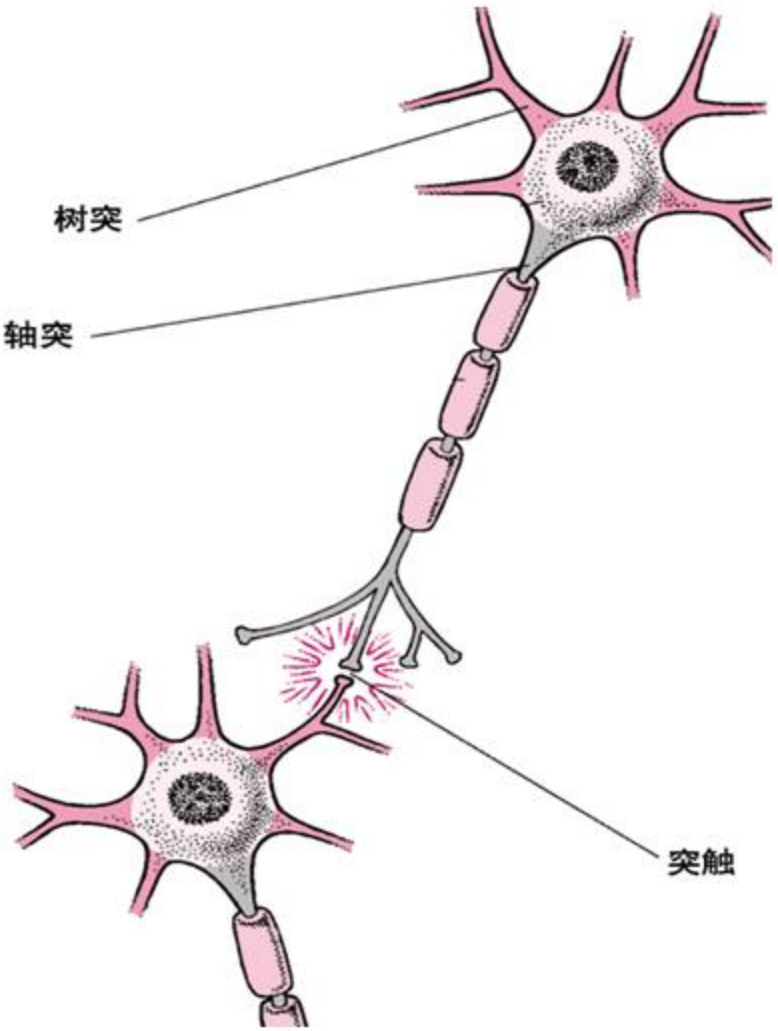
神经网络分类器

神经网络分类器

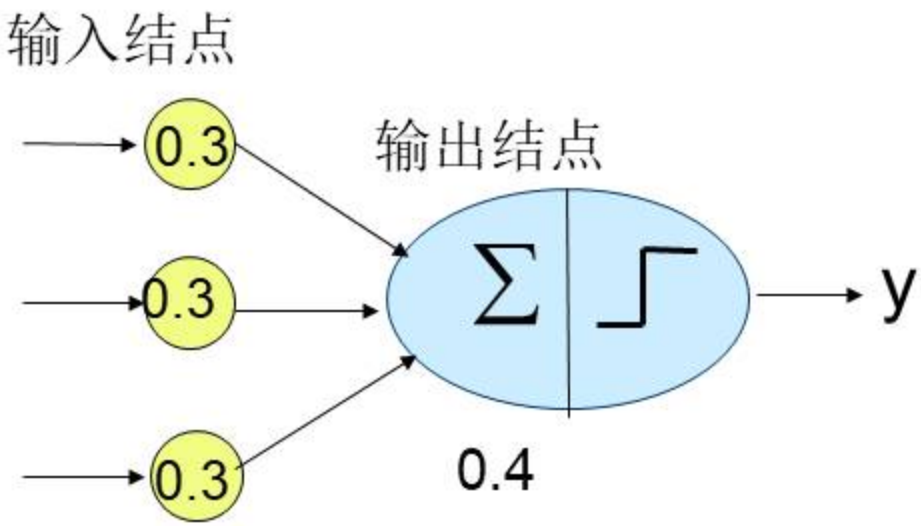
人工神经网络

- 大脑是如何学习的？
  - 神经元传导
  - 神经键改变
- 为什么要模拟人脑？
  - 快速、智能
  - 稳定
- 我们如何模拟人脑的神经元？
- https://blog.csdn.net/zengxiantao1994/article/details/72787849



1.1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



$$\hat{y} = \begin{cases} 1 \ , 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0 \\ -1, 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 < 0 \end{cases}$$

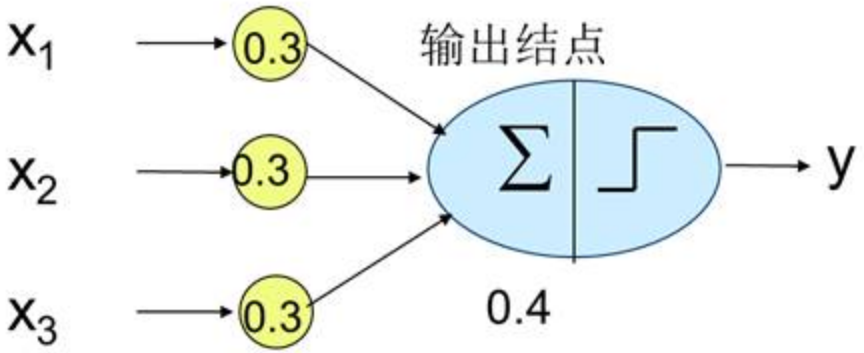
上一个例子的模型：

$$\hat{y}=\begin{cases}1 & ,0.3x_1+0.3x_2+0.3x_3-0.4>0\\ -1 & ,0.3x_1+0.3x_2+0.3x_3-0.4<0\end{cases}$$

化为一般的形式：

$$\hat{y}=sign(w_1x_1+w_2x_2+\cdots+w_dx_d-t)$$

$$\hat{y}=sign(\sum_i w_{ij}x_i-t)$$



多选题 1分

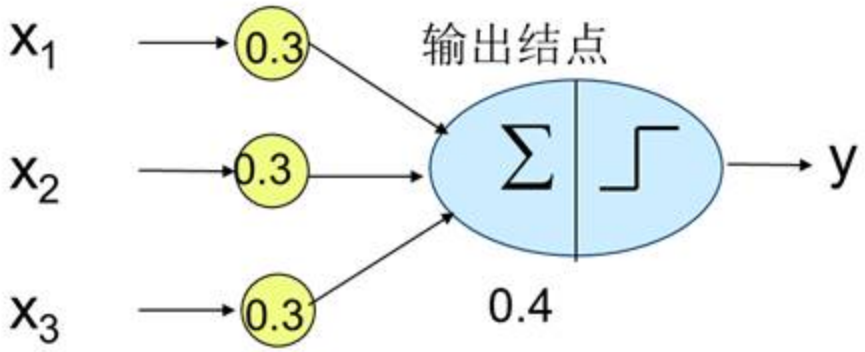
设置

神经元主要包括如下哪几个部分

A 输入权值

B 偏置项bias

C 激活函数



提交

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

}

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

平方误差

$$\hat{y}=sign(w_1x_1+w_2x_2+\cdots+w_dx_d-t)$$

1.1感知机

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

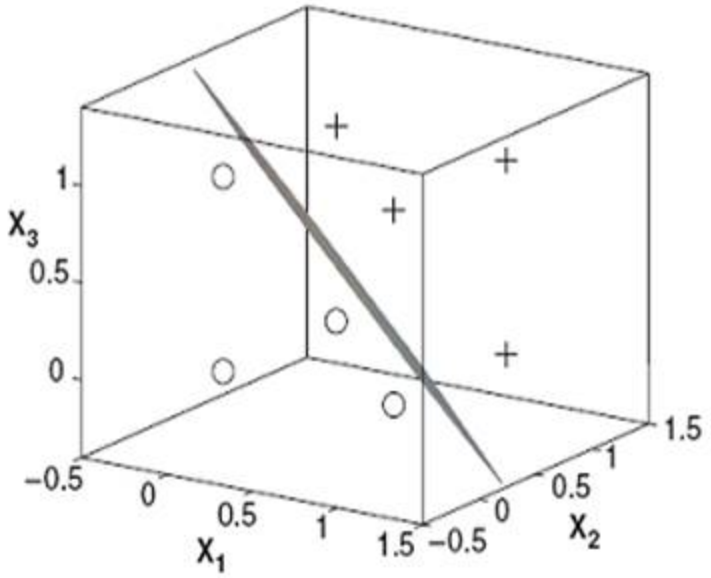
- 如果 $y_i=y_i^{(k)}$， 权值w不变；
- 如果 $y_i=1$（实际类），$y_i^{(k)}=-1$，预测误差为2；

  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；

- 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

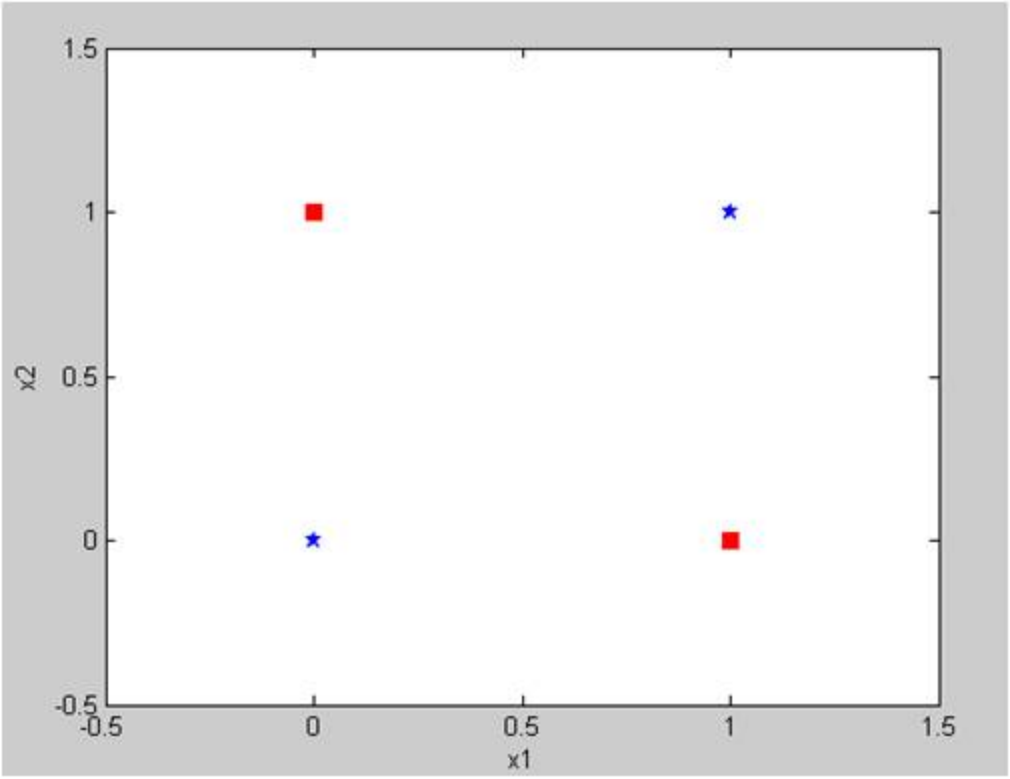


1.1感知机

- Marvin Minsky, *Perceptrons*

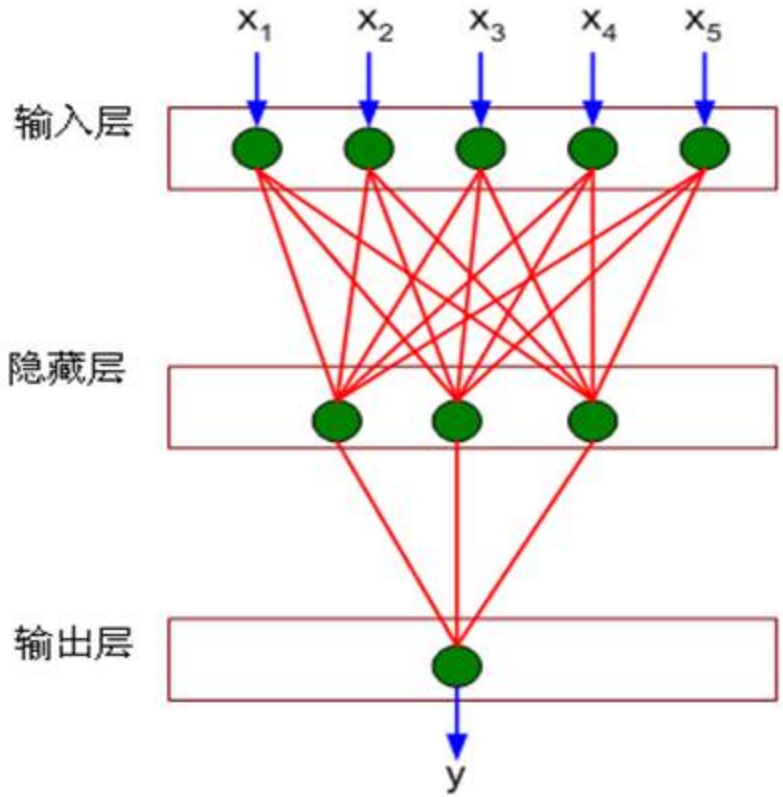
  证明了感知器无法执行“异或问题”

x_1	x_2	Y
0	0	−1
1	0	1
0	1	1
1	1	−1



1.2多层人工神经网络

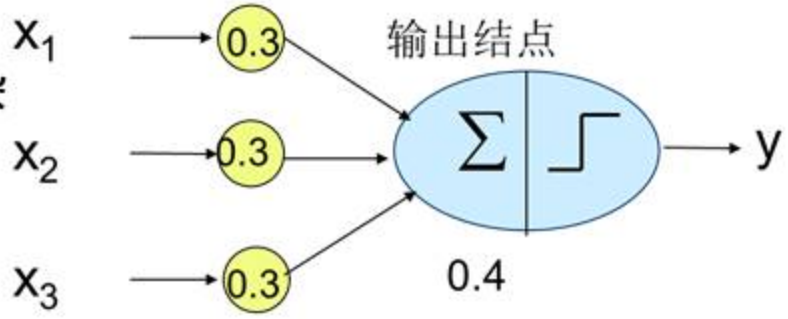
- 人工神经网络比感知机模型复杂
  - 输入层和输出层之间包含隐藏层



1.2多层人工神经网络

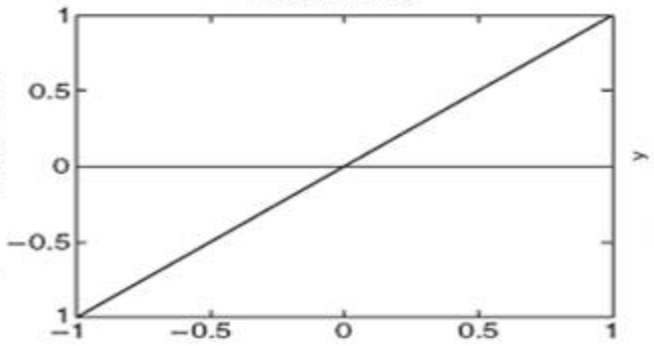
- 人工神经网络比感知机模型复杂
  - 激活函数可以是多种函数

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

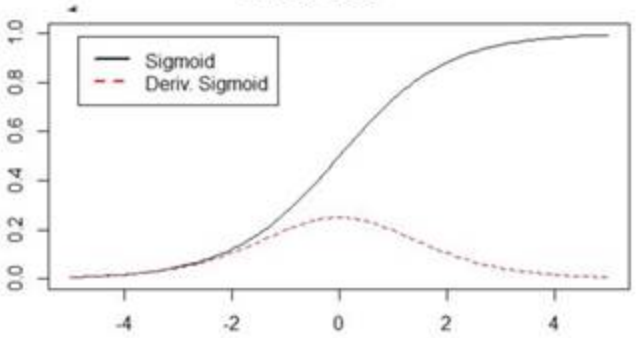


线性函数

线性可分



S型函数

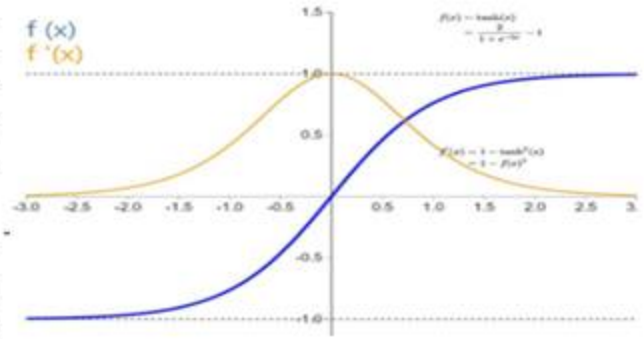


梯度消失

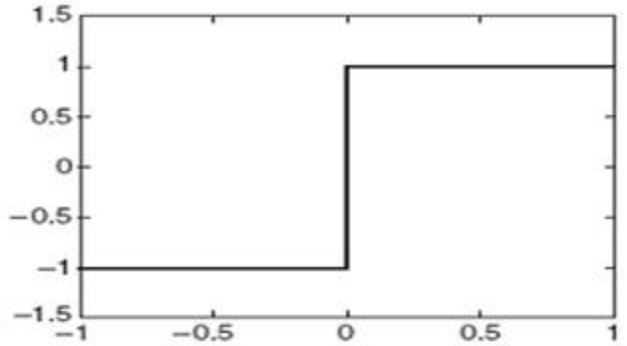
双曲正切函数

梯度消失

收敛比s更快

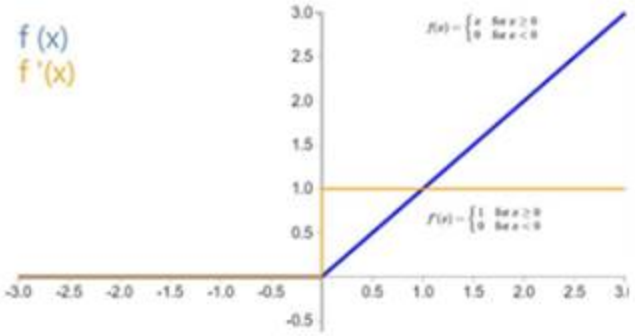


符号函数



1.2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



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	$f(x)=\log\big(1+\exp(x)\big)$	$f'(x)=\frac{1}{1+\exp(-x)}$

