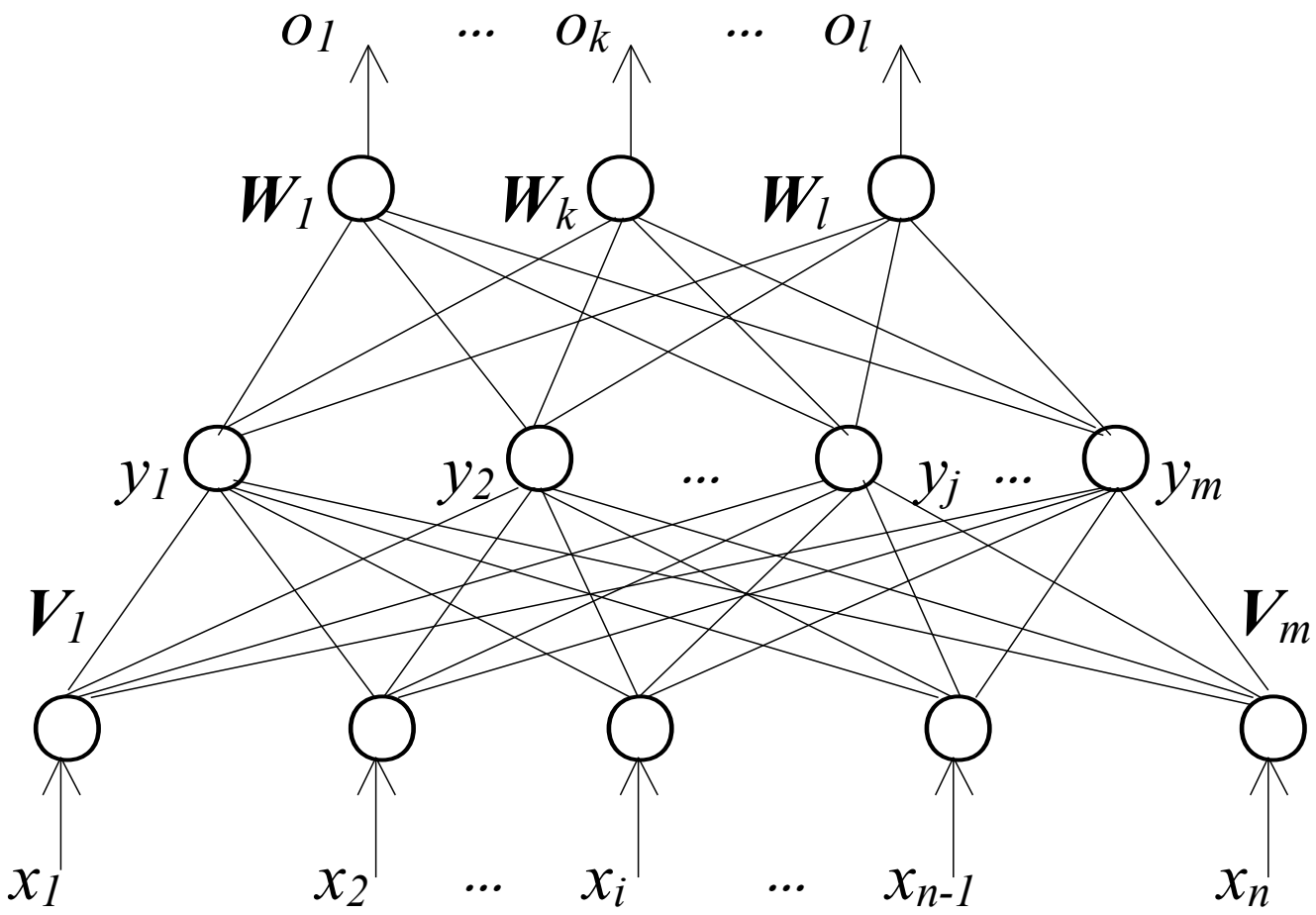


“异或”问题分类

南京大学
NANJING UNIVERSITY

目录

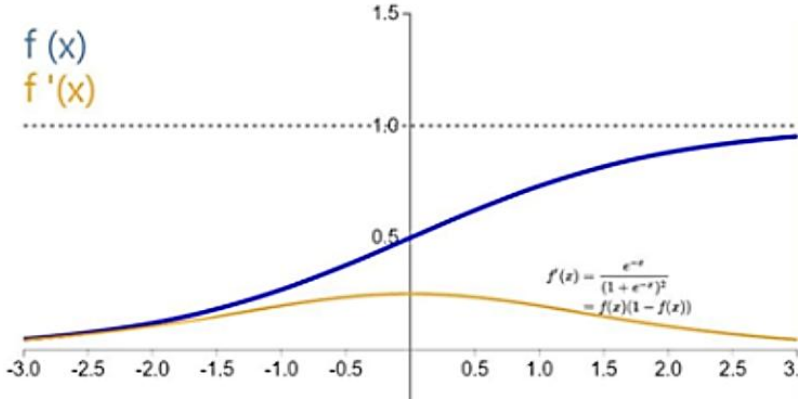
误差反向传播（BP）网络



基于BP算法的多层前馈网络模型

误差反向传播（BP）网络

- **激活函数**
  - 必须处处可导
    - 一般都使用S型函数

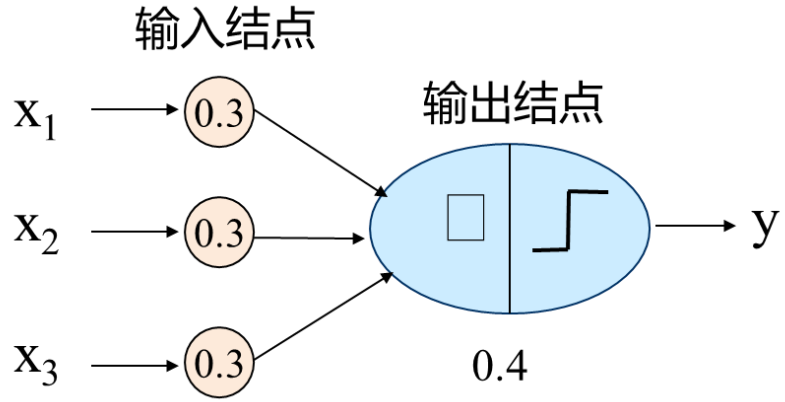


- **使用S型激活函数时BP网络输入与输出关系**
  - 输入

$$net = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

  - 输出

$$y = \mathrm{f}(net) = \frac{1}{1+\mathrm{e}^{-net}}$$

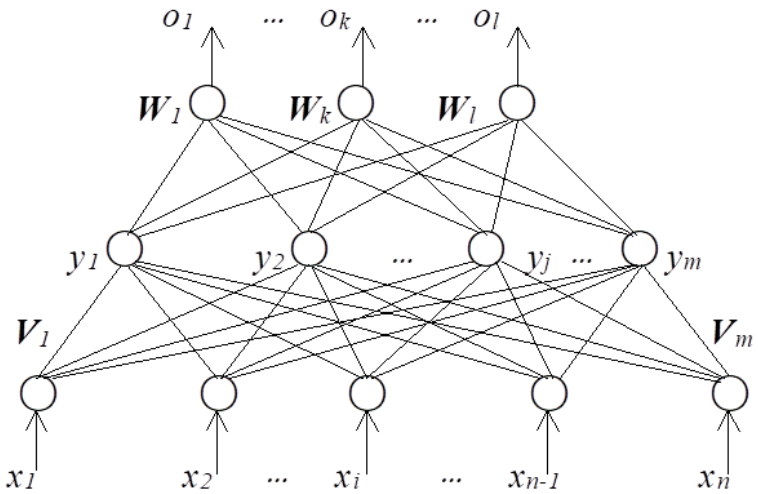


误差反向传播（BP）网络

- **学习的类型：**监督式学习
- **核心思想：**
  - 将输出误差以**某种形式**通过隐层向输入层逐层反传

将误差分摊给各层的所有单元
——各层单元的误差信号 → 修正各单元权值

- **学习的过程：**
  - 信号的正向传播 → 误差的反向传播



误差反向传播（BP）网络

- **正向传播：**
  - 输入样本 — 输入层 — 各隐层 — 输出层
- **判断是否转入反向传播阶段：**
  - 若输出层的实际输出与期望的输出（教师信号）不符
- **误差反传**
  - 误差以某种形式在各层表示 —— 修正各层单元的权值

