



模式识别

主讲：崔林艳&邹征夏

单位：宇航学院

开课专业：飞行器控制与信息工程

第七章 经典神经网络分类器

CONTENTS PAGE

7.1 人工神经元

7.2 感知机

7.3 BP神经网络

7.4 自组织神经网络

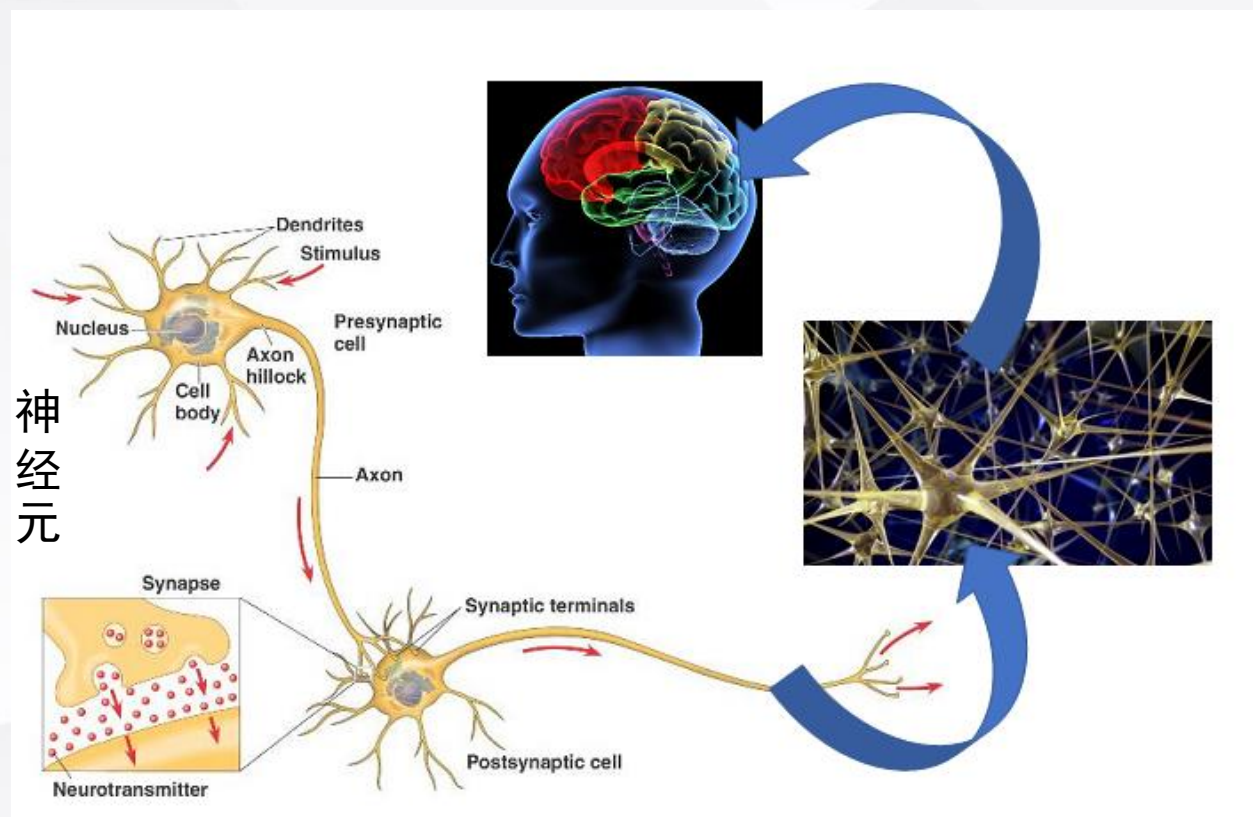
7.1 人工神经元

7.1.1 人工神经元

7.1.2 人工神经元数学模型

7.1 人工神经元

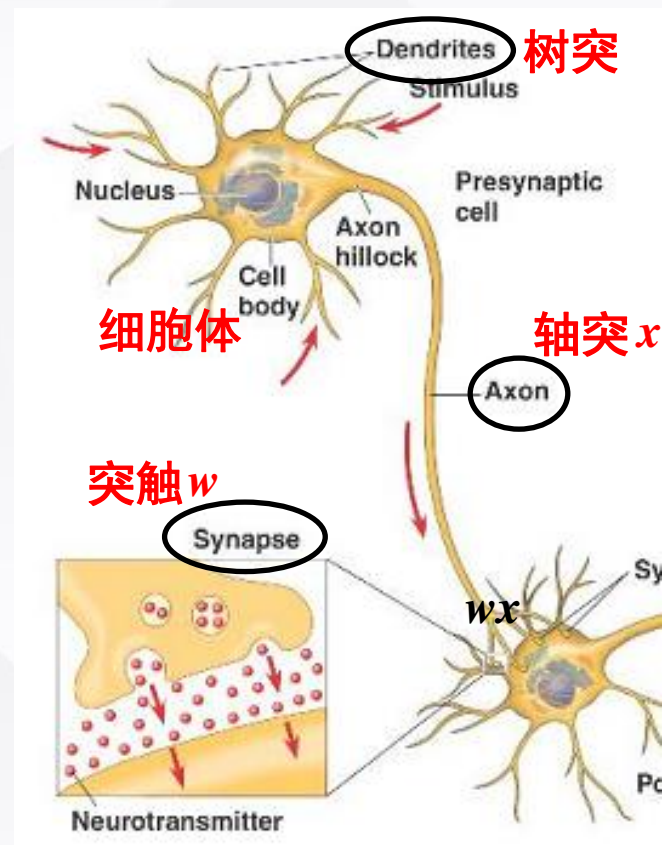
人类大脑有860亿个神经元，几乎所有的神经元都是相互联系在一起，组成复杂网络。



7.1 人工神经元

7.1.1 生物神经元组成

- **细胞体 (cell body)**: 神经细胞的主体, 是神经细胞进行信息加工的主要场所, 可认为是神经元。
