第九讲 神经网络

授课老师：郭 迟 教授

guochi@whu.edu.cn

武汉大学测绘学院

2021.12

神经网络

1 神经网络基本概念

2 感知器与多层网络

3 BP神经网络

4 总结

第九讲 神经网络

神经网络基本概念

- 上世纪40~50年代，人们开始尝试研究神经系统的数学模型
- 上世纪80年代以来，开始大量借用神经生理学的概念研究机器智能，进行人类感知与思维机理的探索，诞生了一门新的学科——人工神经网络（artificial neural networks），为模式识别和机器学习的方法体系注入了新的血液
- 神经网络采用可以实现的器件或计算机，通过硬件或软件的方式，模拟生物体神经系统的某些功能与结构，用以解决机器学习、识别、控制、决策等机器智能问题
- 模拟人脑神经细胞的工作特点：
  - * 单元间的广泛连接；
  - * 并行分布式的信息存贮与处理；
  - * 自适应的学习能力等

第九讲 神经网络

神经网络基本概念

- 第一阶段：启蒙期，始于1943年

  形式神经元的数学模型提出

- 第二阶段：低潮期，始于1969年

  《感知器》(Perceptions)出版

- 第三阶段：复兴期，从1982年到1986年

  Hopfield的两篇论文提出新的神经网络模型

  《并行分布处理》出版，提出反向传播算法

- 第四个阶段：低潮期，始于1990年

  统计学习理论、SVM的兴起

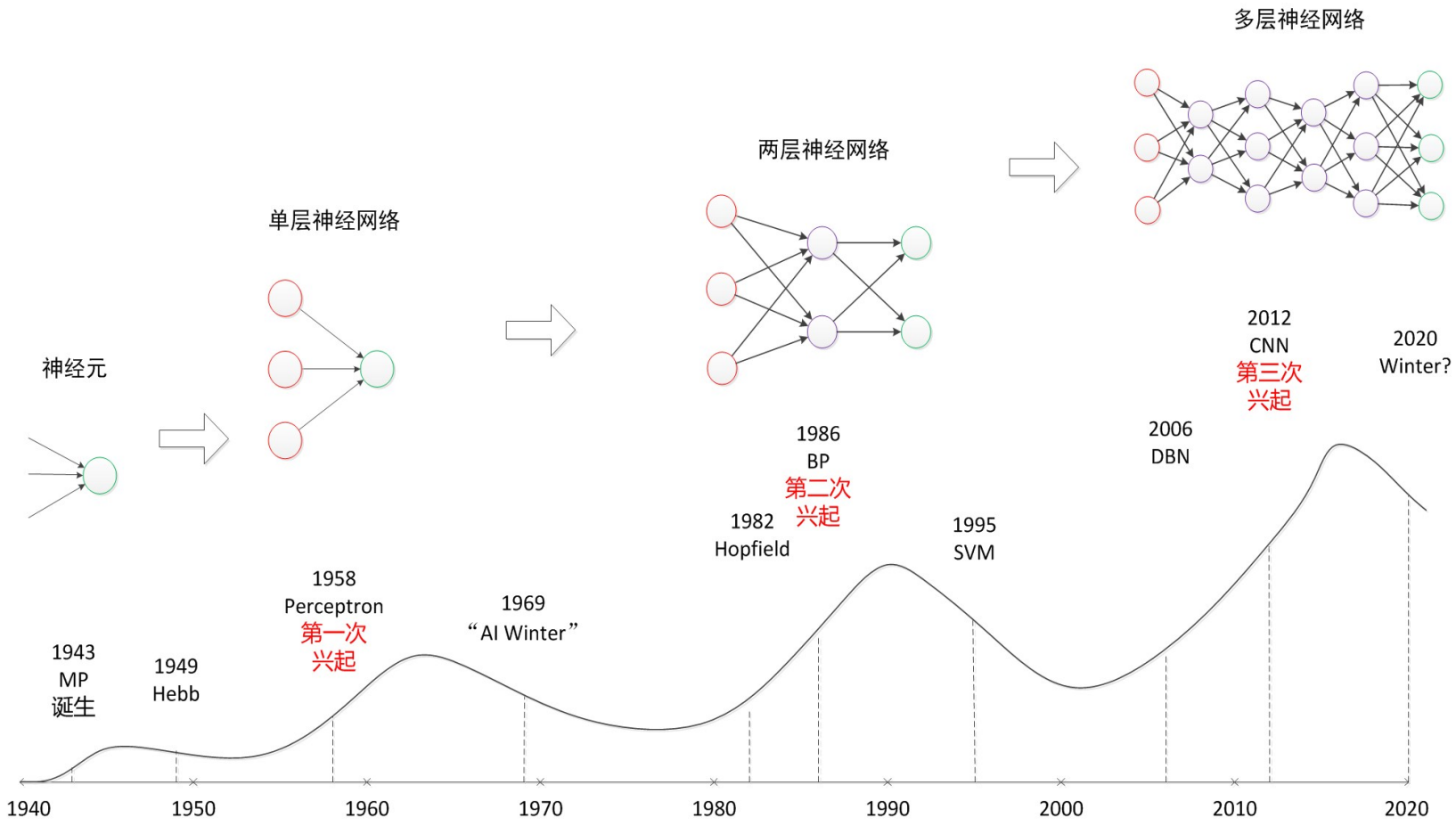
- 第五个阶段：热潮期，2010年后

  深度学习、Google大脑、AlphaGo

第九讲 神经网络

神经网络基本概念

- 神经网络的发展历程：



第九讲 神经网络

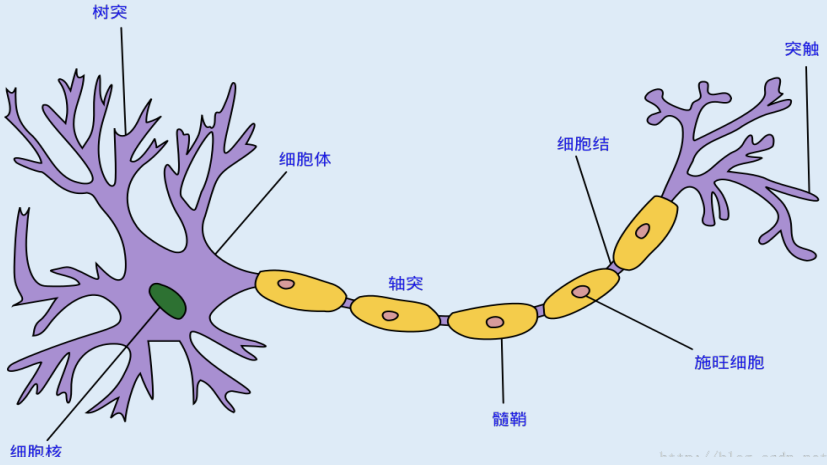
神经网络基本概念

- 生物神经元有兴奋和抑制两种状态，抑制状态的神经元由树突和细胞体接收传来的兴奋电位，当输入兴奋总量超过阈值，神经元被激发进入兴奋状态，产生输出脉冲，由突触传递给其它神经元

- 科学研究发现：小脑中人类的大脑皮质包含大约140-160亿神经元，小脑中包含大约550-700亿神经元，加起来大约是860亿，并且一个立方毫米的神经元可以产生超过1000TB的数据