解决了部分梯度消失问题

收敛速度比双面正切更快

导致神经元死亡

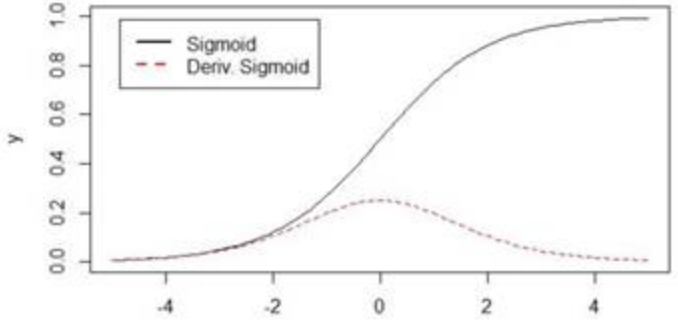
设置较小的学习率

单选题 1分

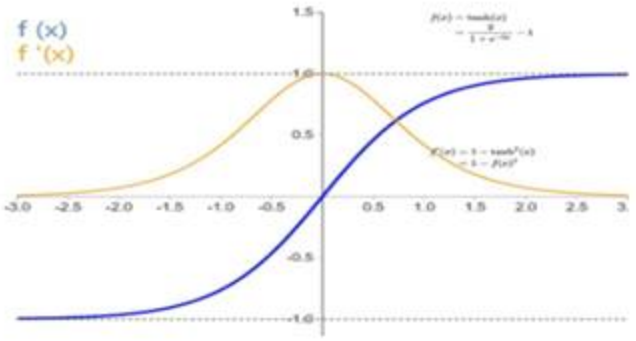
设置

以下哪种激活函数解决了部分梯度消失问题

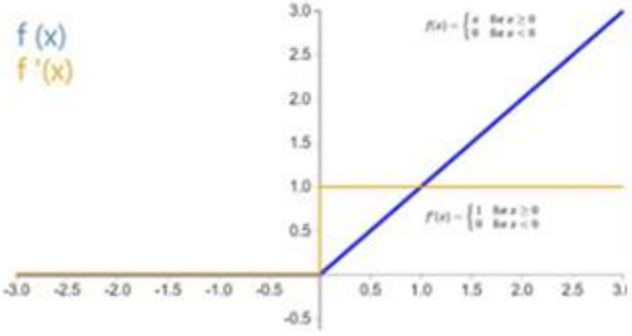
A S型函数sigmoid



B 双面正切函数Tanh



C 线性整流函数ReLU



提交

单选题 1分

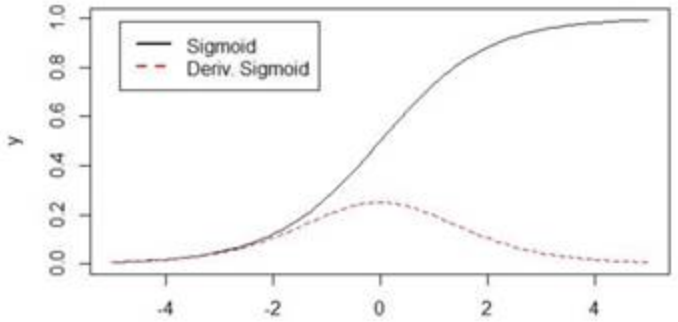
设置

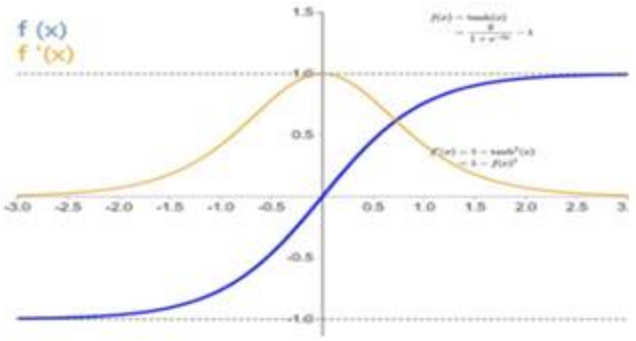
以下哪种激活函数收敛速度最快

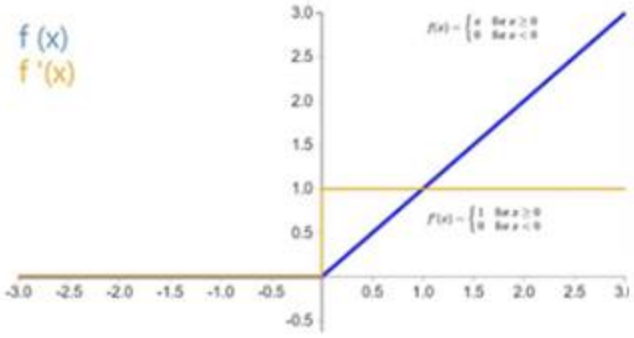
A S型函数sigmoid

B 双面正切函数Tanh

C 线性整流函数ReLU







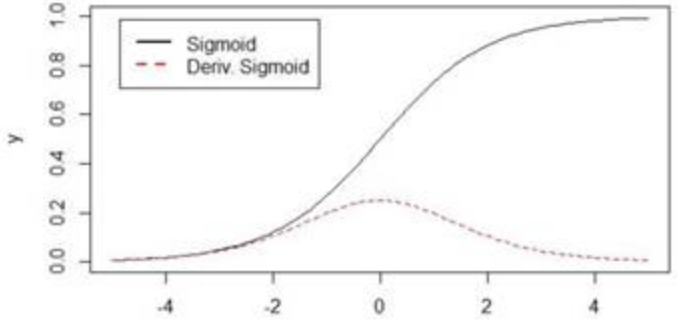
提交

单选题 1分

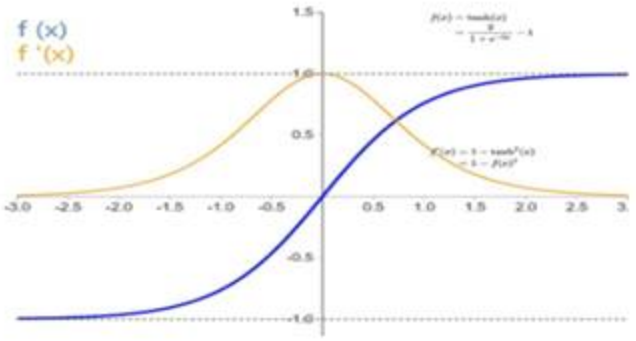
设置

以下哪种激活函数容易导致神经元死亡

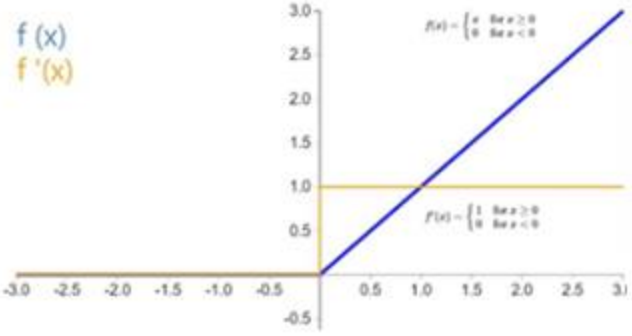
A S型函数sigmoid



B 双面正切函数Tanh



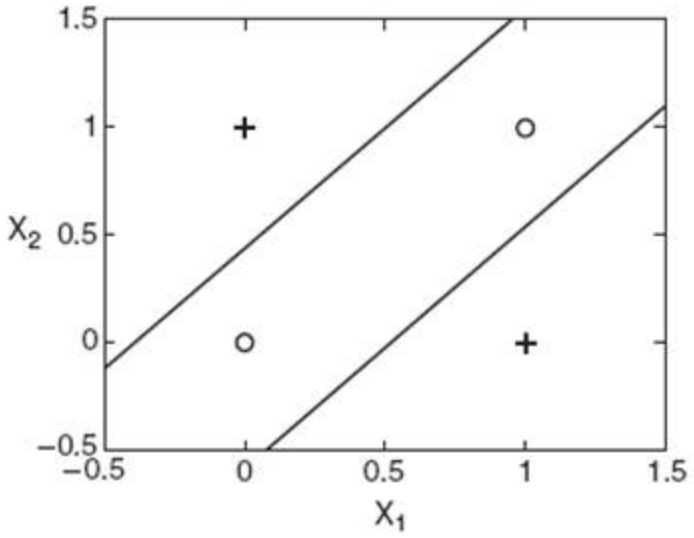
C 线性整流函数ReLU



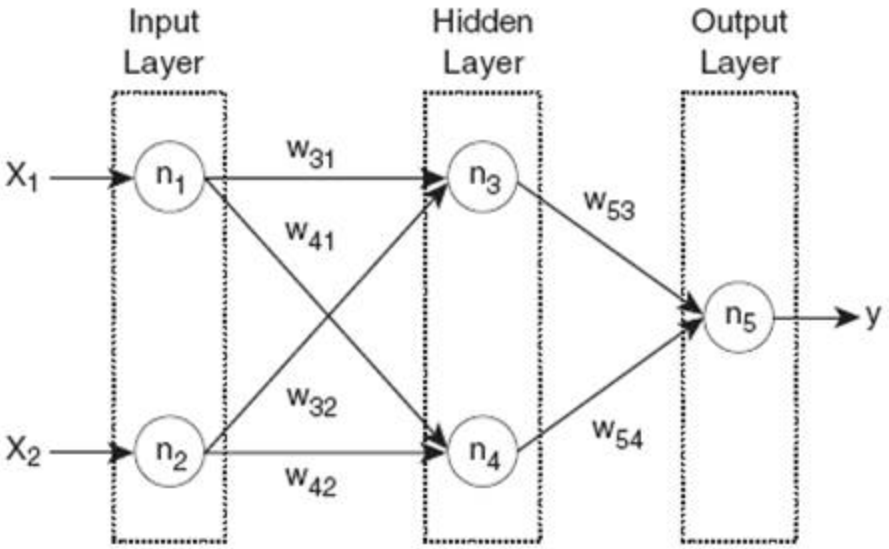
提交

1.2多层人工神经网络

- 例：用两层前馈神经网络解决异或问题



(a) 决策边界



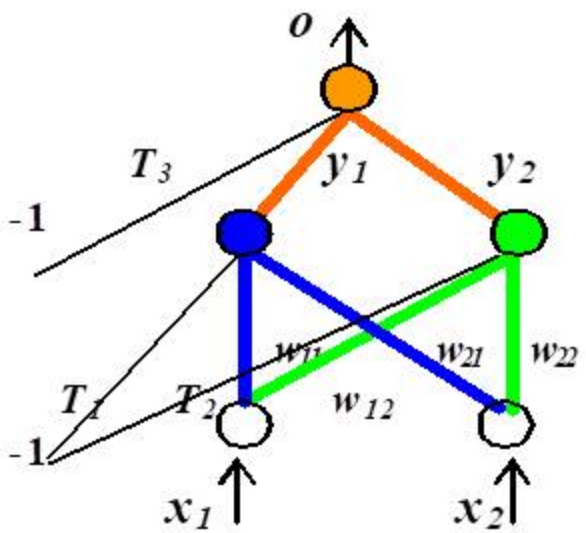
(b) 神经网络拓扑结构

用两计算层感知器解决“异或”问题。

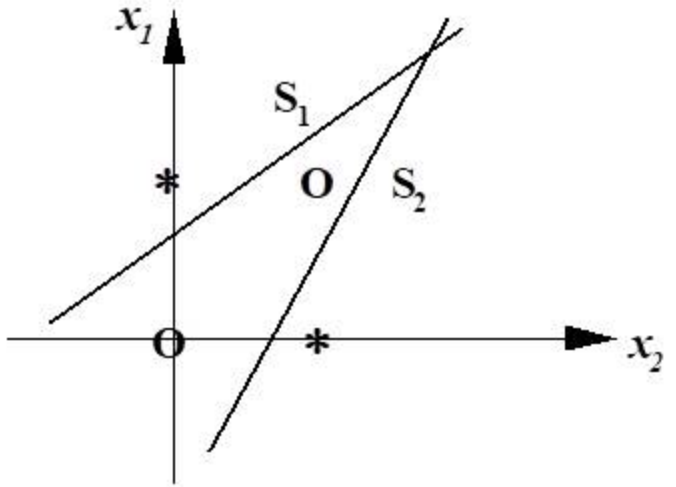
“异或”的真值表

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

双层感知器

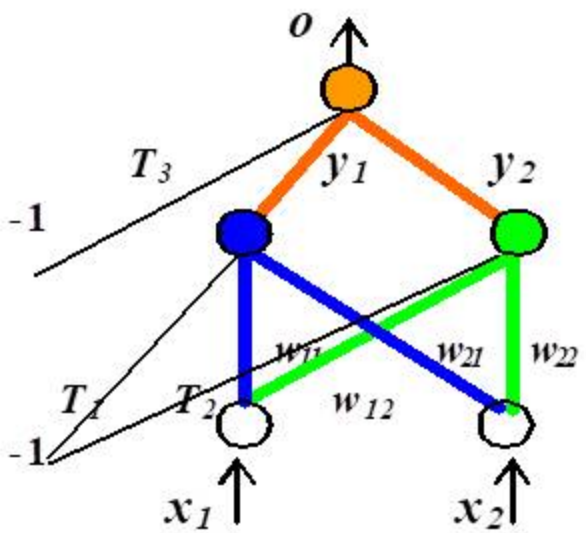


“异或”问题分类



用两计算层感知器解决“异或”问题

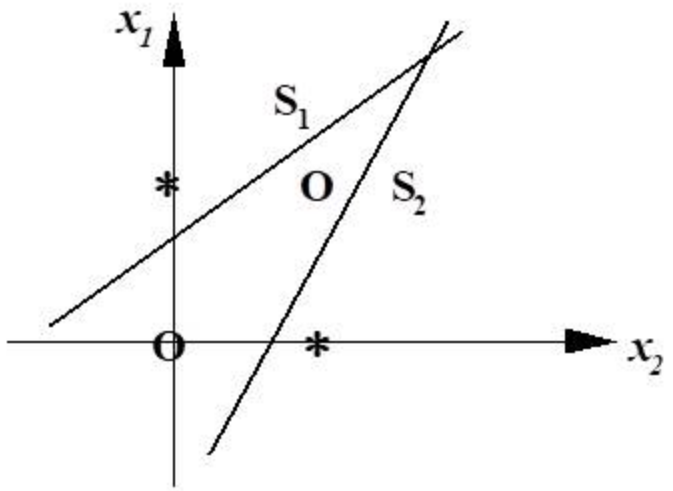
双层感知器



“异或”的真值表

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

“异或”问题分类

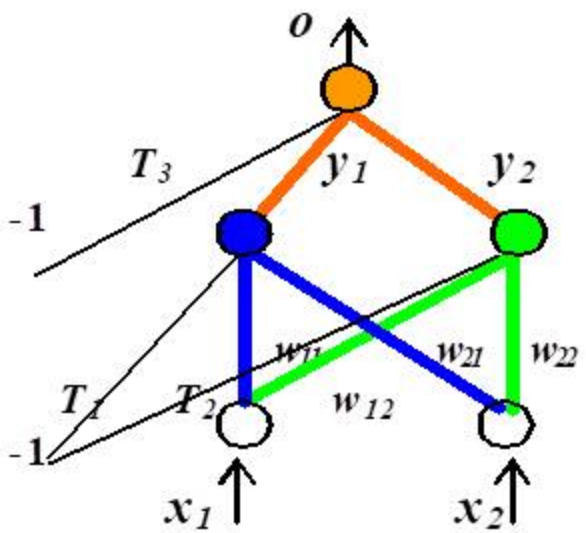


用两计算层感知器解决“异或”问题。

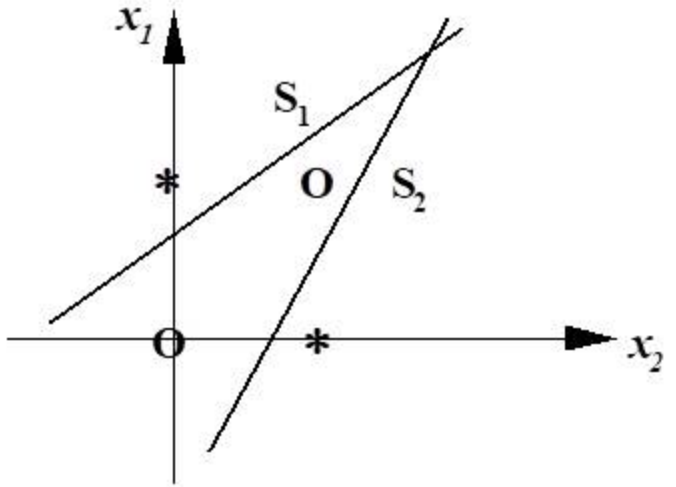
“异或”的真值表

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

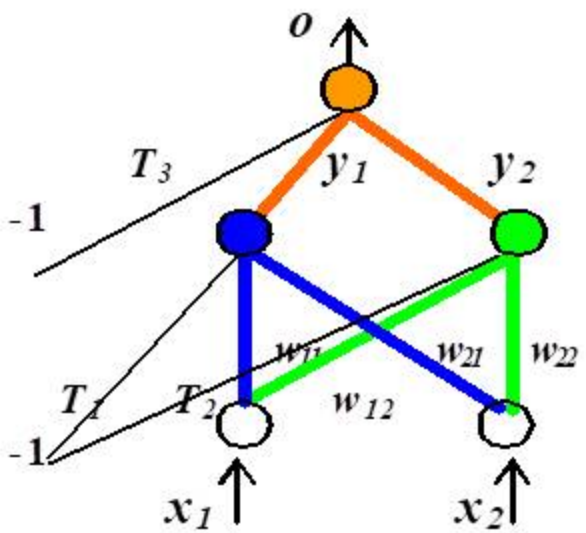


“异或”问题分类



例四 用两计算层感知器解决“异或”问题。

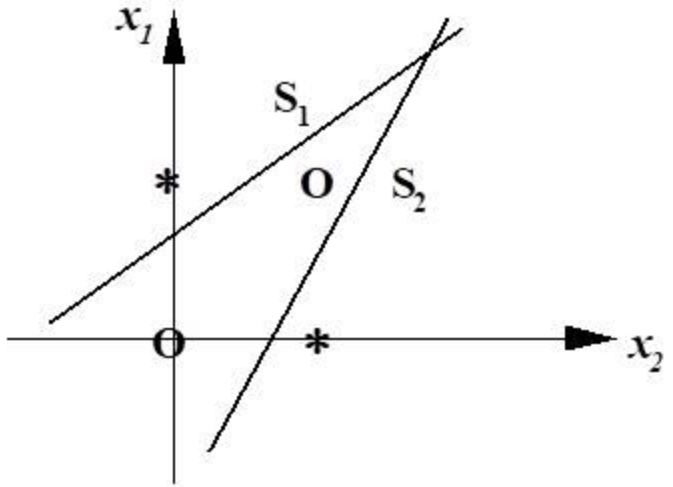
双层感知器



“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

“异或”问题分类

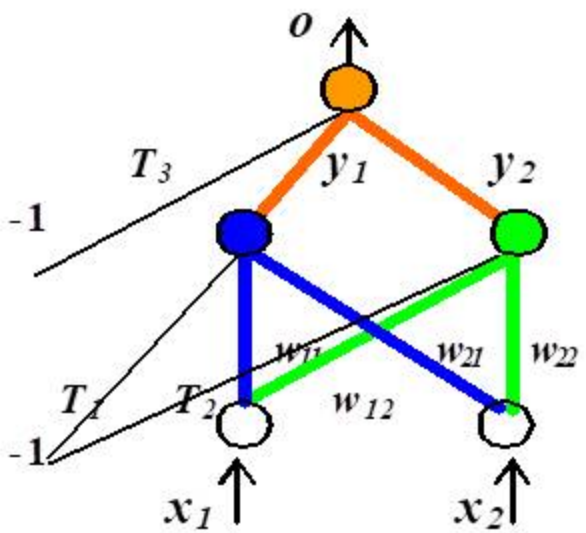


例四 用两计算层感知器解决“异或”问题。