- **网络输出的误差减少到可接受的程度**
- **进行到预先设定的学习次数为止**

南京大学
NANJING UNIVERSITY

目录

后向传播算法

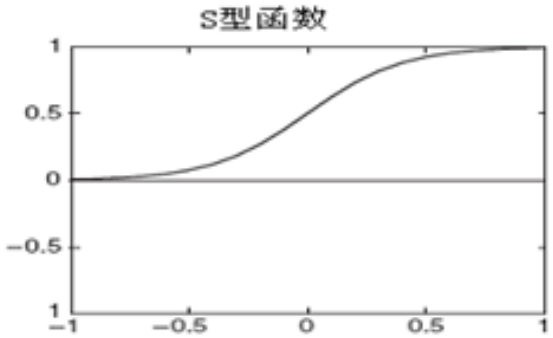
1、初始化权重

- 循环以下两步，直到满足条件

2、向前传播输入

- 在每个节点加权求和，再代入激活函数

$$I_j = \sum_i w_{ij} x_i \qquad y = \frac{1}{1+e^{-x}}$$



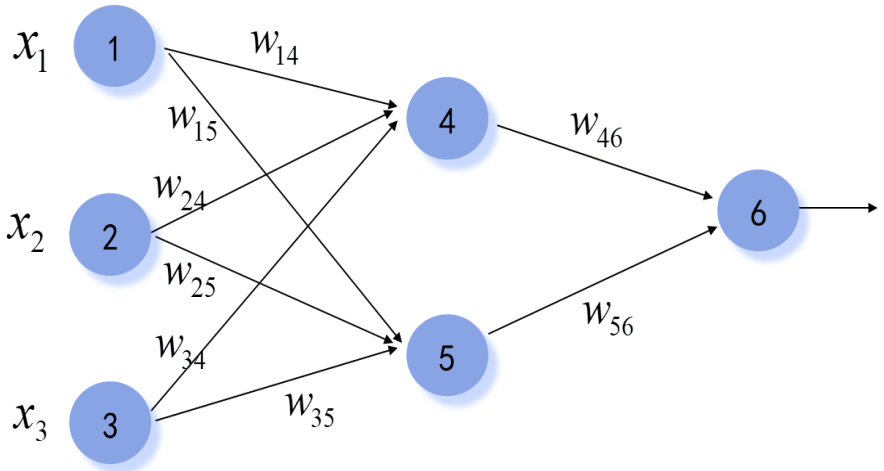
3、向后传播误差

$$Err(j) = O_j(1-O_j)(T_j - O_j)$$

$$Err(j) = O_j(1-O_j)\sum_k Err(k)w_{jk}$$

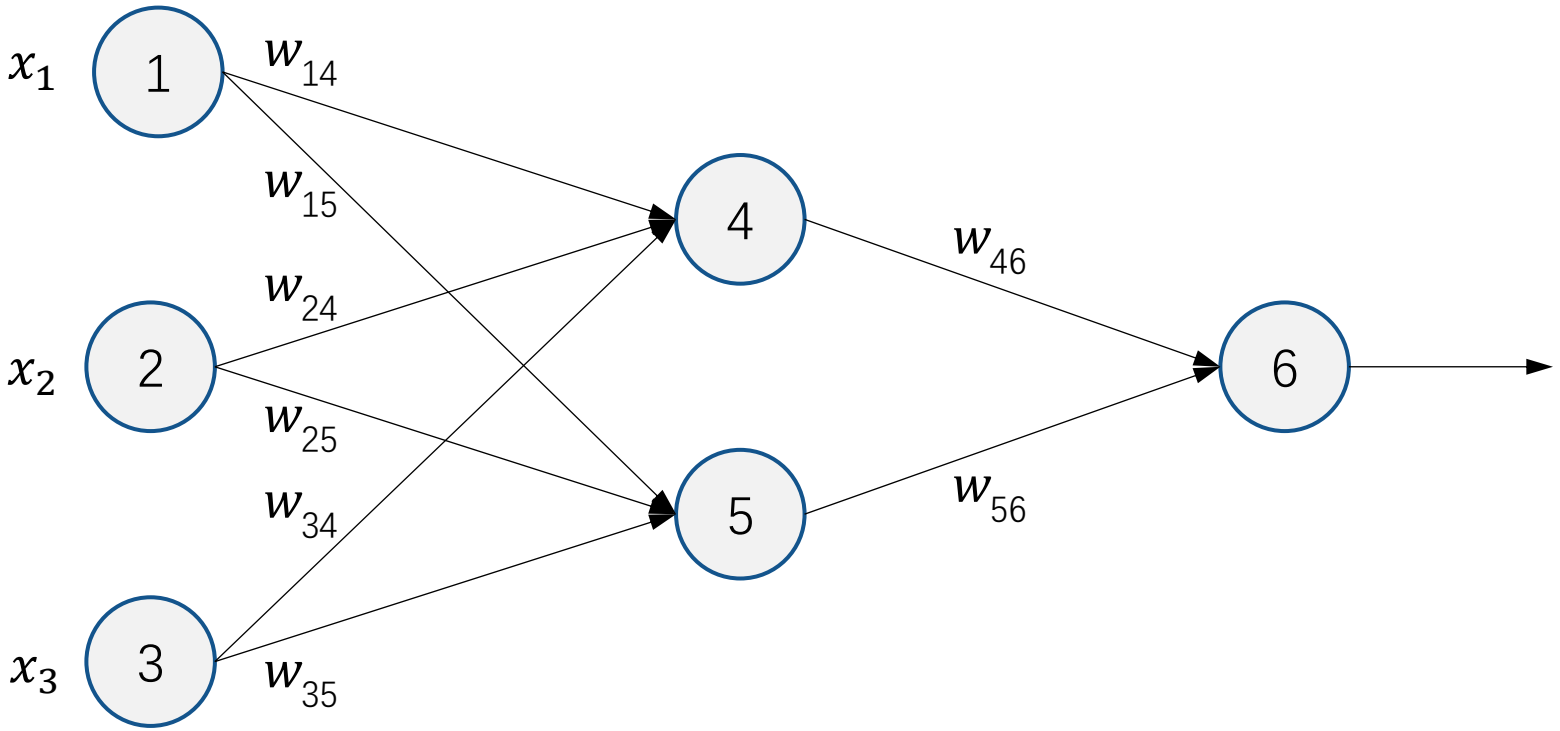
$$w_{ij} = w_{ij} + (l)Err_j O_i$$

$$\theta_j = \theta_j + (l)Err_j$$



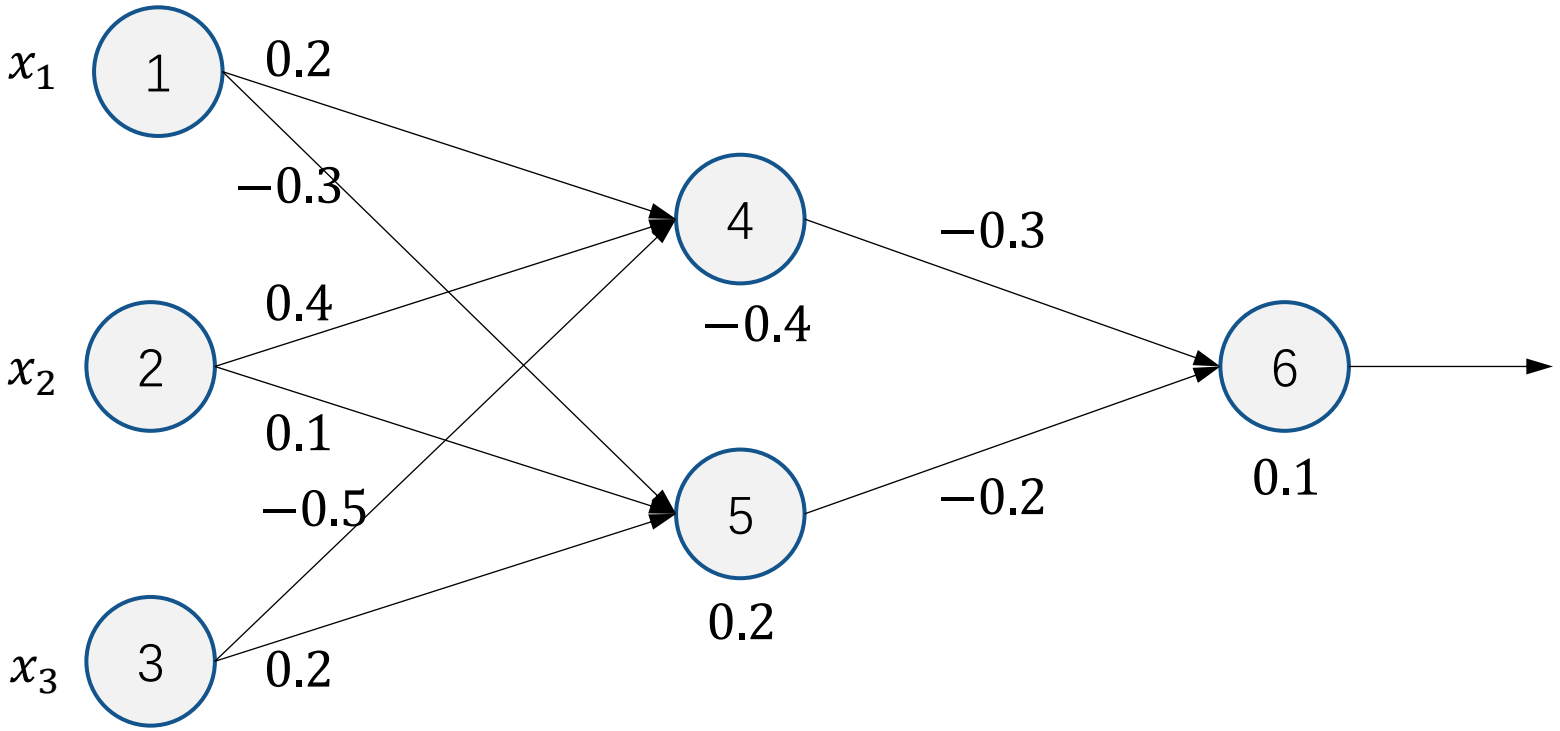
后向传播算法

1、初始化权重和偏置

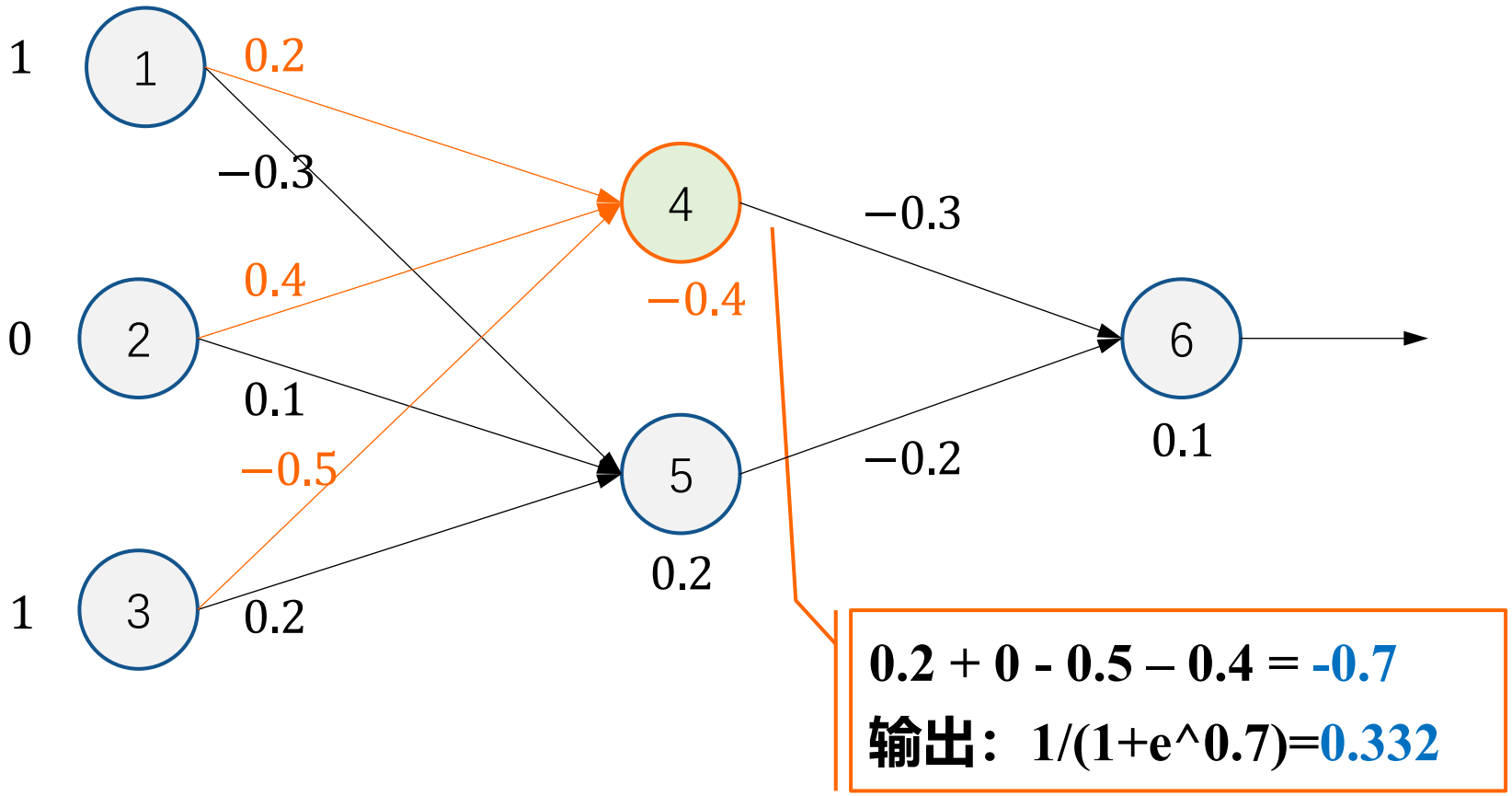


后向传播算法

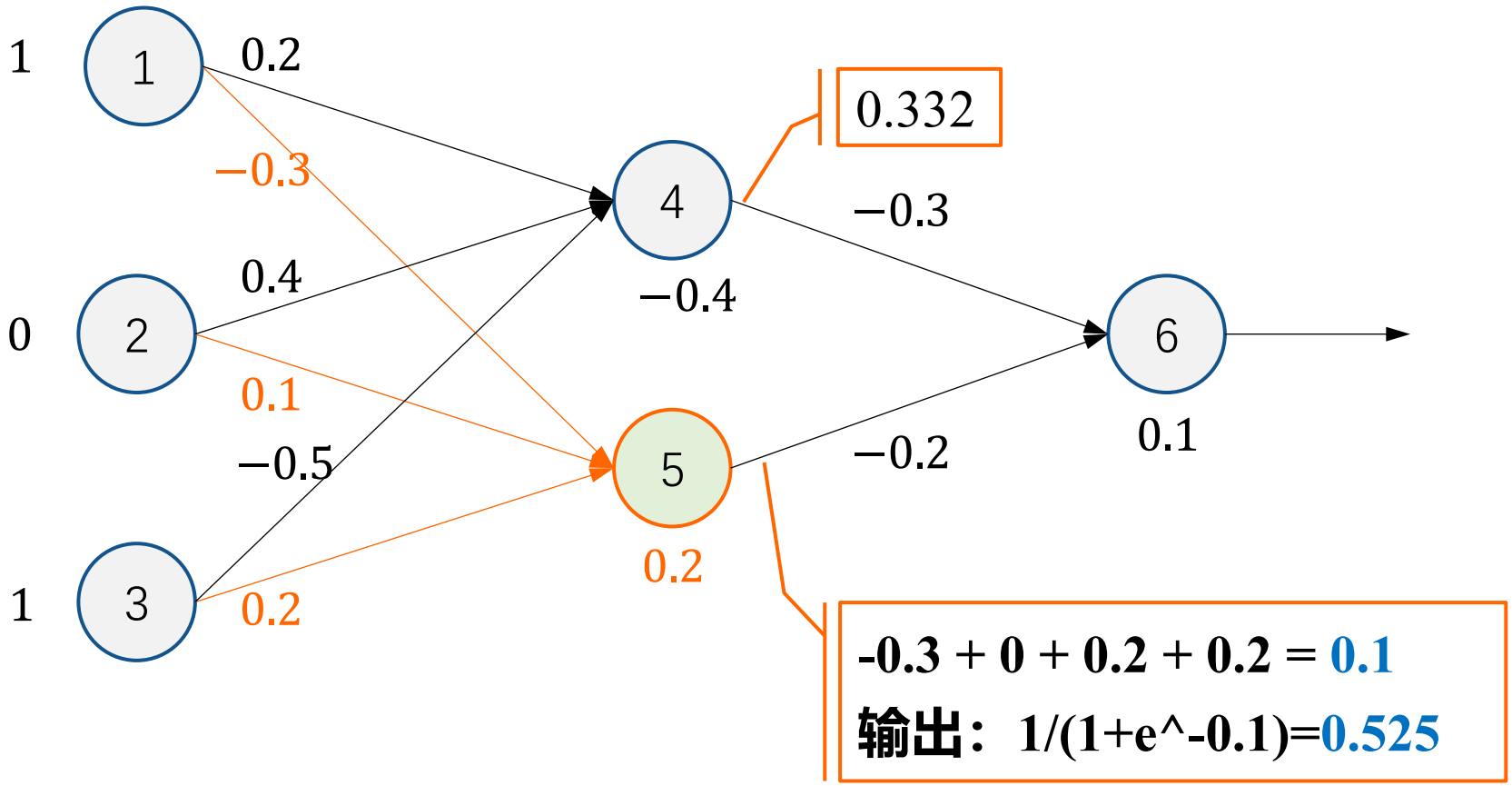
2、向前传播输入



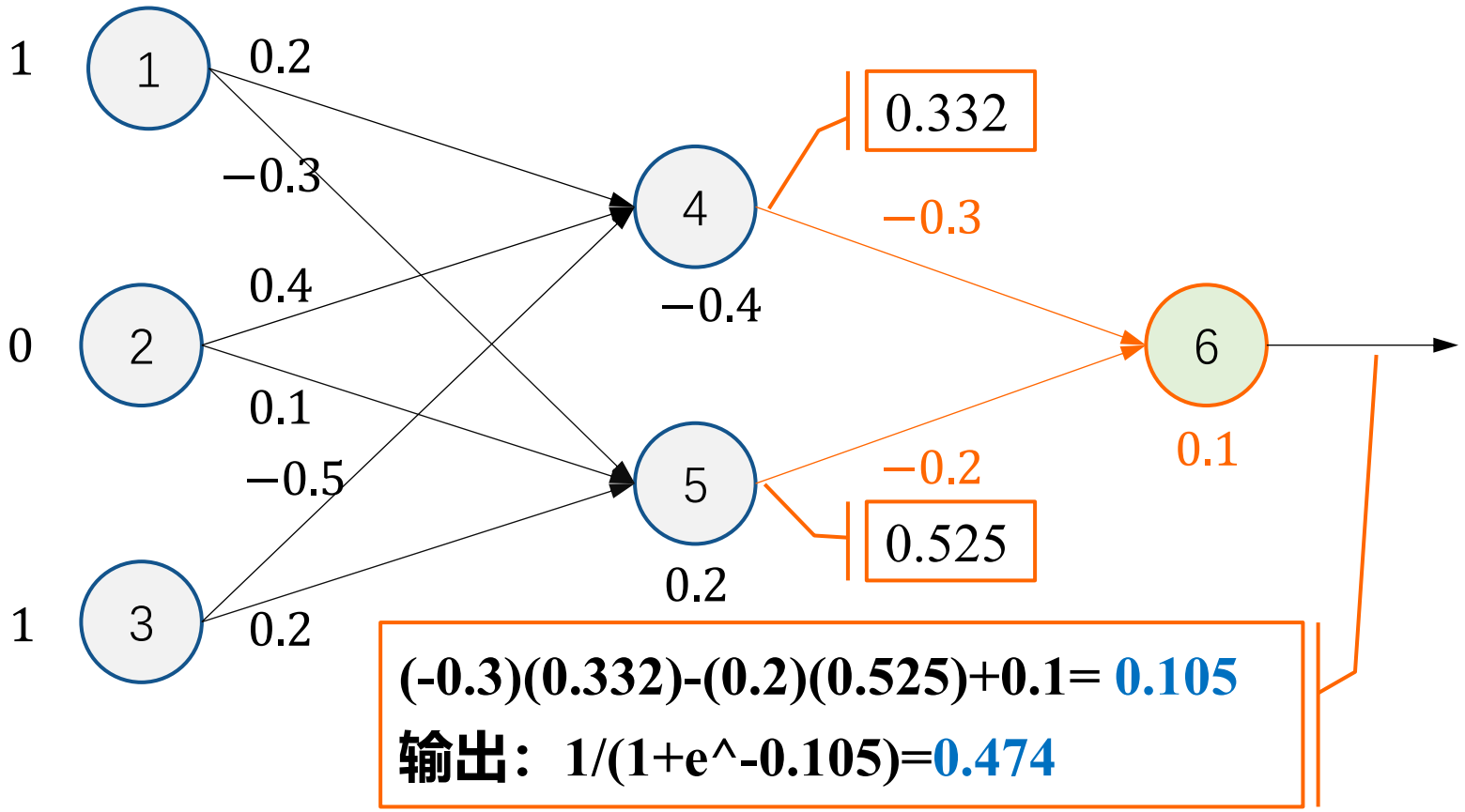
前向转播


1
0
1
1
2
3
0.2
−0.3
0.4
0.1
−0.5
0.2
4
−0.4
5
0.2
−0.3
−0.2
6
0.1
0.2 + 0 - 0.5 – 0.4 = -0.7
输出： 1/(1+e^0.7)=0.332