生物神经元的基本组成：

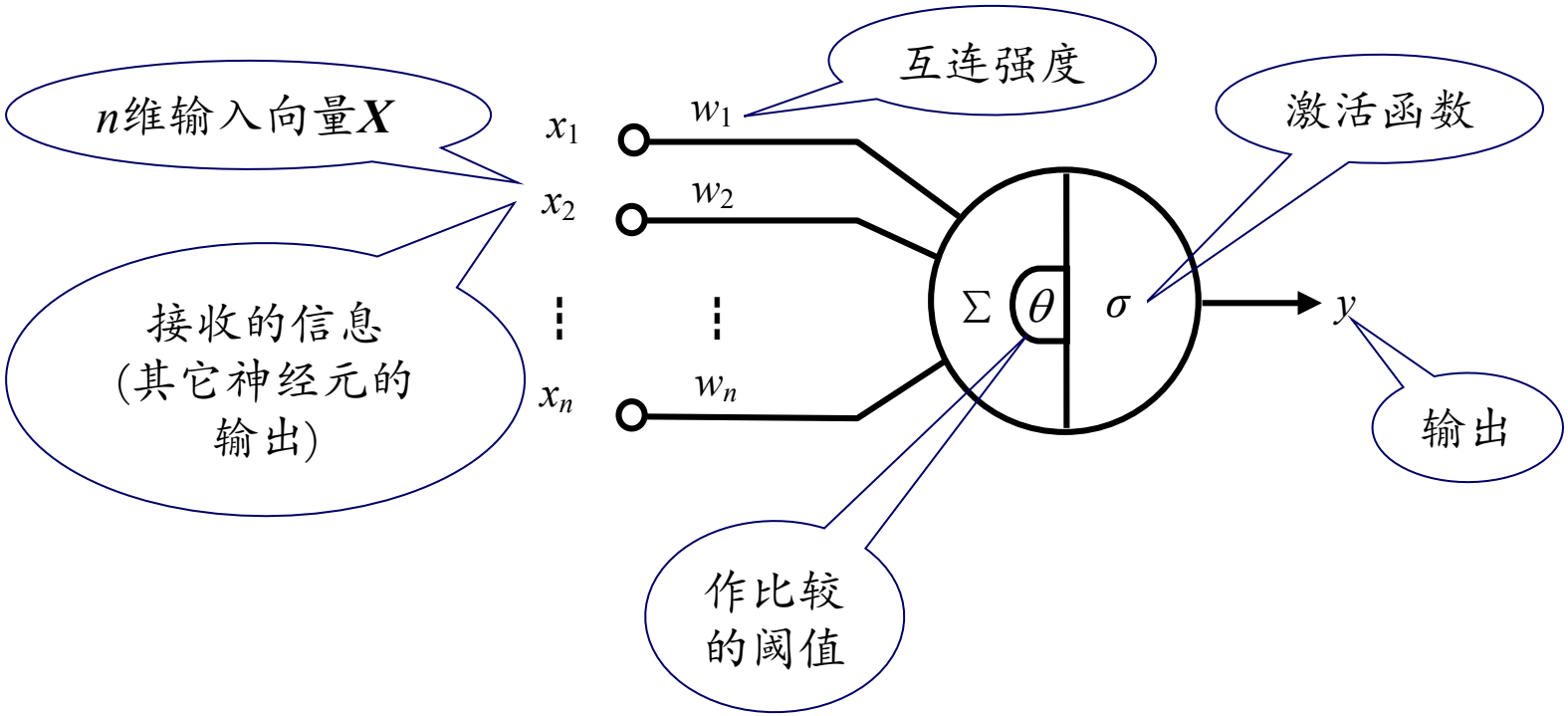
1. 细胞体
2. 树突
3. 轴突
4. 突触



第九讲 神经网络

神经网络基本概念

- 通过对生物神经元的模拟，得到人工神经网络的基本神经元结构：



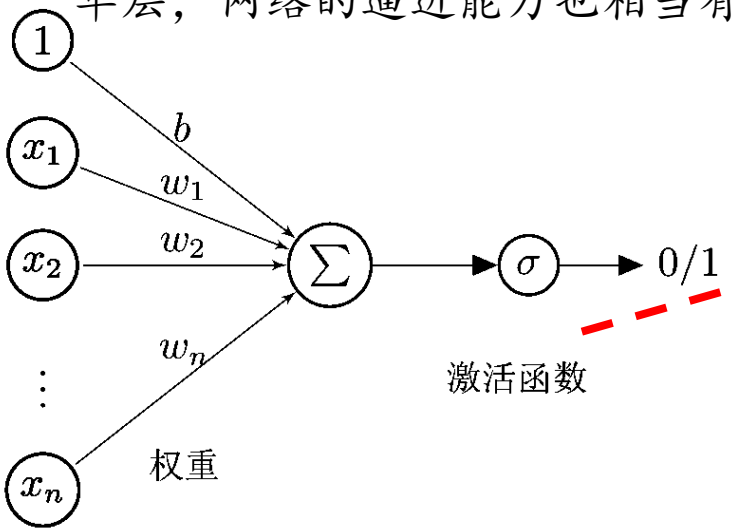
人工神经元间的互连：信息传递路径轴突-突触-树突的简化；

连接的权值：两个互连的神经元之间相互作用的强弱

神经网络基本概念

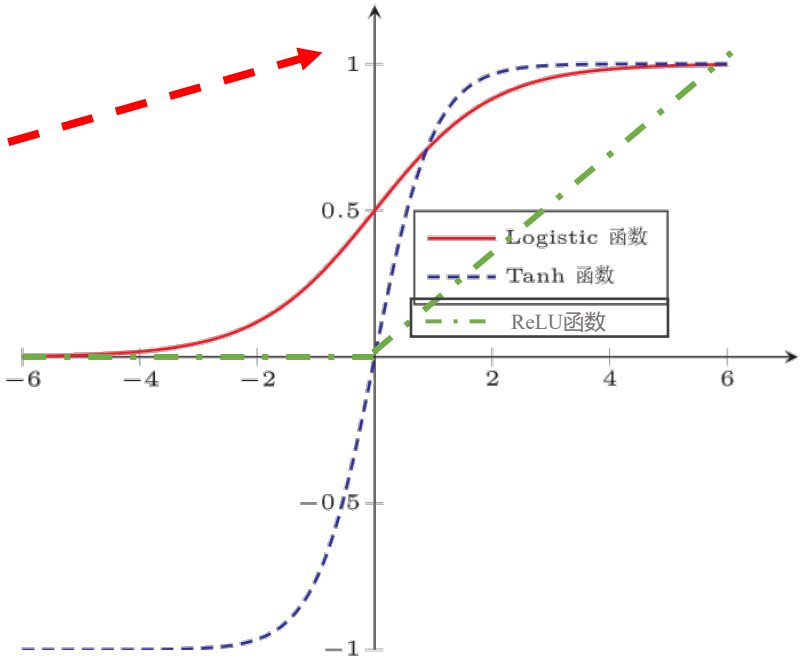
- 通过对生物神经元的模拟，得到人工神经网络的基本神经元结构：

  如果不用激活函数，每一层节点的输入都是上层输出的线性函数，很容易验证，无论你神经网络有多少层，输出都是输入的线性组合，可以合并为单层，网络的逼近能力也相当有限



$$\text{net}_i = \sum_{j=1}^{n} w_{ij} x_j + b$$

$$y_i = \sigma(\text{net}_i)$$

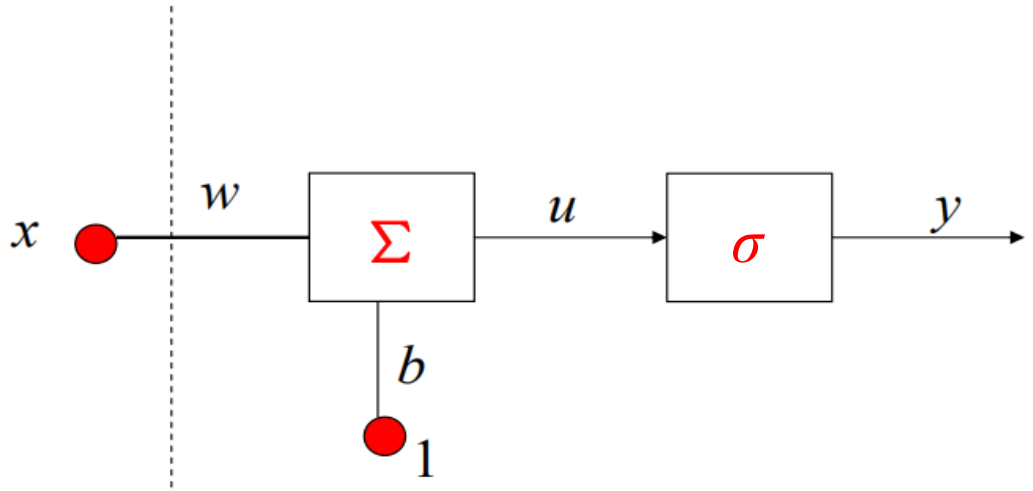


第九讲 神经网络

神经网络基本概念

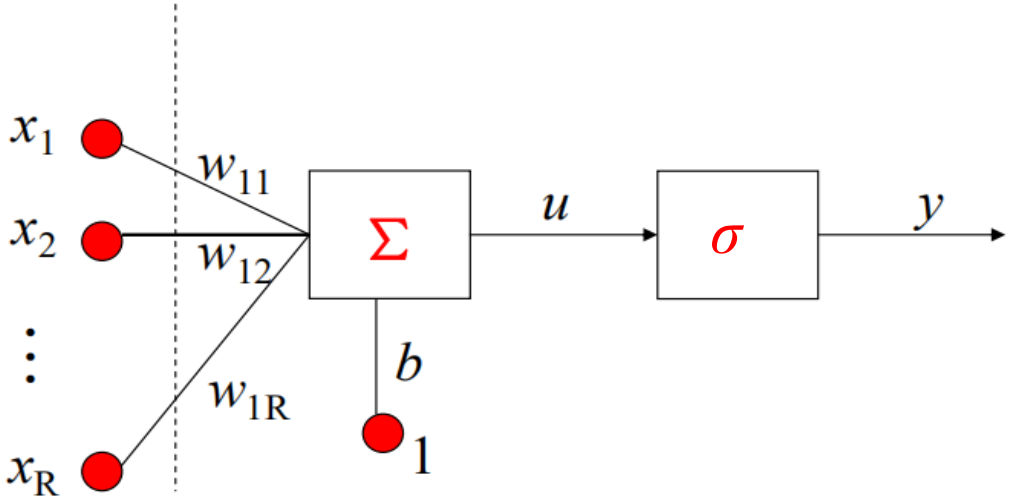
单输入单个神经元

$$y = f(wx + b)$$



多输入单个神经元