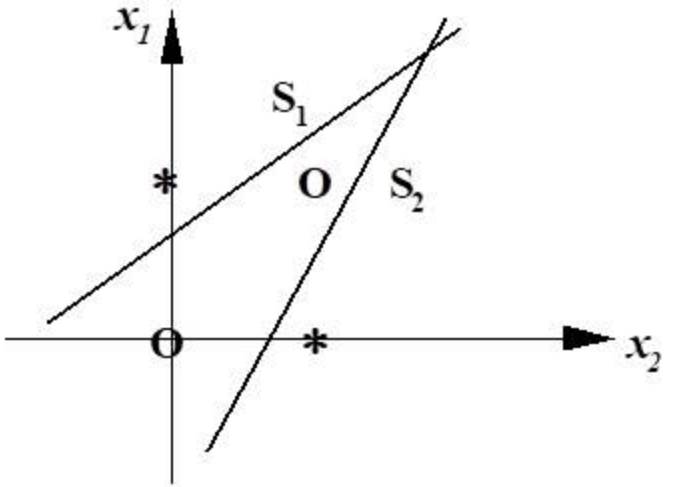
“异或”的真值表

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

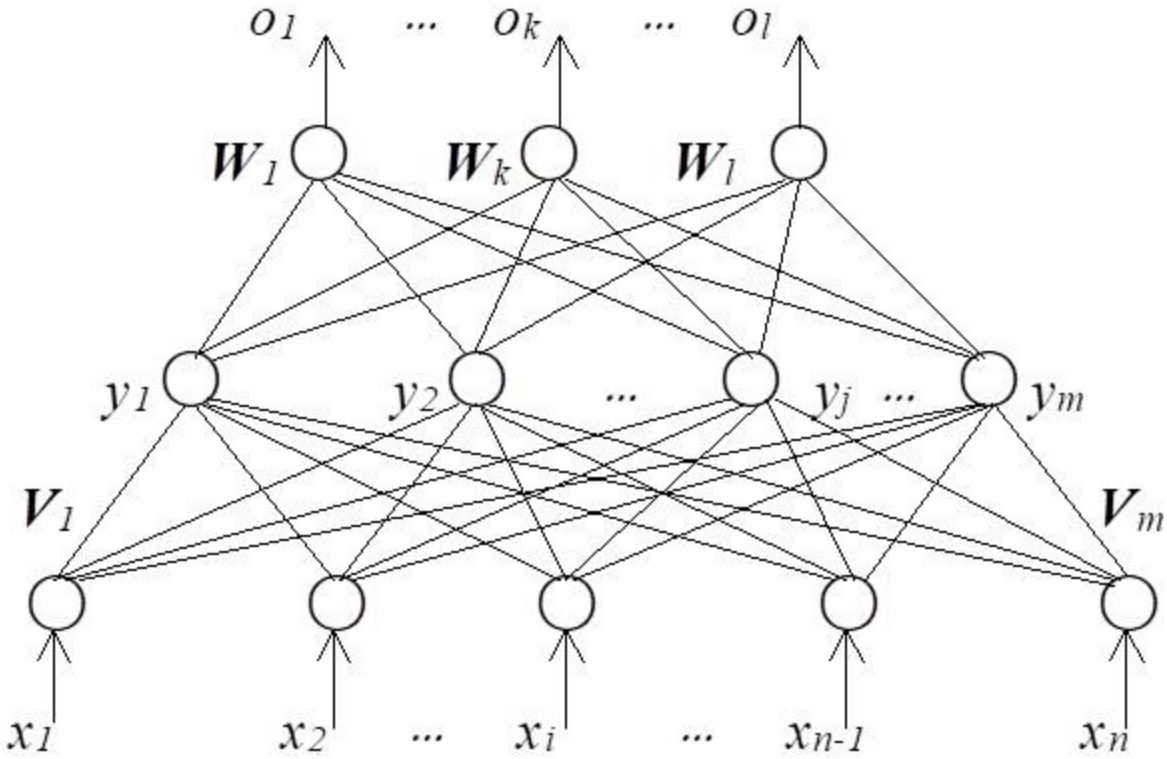
双层感知器



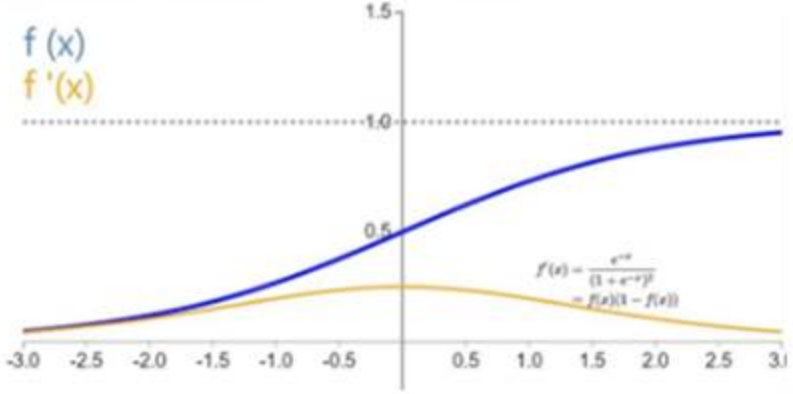
“异或”问题分类



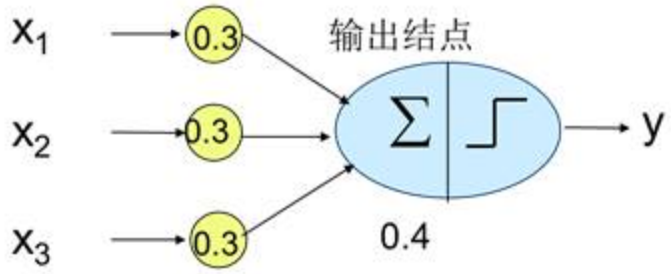
基于BP算法的多层前馈网络模型



1.4误差反向传播（BP）网络

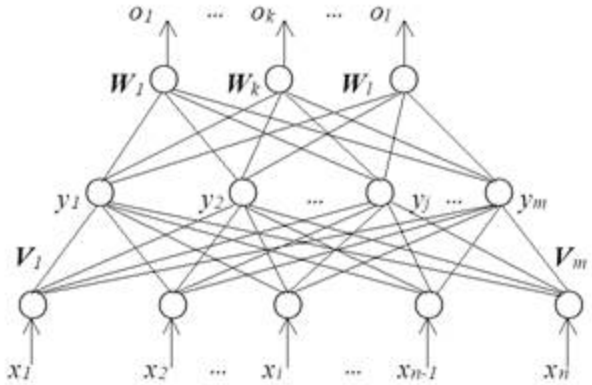


- 激活函数
  - 必须处处可导
    - 一般都使用S型函数
- 使用S型激活函数时BP网络输入与输出关系
  - 输入 $net = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$
  - 输出 $y = \mathrm{f}(net) = \dfrac{1}{1+\mathrm{e}^{-net}}$



1.4误差反向传播（BP）网络

- 学习的过程：
  - 神经网络在外界输入样本的刺激下不断改变网络的连接权值，以使网络的输出不断地接近期望的输出。
- 学习的本质：
  - 对各连接权值的动态调整
- 学习规则：
  - 权值调整规则，即在学习过程中网络中各神经元的连接权变化所依据的一定的调整规则。

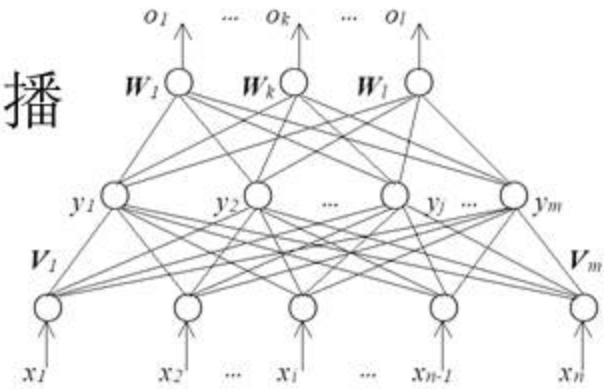


1.4误差反向传播（BP）网络

- 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



1.4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

1.5后向传播算法

1. 初始化权重

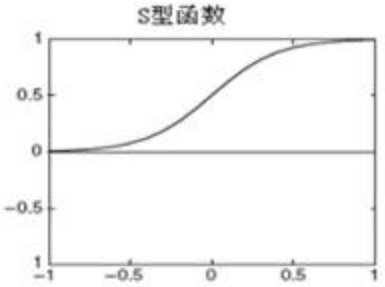
循环以下两步，直到满足条件

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

1. 向前传播输入

在每个节点加权求和，再代入激活函数

$$\hat{y} = sign(\sum_i w_{ij} x_i - t_j)$$



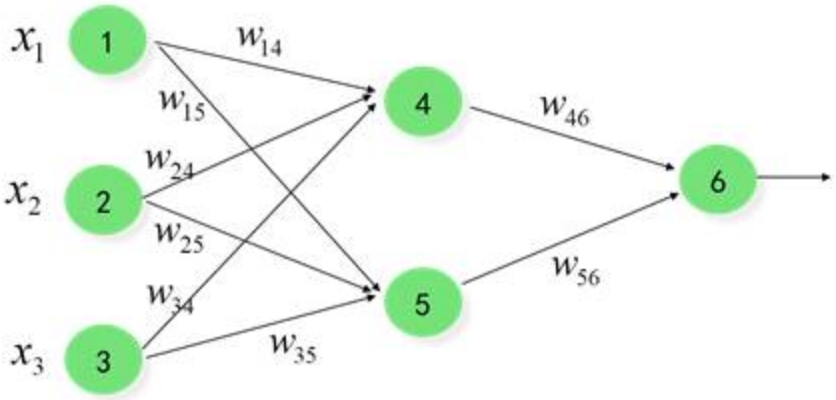
2. 向后传播误差

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

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

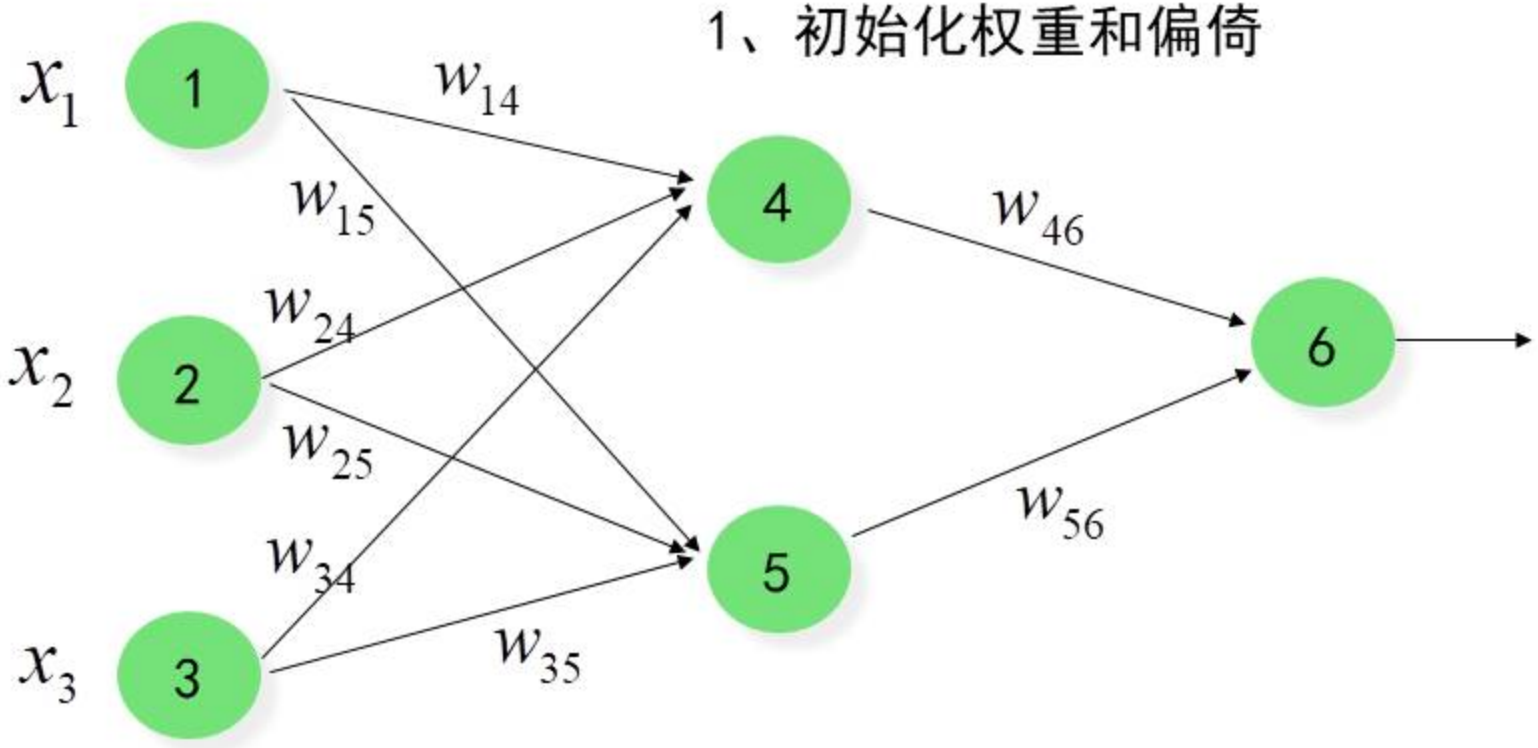
$$w_{ij} = w_{ij} + \lambda Err_j y_i$$

$$t_j = t_j + \lambda Err_j$$

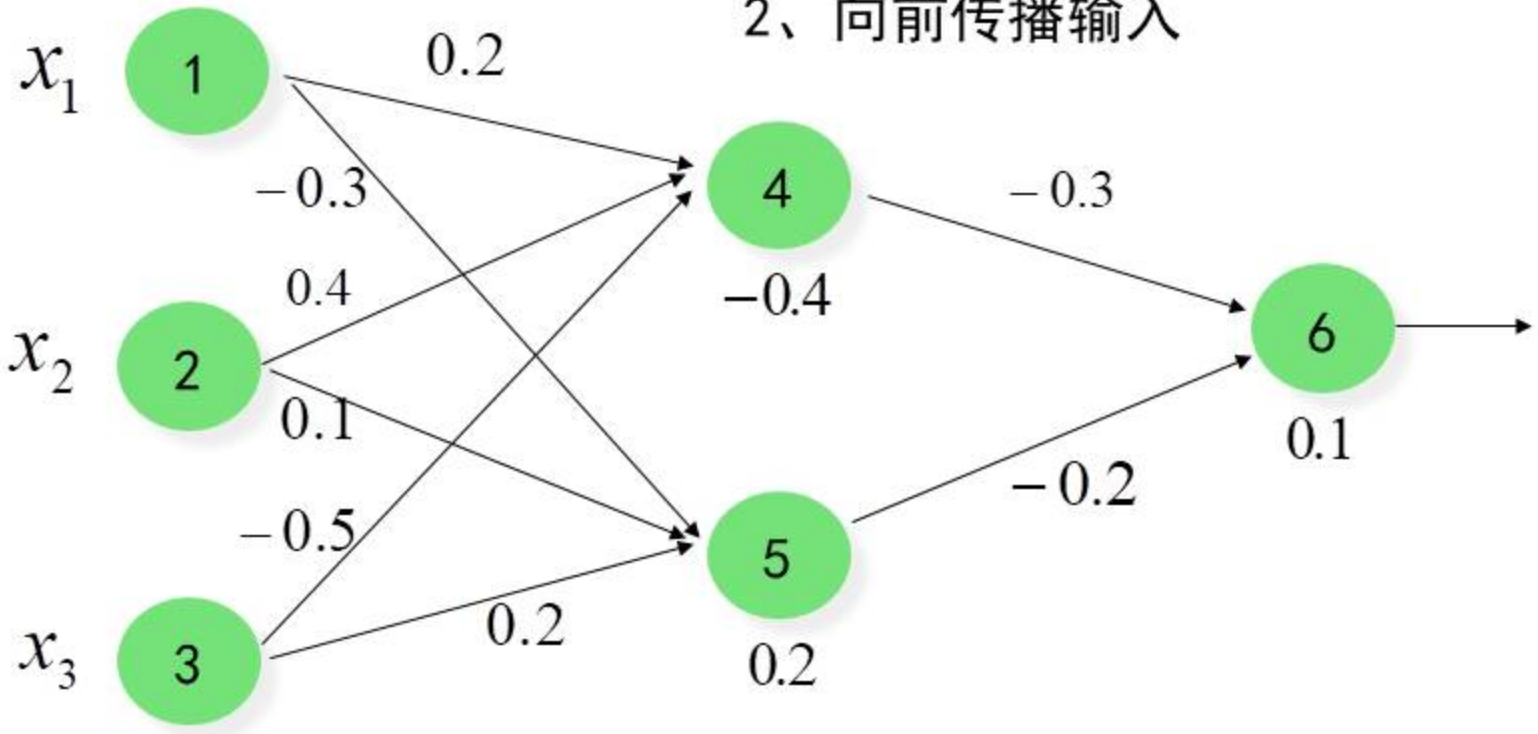


1.5后向传播算法

1、初始化权重和偏倚

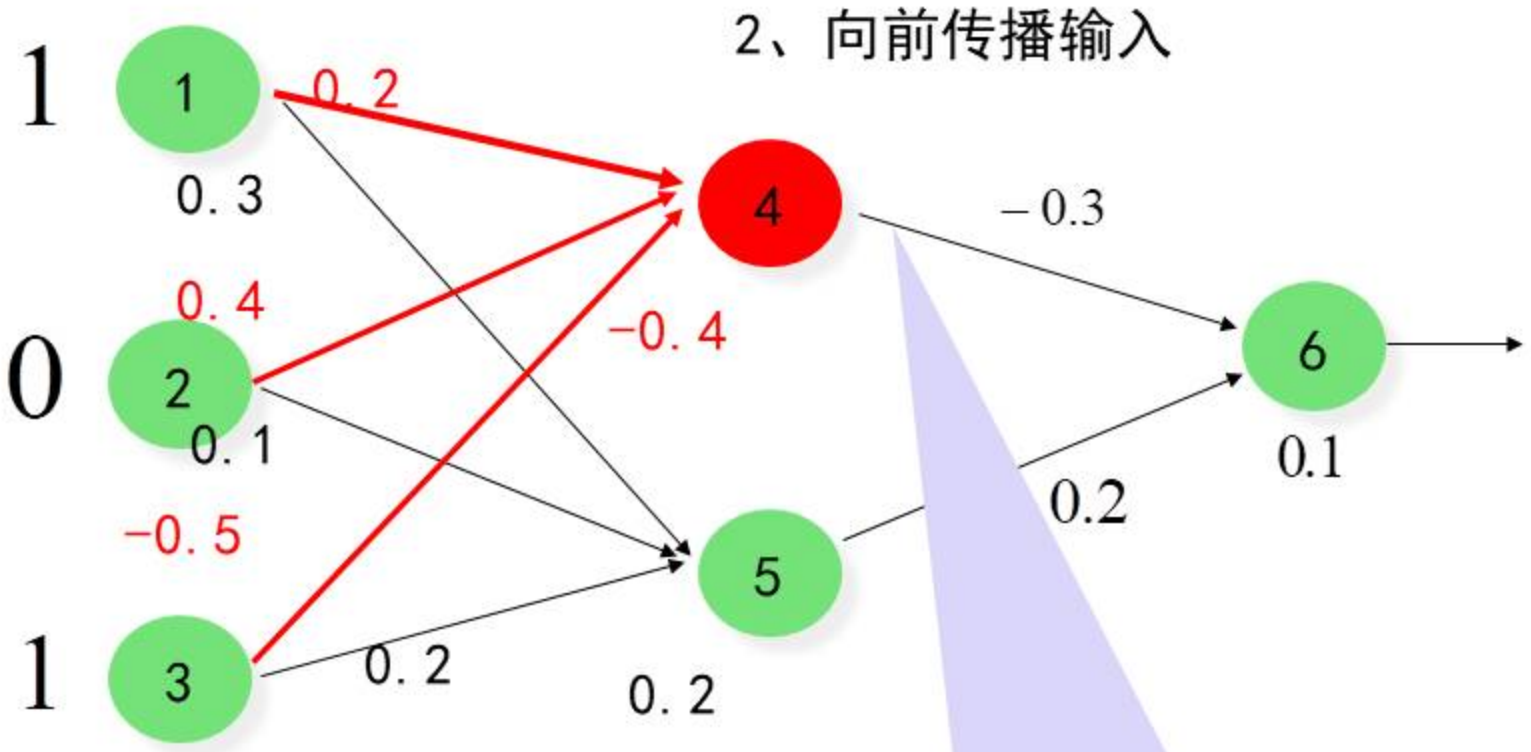


2、向前传播输入



填空题 1分

设置



0.2 + 0 − 0.5 − 0.4 = [填空1]

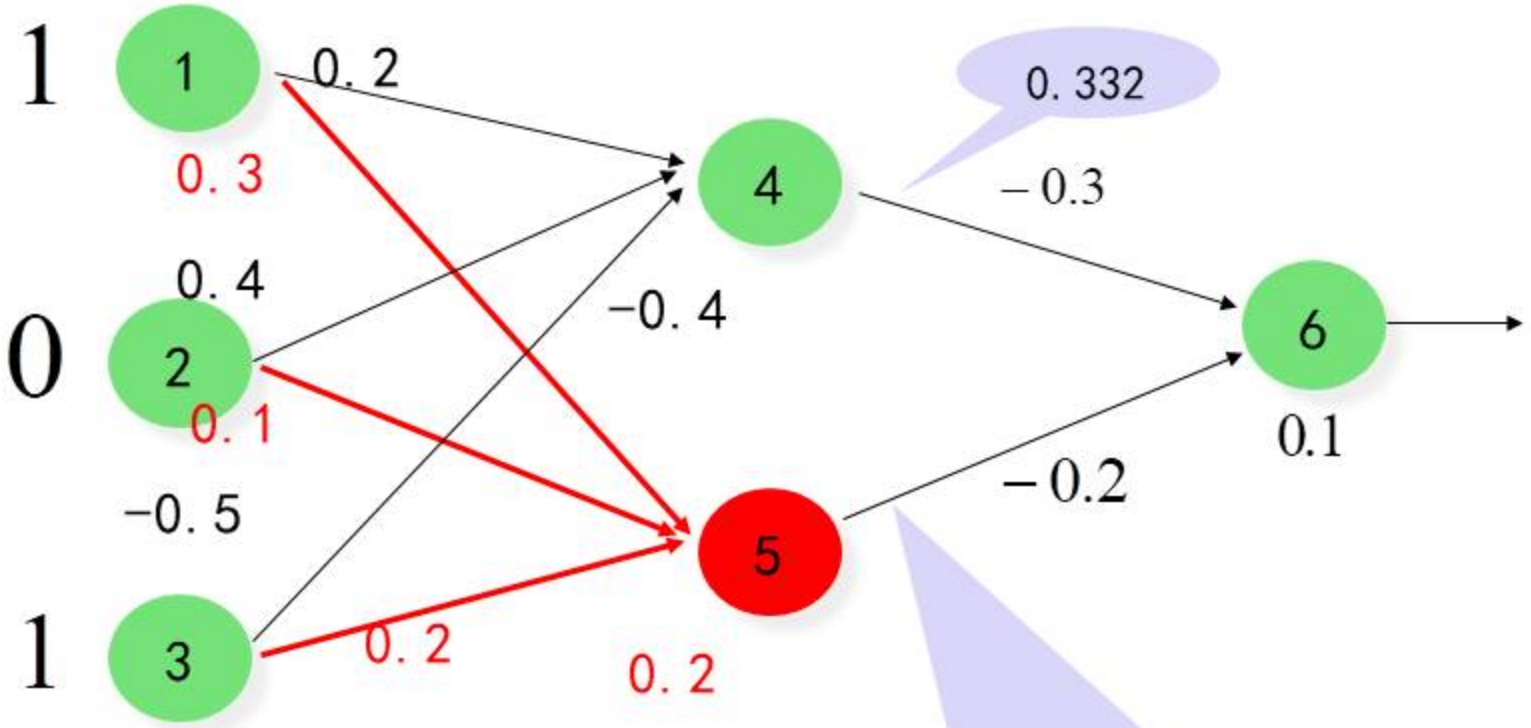
输出：1/(1+e^0.7)=0.332

正常使用填空题需3.0以上版本雨课堂

作答

填空题 1分

设置



-0.3 + 0 + 0.2 + 0.2 =[填空1]