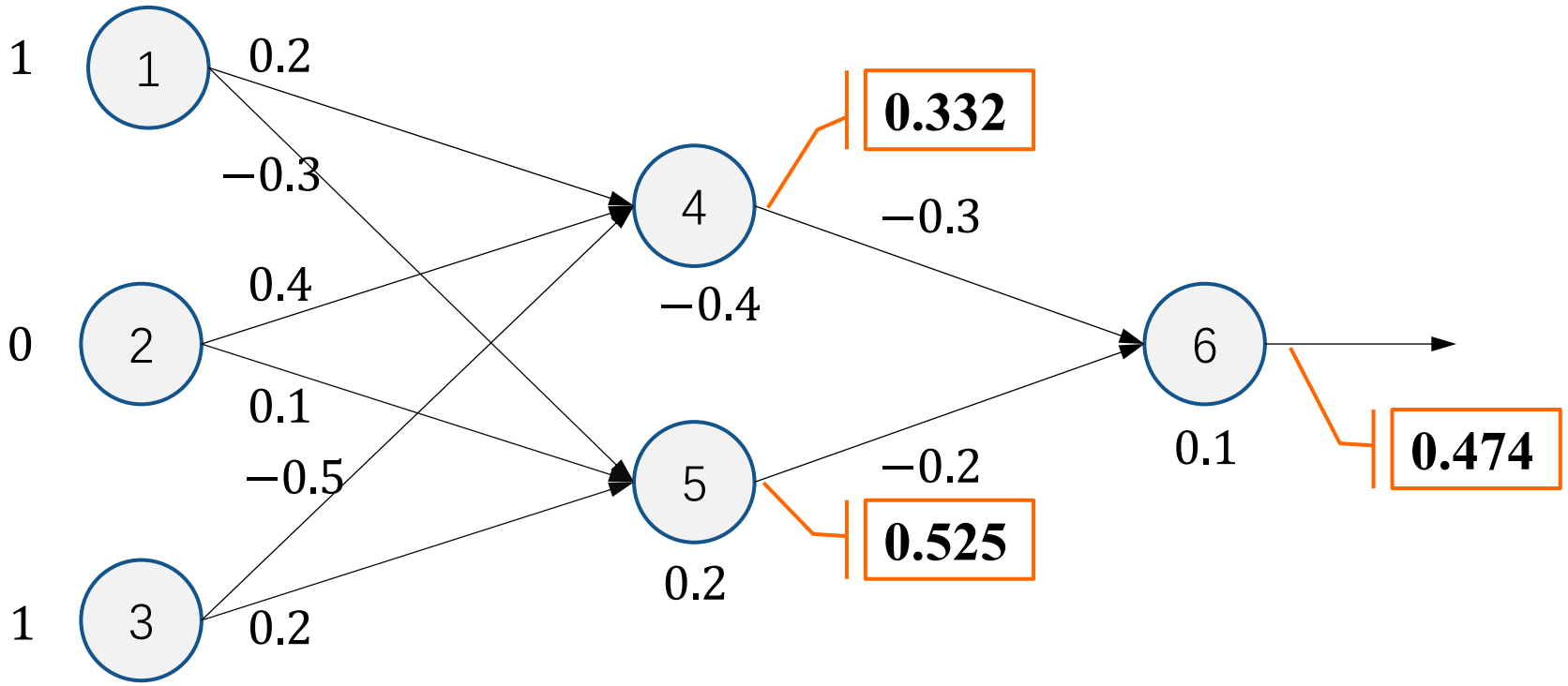
前向转播


1
0
1
1
2
3
0.2
−0.3
0.4
0.1
−0.5
0.2
4
5
−0.4
0.2
0.332
−0.3
−0.2
6
0.1
-0.3 + 0 + 0.2 + 0.2 = 0.1
输出：1/(1+e^-0.1)=0.525

前向转播


1
0
1
1
2
3
0.2
−0.3
0.4
0.1
−0.5
0.2
4
5
−0.4
0.2
0.332
−0.3
−0.2
0.525
6
0.1
(-0.3)(0.332)-(0.2)(0.525)+0.1= 0.105
输出：1/(1+e^-0.105)=0.474

前向转播



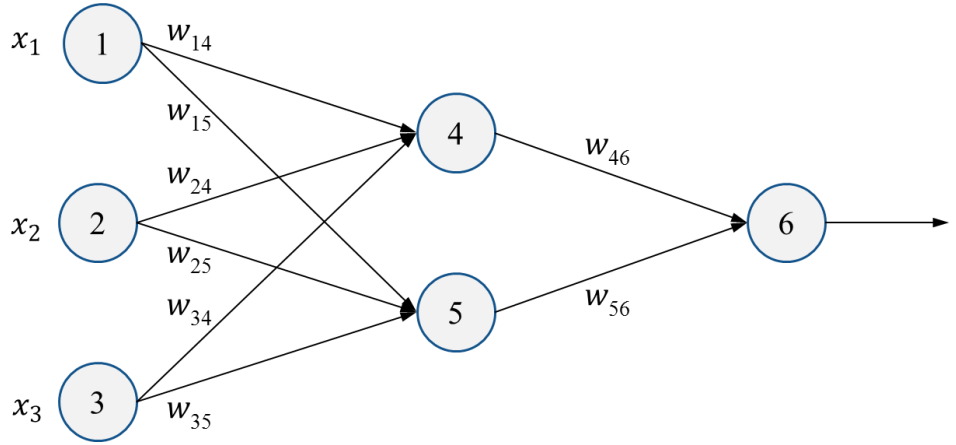
前向转播

- 初始随机设置参数

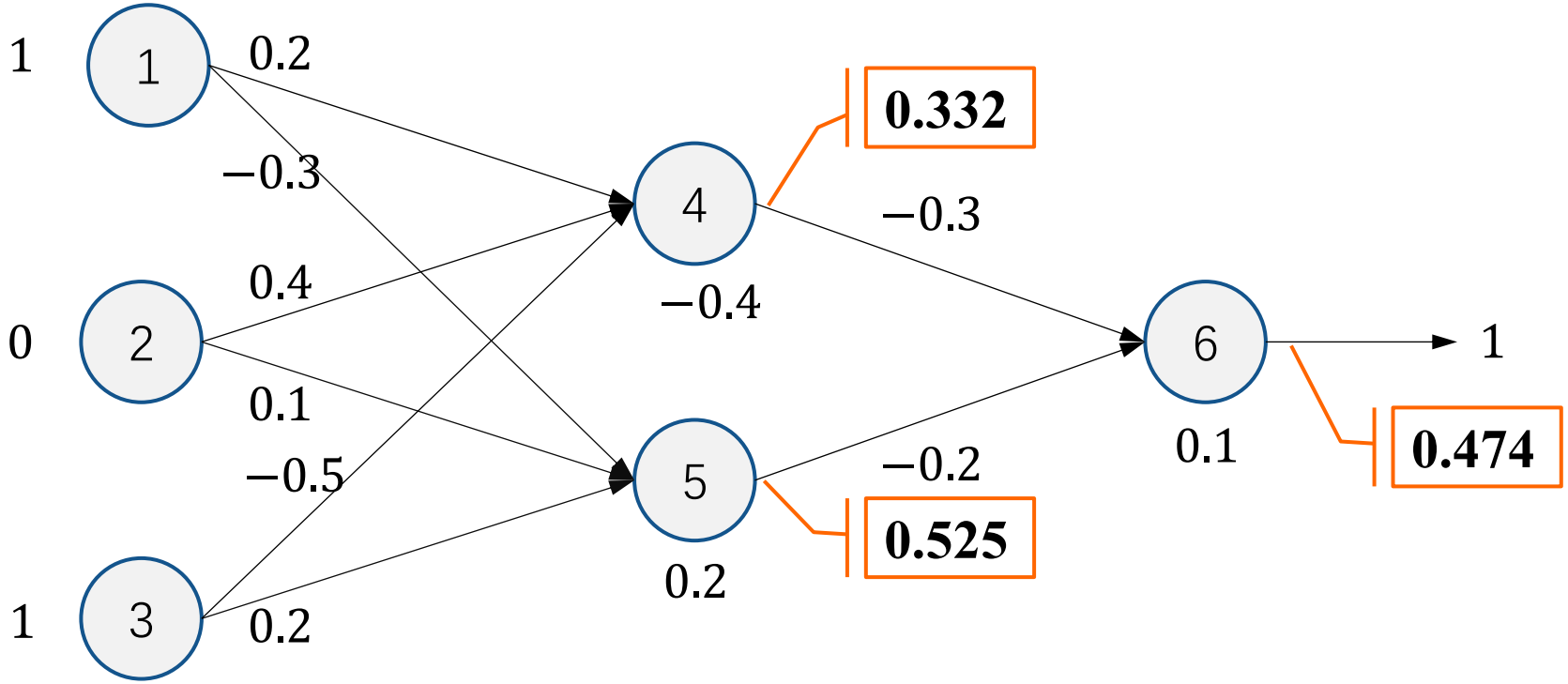
x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

- 输入和输出的表格

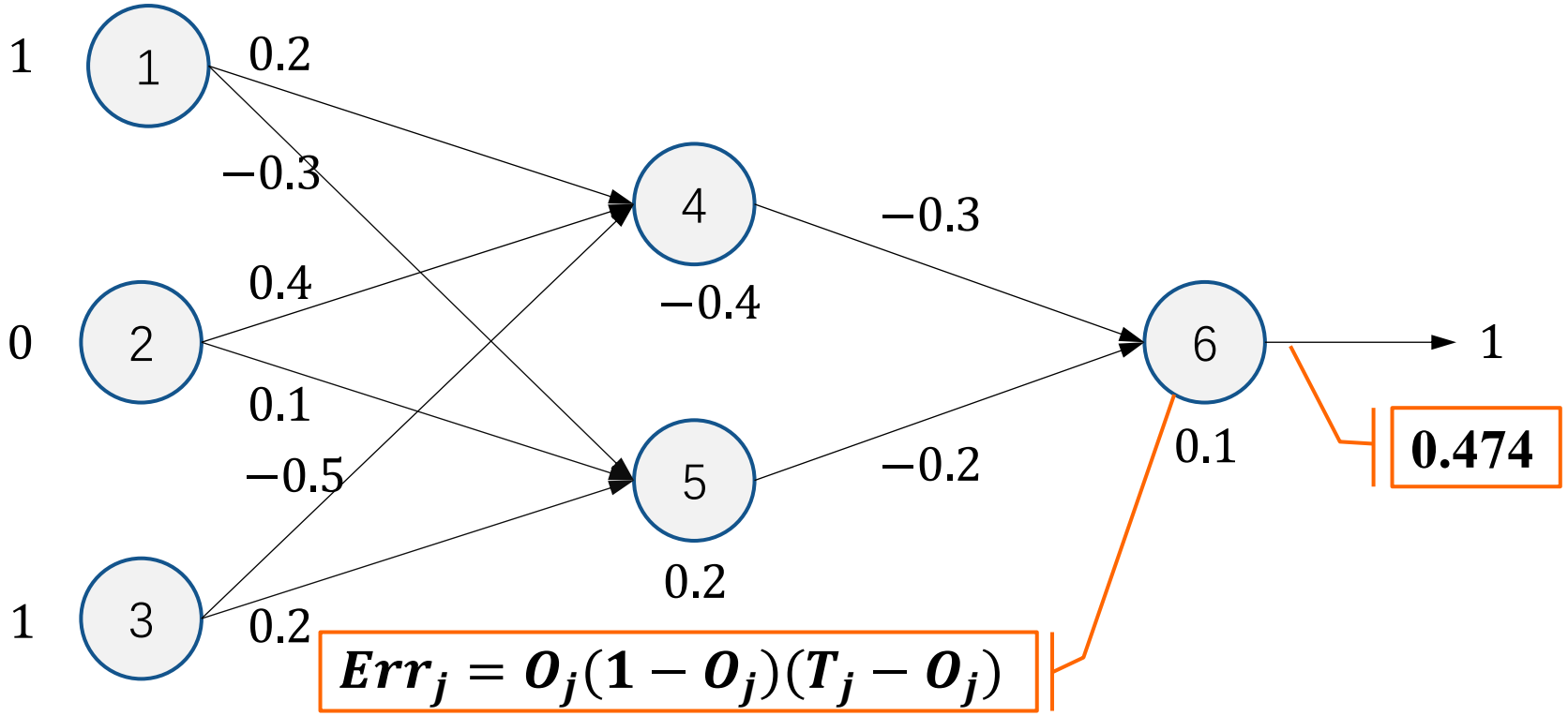
Unit j	*Net input,* I_j	*Output,* O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$