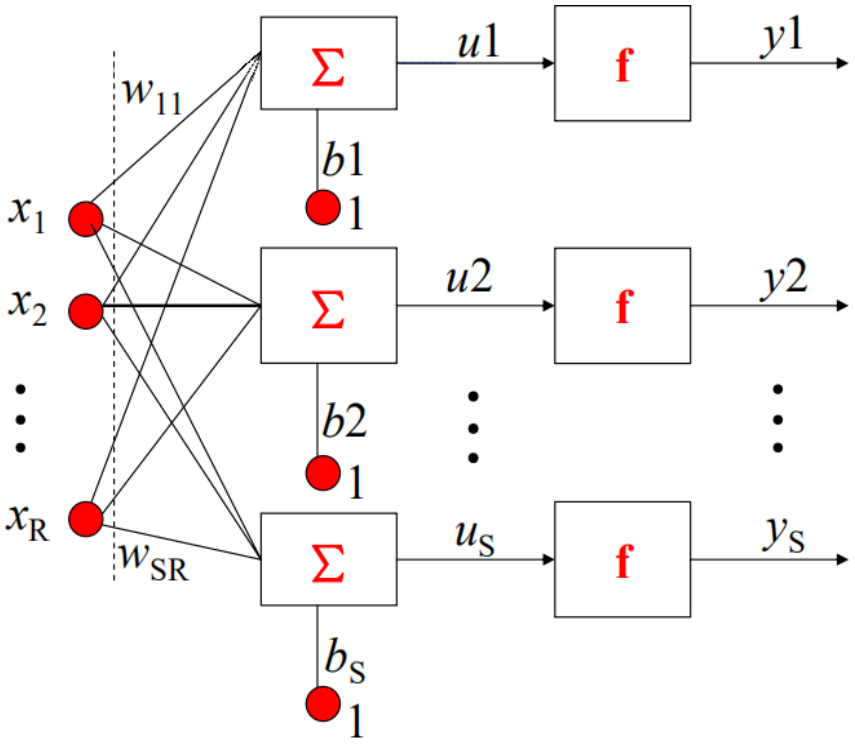
$$y = f(w^T x + b)$$



神经网络基本概念

- **神经元的层**：一般来说，有多个输入的单个神经元并不能满足实际应用的要求。在实际应用中需要有多个并行神经元。我们将这些可以并行操作的神经元组成的集合称为“层”

- 由S个神经元组成的单层网络；
- R个输入中的每一个均与每个神经元相连；
- 权值矩阵有S行R列



神经网络

1 神经网络基本概念

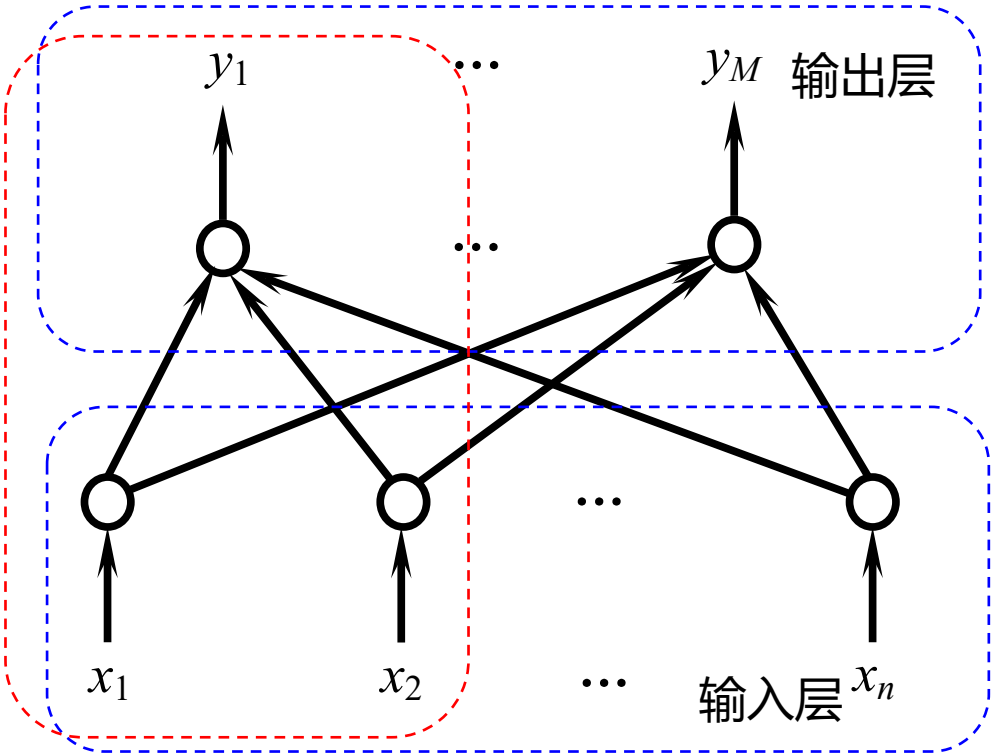
2 感知器与多层网络

3 BP神经网络

4 总结

感知器与多层网络

- 感知器：F．Rosenblatt于1957年提出，学习的目标是通过改变权值使神经网络由给定的输入得到给定的输出，或称阈值逻辑单元



感知器结构示意图

结构特点：

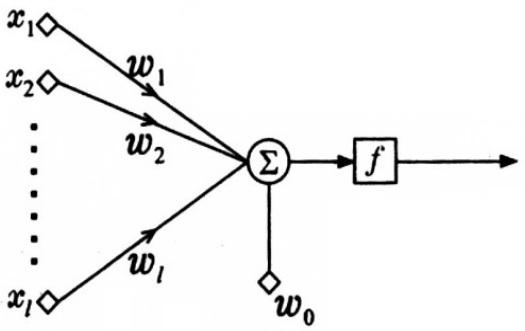
* 双层（输入层、输出层）；
* 两层单元之间为全互连；
* 连接权值可调

* 输出层神经元个数等于类别数（两类问题时输出层为一个神经元）

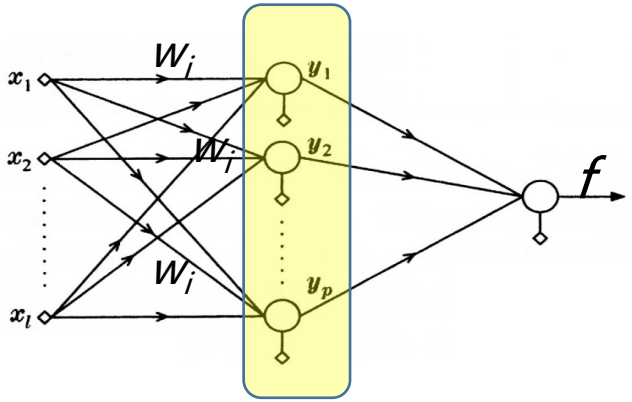
第九讲 神经网络

感知器与多层网络

- 感知器：F．Rosenblatt于1957年提出，学习的目标是通过改变权值使神经网络由给定的输入得到给定的输出。两层感知器的结构与单层感知器相比增加了一个隐藏层