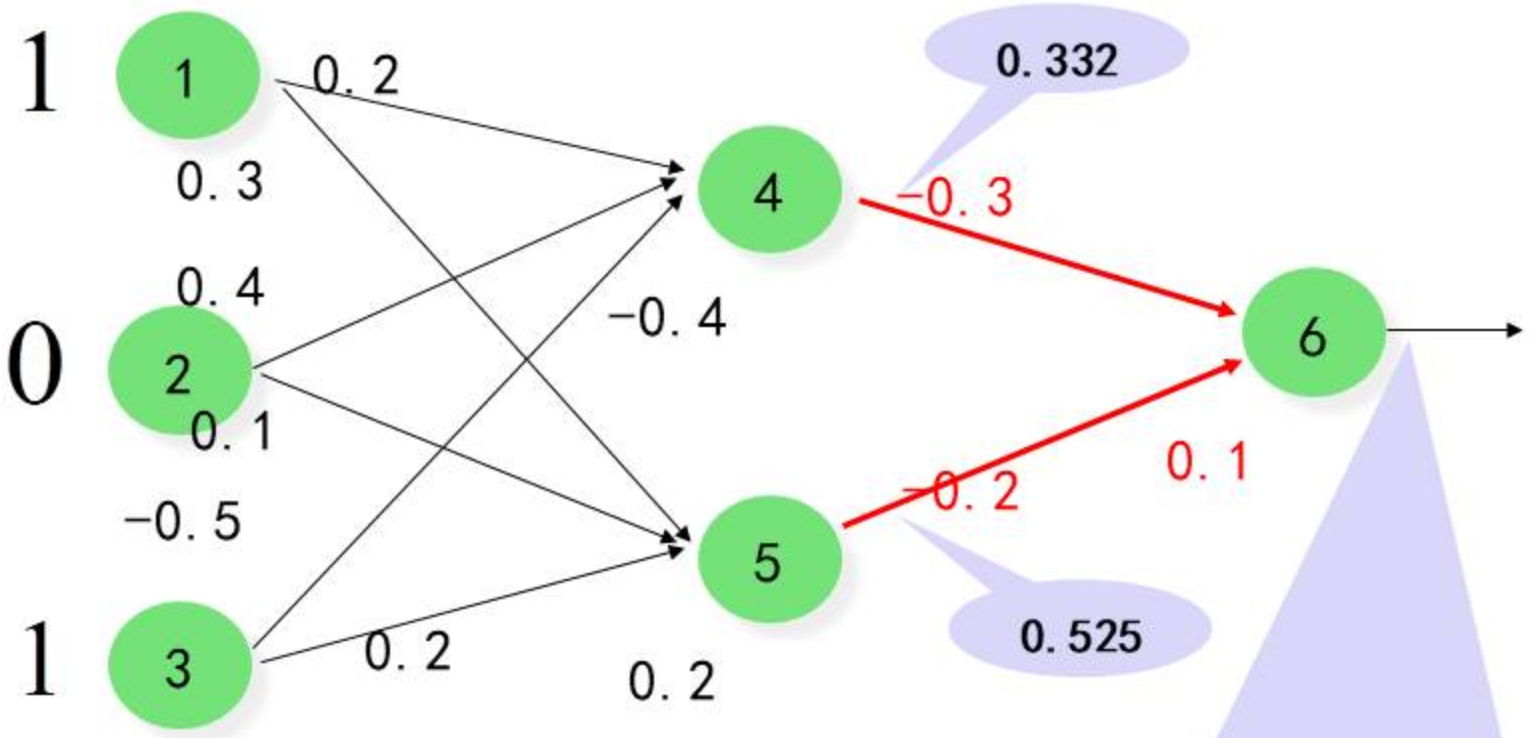
输出：1/(1+e^-0.1)=0.525

正常使用填空题需3.0以上版本雨课堂

作答

填空题 1分

设置



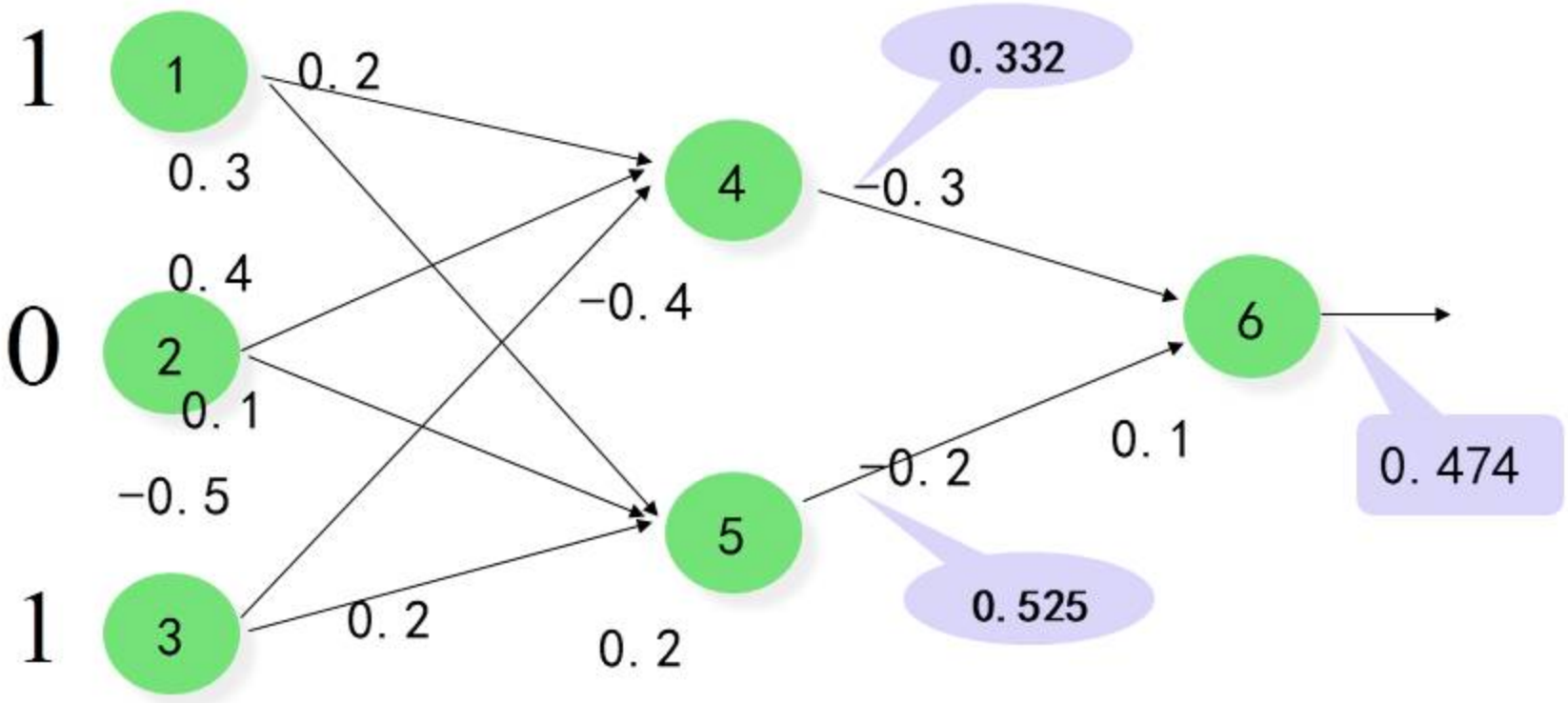
(−0.3)(0.332)−(0.2)(0.525)+0.1= [填空1]

输出：1/(1+e^−0.105)=0.474

正常使用填空题需3.0以上版本雨课堂

作答

1.5后向传播算法

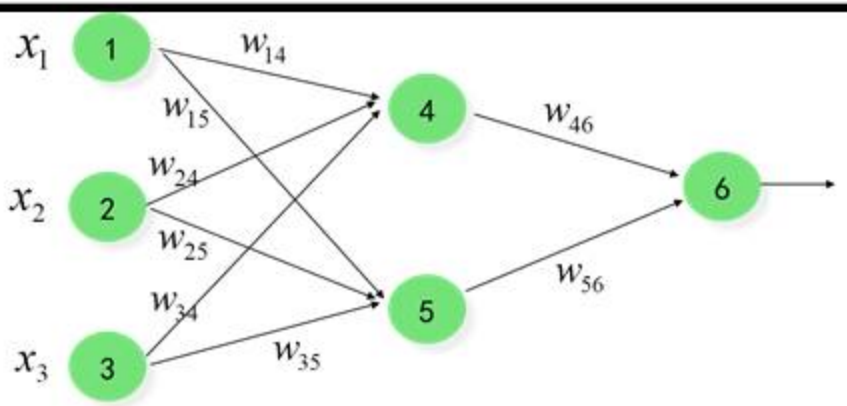


初始随机设置参数

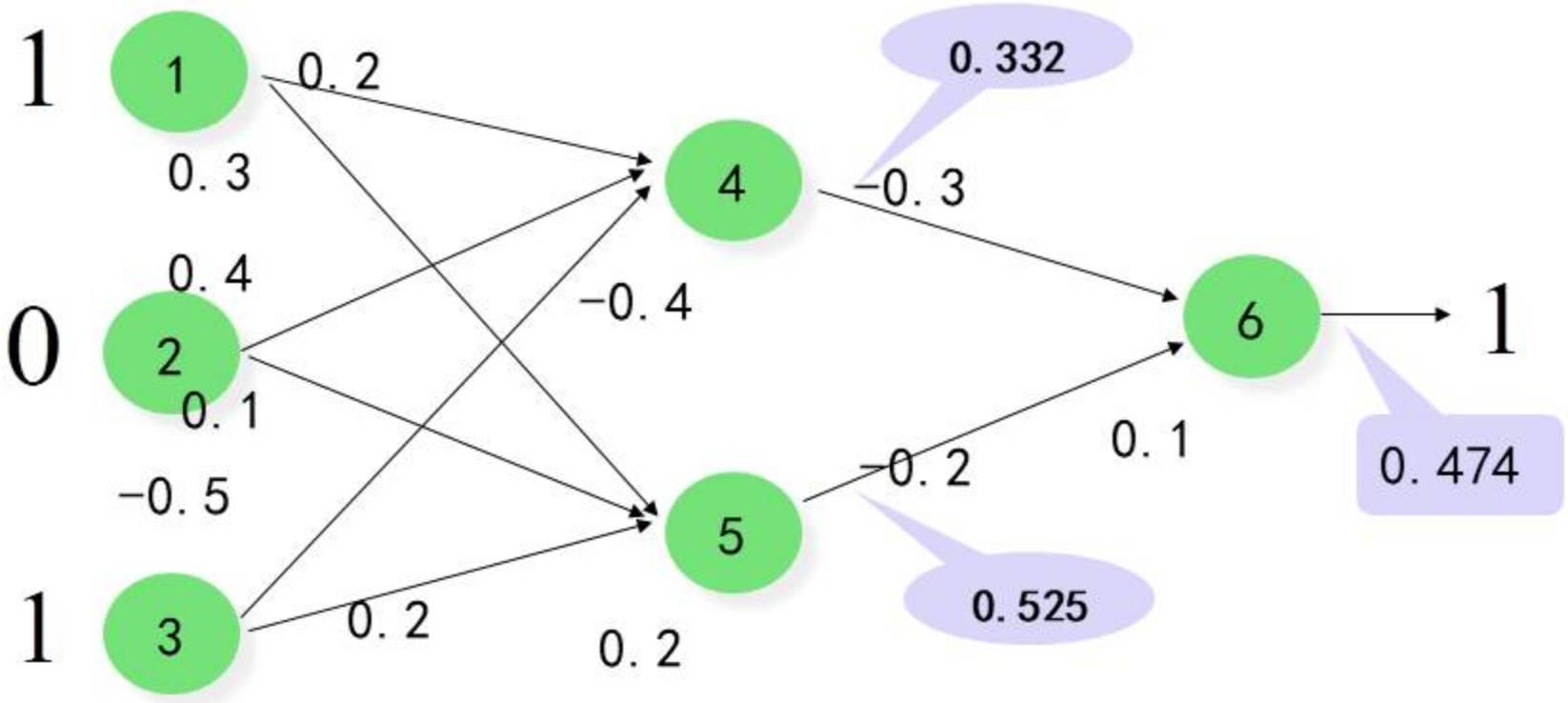
x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

静输入和输出的表格

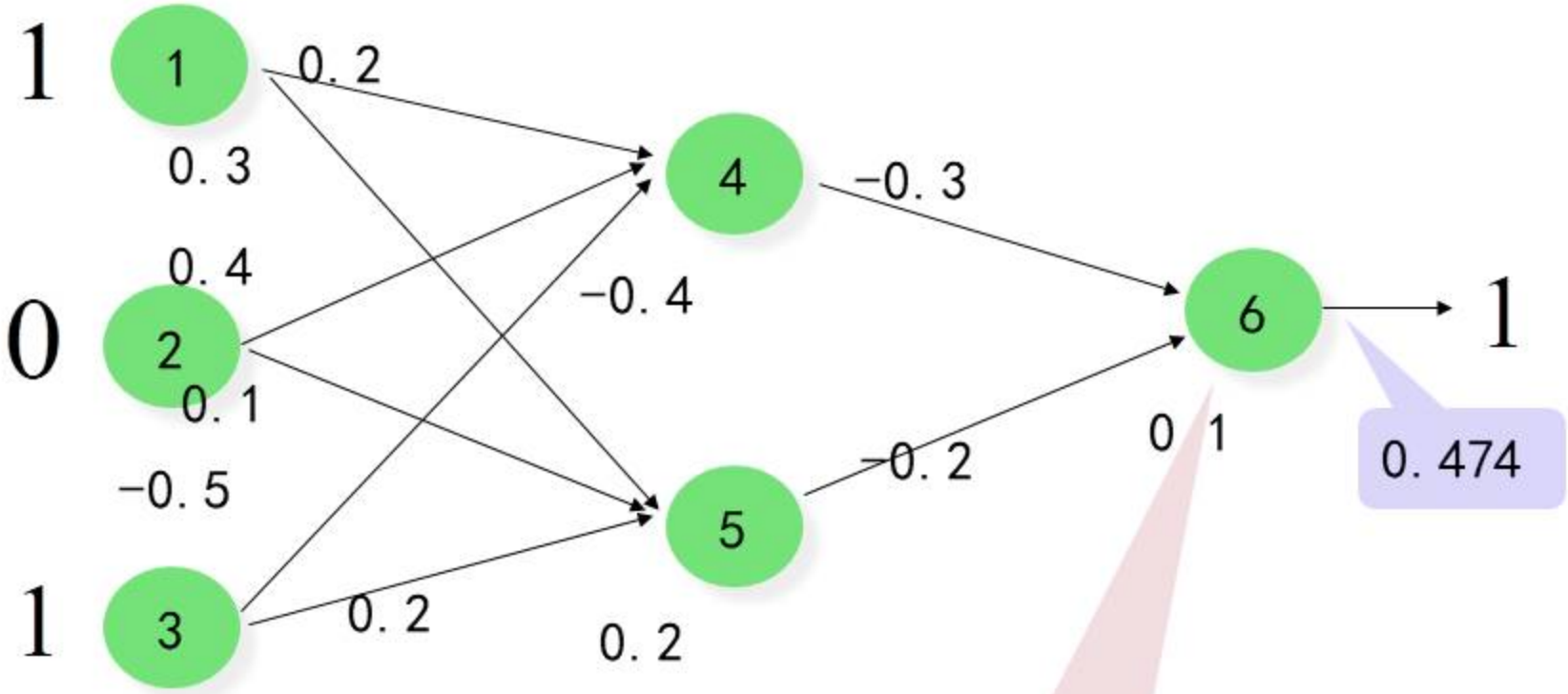
Unit j	*Net input,* I_j	*Output,* O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