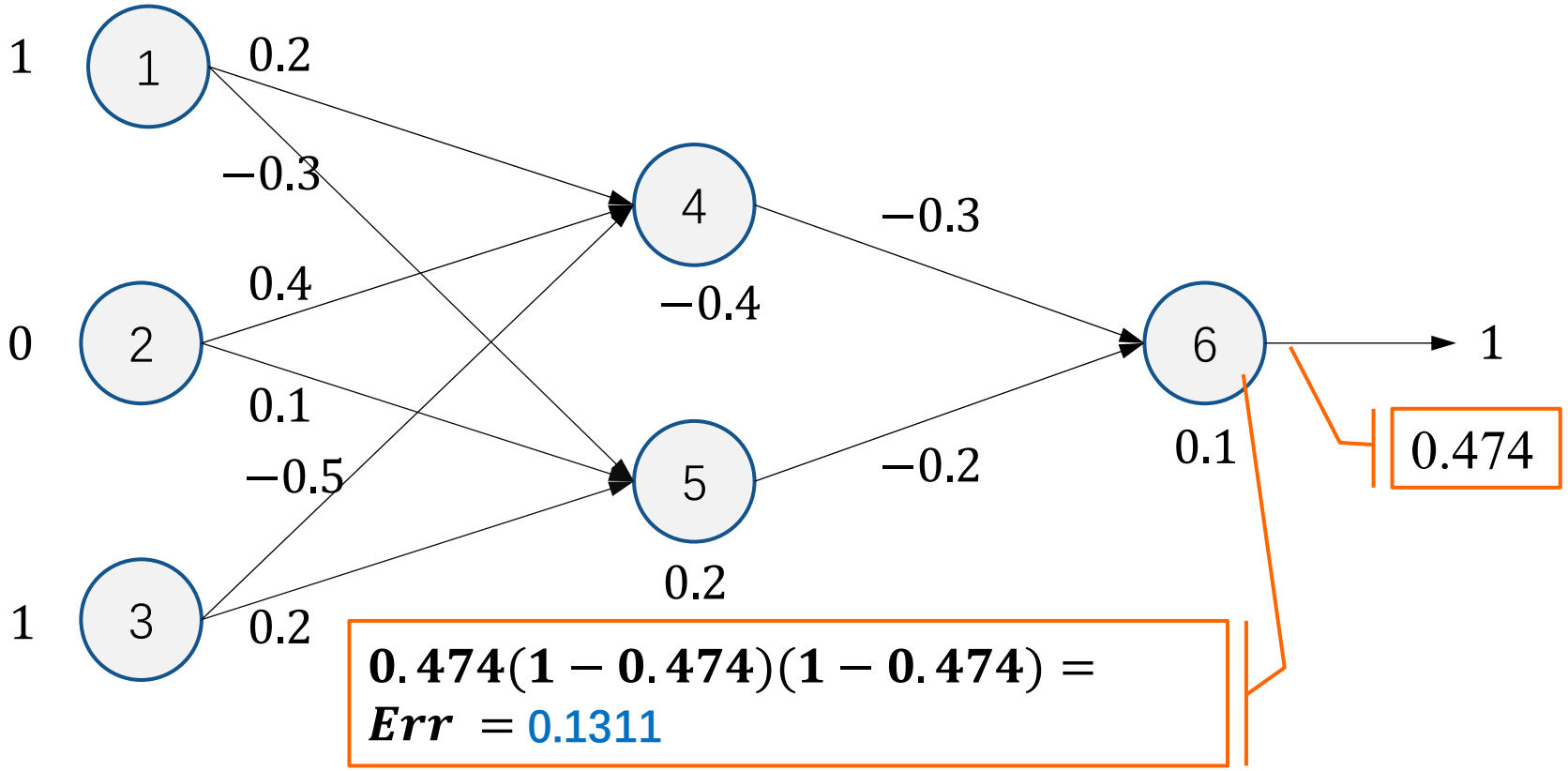
前向转播



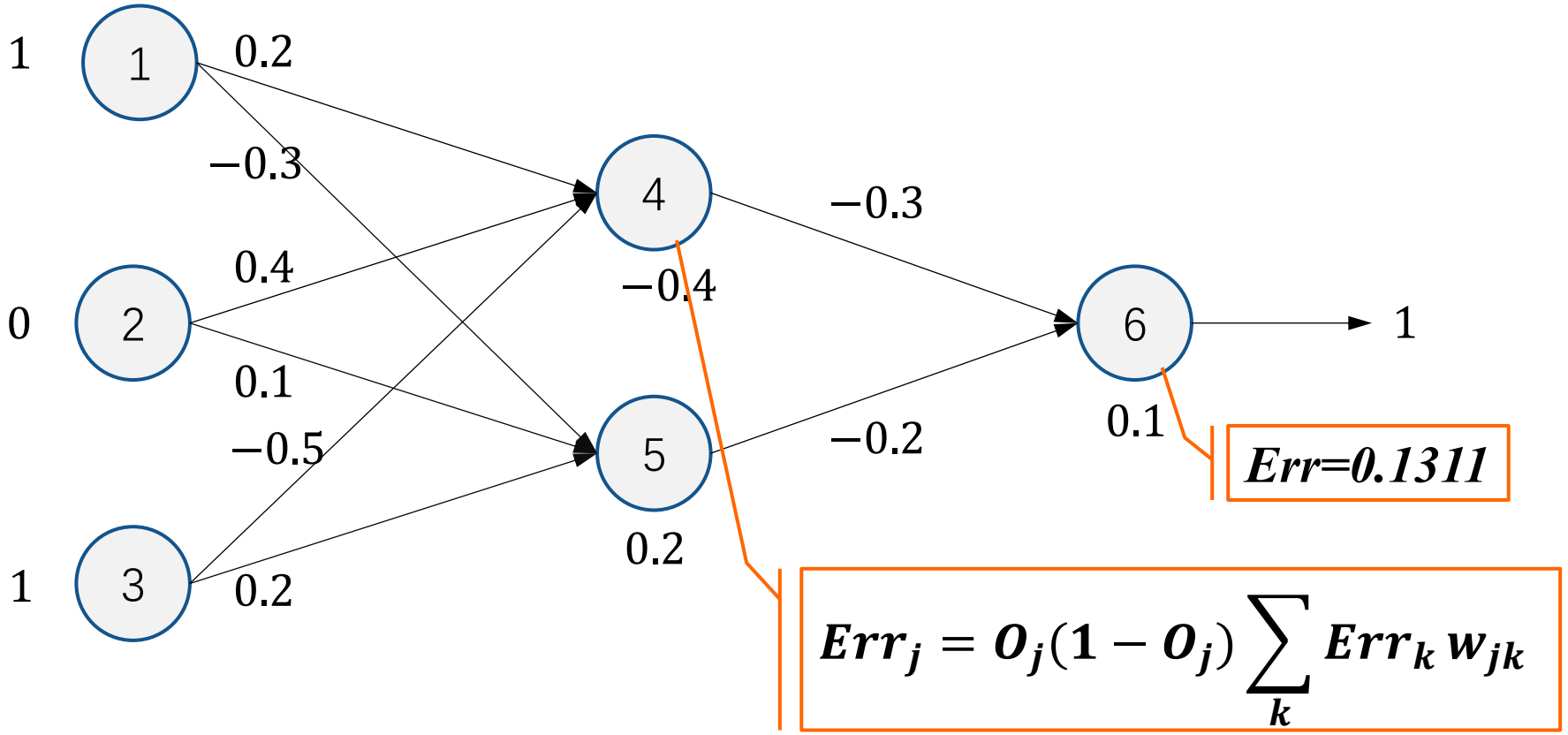
后向传播


1
1
0
2
1
3
1
0.2
−0.3
0.4
0.1
−0.5
0.2
4
−0.4
5
0.2
−0.3
−0.2
6
0.1
1
0.474
$Err_j = O_j(1 - O_j)(T_j - O_j)$

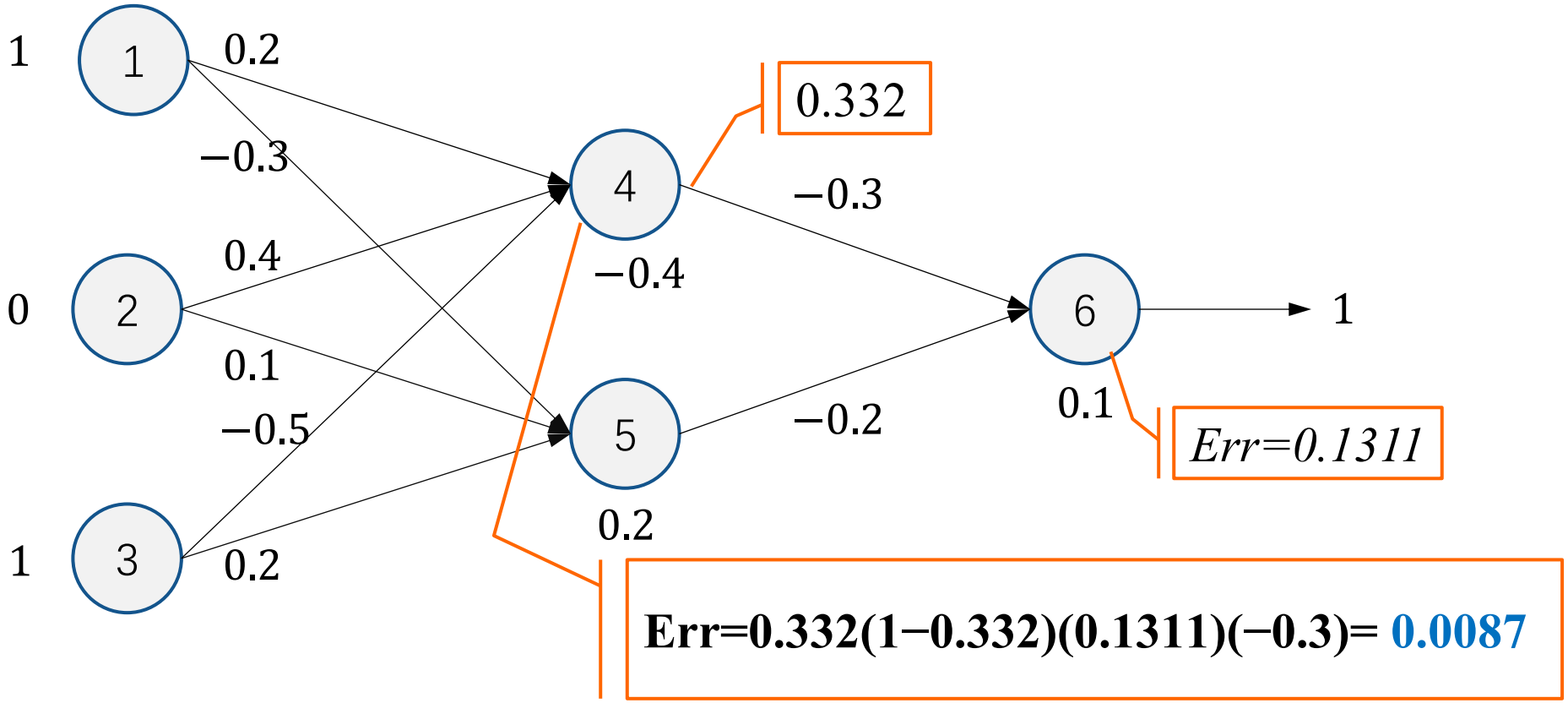
后向传播


1
1
0
2
1
3
1
0.2
−0.3
0.4
0.1
−0.5
0.2
4
−0.4
5
0.2
−0.3
−0.2
6
0.1
1
0.474
$\mathbf{0.474(1-0.474)(1-0.474)} =$
$\boldsymbol{Err} = 0.1311$

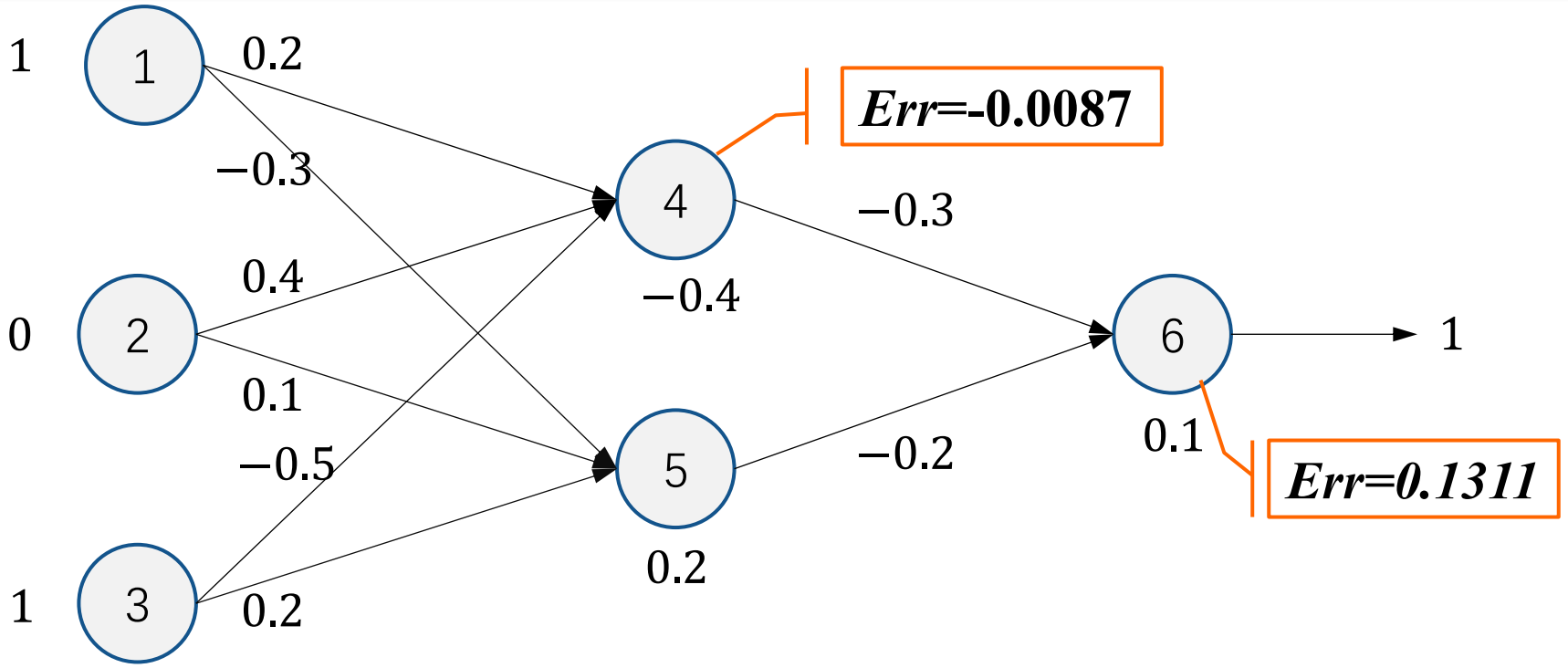
后向传播


1
0
1
1
2
3
4
5
6
0.2
−0.3
0.4
0.1
−0.5
0.2
−0.4
0.2
−0.3
−0.2
0.1
1
Err=0.1311
$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$