单层感知器结构



两层感知器结构

提问：这个结构你们熟悉吗？想想logistic回归

感知器与多层网络

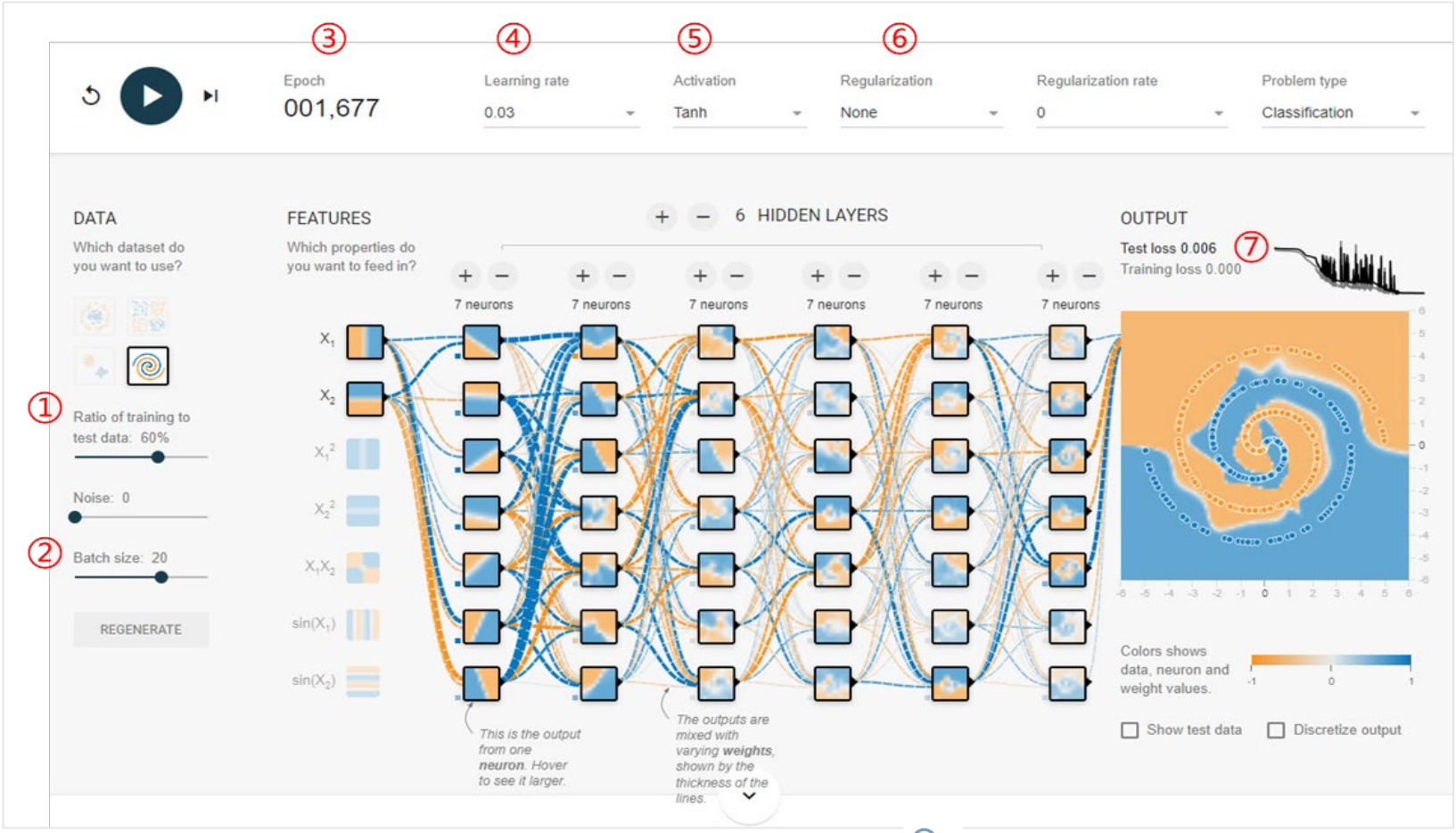
- 只要其隐藏层神经元的数量足够，它可以以任意精度来近似任何从一个定义在实数空间中的有界闭集函数。一个两层的神经网络可以模拟任何函数：

http://neuralnetworksanddeeplearning.com/chap4.html

第九讲 神经网络

感知器与多层网络

- 进一步理解多层感知器



多层神经网络 By ®Tensorflow

神经网络

1 神经网络基本概念

2 感知器与多层网络

3 BP神经网络

4 总结

第九讲 神经网络

BP神经网络

1. 神经网络的学习规则

- **Hebb学习规则**

由加拿大著名生理心理学家*Hebb*根据生物学中的条件反射机理，于**1949**年提出的神经元连接强度变化规则，属于无监督学习

- 受巴甫洛夫的条件反射实验的启发，Hebb认为在同一时间被激发的神经元间的联系会被强化。相反，如果两个神经元总是不能同步激发，那么它们间的联系将会越来越弱

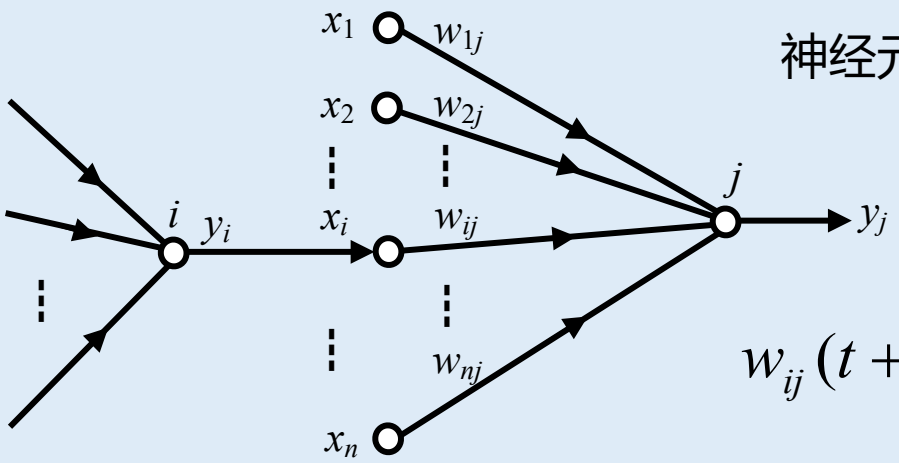
- 如果神经网络中某一神经元与另一直接与其相连的神经元同时处于兴奋状态，那么这两个神经元之间的连接强度应该加强

第九讲 神经网络

BP神经网络

1. 神经网络的学习规则

- **Hebb学习规则**



神经元间的连接

$$w_{ij}(t+1) = w_{ij}(t) + \eta[y_j(t)y_i(t)]$$

$w_{ij}(t+1)$：修正一次后的某一权值；

η：学习率，表示学习速率的比例常数；

$y_j(t)$，$y_i(t)$：分别表示t时刻第j个和第i个神经元的状态（输出）

- 如果y_i与y_j同时被激活，即y_i与y_j同时为正，那么w_{ij}将增大；
- 如果y_i被激活，而y_j处于抑制状态，即y_i为正y_j为负，那么w_{ij}将变小

1. 神经网络的学习规则

➢ Perceptron学习规则

1958年，美国学者***Frank Rosenblatt*** 首次提出感知器，感知器的学习规则是根据网络的输出误差对神经元的连接权值进行修正，属于有监督学习

设d_i为神经元i的期望输出，y_i为神经元i的实际输出，d_i-y_i为误差信号或学习信号，神经元i到神经元j的连接权为w_{ij}。在**Hebb**学习规则中引入监督信号，将**Hebb**学习规则公式中的yi换成神经元i期望目标输出d_i 与神经元i实际输出y_i之差，即为有监督*Perceptron*学习规则

➢ *Perceptron* 规则简单来讲就是：若神经元实际输出比期望输出大，则减少输入为正的连接的权重，增大所有输入为负的连接的权重。反之，则增大所有输入为正的连接权的权重，减少所有输入为负的连接权的权重

BP神经网络

1. 神经网络的学习规则

- **Perceptron学习规则**

(1) 选择一组初始权值w_{ij};

(2) 计算某一输入模式对应的实际输出与期望输出的误差;

(3) 更新权值，阈值可视为输入恒为（-1）的一个权值;

$$w_{ij}(t+1) = w_{ij}(t) + \eta[d_j - y_j(t)]x_i(t)$$

式中，η：学习率;

d_j，$y_j(t)$：第j个神经元的期望输出与实际输出;