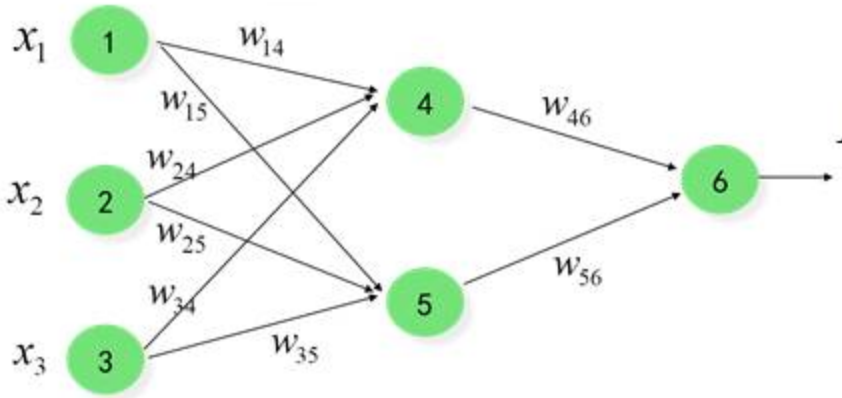


1.5后向传播算法



1.5后向传播算法

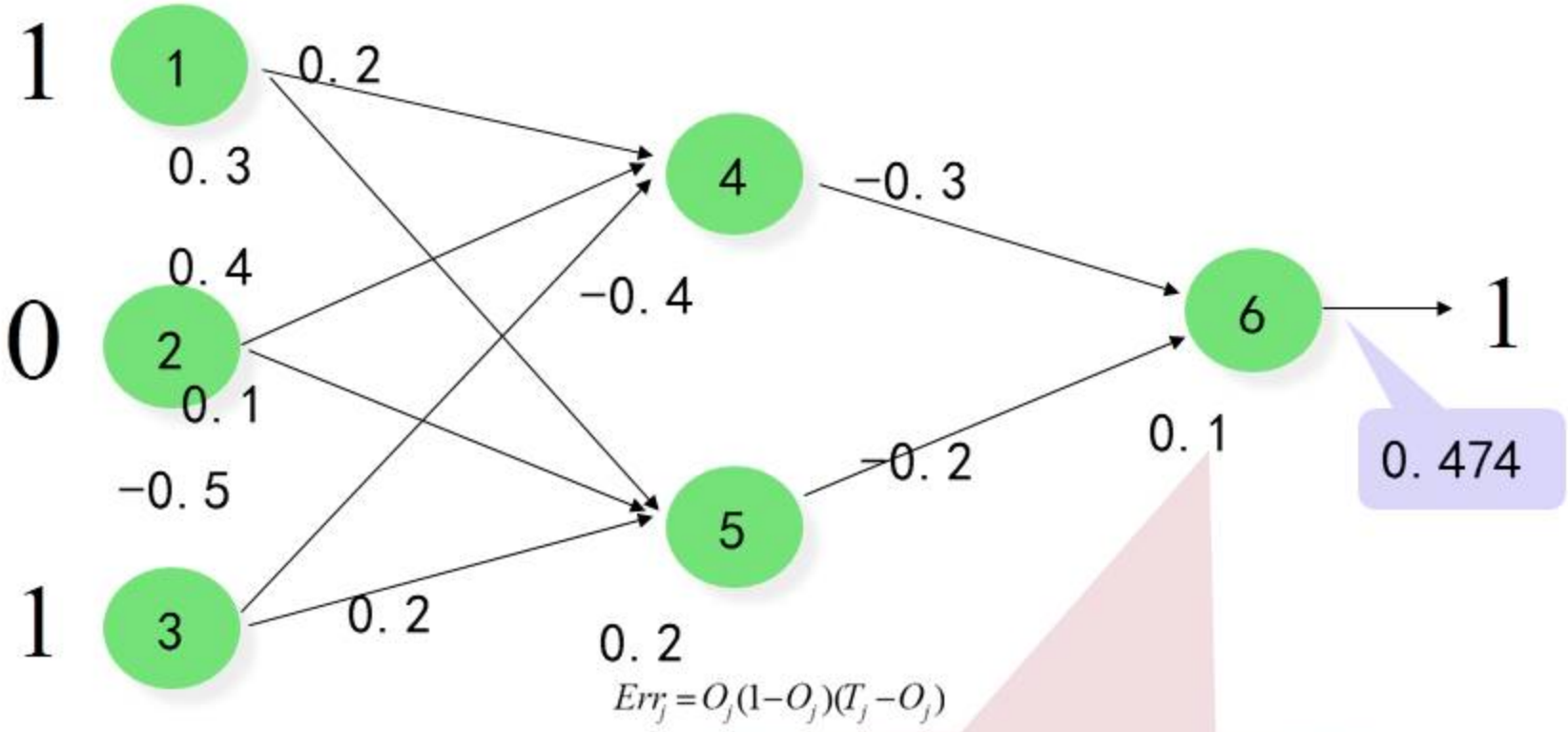


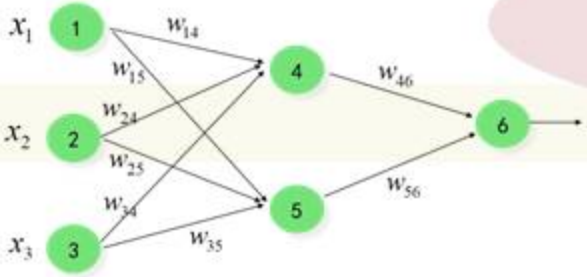


$$Err_j = O_j(1-O_j)(T_j-O_j)$$

填空题 1分

设置



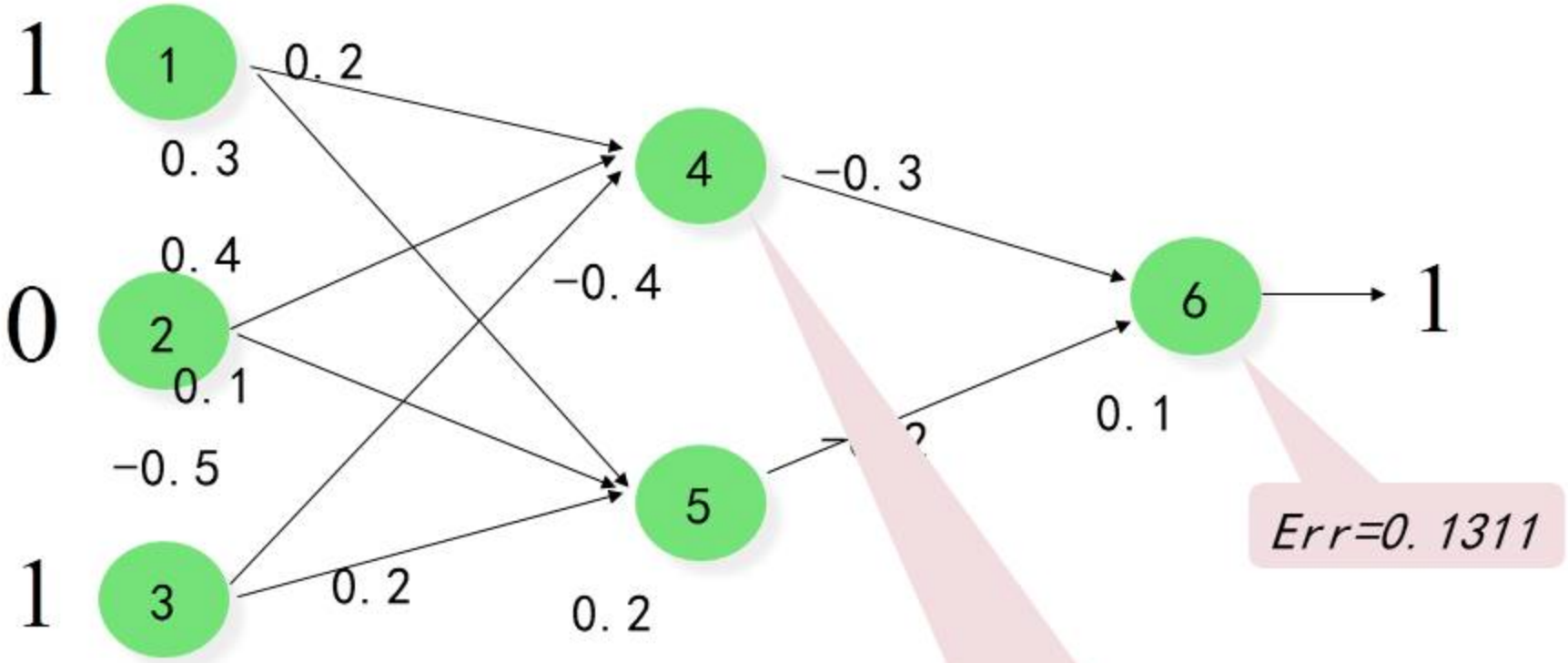


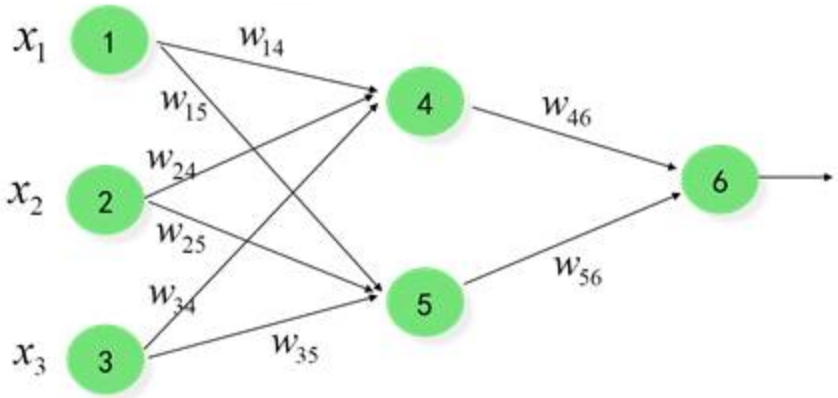
Err =

0.474(1−0.474) (1−0.474)= [填空1]

作答

1.5后向传播算法

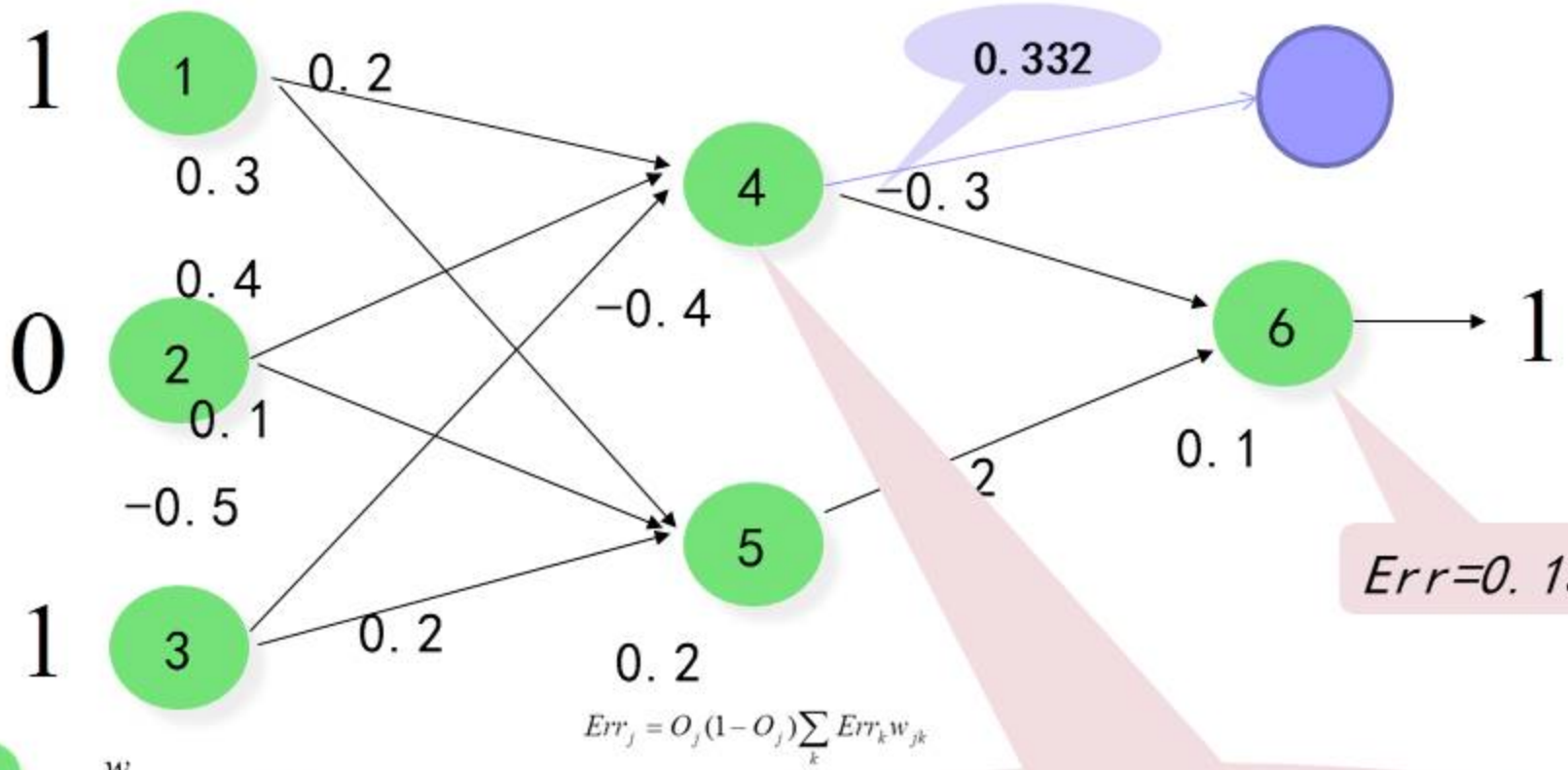




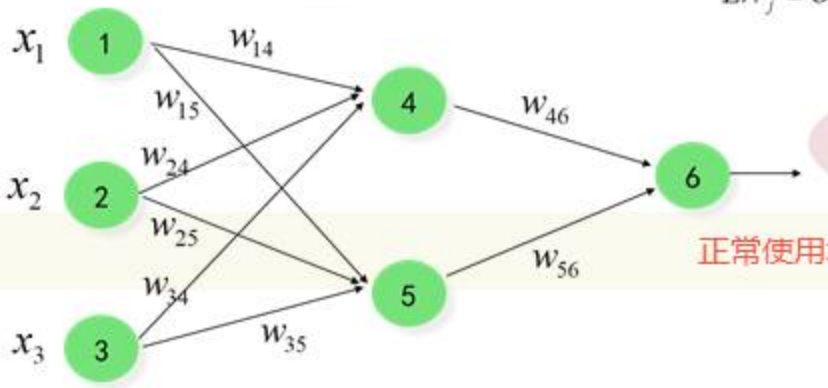
$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

填空题 1分

设置



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



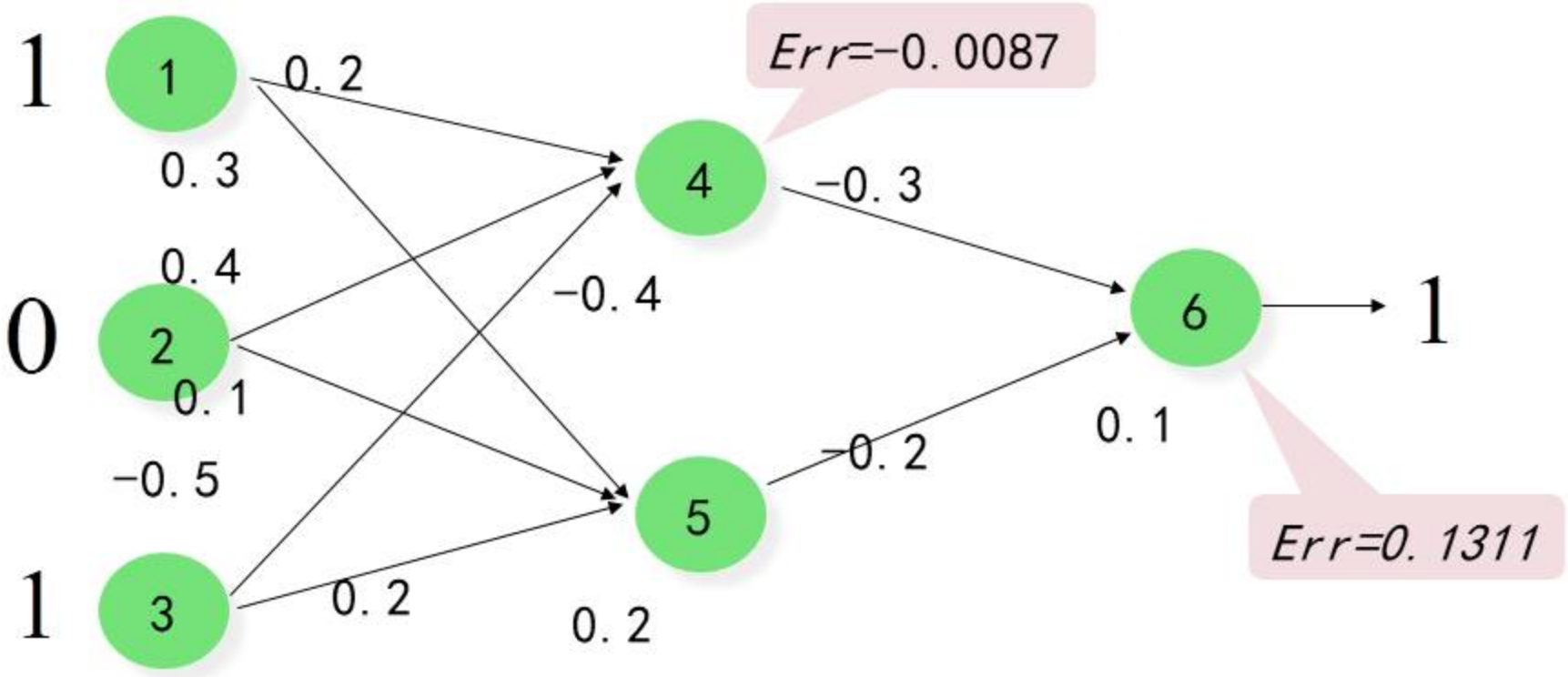
Err=

0.332 (1−0.332) (0.1311) (−0.3)= [填空1]

正常使用填空题需3.0以上版本雨课堂

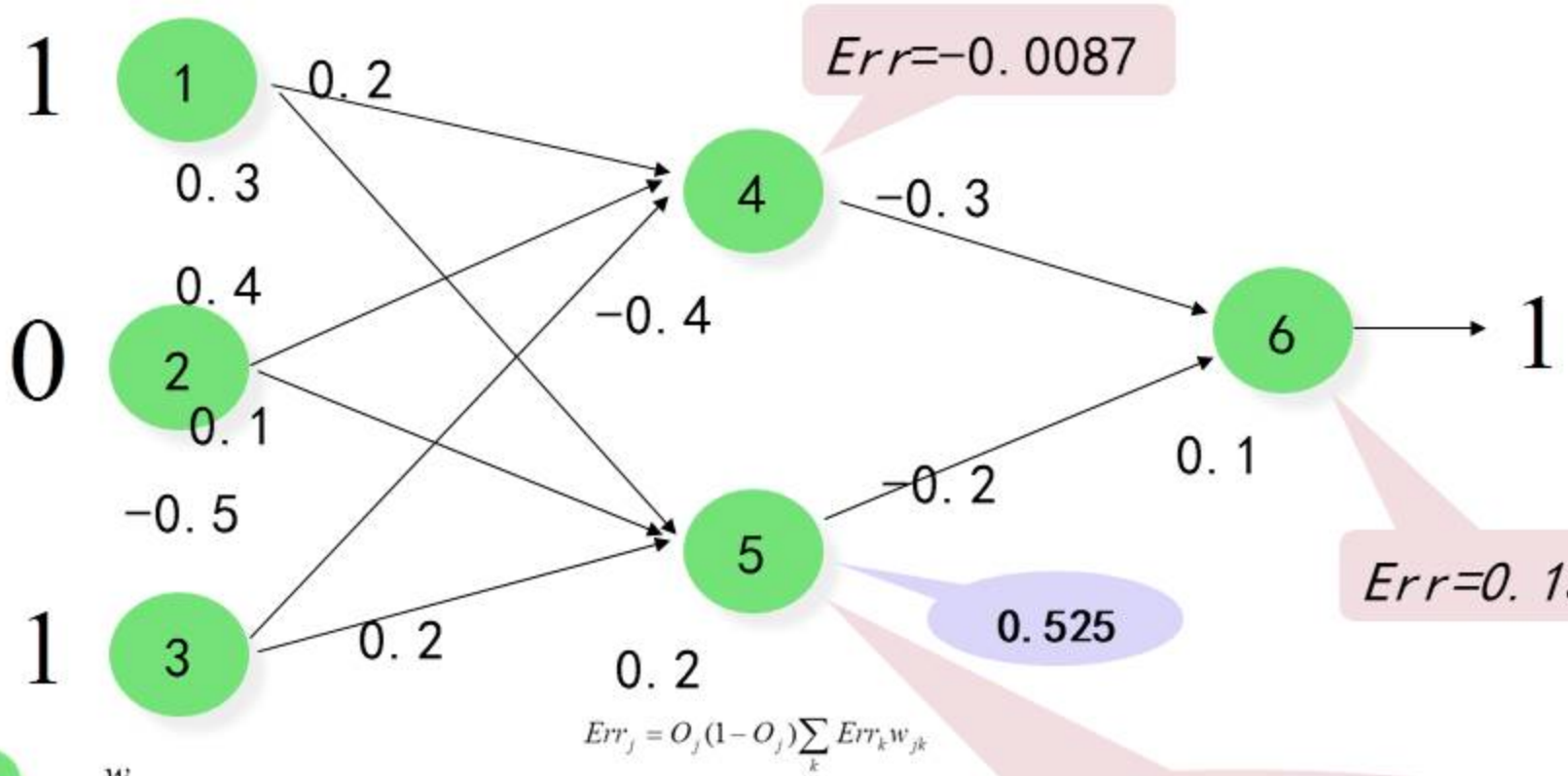
作答

1.5后向传播算法

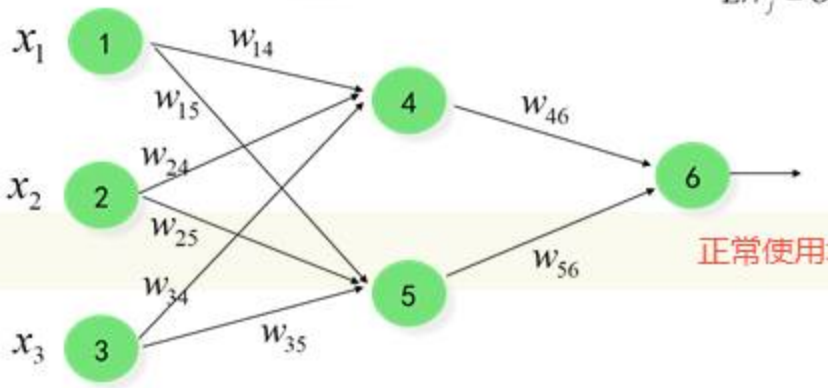


填空题 1分

设置



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

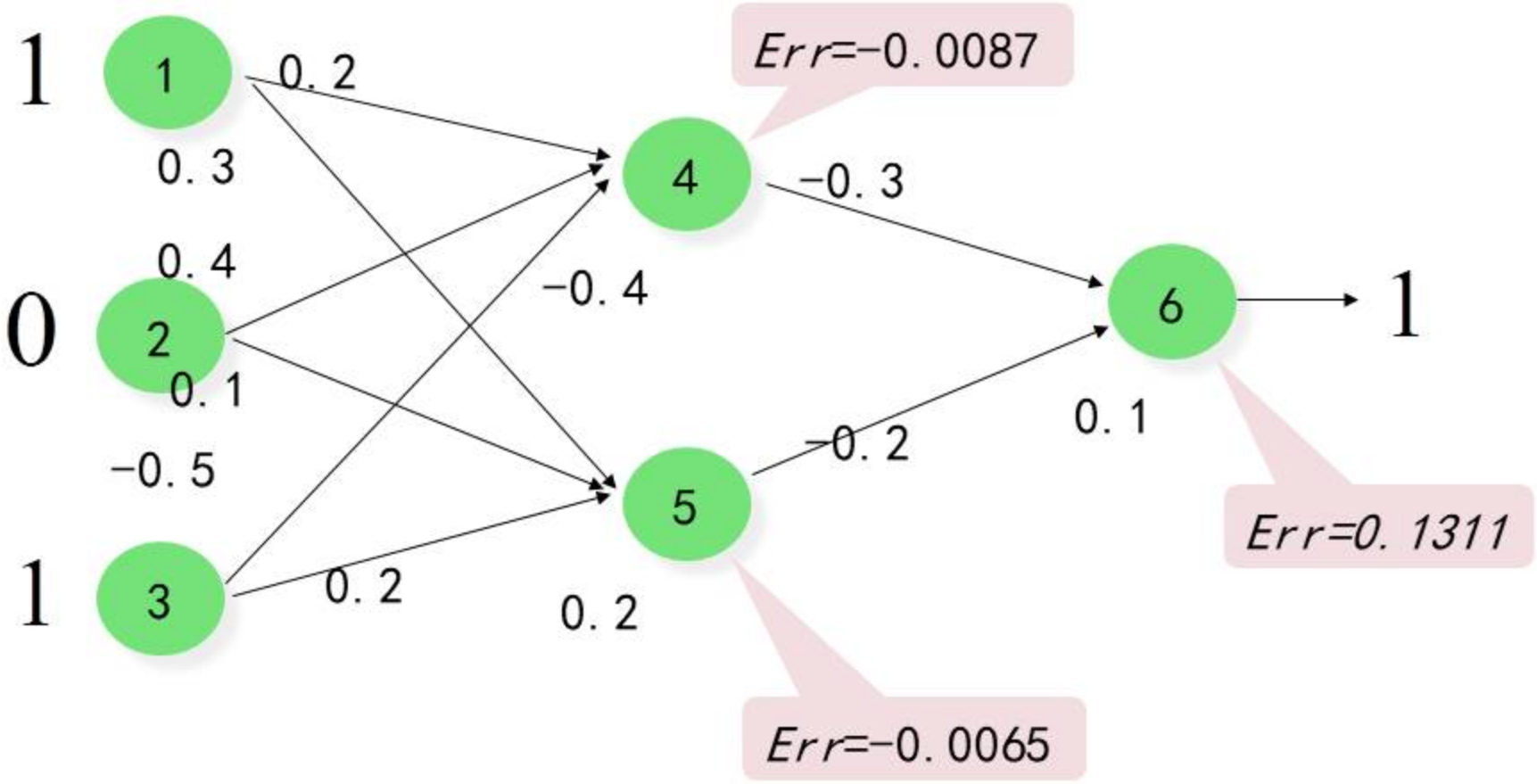


Err=

0.525 (1−0.525) (0.1311) (−0.2)=[填空1

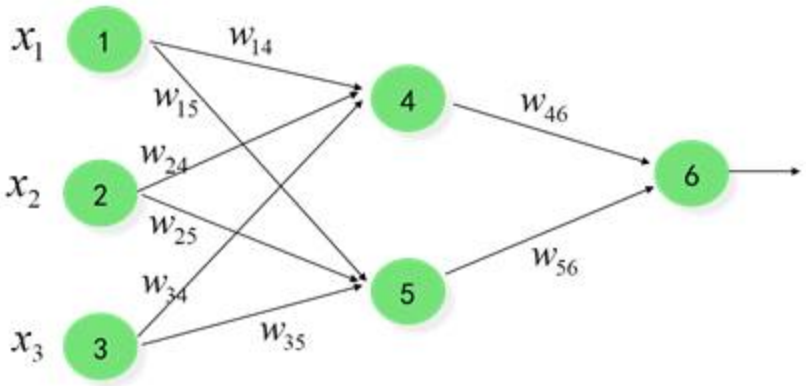
正常使用填空题需3.0以上版本雨课堂

作答


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=-0.0087
Err=0.1311
Err=-0.0065