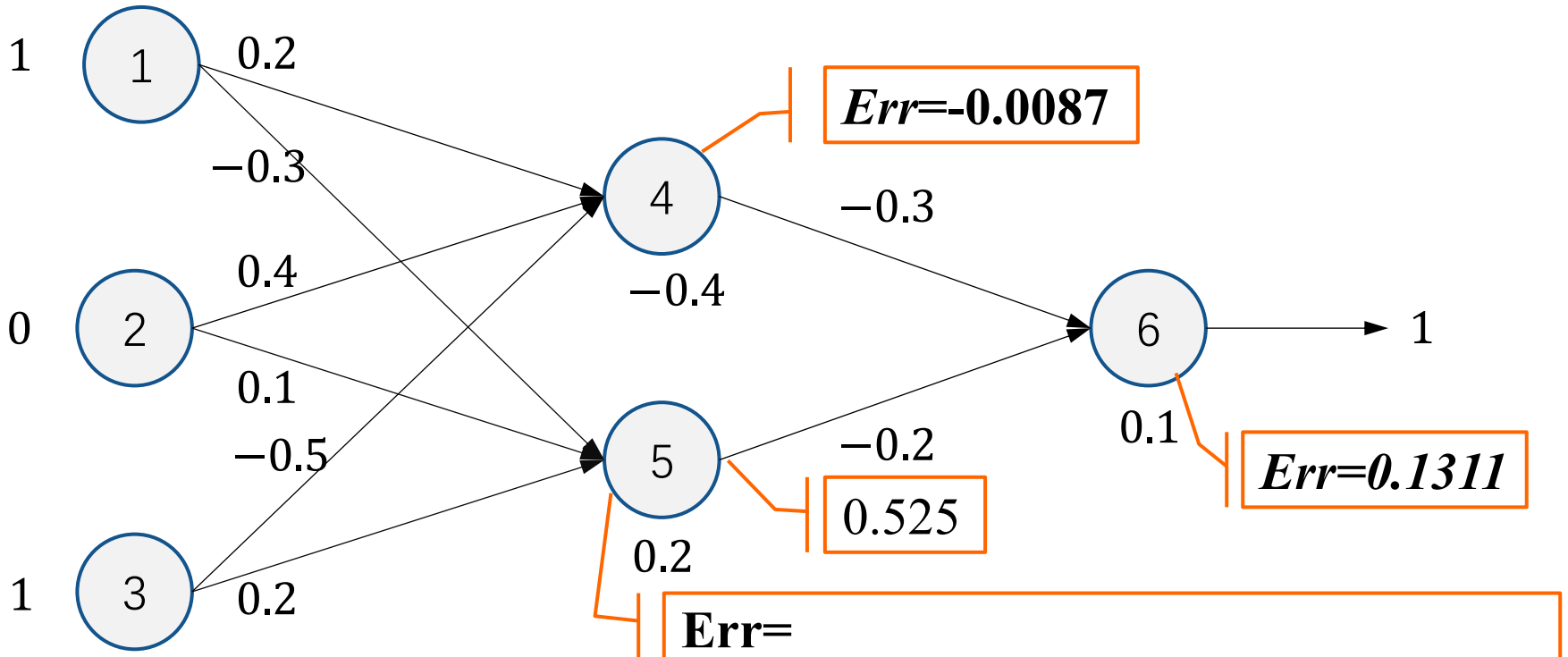
后向传播


1
1
0
1
2
3
0.2
−0.3
0.4
0.1
−0.5
0.2
4
5
−0.4
0.2
0.332
−0.3
−0.2
6
0.1
1
Err=0.1311
Err=0.332(1−0.332)(0.1311)(−0.3)= 0.0087

后向传播

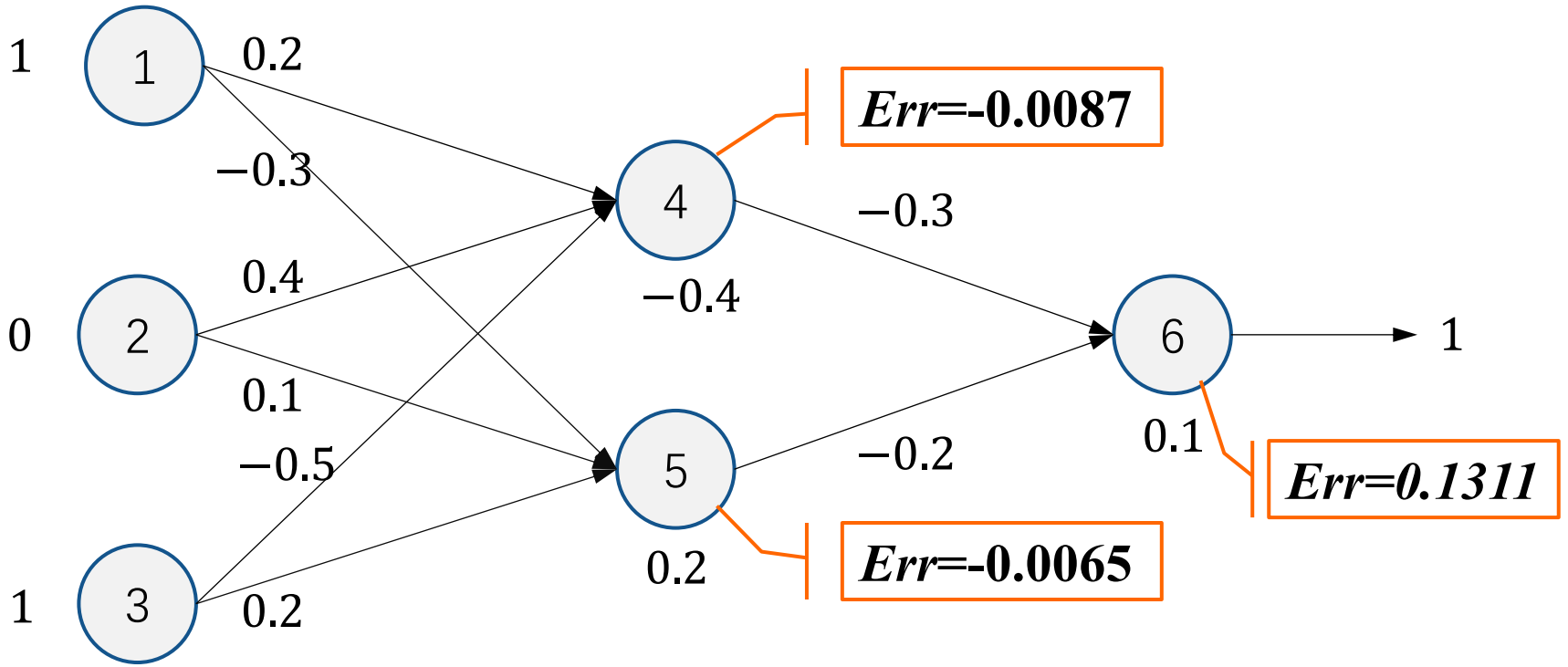


后向传播



$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

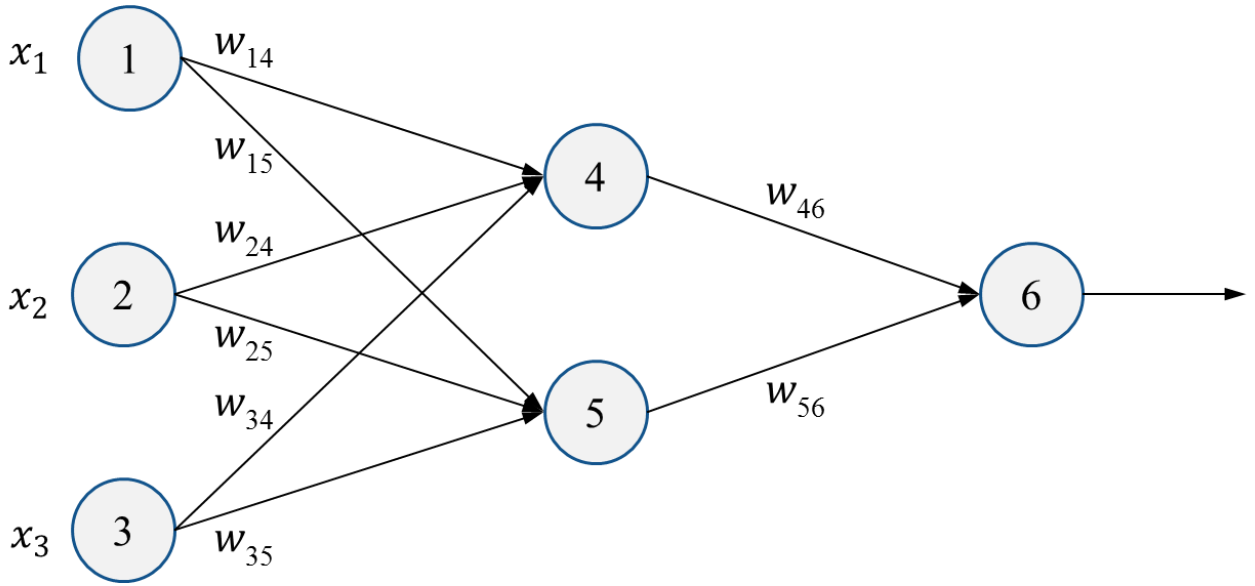
后向传播


1
1
0
2
1
3
1
0.2
−0.3
0.4
0.1
−0.5
0.2
4
−0.4
5
0.2
Err=-0.0087
−0.3
−0.2
Err=-0.0065
6
0.1
1
Err=0.1311