$x_i(t)$：第j个神经元的状态，即第i个输入

(4) 返回(2)，直到对所有训练模式网络输出均能满足要求

1. 神经网络的学习规则

➢ δ学习规则与梯度下降法

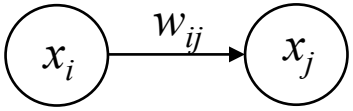
1986年，认知心理学家*McClelland* 和 *Rumelhart*在神经网络训练中引入了δ规则，称为连续感知器学习规则，以目标的负梯度方向对参数进行调整

$$w_{ij}(t+1) = w_{ij}(t) - \eta \triangledown E$$

其中梯度$\triangledown E$：

$$\sigma_j(t)x_i(t)$$

σ_j是误差在x_j上的梯度



δ规则也是BP神经网络的参数更新规则，有三个要点：

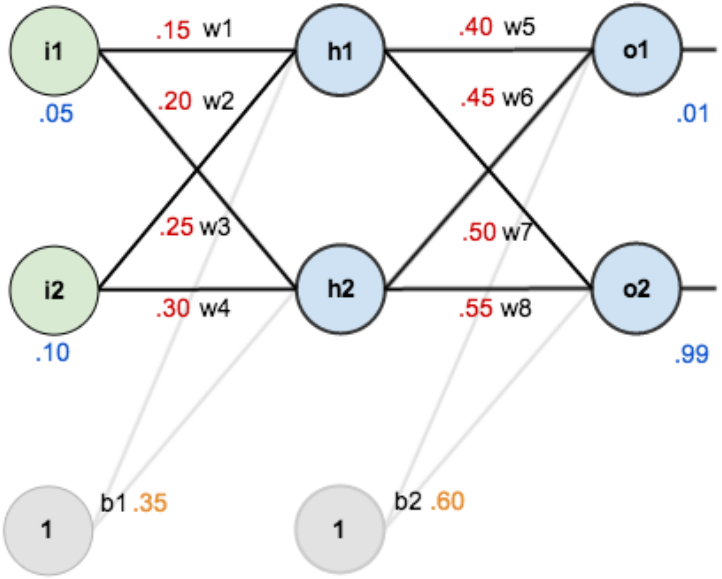
- 要能够计算神经元输出值与期望值之间的误差（定义损失函数）；
- 要能够计算梯度（即损失函数可导、激活函数可导）
- 由学习率η控制参数更新的速度

BP神经网络

2. 反向误差传播

- BP神经网络1986年由*McClelland* 和 *Rumelhart*为首的科学家提出的概念，是一种按照误差反向传播算法训练的多层感知器，是应用最广泛的神经网络模型之一。其核心就是BP算法（Back-Propagation Training Algorithm）

一个BP神经网络训练的例子



- 根据初始参数进行网络前向计算：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$
$$= 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

记得这是什么吗 $f(x) = \frac{1}{1+e^{-x}}$

$$out_{h2} = 0.596884378$$

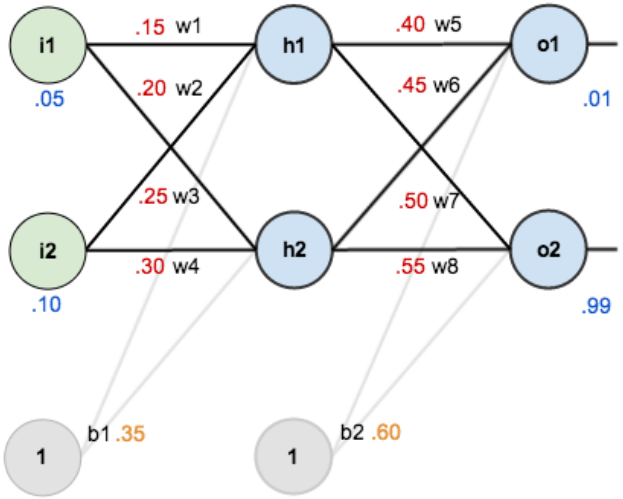
$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$
$$= 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.7513650$$

$$out_{o2} = 0.772928465$$

BP神经网络

2. 反向误差传播



- 定义损失函数：最小平方误差函数

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

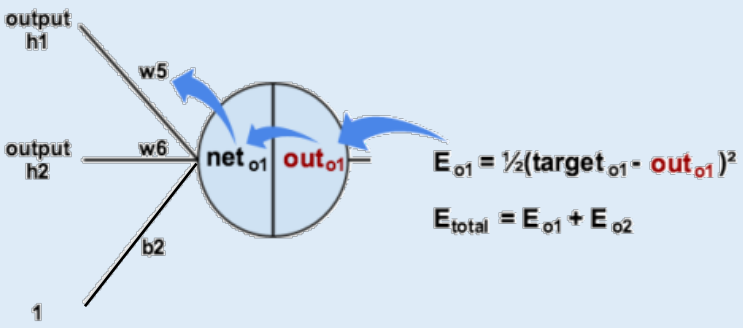
便于求导后消除系数

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{o2} = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

- 参数$w5$的更新过程：$\Delta w_j = -\eta \nabla E$



误差在参数$w5$方向上的梯度：

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

① ② ③

$$= 0.74136507 * 0.186815602 * 0.593269992$$

$$= 0.082167041$$

BP神经网络

2. 反向误差传播

- 参数$w5$的更新过程：$\Delta w_j = -\eta \nabla E$

① $\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$

$= -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$

②

sigmoid函数具有以下性质：$f'(x)=f(x)(1-f(x))$

$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$

$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$

③

$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$

⇒

$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$

BP神经网络

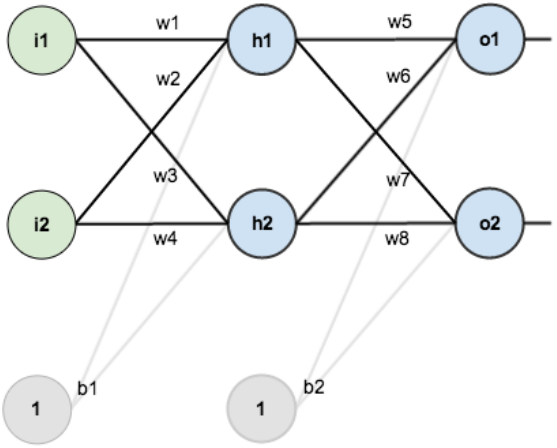
2. 反向误差传播

- 参数$w5$的更新过程：$\Delta w_j = -\eta \nabla E$

$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1}$$



δ学习规则在BP神经网络上的体现

BP神经网络

2. 反向误差传播

- 参数$w1$的更新过程： $\Delta w_j = -\eta \nabla E$

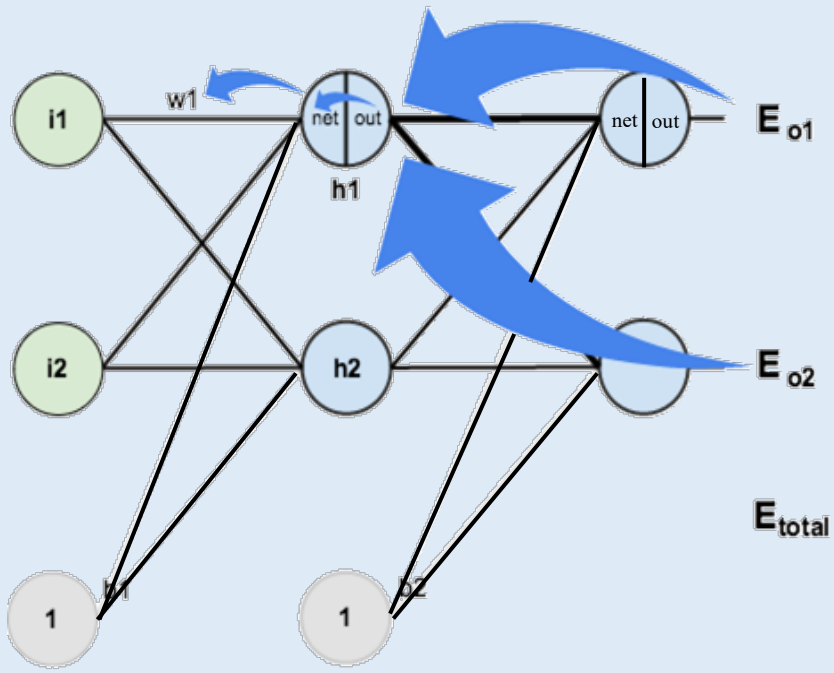
误差在参数$w1$方向上的梯度：

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

③ ④

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

① ②



① $\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

BP神经网络

2. 反向误差传播

- 参数$w1$的更新过程： $\Delta w_j = -\eta \nabla E$

① $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

② $$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

③ $$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

$f'(x)=f(x)(1-f(x))$

④ $net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

⇒

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

BP神经网络

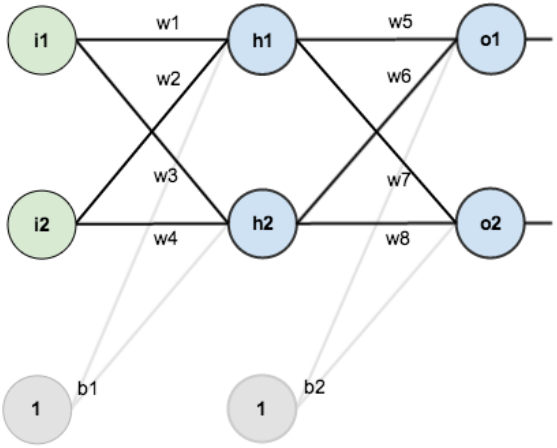
2. 反向误差传播

- 参数$w1$的更新过程： $\Delta w_j = -\eta \nabla E$

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}}\right) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \delta_o * w_{ho}\right) * out_{h1}(1 - out_{h1}) * i_1$$

$$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1$$ （不失一般性写为：$\delta_j o_i$）