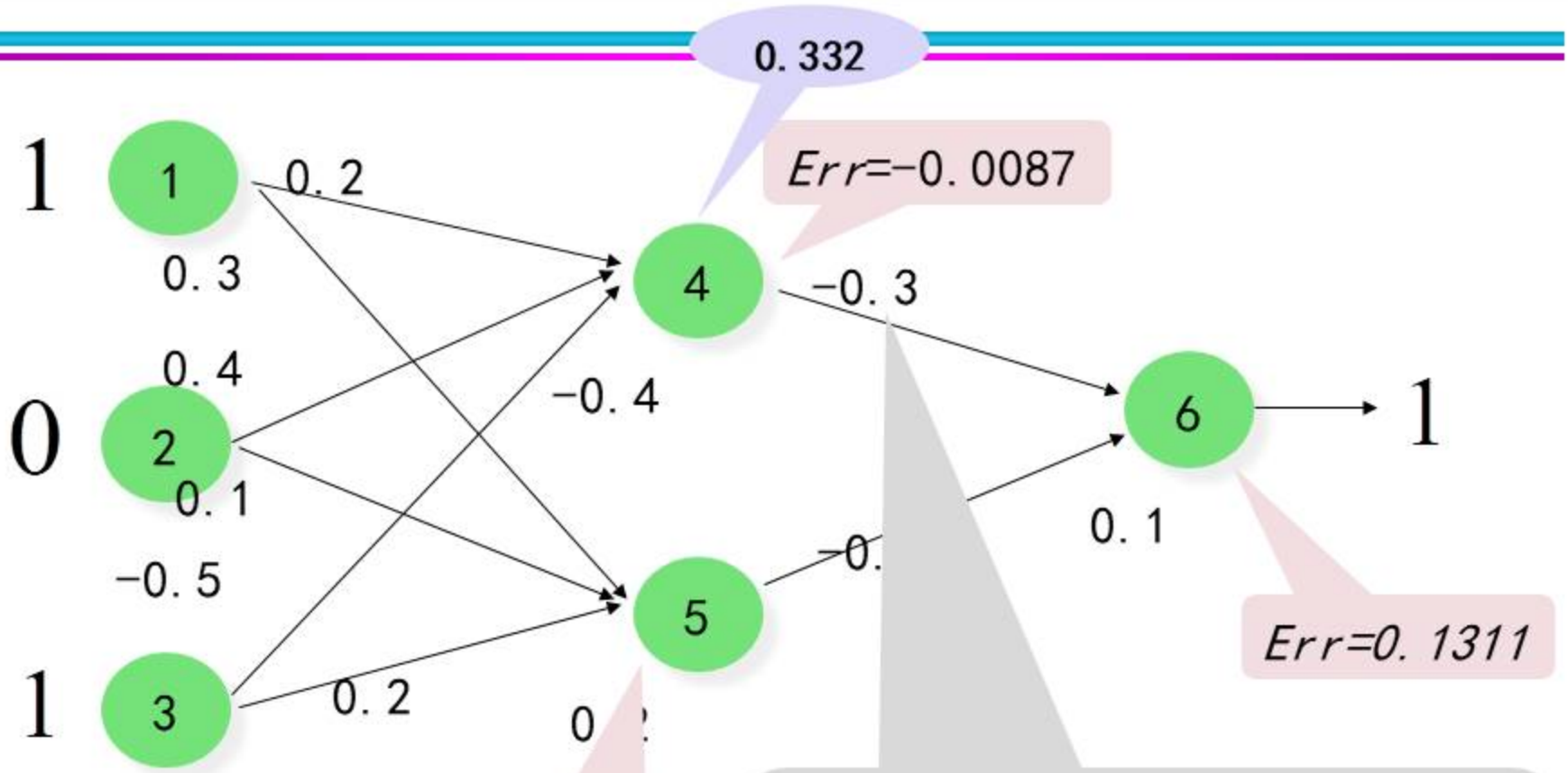
计算各结点的误差

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	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$



填空题 1分

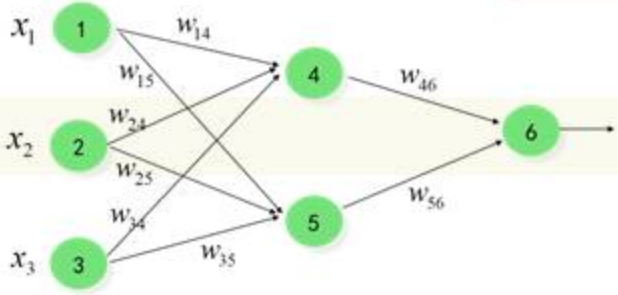
设置



$w' = w + \lambda * Err * 0_j$

$w' = -0.3 + 0.9 * 0.1311 * 0.332$

$= $ [填空1]

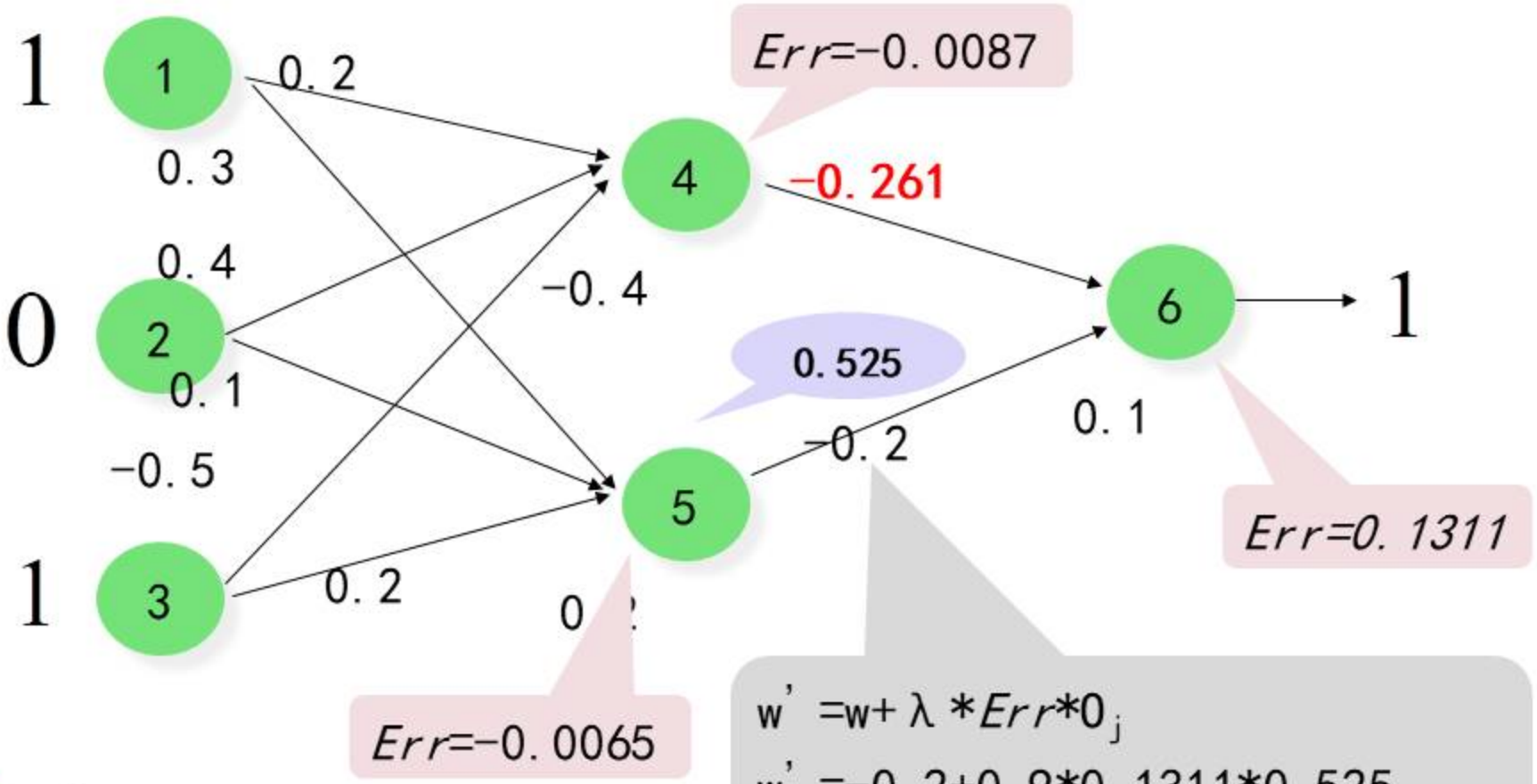


正常使用填空题需3.0以上版本雨课堂

作答

填空题 1分

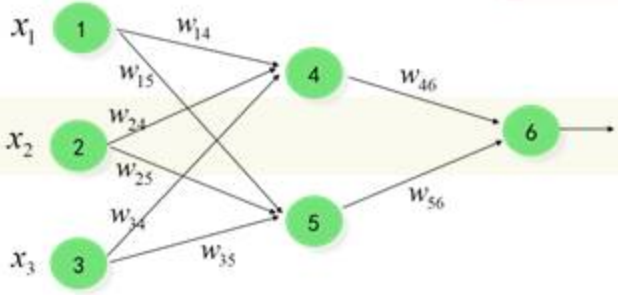
设置



$w' = w + \lambda * Err * 0_j$

$w' = -0.2 + 0.9 * 0.1311 * 0.525$

$= $ [填空1]

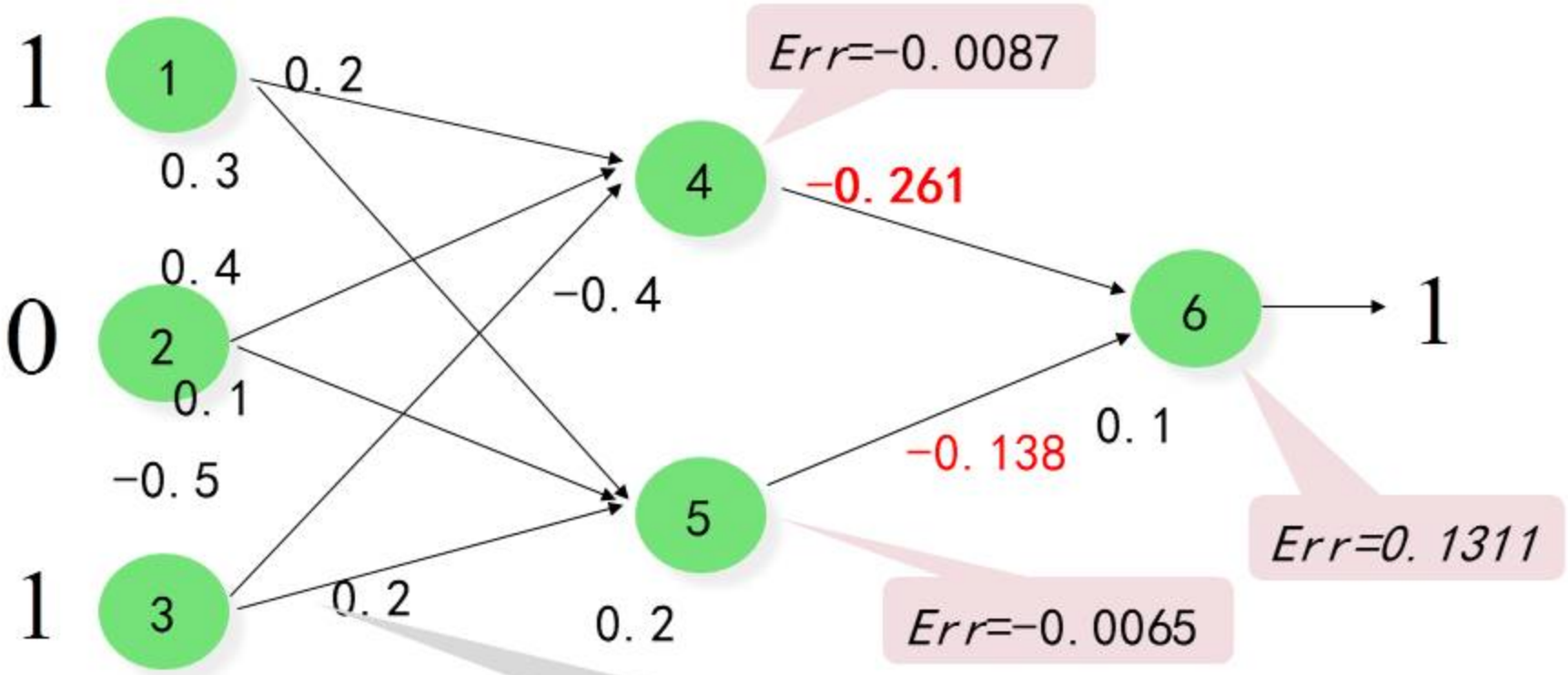


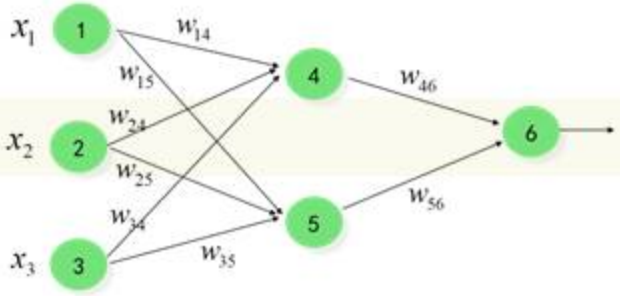
正常使用填空题需3.0以上版本雨课堂

作答

填空题 1分

设置

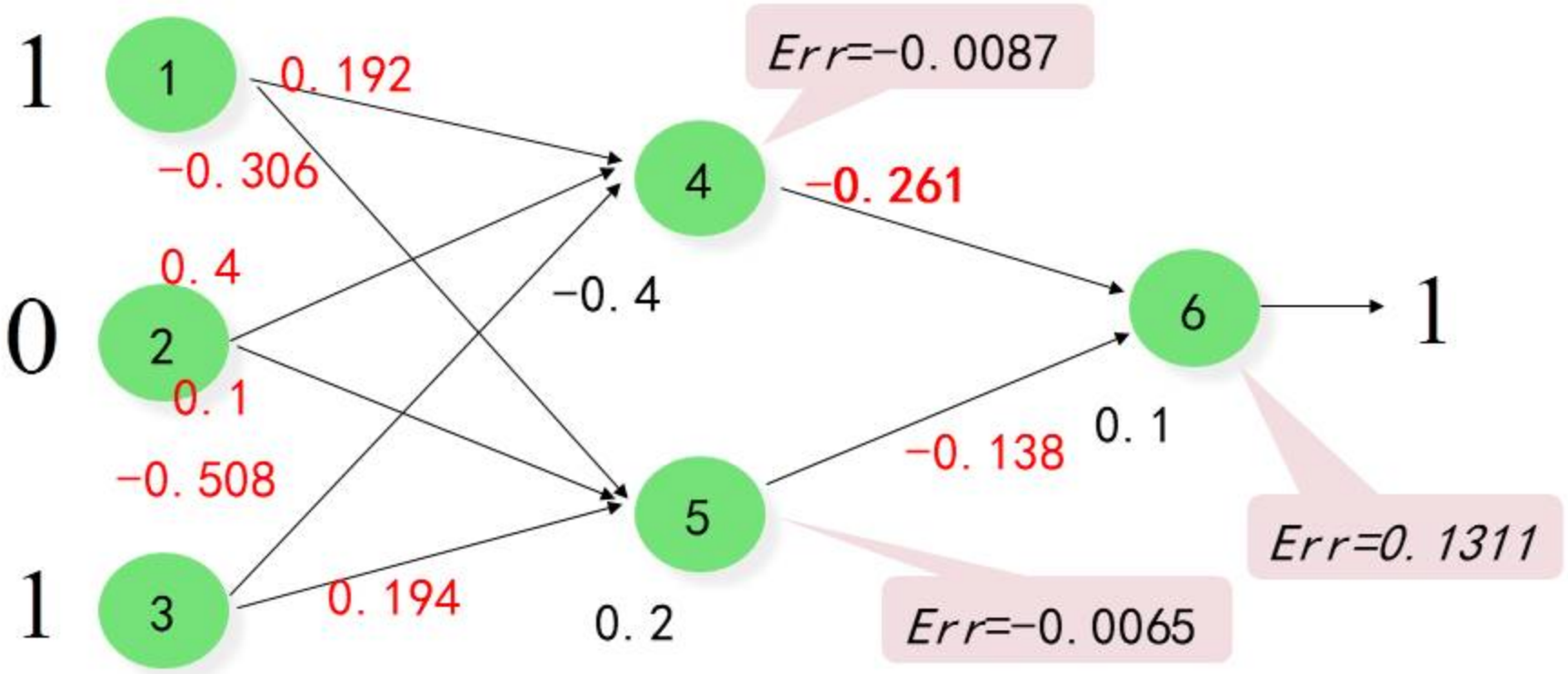




$w' = w + \lambda * Err * 0_j$

$w' = 0.2 + 0.9 * -0.0065 * 1 =$ [填空1]

作答


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

1.5后向传播算法

根据误差调整新的权重 和偏倚

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

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

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

$$t_j = t_j + \lambda Err_j$$

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	$-0.4 + (0.9)(-0.0087) = -0.408$