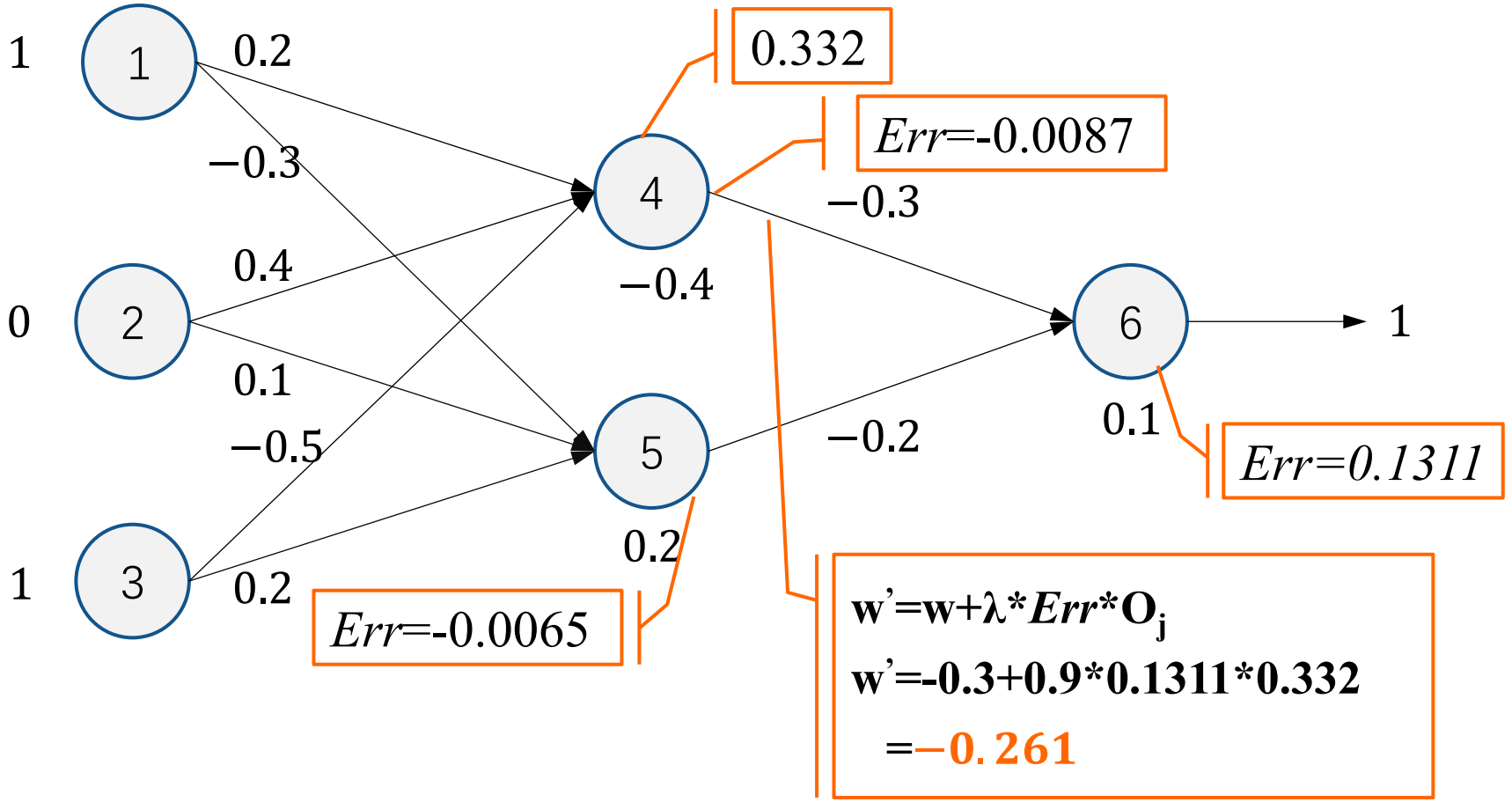
后向传播

- 计算各结点的误差

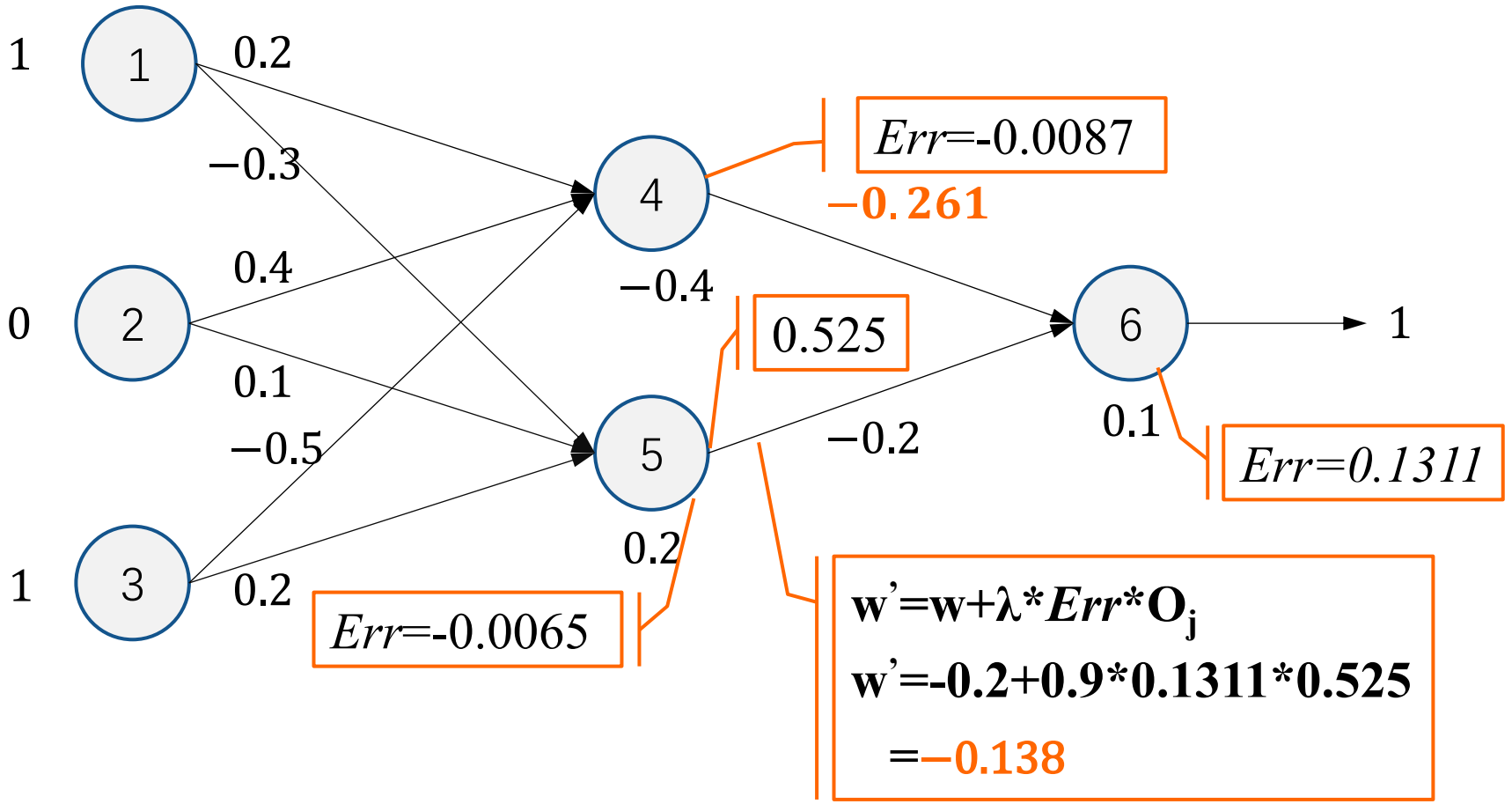
Unit j	Err_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$



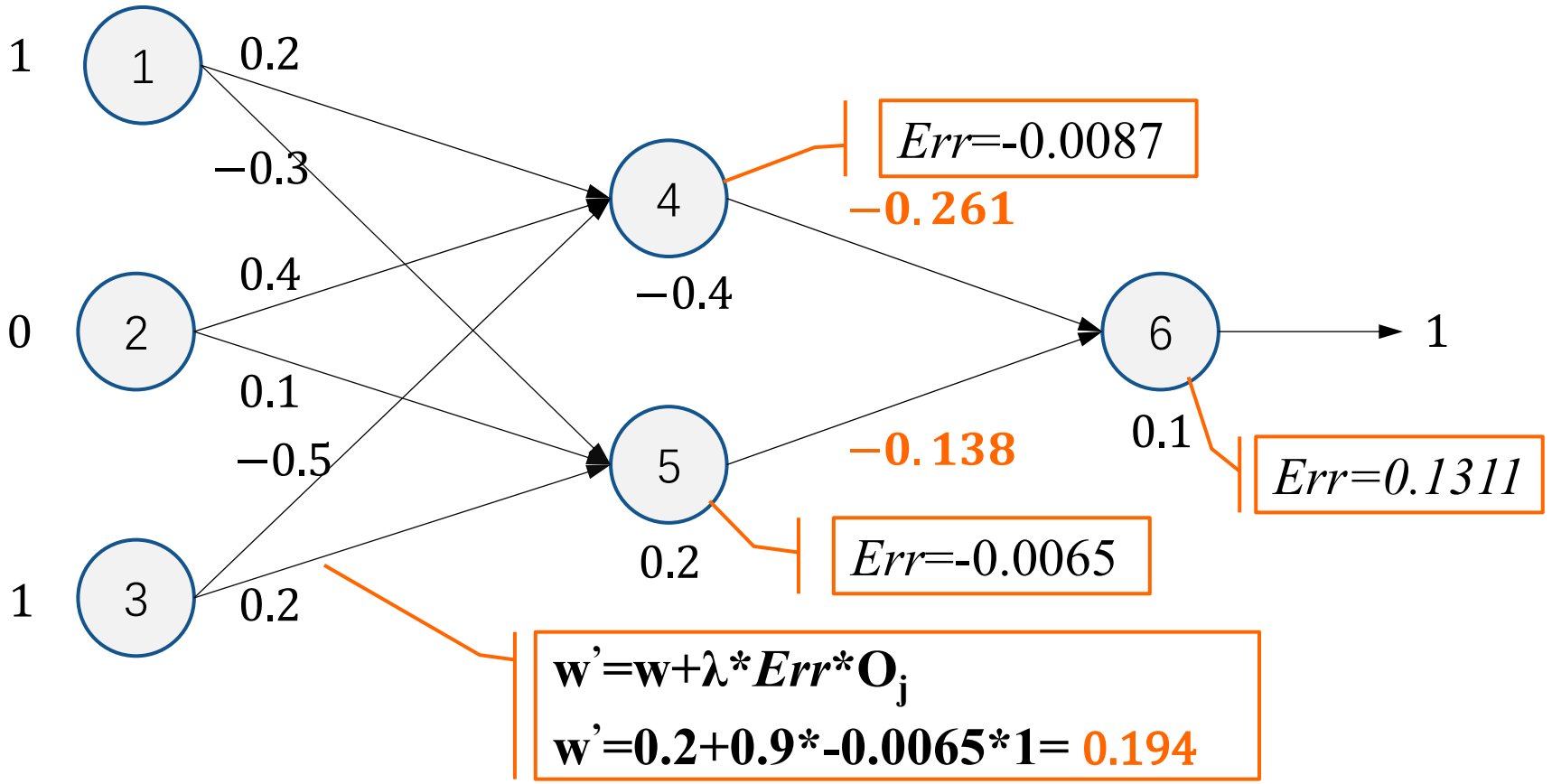
权重更新


1
1
0
2
1
3
1
0.2
−0.3
0.4
0.1
−0.5
0.2
4
5
0.332
Err=-0.0087
−0.3
−0.4
0.2
Err=-0.0065
6
1
−0.2
0.1
Err=0.1311
w'=w+λ*Err*O_j
w'=-0.3+0.9*0.1311*0.332
=−0.261

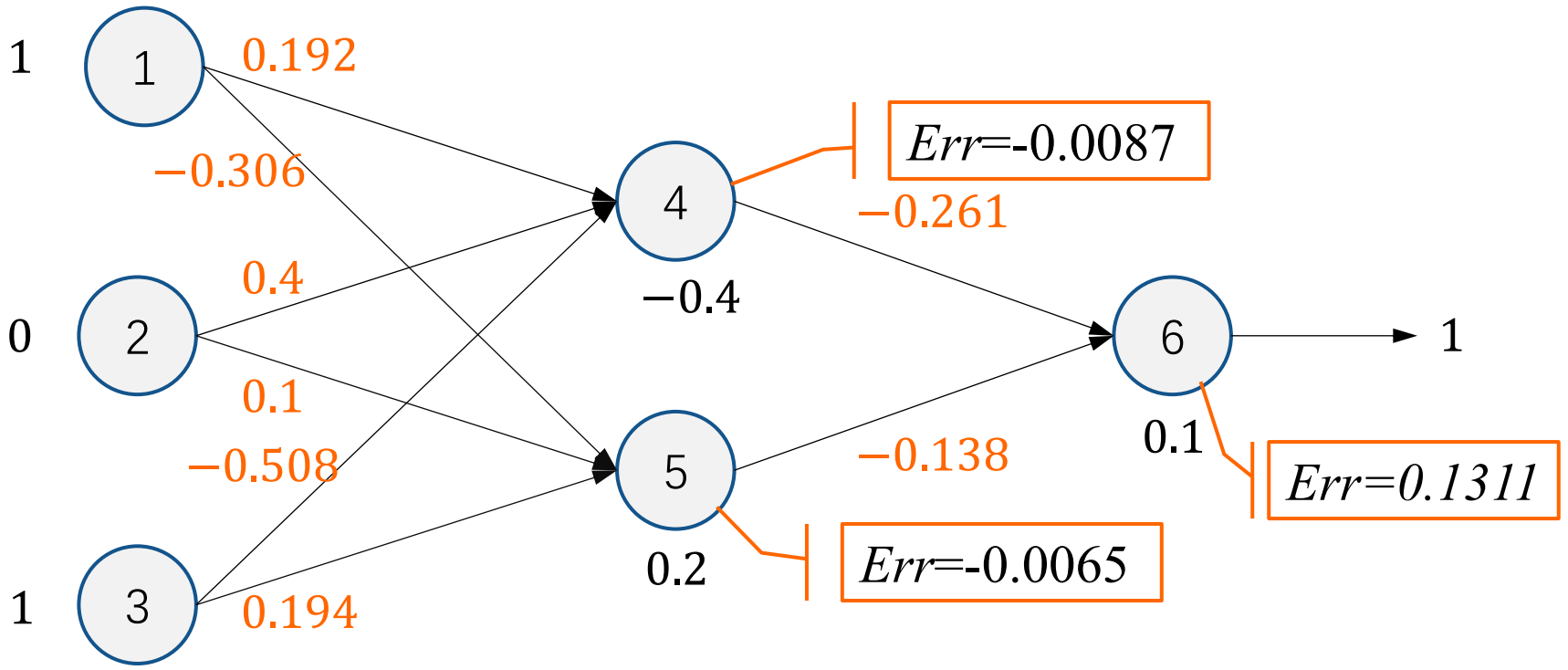
权重更新


1
0
1
1
2
3
0.2
−0.3
0.4
0.1
−0.5
0.2
4
5
−0.4
0.2
Err=-0.0087
−0.261
0.525
Err=-0.0065
6
−0.2
0.1
1
Err=0.1311
w'=w+λ*Err*O_j
w'=-0.2+0.9*0.1311*0.525
=−0.138

权重更新


1
1
0
1
2
3
0.2
−0.3
0.4
0.1
−0.5
0.2
4
5
−0.4
0.2
Err=-0.0087
−0.261
−0.138
6
1
0.1
Err=0.1311
Err=-0.0065
w'=w+λ*Err*O_j
w'=0.2+0.9*-0.0065*1= 0.194

后向传播算法


1
1
0.192
−0.306
0
2
0.4
0.1
−0.508
1
3
0.194
4
−0.4
5
0.2
Err=-0.0087
−0.261
6
1
0.1
−0.138
Err=0.1311
Err=-0.0065

后向传播算法

- **根据误差调整新的权重和偏置**

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

$$w_{ij} = w_{ij} + \lambda Err_j y_i$$

$$t_j = t_j + \lambda Err_j$$

Weight or bias	*New value*
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
w_{35}	$0.2 + (0.9)(-0.0065)(1) = 0.194$
θ_6	$0.1 + (0.9)(0.1311) = 0.218$
θ_5	$0.2 + (0.9)(-0.0065) = 0.194$
θ_4	$-0.4 + (0.9)(-0.0087) = -0.408$