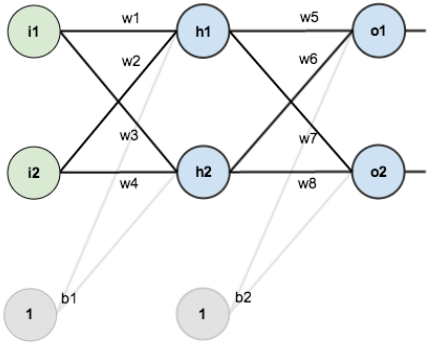


δ学习规则在BP神经网络上的体现

BP神经网络

2. 反向误差传播

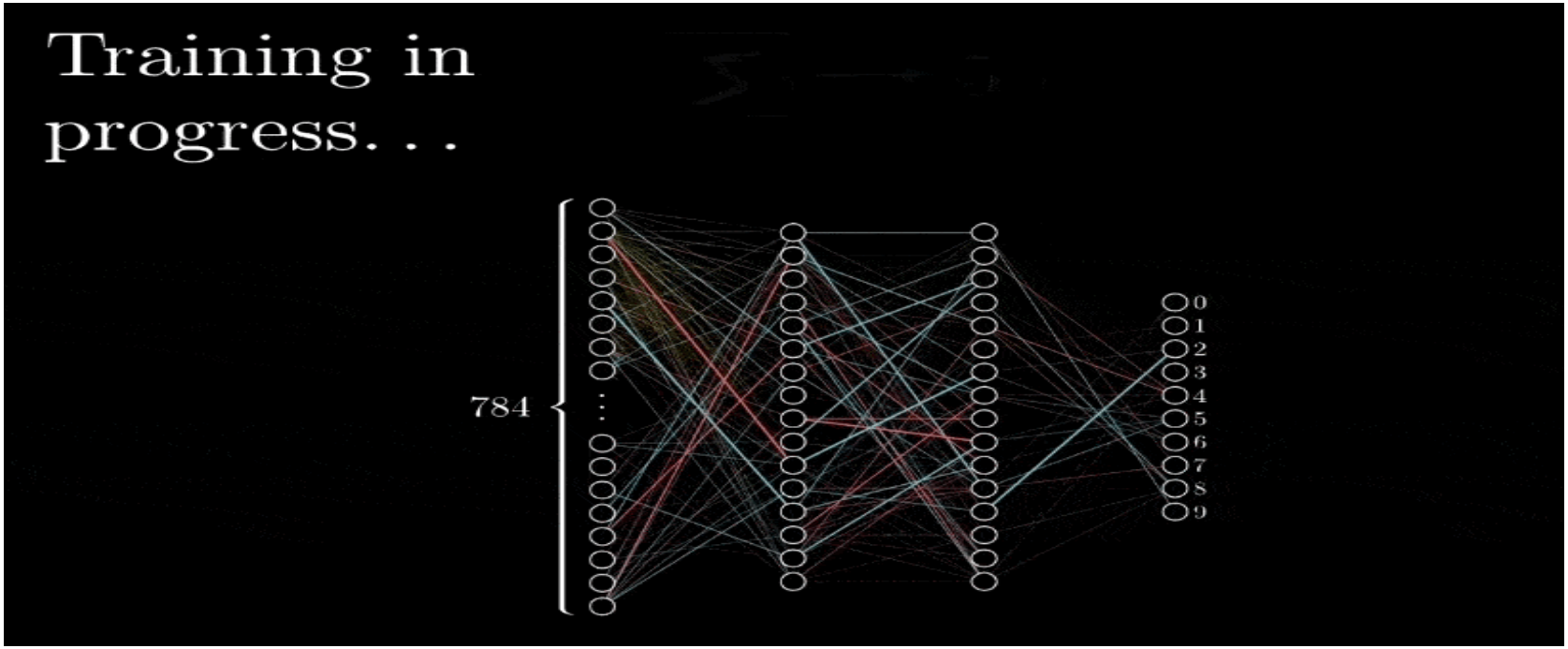
《西瓜书》公式5-10、5-15



- 比较一下δ_o与δ_h

$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

$$\delta_{h1} = (\sum_{o} \delta_o * w_{ho}) * out_{h1}(1 - out_{h1})$$



BP神经网络

2. 反向误差传播

BP算法程序实现

(1)初始化；

(2)输入训练样本对$X \Leftarrow X^p$、$d \Leftarrow d^p$计算各层输出；

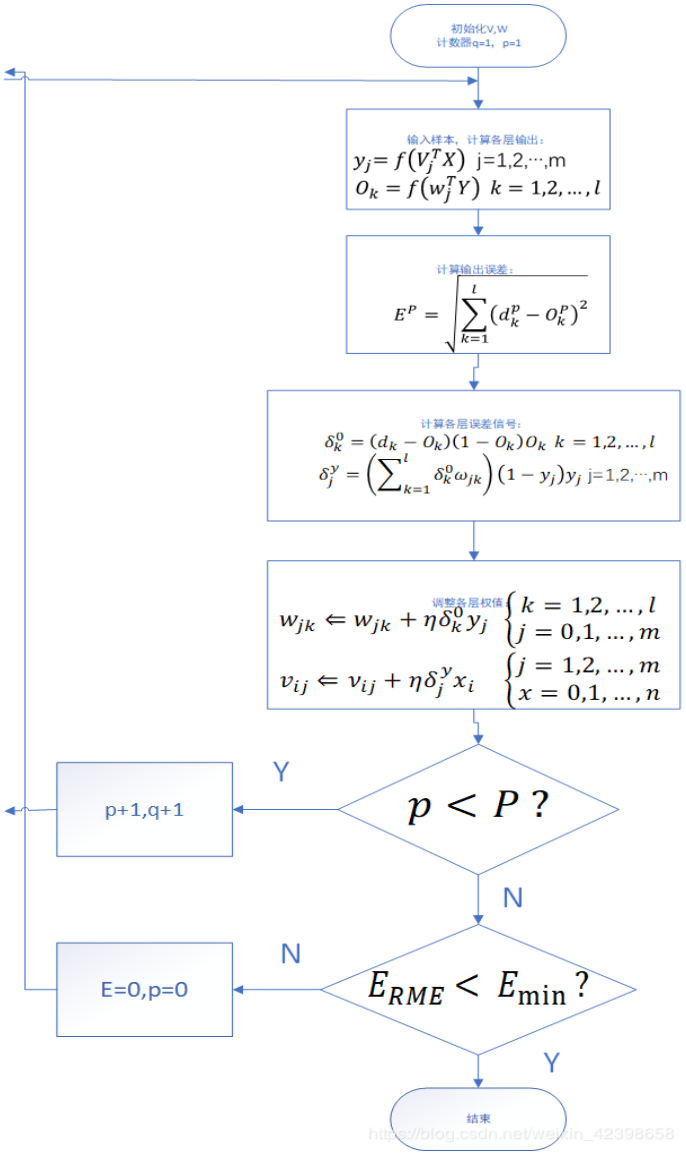
(3)计算网络输出误差；

(4)计算各层误差信号；

(5)调整各层权值；

(6)检查是否对所有样本完成一次轮训；

(7)检查网络总误差是否达到精度要求。



BP神经网络

3. 损失函数

- 前面的推导中，我们采用了均方误差作为损失函数。在实际工作中，神经网络可以考虑这些损失函数：

均方误差损失也是一种比较常见的损失函数： $MSE = \frac{1}{n}\sum_{i}^{n}(\hat{y}_i - y_i)^2$

交叉熵损失函数： $C = -\sum_{k} y_k \log \hat{y}_k$

- 交叉熵损失函数经常用于分类问题中，特别是在神经网络做分类问题时，也经常使用交叉熵作为损失函数，此外，由于交叉熵涉及到计算每个类别的概率，所以交叉熵几乎每次都和sigmoid(或softmax)函数一起出现