1.5后向传播算法

1. 初始化权重

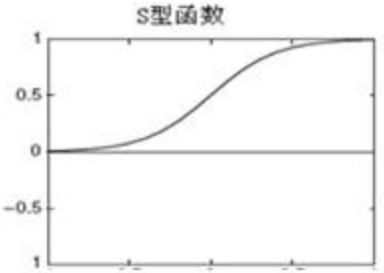
循环以下两步，直到满足条件

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

1. 向前传播输入

在每个节点加权求和，再代入激活函数

$$\hat{y} = sign(\sum_i w_{ij} x_i - t_j)$$



2. 向后传播误差

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

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

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

$$t_j = t_j + \lambda Err_j$$

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

推导参考：

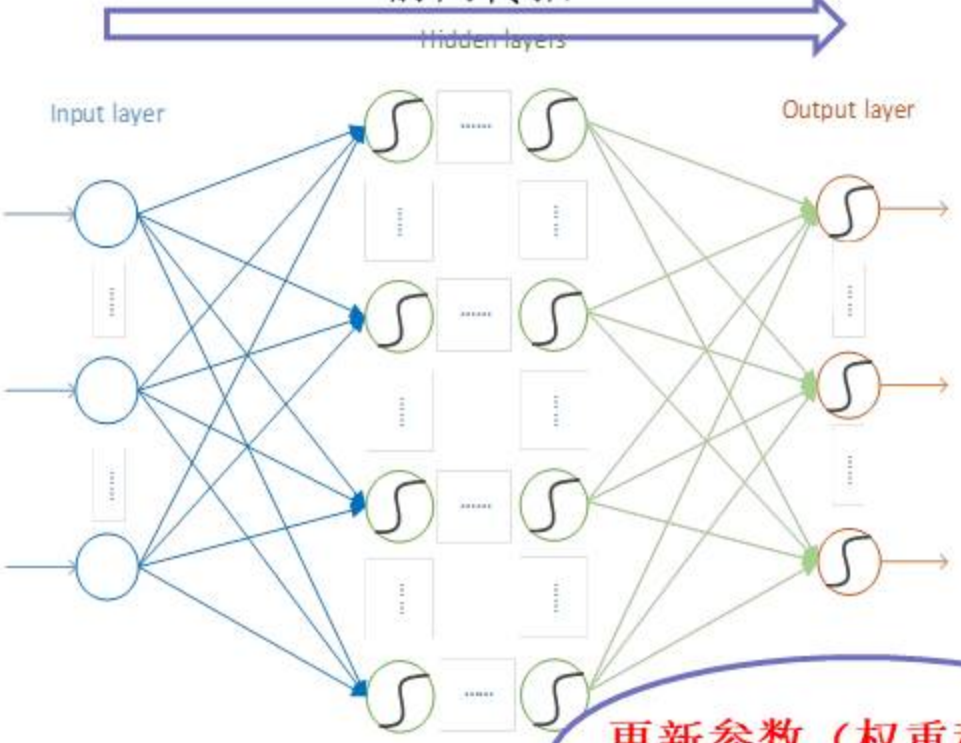
https://baijiahao.baidu.com/s?id=1589633691348109038&wfr=spider&for=pc

1.6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



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

从训练集中选择一部分样本（Batch）

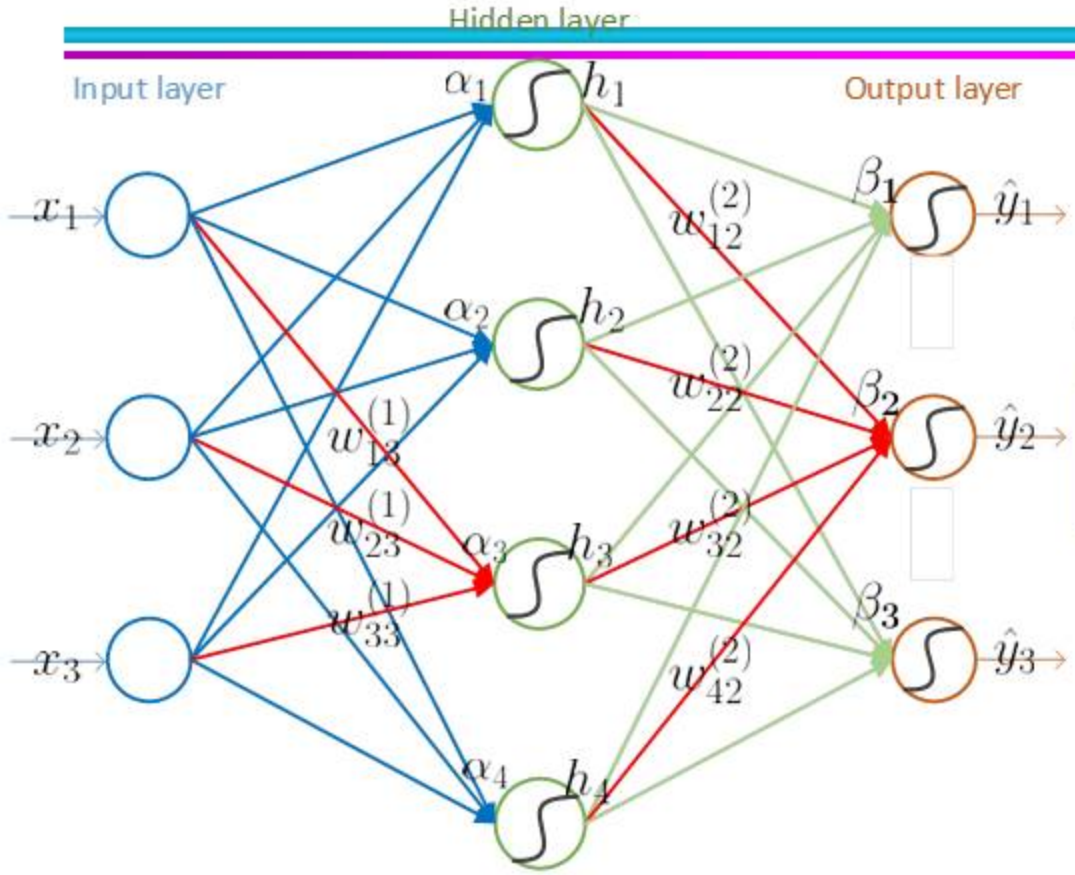
损失函数

最小化损失

更新参数（权重和阈值）

迭代

示例



超参数

- 激活函数：Sigmoid
- 损失函数：均方差损失
- Batch size: 1

训练的参数

Weights $\vec{w}$, Bias $\vec{b}$ 随机初始化

最小化 $E = \frac{1}{2}\sum_{j=1}^{3}(\hat{y}_j - y_j)^2$

梯度下降法（Gradient Descent）

$$D = \{(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), \cdots, (\vec{x}_{1000}, \vec{y}_{1000})\}, \vec{x} \in R^3, \vec{y} \in R^3$$

Input: $\vec{x} = (x_1, x_2, x_3)$

Output: $\vec{\hat{y}} = (\hat{y}_1, \hat{y}_2, \hat{y}_3)$

Real: $\vec{y} = (y_1, y_2, y_3)$

$$\alpha_3 = \sum_{k=1}^{3} w_{k3}^{(1)} x_k \qquad \beta_2 = \sum_{i=1}^{4} w_{i2}^{(2)} h_i$$

$$h_3 = f(\alpha_3 - b_3^{(1)}) \qquad \hat{y}_2 = f(\beta_2 - b_2^{(2)})$$

1.6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

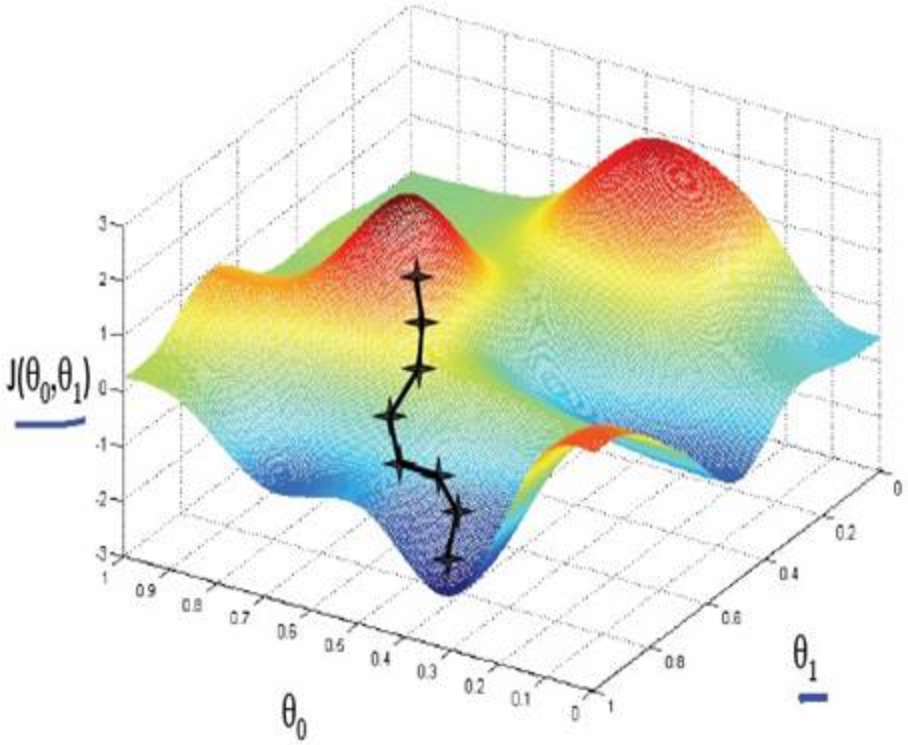
最小化 $f(\theta_0, \theta_1)$ $\Longrightarrow$ $\boxed{\vec{\theta} \leftarrow \vec{\theta} - \eta \nabla f(\vec{\theta})}$ $\Longrightarrow$

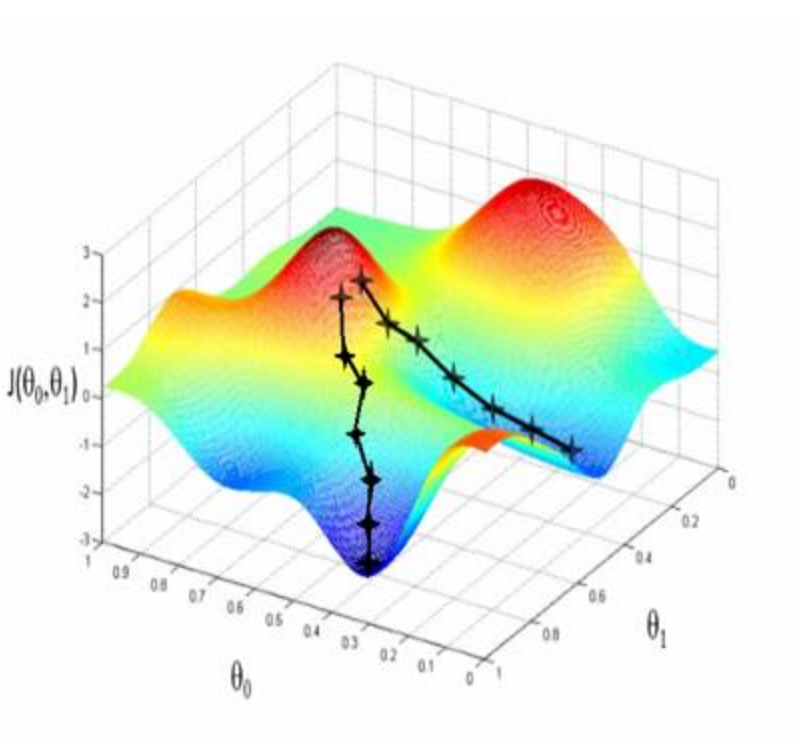
$$\theta_0 \leftarrow \theta_0 - \eta \frac{\partial f}{\partial \theta_0}$$

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

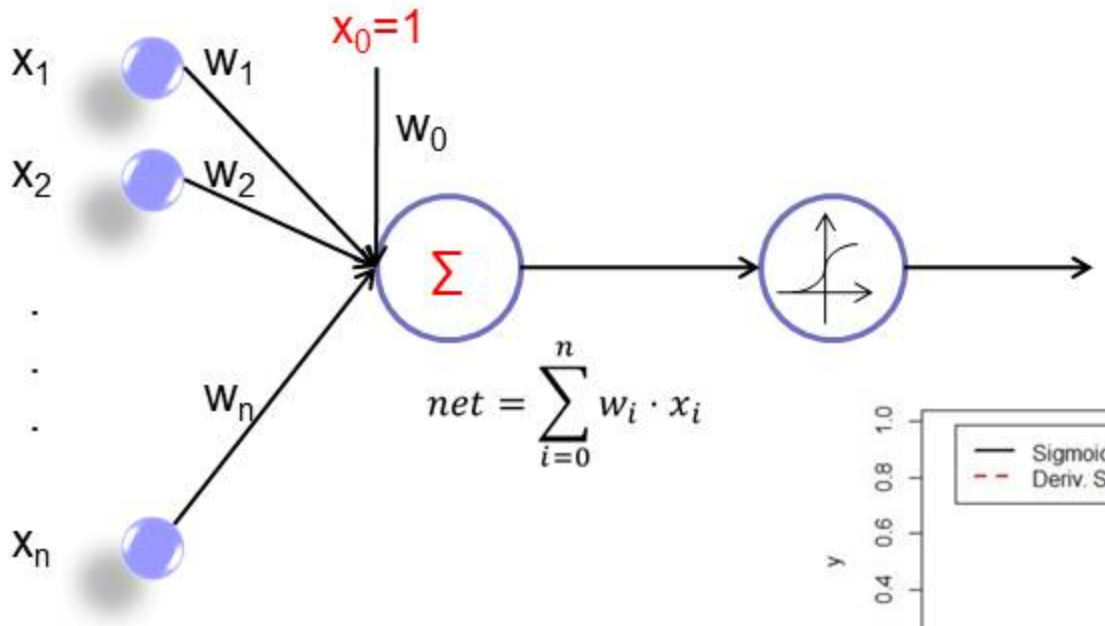
1.6后向传播算法推导

梯度下降是求一个函数最小值的一阶迭代优化算法



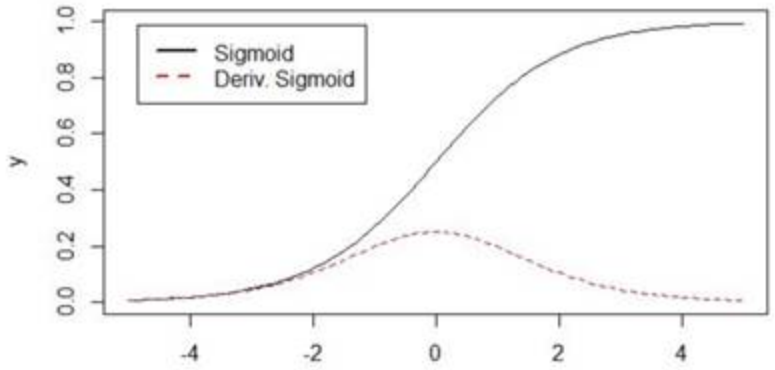


1.6后向传播算法推导



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

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

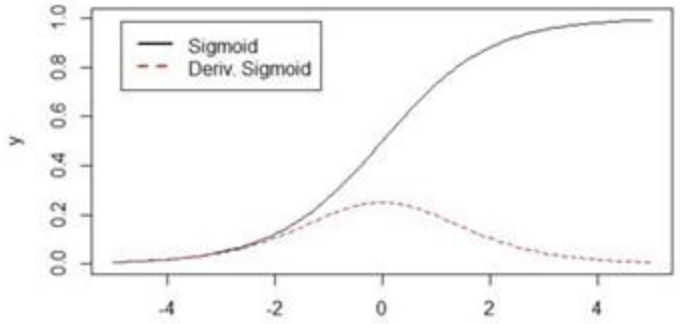


logistic函数（sigmoid函数）

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

1.6后向传播算法推导

Logistic函数求导 $y = \frac{1}{1+e^{-x}}$



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

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

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

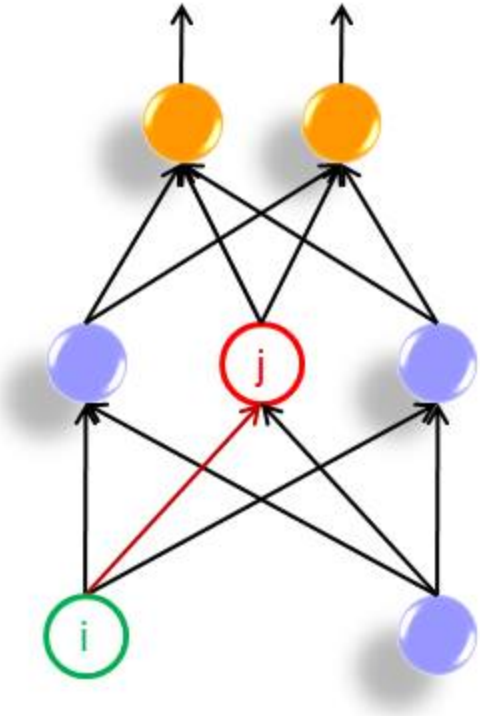
$$= \frac{1}{1+e^{-x}}(1 - \frac{1}{1+e^{-x}}) \tag{4}$$

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

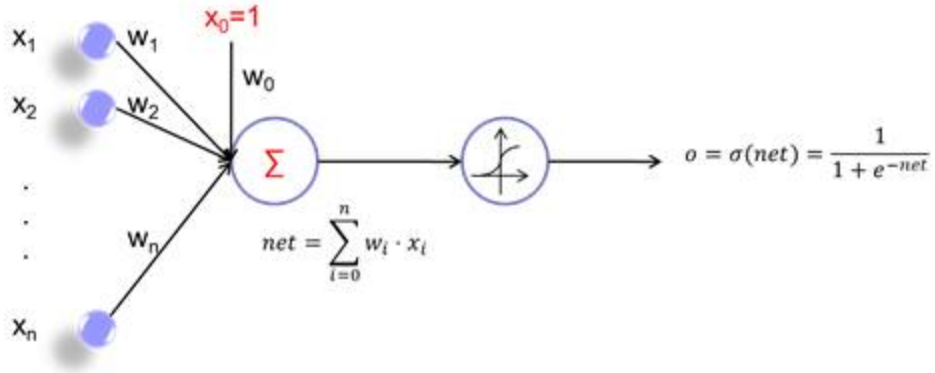
1.6后向传播算法推导

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

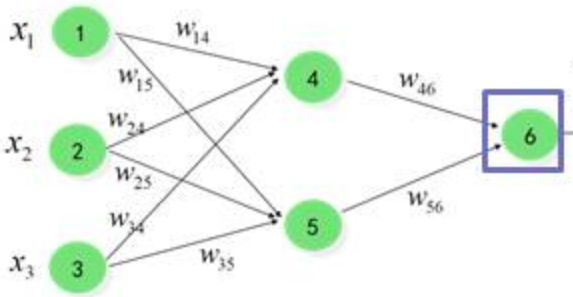
$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j} x_{ij}$$



- $net_j = \sum w_{ij} x_{ij}$ （ 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



1.6.1Output 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

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

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

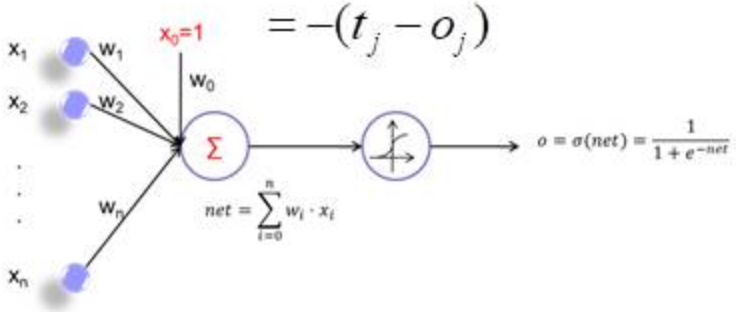
$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1-o_j)$$

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

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

$$= -(t_j - o_j)$$

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

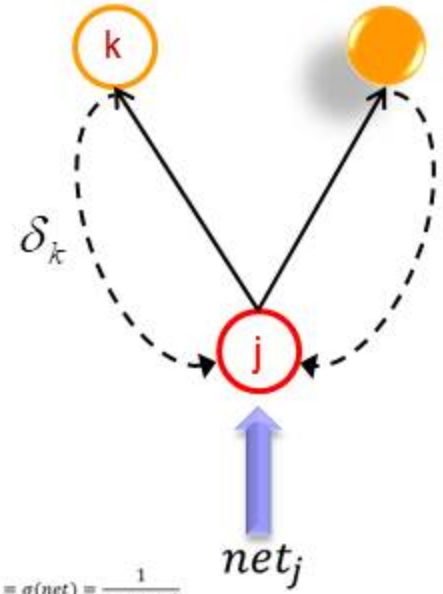


$$\Delta w_{ij} = -\eta\frac{\partial E_d}{\partial w_{ij}} = \eta(t_j - o_j)o_j(1-o_j)\,x_{ij}$$

1.6.2Hidden Units 隐藏层神经元的训练

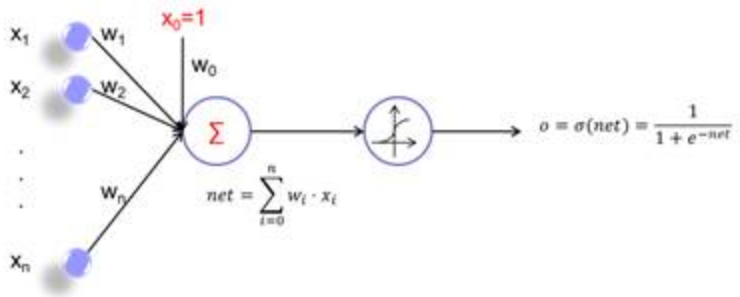
$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