后向传播算法

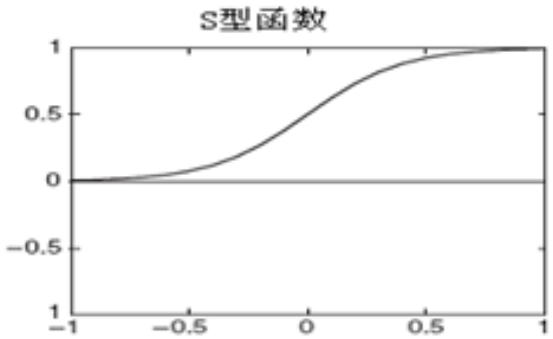
1、初始化权重

- 循环以下两步，直到满足条件

2、向前传播输入

- 在每个节点加权求和，再代入激活函数

$$I_j = \sum_i x_i w_{ij} \qquad y = \frac{1}{1+e^{-x}}$$



3、向后传播误差

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta w_{ij} = (l)Err_j O_i$$

$$\theta_j = \theta_j + (l)Err_j$$

均方误差

$$J(w,b) = \frac{1}{n}\sum \frac{1}{2}(Y - Y^{hat})^2$$

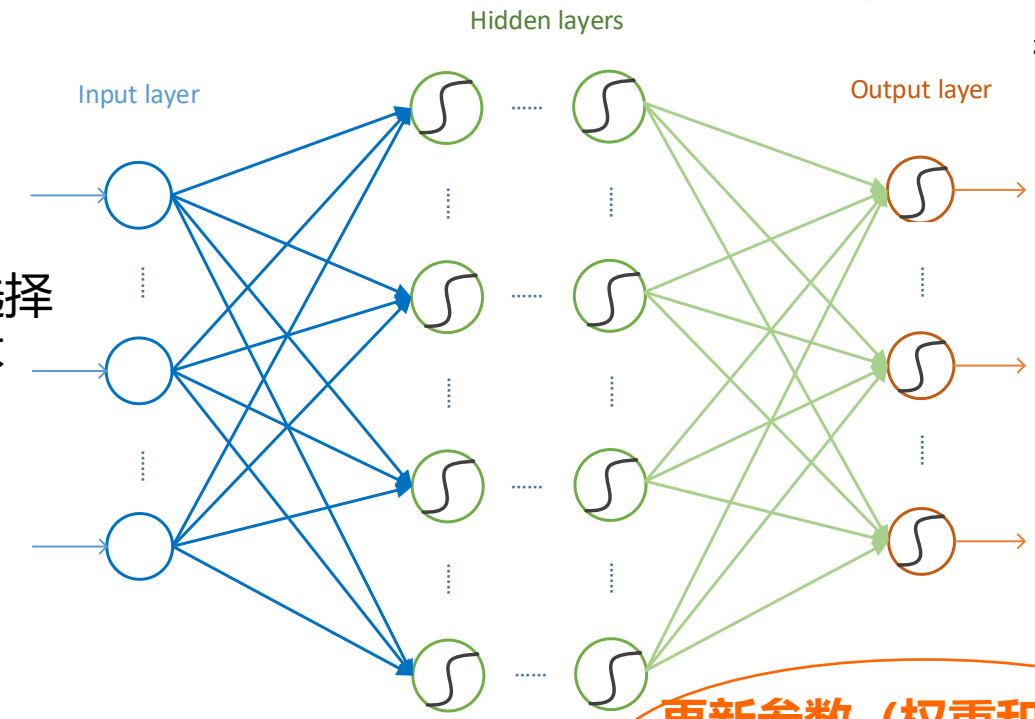
权值迭代，梯度下降法

后向传播算法推导

$$D = \{(\vec{x_1}, \vec{y_1}), (\vec{x_2}, \vec{y_2}), \cdots, (\vec{x_m}, \vec{y_m})\}, \vec{x_i} \in R^d, \vec{y_i} \in R^l$$

训练集　　验证集

前向传播



从训练集中选择一部分样本 (Batch)

每个样本产生的预测和其真实标签存在一定误差

预测标签 ≠ 真实标签

损失函数

最小化损失

更新参数（权重和阈值）

迭代

后向传播算法推导

- 什么是梯度?

  For example: $f(\theta_0, \theta_1) \xrightarrow{\nabla} (\frac{\partial f}{\partial \theta_0}, \frac{\partial f}{\partial \theta_1})^T$

- 梯度 ∇f 指向函数增长最快的方向

- $-\nabla f$ 指向函数下降最快的方向

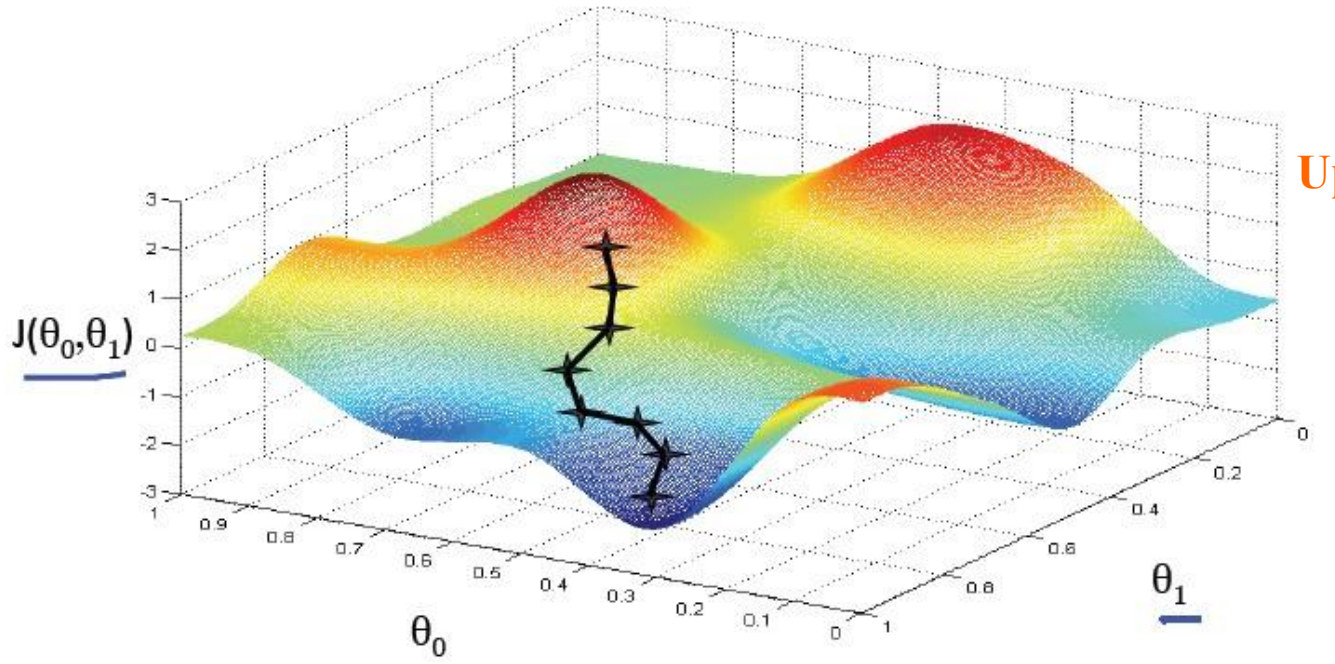
最小化 $f(\theta_0, \theta_1)$ → $\vec{\theta} \leftarrow \vec{\theta} - \eta \nabla f(\vec{\theta})$ →

$$\theta_0 \leftarrow \theta_0 - \eta \frac{\partial f}{\partial \theta_0}$$

$$\theta_1 \leftarrow \theta_1 - \eta \frac{\partial f}{\partial \theta_1}$$

后向传播算法推导

梯度下降是求一个函数最小值的一阶迭代优化算法



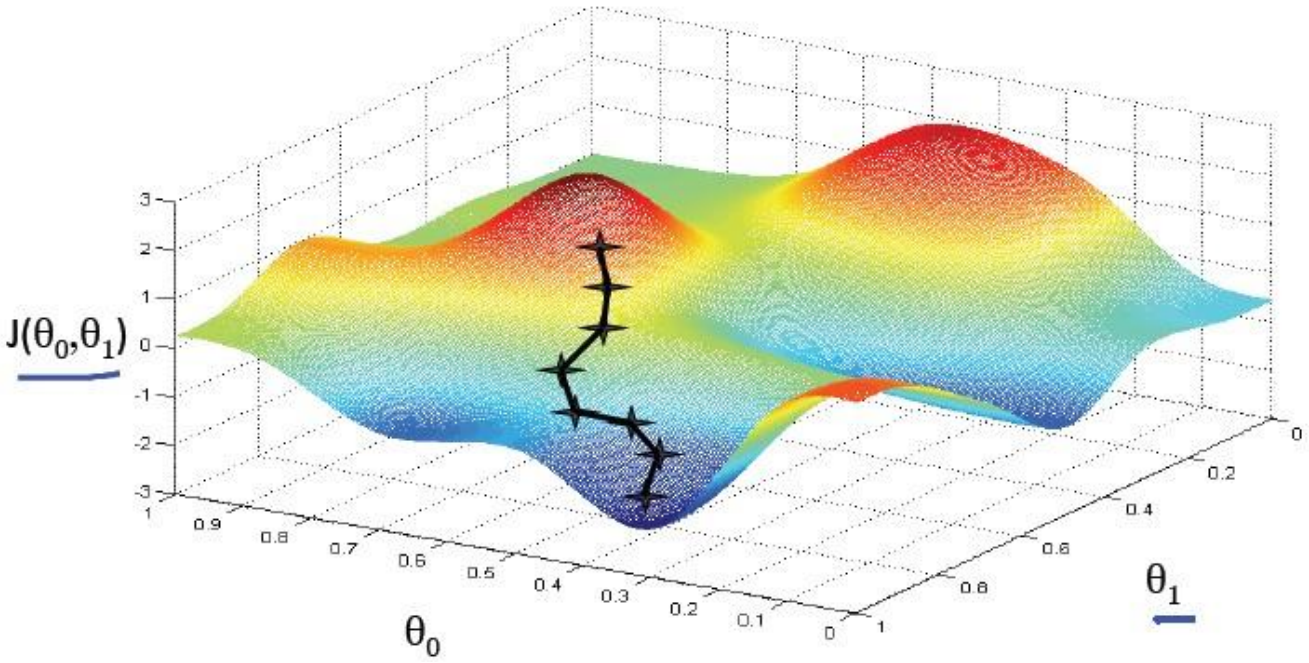
Update: $\vec{\theta} \leftarrow \vec{\theta} - \eta \nabla f(\vec{\theta})$

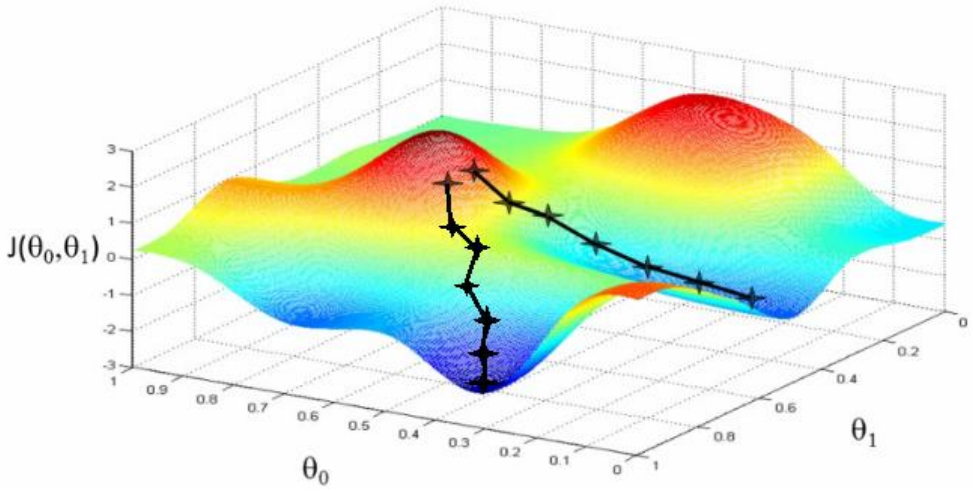
$$\theta_0 \leftarrow \theta_0 - \eta \frac{\partial f}{\partial \theta_0}$$

$$\theta_1 \leftarrow \theta_1 - \eta \frac{\partial f}{\partial \theta_1}$$

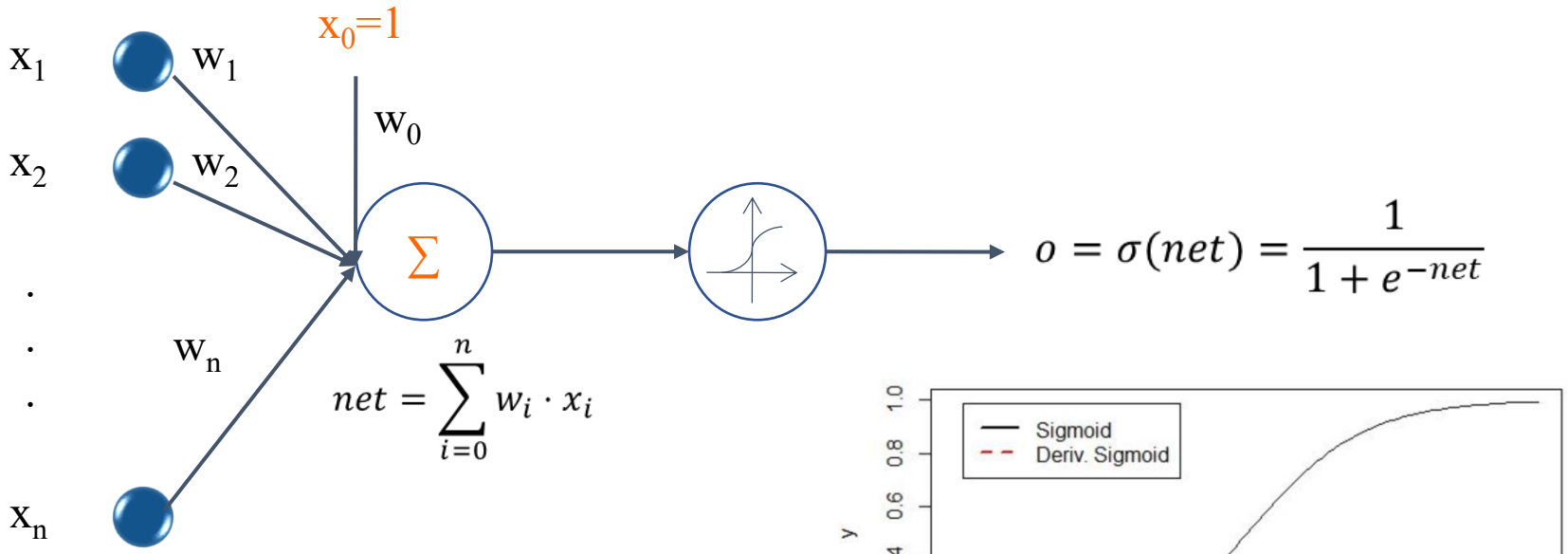
后向传播算法推导

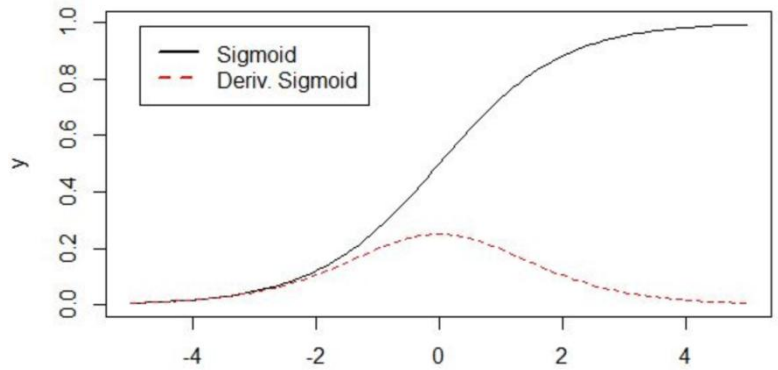
梯度下降是求一个函数最小值的一阶迭代优化算法





后向传播算法推导





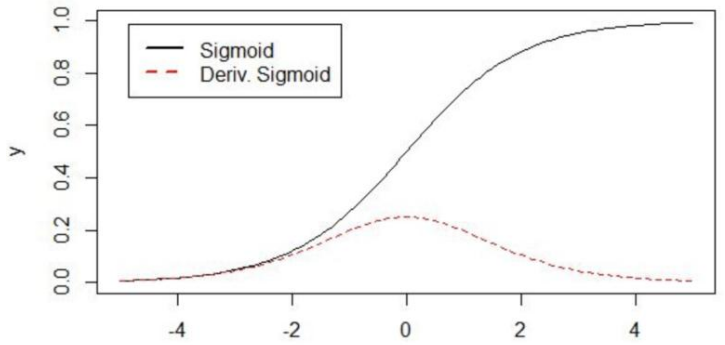
logistic函数（sigmoid函数）

$$\sigma(y)=\frac{1}{1+e^{-y}} \qquad \frac{d\sigma(y)}{dy}=\sigma(y)\cdot(1-\sigma(y))$$