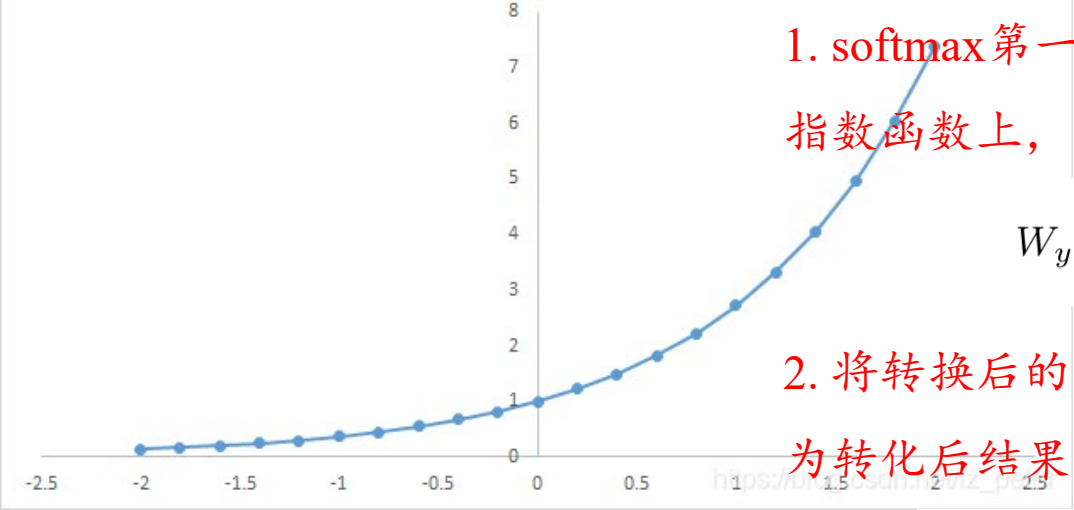
$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

$$= -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

MSE+sigmoid

3. 损失函数

- 感知器的输出层：输出值既可以是一个连续的无范围约束的数值（回归问题），也可以是一个离散的范围被限制（一般在0至1之间）的数值（分类问题）。所以针对不同的问题，对输出层的设计即激活函数的设计应不同。一般情况下，在神经网络中，回归问题选择恒等函数作为激活函数，分类问题选择softmax函数作为激活函数



1. softmax第一步将模型的预测结果转化到指数函数上，这样保证了概率的非负性

$$W_y.x = \sum_{i=1}^{d} W_{yi}x_i = f_y = Z$$

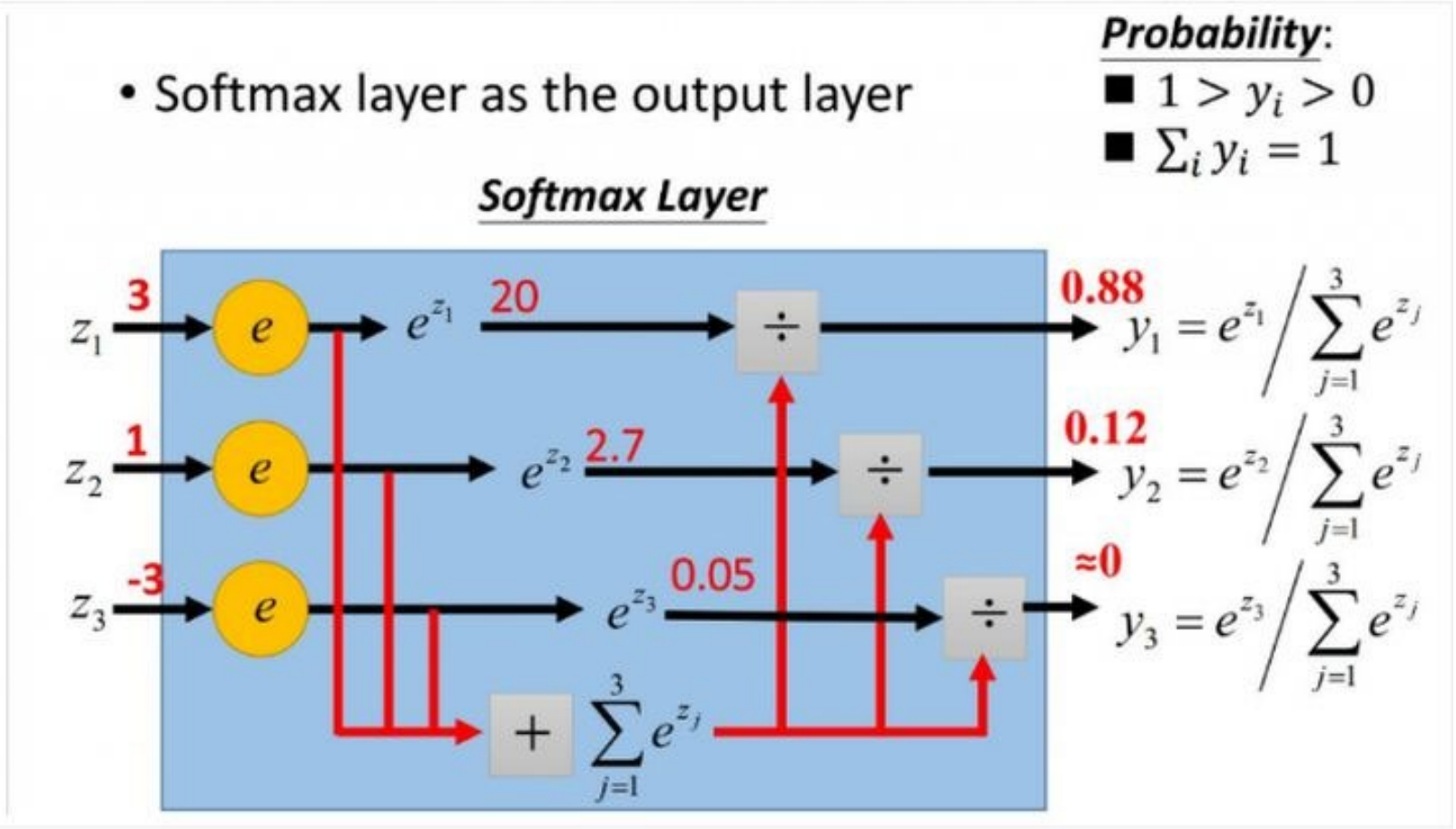
2. 将转换后的结果进行归一化处理，理解为转化后结果占总数的百分比

$$p(y|x) = \frac{\exp(f_y)}{\sum_{c=1}^{C}\exp(f_c)} = softmax(f)_y$$

BP神经网络

3. 损失函数

- Softmax函数：$a_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$



BP神经网络

3. 损失函数

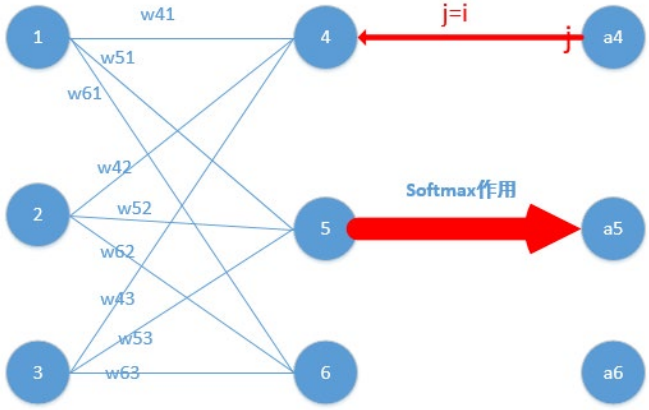
- Softmax的导函数：计算 $\frac{\partial out_{o1}}{\partial net_{o1}}$

if $j=i$:

$$\frac{\partial a_j}{\partial z_i}=\frac{\partial}{\partial z_i}\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right)$$

$$=\frac{(e^{z_j})'\cdot\sum_k e^{z_k}-e^{z_j}\cdot e^{z_j}}{\left(\sum_k e^{z_k}\right)^2}$$

$$=\frac{e^{z_j}}{\sum_k e^{z_k}}-\frac{e^{z_j}}{\sum_k e^{z_k}}\cdot\frac{e^{z_j}}{\sum_k e^{z_k}}=a_j(1-a_j)$$