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



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

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



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

1.6后向传播算法推导

1. 初始化权重

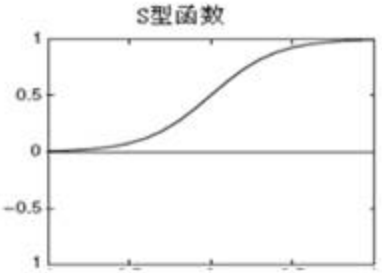
循环以下两步，直到满足条件

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

1. 向前传播输入

在每个节点加权求和，再代入激活函数

$$\hat{y} = sign(\sum_i w_{ij} x_i - t_j)$$



2. 向后传播误差

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

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

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

$$t_j = t_j + \lambda Err_j$$

均方误差

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

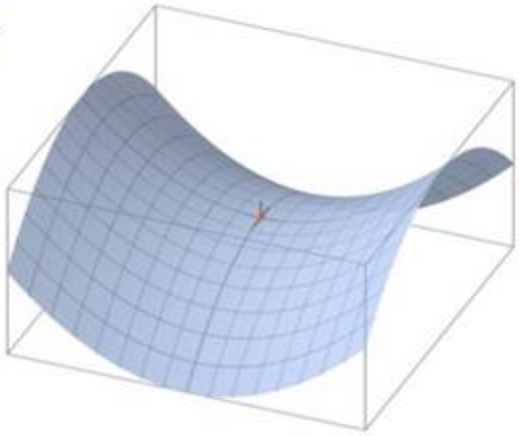
权值迭代，梯度下降法

推导参考：

https://baijiahao.baidu.com/s?id=1589633691348109038&wfr=spider&for=pc

1.7后向传播BP网络注意事项 人工神经网络

- 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

人工神经网络

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

多选题 1分

设置

神经网络分类器的特点

A 普适近似，精度较高

B 噪声敏感

C 训练非常耗时

提交

神经网络分类编程实践

- https://scikit-learn.org/stable/modules/neural_networks_supervised.html

- 同学们可以尝试利用python读入本地iris数据集，来完成神经网络分类，分析其分类效果

第13次课后作业

- 第十三次课后作业-在educoder平台上完成作业
- https://www.educoder.net/shixuns/tf9qcivo/challenges

提交作业截至时间：2020年4月2日

题目来源

- 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。
- 在这个挑战中，我们要求你完成对什么样的人可能生存的分析。

题目内容

- 目标：预测泰坦尼克号上的乘客是幸存还是遇难，每个乘客对应一个乘客Id，用0表示遇难，用1表示幸存。
- 成绩：用正确预测的百分比来表示你的成绩。

题目数据

- 题目数据被分为两组:
- 训练集(train.csv)
- 测试集(test.csv)

训练集中包含每个乘客的生存结果以及各项特征，以此来学习规律。

测试集中不包含乘客的生存结果，根据各项特征预测乘客的生存情况。

求解思路

- 下载数据和工具
- 认识数据
- 选取特征、数据预处理
- 利用模型求解
- 优化结果

认识数据

- 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。
- 训练集的数据形式

	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

认识数据

- 训练集中的数据信息

```
train.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

认识数据

- 空缺数据统计：年龄缺少177个 船舱号缺少687个 存活率： 38.4%

```
print(train.isnull().sum())
print(test.info())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
```

```
surv = train[train['Survived']==1]
nosurv = train[train['Survived']==0]
surv_col = "blue"
nosurv_col = "red"

print("Survived: %i (%.1f percent), Not Survived: %i (%.1f percent), Total: %i"\
      %(len(surv), 1.*len(surv)/len(train)*100.0,\
        len(nosurv), 1.*len(nosurv)/len(train)*100.0, len(train)))
```

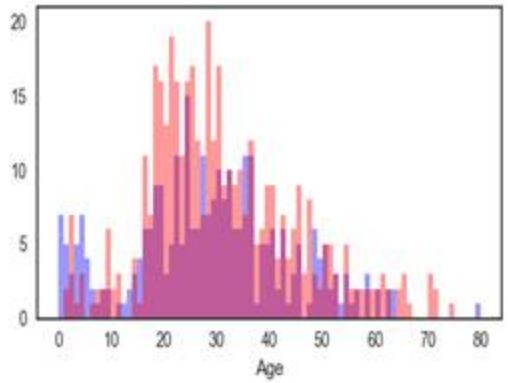
```
Survived: 342 (38.4 percent), Not Survived: 549 (61.6 percent), Total: 891
```

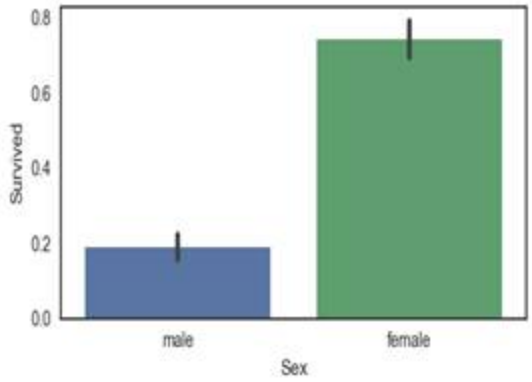
Sibsp数据分布

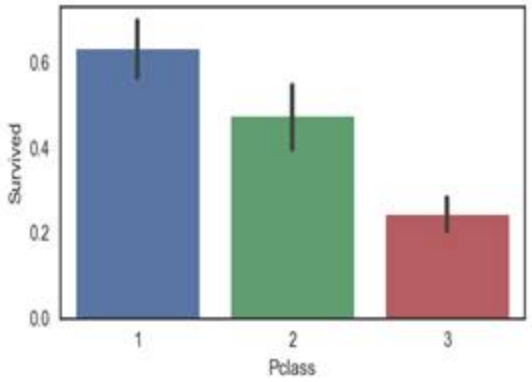
```
tab = pd.crosstab(train['SibSp'], train['Survived'])
print(tab)
```

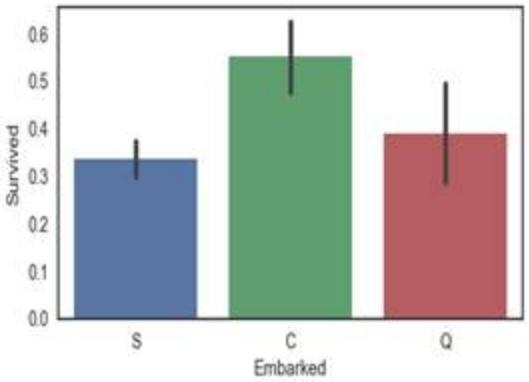
```
Survived    0    1
SibSp
0         398  210
1          97  112
2          15   13
3          12    4
4          15    3
5           5    0
8           7    0
```

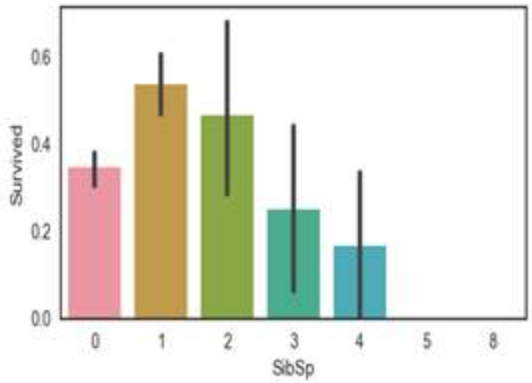
认识数据

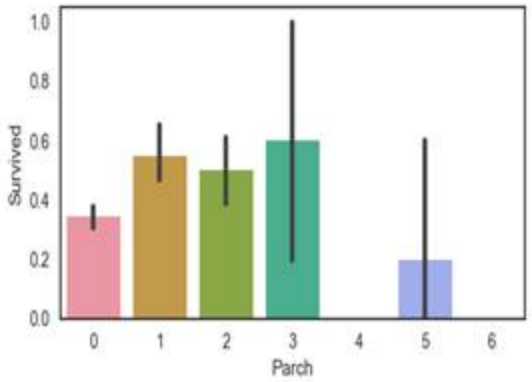

Age

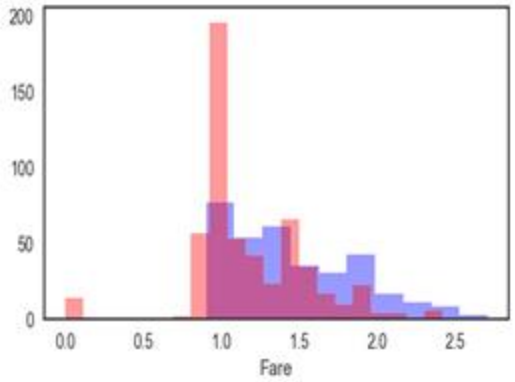

Survived
male
female
Sex


Survived
Pclass


Survived
S
C
Q
Embarked


Survived
SibSp


Survived
Parch


Fare

认识数据

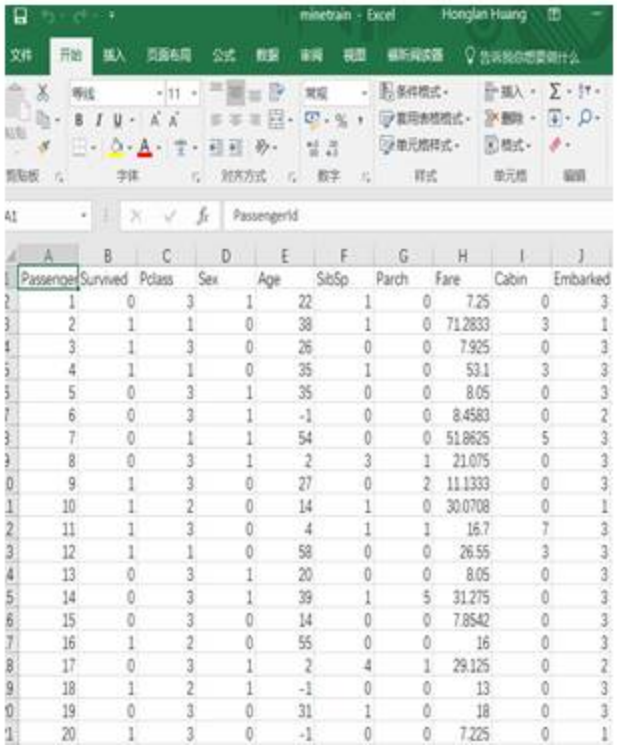
- 通过初步观察数据我们可以得到以下几点结论：
- 1、女性存活率明显比男性高→性别(Sex)
- 2、1等舱的乘客明显存活率高→等级(Class)
- 3、孩子存活率较高→年龄(Age)
- 4、有少数同行人员存活率较高→兄弟姐妹(SibSp)、父母子女(Parch)
- 5、票价、船舱号和等级相对应，同样影响生存率→票价(Fare)、船舱号(Cabin)
- 6、C港口登船的人存活率较高→登陆港口(Embarked)

由此选出所需的特征

数据处理

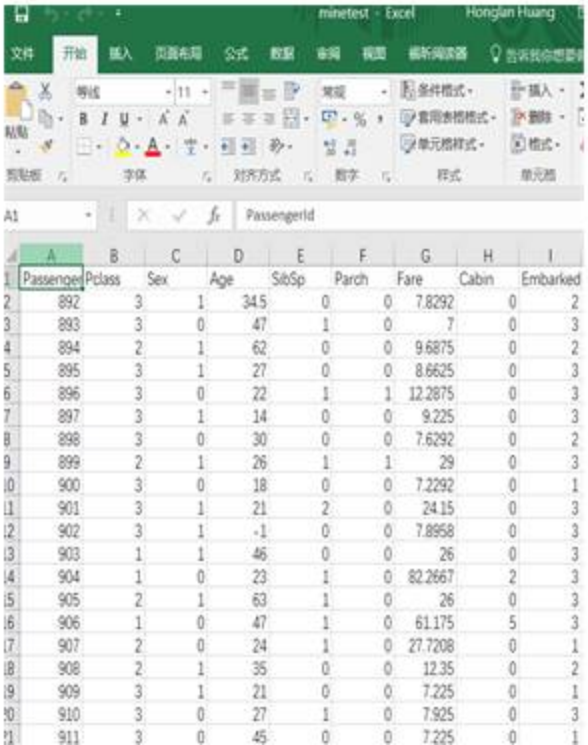
- 性别：男性→1 女性→0
- 船舱：A →1　B →2　C →3　D →4　E →5　F →6　G →7　T →8　空缺→0
- 港口：C →1　Q →2　S →3　空缺→0
- 年龄：空缺→-1

train



PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
1	0	3	1	22	1	0	7.25	0	3
2	1	1	0	38	1	0	71.2833	3	1
3	1	3	0	26	0	0	7.925	0	3
4	1	1	0	35	1	0	53.1	3	3
5	0	3	1	35	0	0	8.05	0	3
6	0	3	1	-1	0	0	8.4583	0	2
7	0	1	1	54	0	0	51.8625	5	3
8	0	3	1	2	3	1	21.075	0	3
9	1	3	0	27	0	2	11.1333	0	3
10	1	2	0	14	1	0	30.0708	0	1
11	1	3	0	4	1	1	16.7	7	3
12	1	1	0	58	0	0	26.55	3	3
13	0	3	1	20	0	0	8.05	0	3
14	0	3	1	39	1	5	31.275	0	3
15	0	3	0	14	0	0	7.8542	0	3
16	1	2	0	55	0	0	16	0	3
17	0	3	1	2	4	1	29.125	0	2
18	1	2	1	-1	0	0	13	0	3
19	0	3	0	31	1	0	18	0	3
20	1	3	0	-1	0	0	7.225	0	1

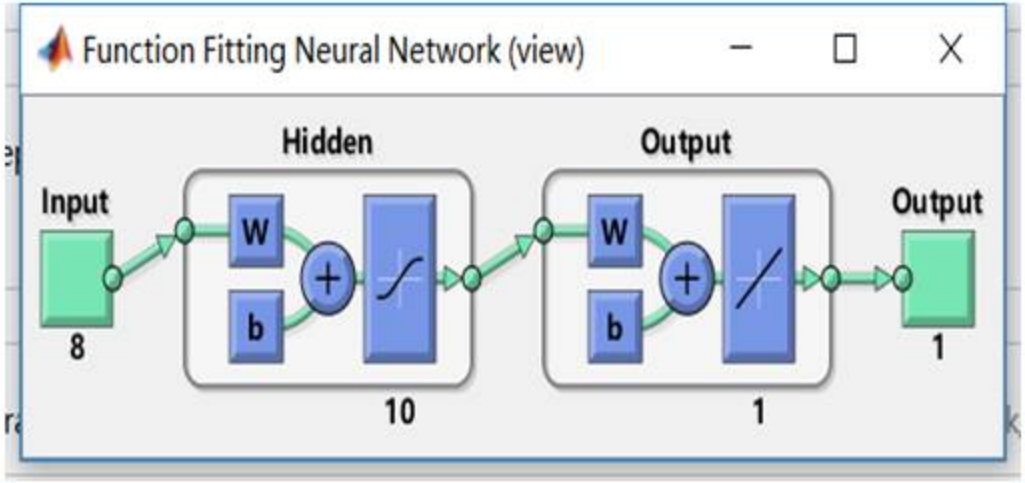
test



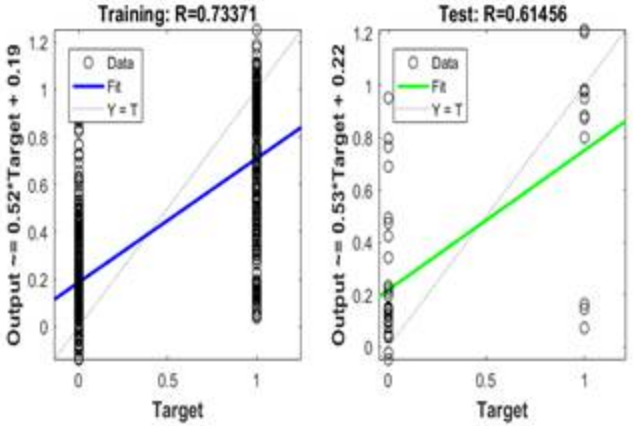
PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
892	3	1	34.5	0	0	7.8292	0	2
893	3	0	47	1	0	7	0	3
894	2	1	62	0	0	9.6875	0	2
895	3	1	27	0	0	8.6625	0	3
896	3	0	22	1	1	12.2875	0	3
897	3	1	14	0	0	9.225	0	3
898	3	0	30	0	0	7.6292	0	2
899	2	1	26	1	1	29	0	3
900	3	0	18	0	0	7.2292	0	1
901	3	1	21	2	0	24.15	0	3
902	3	1	-1	0	0	7.8958	0	3
903	1	1	46	0	0	26	0	3
904	1	0	23	1	0	82.2667	2	3
905	2	1	63	1	0	26	0	3
906	1	0	47	1	0	61.175	5	3
907	2	0	24	1	0	27.7208	0	1
908	2	1	35	0	0	12.35	0	2
909	3	1	21	0	0	7.225	0	1
910	3	0	27	1	0	7.925	0	3
911	3	0	45	0	0	7.225	0	1

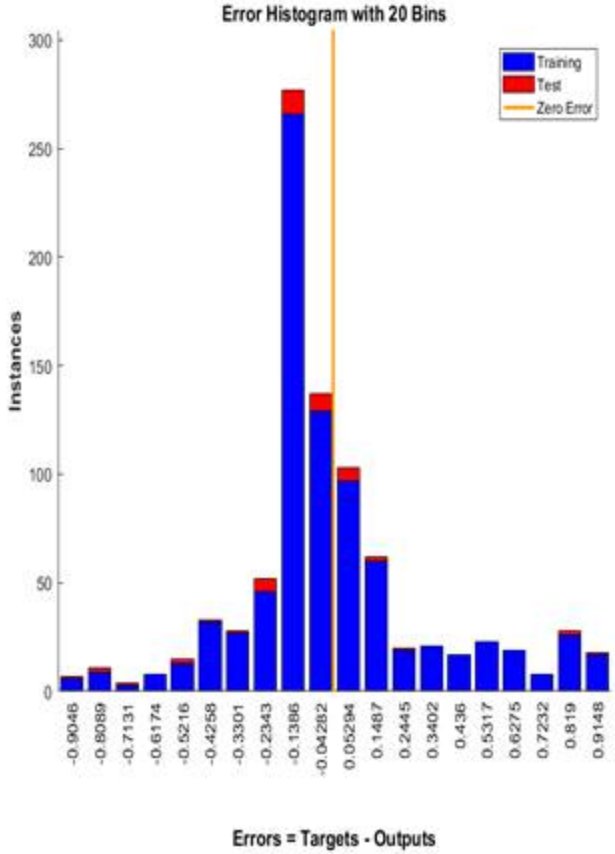
模型求解

- 利用神经网络模型，模型设置为对train.csv的数据按照70%比例的数据参与模型的训练，剩余30%作为测试数据集。



模型求解





神经网络在训练集上的准确率为93%左右，那么在测试集上呢?

实验结果

- 进行多次训练和调整层数，将测试集输入到训练好的网络上。

```
D=myNeuralNetworkFunction8(c);
Y=ones(418,10);
for j = 1:10
    for i =1:418
        if D(i)<=0.4+j/100
            Y(i,j)=0;
        else Y(i,j)=1;
        end
    end
end
```

	Passenger	Survived
2	892	0
3	893	0
4	894	0
5	895	0
6	896	0
7	897	0
8	898	1
9	899	0
10	900	1

- 结果在测试集上目前准确率最好的成绩为82.775%，在排行榜上排140名左右。

优化结果

- 1、投票法
- 2、…

13	▲14	John Johns		1.00000
14	▲2740	boiledeggs		1.00000
15	new	Kevin McClain		
16	▼3	NUDT丁兆云DM课		
17	▼3	krish95chs		

1	–	RANDOF Consulting		1.00000	2	2mo
2	–	ryemitan		1.00000	1	2mo
3	–	Jason Zutty		1.00000	1	2mo
4	–	tushky01		1.00000	3	2mo
5	–	Nostradamus		1.00000	12	1mo
6	–	Abhinav Sharma		1.00000	1	1mo
7	–	giim		1.00000	20	1mo
8	–	Aidyn K		1.00000	11	1mo
9	–	keyihuang		1.00000	6	1mo
10	–	muni0893 3		1.00000	5	1mo
11	–	itaibl		1.00000	25	23d
12	–	AaronTa		1.00000	1	7d
13	▲15	John Johns		1.00000	11	5d
14	▲3381	boiledeggs		1.00000	8	3d

总结

- 从结果来看，妇女儿童的存活率明显很高。面对沉船灾难时，船长爱德华·约翰·史密斯（Edward J. Smith）在最后的时刻下命令，命令先让妇女和儿童上救生艇。
- 但人类毕竟不是机器，并不是会划定一个规则就能完全预测的，深入了解那些算法难以预测的离群点，背后往往都蕴藏着非常动人的故事。

Evans turned to Brown and said, 'You go first, you have children waiting at home.'

Any Questions?

谢谢！