后向传播算法推导

Logistic函数求导

$$y = \frac{1}{1+e^{-x}}$$



$$y'_x = \frac{e^{-x}}{(1+e^{-x})^2} \tag{1}$$

$$= \frac{1+e^{-x}-1}{(1+e^{-x})^2} \tag{2}$$

$$= \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2} \tag{3}$$

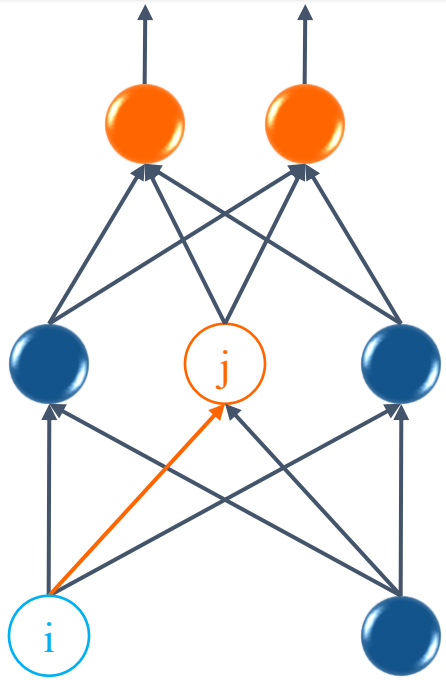
$$= \frac{1}{1+e^{-x}}(1 - \frac{1}{1+e^{-x}}) \tag{4}$$

$$= y(1-y) \tag{5}$$

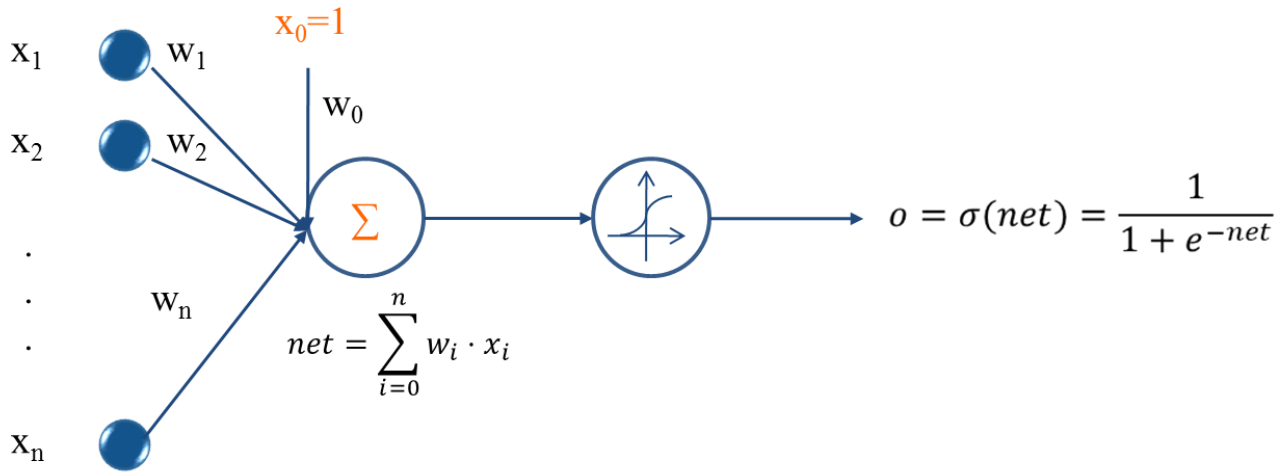
后向传播算法推导

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

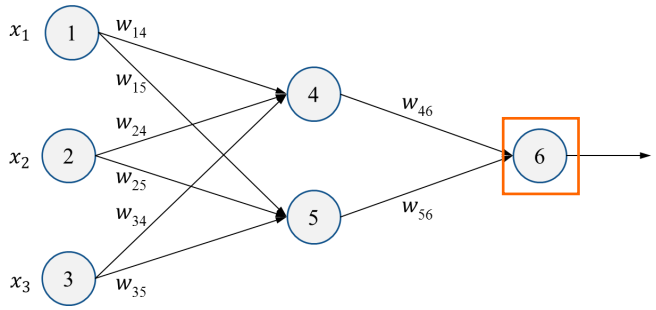
$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j}x_{ij}$$



- $net_j = \sum w_{ij}x_{ij}$ （j神经元输入的加权和）
- 激活函数，the sigmoid function
- *outputs* = 网络最后一层中的神经元
- *Downstream* (*j*) = 将j神经元的输出作为输入的神经元集合



Output Units 输出神经元的训练



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

Sigmoid Function

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

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

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

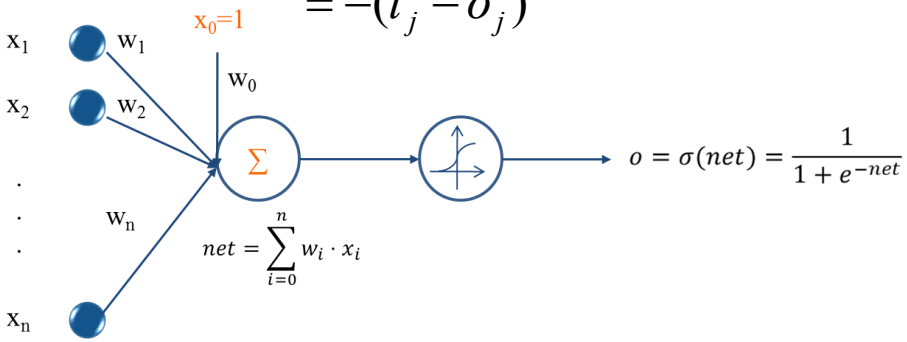
$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2$$

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

$$= -(t_j - o_j)$$

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

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

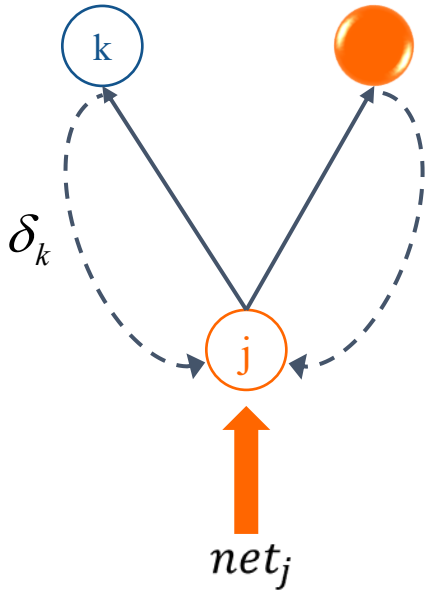


$$\Delta w_{ij} = -\eta \frac{\partial E_d}{\partial w_{ij}} = \eta (t_j - o_j) o_j (1-o_j) x_{ij}$$

Hidden Units 隐藏层神经元的训练

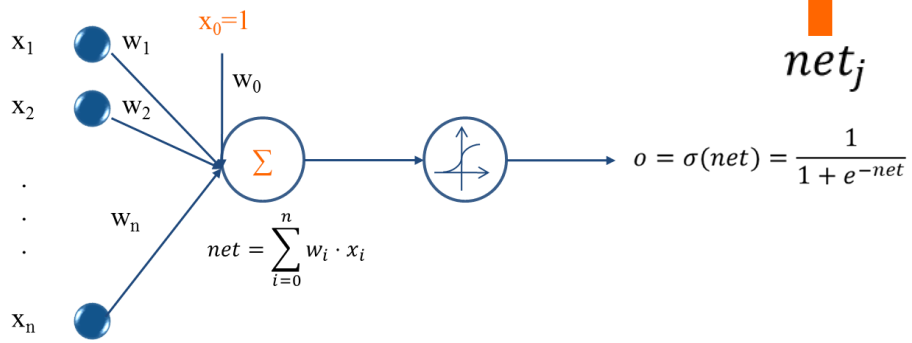
$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

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



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

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



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

后向传播算法推导

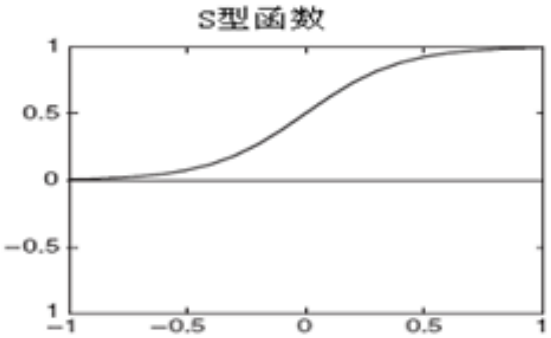
1、初始化权重

- 循环以下两步，直到满足条件

2、向前传播输入

- 在每个节点加权求和，再代入激活函数

$$I_j = \sum_i w_{ij} x_i \qquad y = \frac{1}{1+e^{-I}}$$



3、向后传播误差

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

$$w_{ij} = w_{ij} + (l)Err_j O_i$$

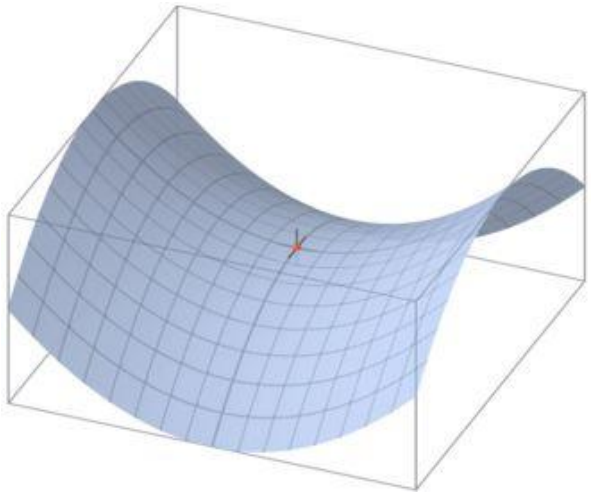
$$\theta_j = \theta_j + (l)Err_j$$

均方误差

$$J(w,b) = \frac{1}{n}\sum \frac{1}{2}(Y - Y^{hat})^2$$

后向传播BP网络注意事项

- 初始值选择
  - 权值向量以及阀值的初始值应设定在**一均匀分布的小范围内**
  - 初始值**不能为零**，否则性能曲面会趋向于**鞍点**
  - 初始值**不能太大**，否则远离优化点，导致性能曲面平坦，学习率很慢
- 训练样本输入次序
  - 不同，也会造成不一样的学习结果
  - 在每一次的学习循环中，输入向量输入网络的**次序应使其不同**
- BP算法的学习过程的终止条件
  - 权值向量的梯度 < **给定值**
  - 均方误差值 < **给定误差容限值**
  - 若其**推广能力达到目标**则予终止
  - 可以结合上述各种方式



总 结

- 普适近似，精度较高
- 噪声敏感
- 训练非常耗时，但对目标分类较快

作业：泰坦尼克号幸存者预测

- RMS泰坦尼克号的沉没是历史上最著名的沉船之一。 1912年4月15日，在首航期间，泰坦尼克号撞上冰山后沉没，2224名乘客和机组人员中有1502人遇难。 这一耸人听闻的悲剧震撼了国际社会，也催生了更完备的船舶安全条例。
- 没有足够的救生艇是泰坦尼克号存活率低的重要原因， 虽然幸存下来的运气有一些因素，但有些人比其他人更有可能生存，比如妇女，儿童和上层阶级。
- 在这个作业中，我们要求你完成对什么样的人可能生存的分析。

作业：泰坦尼克号幸存者预测

- 尽可能从不同的方面来认识数据。有一些规律和性质能被很容易地发现，而一些隐藏的关系却不能被一眼看出，需要从某些特殊的角度才能观察到。
- 数据集介绍及下载：https://www.kaggle.com/competitions/titanic/data
- 数据形式

	A	B	C	D	E	F	G	H	I	J	K	L
1	Passenge	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings,	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen	female	26	0	0	STON/O2	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, M	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson,	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, M	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstror	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S

作业：泰坦尼克号幸存者预测

具体要求：

- 进行数据探索，包括但不限于统计描述、缺失值检查、异常值检测等，分析各个变量与生存率之间的关系。
- 数据集使用Kaggle网站提供的数据集（train.csv、test.csv，不要修改文件名），文件路径设置为当前代码文件路径（如“./test.csv”）。
- 构建神经网络模型预测泰坦尼克号上的乘客是幸存还是遇难。
- 计算模型的准确率（需不低于90%）等评价指标。

作业：泰坦尼克号幸存者预测

上交形式

- 提交完整的代码文件（使用Jupyter Notebook格式）。
- 文件命名为“**学号-姓名-准确率.ipynb**”（如**602024710021-张三-97.8.ipynb**）。
- 不需要提交数据集，只提交Jupyter文件。

注意事项

- 确保代码具有良好的可读性，适当添加注释说明。
- 遵守学术诚信原则，严禁抄袭他人作品。