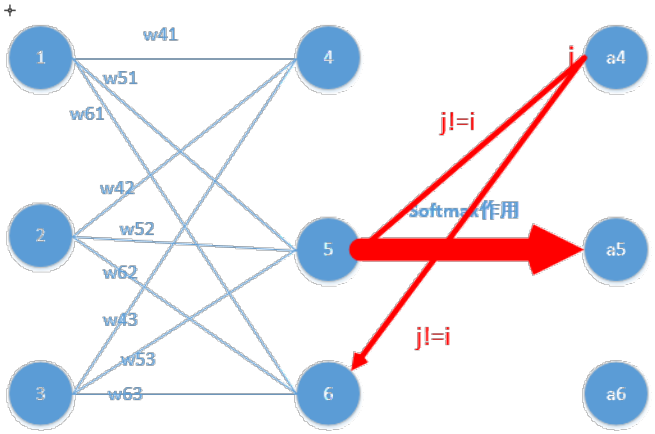


if $j\neq i$:

$$\frac{\partial a_j}{\partial z_i}=\frac{\partial}{\partial z_i}\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right)$$

$$=\frac{0\cdot\sum_k e^{z_k}-e^{z_j}\cdot e^{z_i}}{\left(\sum_k e^{z_k}\right)^2}$$

$$=-\frac{e^{z_j}}{\sum_k e^{z_k}}\cdot\frac{e^{z_i}}{\sum_k e^{z_k}}=-a_j a_i$$



BP神经网络

3. 损失函数

- 交叉熵的导函数：计算 $\frac{\partial E_{total}}{\partial out_{o1}}$

$C = -\sum_k y_k \, lna_j$ 设训练数据的真实输出y_j个为1，其它均为0，则：

$$Loss = -y_j lna_j = -lna_j$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial Loss}{\partial a_j} = -\frac{1}{a_j}$$

- 因此在输出层采用交叉熵损失函数+softmax激活函数后，输出层节点的δ:

$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \begin{cases} a_i \\ a_j - 1 \end{cases}$$

- MSE+sigmoid的输出层梯度与激活函数的导数成正比。如果起始输出值比较大，也即激活函数的导数比较小，那么整个梯度幅度更新幅度都比较小，收敛时间很长；交叉熵+softmax则没有这个问题，收敛速度快

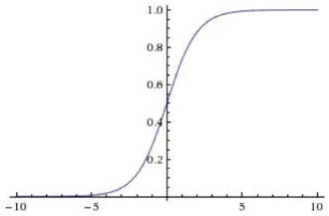
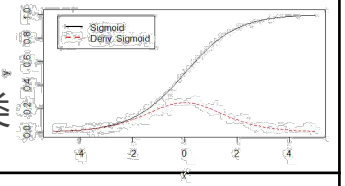
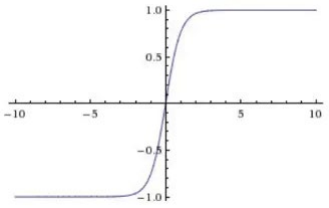
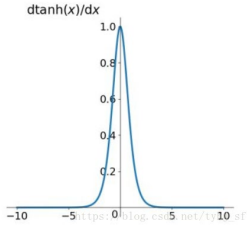
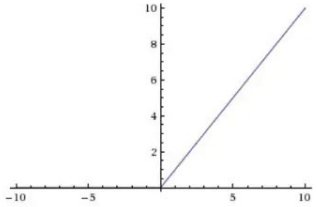
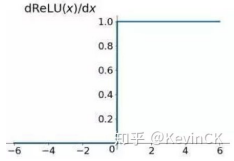
3. 激活函数

- 给神经网络引入非线性元素，使神经网络可以完成非线性映射。如果不使用激活函数，无论神经网络有多少层，都只是线性组合而已
- 常见的激活函数及其导函数：

激活函数	形式	导数形式
Sigmoid	$f(x)=\frac{1}{1+e^{-x}}$	$f'(x)=f(x)(1-f(x))$
Tanh	$f(x)=tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$	$f'(x)=1-(f(z))^2$
ReLU	$f(x)=max(0,x)=\begin{cases}0 & x\leq 0\\ x & x>0\end{cases}$	$f'(x)=\begin{cases}0 & x\leq 0\\ 1 & x>0\end{cases}$
Leaky ReLU	$f(x)=max(0.001x,x)=\begin{cases}0.001x & x\leq 0\\ x & x>0\end{cases}a$	$f(x)=max(0.001x,x)=\begin{cases}0.001 & x\leq 0\\ 1 & x>0\end{cases}$
PReLU	$f(x)=max(\alpha x,x)=\begin{cases}\alpha x & x\leq 0\\ x & x>0\end{cases}$	$f(x)=max(\alpha x,x)=\begin{cases}\alpha & x\leq 0\\ 1 & x>0\end{cases}$
RReLU	PReLU中的 α 随机取值	
ELU	$f(x)=\begin{cases}x & x\geq 0\\ \alpha(e^x-1) & x<0\end{cases}$	$f(x)=\begin{cases}1 & x\geq 0\\ \alpha e^x & x<0\end{cases}$
Maxout	$f(x)=max(w_1^Tx+b_1,w_2^Tx+b_2)$	$f(x)=max(w_1,w_2)$

3. 激活函数

➢ 几个经典的激活函数的比较

 sigmoid	• 激活函数计算量大，反向传播求误差梯度时，求导涉及幂运算 • Sigmoid 的输出不是0均值的，很容易就会出现梯度弥散甚至梯度消失的情况，从而无法完成深层网络的训练 • 所有数据映射成了（0,1）之间的数适合二分类 
 tanh	• 是0均值的，在实际中tanh会比sigmoid收敛速度快 • 仍然存在梯度弥散的问题 
 Relu	• 正方向上，梯度不变，解决了梯度消失问题 (在正区间) • 计算速度非常快，只需要判断输入是否大于0，收敛速度非常快 • 如果输入是负值，那么神经元输出为0 ，造成神经元死亡 

4. 学习率

- 学习率$\eta \in (0,1)$控制着算法每一轮迭代中的更新步长。若太大则容易振荡，太小则收敛速度又会过慢。有时为了加快收敛速度，要加入前一次的修正量，可增加惯性系数；或者对输出层和隐藏层的设置不同学习率

$$\Delta w_{ij}(t) = -\eta \delta_j O_i + \alpha \Delta w_{ij}(t-1)$$

- Batch Size定义：一次训练所选取的样本数。Batch Size的大小影响模型的优化程度和速度。在没有使用Batch Size之前，这意味着网络在训练时把所有的样本输入网络中，然后计算它们的梯度进行反向传播，由于在计算梯度时使用了整个数据库，所以计算得到的梯度方向更为准确。但在这情况下，计算得到不同梯度值差别巨大，难以使用一个全局的学习率，网络很难收敛

5. 神经网络的过拟合

- 由于多层感知器强大的表示能力，BP神经网络经常遭遇过拟合。其训练误差持续降低，但测试误差却可能上升。有两种策略常用来缓解BP网络的过拟合：

- 第一种策略是“早停”(early stopping)：将数据分成训练集和验证集。训练集用来计算梯度、更新连接权和阈值。验证集用来估计误差。若训练集误差降低但验证集误差升高，则停止训练；同时返回具有最小验证集误差的参数
- 第二种策略是“正则化”（regularization），在损失函数中增加描述网络复杂度的部分，如L2正则化

$$E = \lambda \frac{1}{m} \sum_{k=1}^{m} E_k + (1-\lambda) \sum_i w_i^2$$

复习：logistic回归的正则化

神经网络

1 神经网络基本概念

2 感知器与多层网络

3 BP神经网络

4 总结

总结

（1）非线性的映射能力

该能力主要表现在复杂的权值上，而权值的变化就是BP的学习过程，因此可以存储大量的输入输出模式映射关系，而无需给出给出具体的映射方程。现实中我们经常遇到这一类问题就是，有大量的输入输出的数据，却无法得到一个闭合的数学表达式来表达他们，此时BP神经网络就很有用处了，通过大量数据的训练，最后会拟合出数据输入输出的非线性关系，这一类问题有共同的特点：1.无法得到解析解，2.缺乏专家经验 3.能够表示和转化模式识别或非线性问题，对于这样性质的问题，BP具有很大的优势

特征学习和表示学习的能力超强

总结

（2） 泛化能力

BP网络训练过程（学习过程）就是权值调整过程，它从数据中提取非线性映射关系存储在权值矩阵中，在其后的工作阶段，当向网络输入训练时未曾见过的非样本数据时，网络也能完成由输入空间到输出空间的正确映射，这种能力称为多层感知器的泛化能力

（3）容错能力

BP网络的魅力还在于，允许输入样本中带有较大误差甚至个别错误。因为输入输出是通过大量的权值进行决定的，反映正确规律的知识来自全样本，个别样本的误差对整体影响不大，因此具有很好的容错能力

总结

- 神经网络发展的趋势：越来越深（deep learning）

今天深度学习已经取得了很多的成功，但是有一个很大的问题，就是**理论基础不清楚**。第一，我们要有逐层的处理；第二，我们要有特征的内部变换；第三，我们要有足够的模型复杂度。这三件事情是我们认为深度神经网络为什么能够成功的比较关键的原因

——周志华《深度学习为什么深》