- **树突 (dendrites)**: 细胞体 (神经元) 外围的大量微小分支, 负责从外界获得输入信号, 然后将输出信号传给它唯一的轴突 (axon)。
- **轴突 (axon)**: 细胞的输出装置, 通过突触 (Synapse) 将信号 x 传输给其他神经细胞;
- **突触 (Synapse)**: 一个神经元的轴突与另一个神经元的树突相连接的部位。假设以 w 表示, 则传到下一个神经元树突处的信号变成了 wx 。



其中：突触强弱(参数 w)是可学的，它控制了一个神经元对另一个神经元影响的大小和方向（正负）。

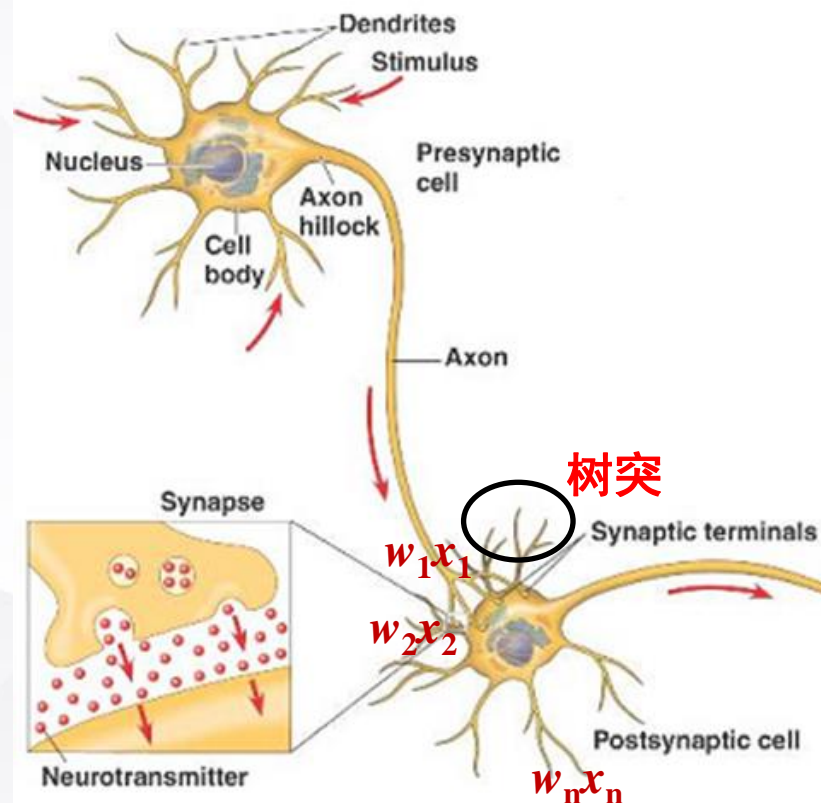
7.1 人工神经元

7.1.1 生物神经元组成

- 树突接收到的信号 ($w x$) ,
传递到神经元(细胞体)内部,
与其他树突传递过来的信号
($w_2 x_2, \dots, w_n x_n$) 进行加和:

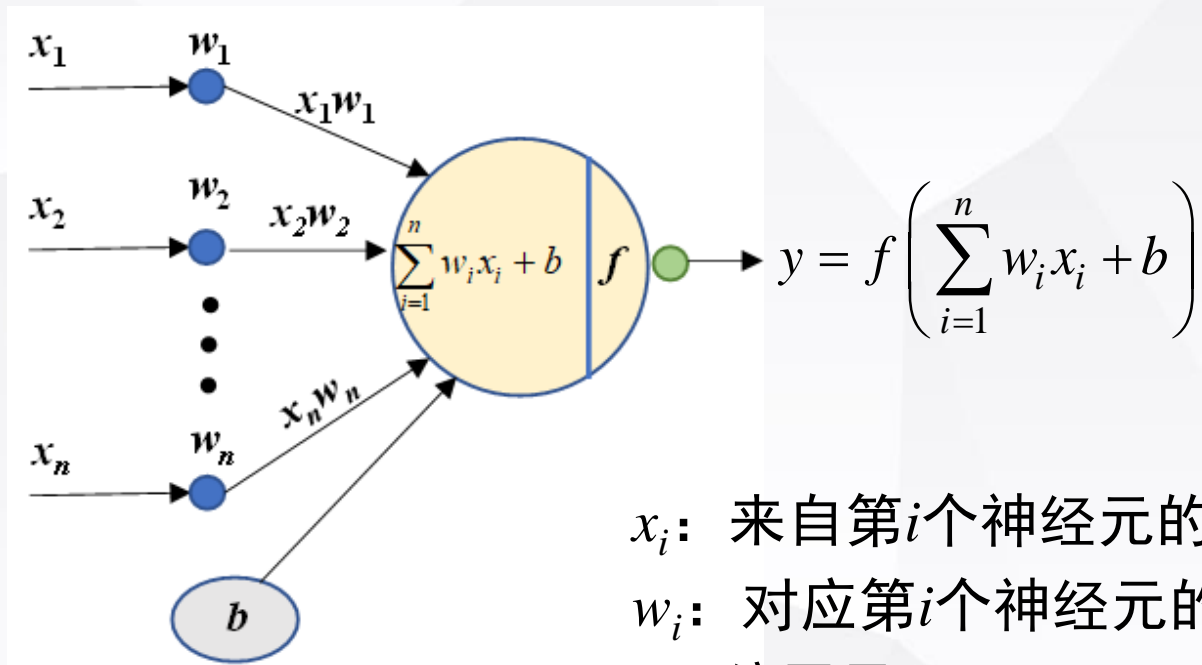
$$w_1 x_1 + w_2 x_2 + \dots + w_n x_n;$$

- 神经元被激活: 当 ($w_1 x_1 + w_2 x_2 + \dots + w_n x_n$) > 某一个固定
阈值 θ , 神经元会被激活, 输出
信号 y , 传递给树突。



7.1 人工神经元

7.1.2 人工神经元数学模型



x_i : 来自第 i 个神经元的输入

w_i : 对应第 i 个神经元的连接权重

b : 偏置量

f : 激活函数（是否激活神经元的函数）

y : 神经元输出

激活函数(activation function)：指的是否激活神经元的函数。在人工神经网络中，神经元节点的激活函数定义了对神经元输出的映射。

7.2 感知机

7.2.1 感知机结构

7.2.2 感知机模型求解

7.2.3 感知机参数学习基本原理

7.2.4 感知机参数学习详细过程

7.2.5 感知机算例

7.2.6 多层感知机

7.2 感知机

7.2.1 感知机结构

最简单的前馈式人工神经网络。

- 输入：可接收多个输入

$$(x_1, x_2, \dots, x_n \mid x_i \in \mathbb{R})$$

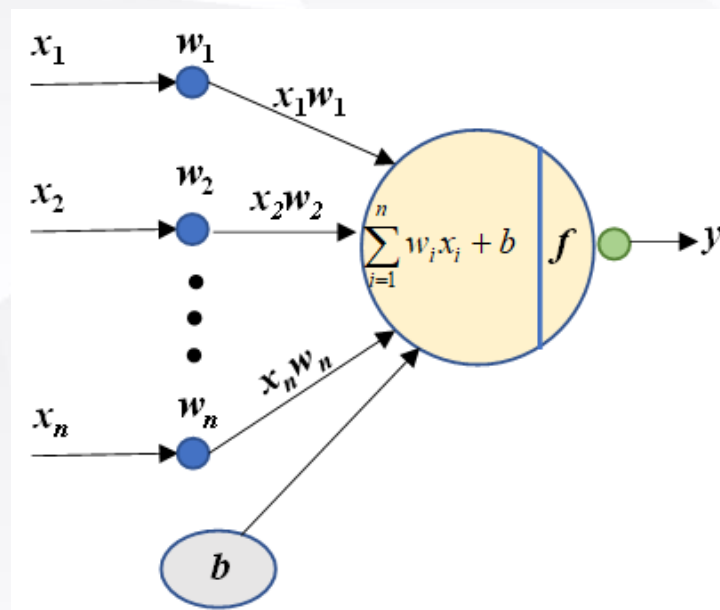
- 输入对应权值： $w_i \in \mathbb{R}$

- 偏置项： $b \in \mathbb{R}$

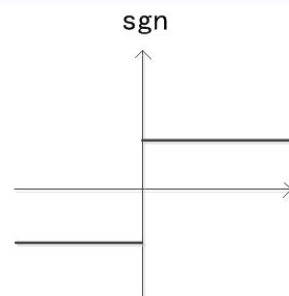
- 激活函数：

通常选择**阶跃函数**作为激活函数

- 输出 y : $y = \text{sgn}\left(\sum_{i=1}^n w_i x_i + b\right)$
 $y \in \{+1, -1\}$



具有一层结构的感知机

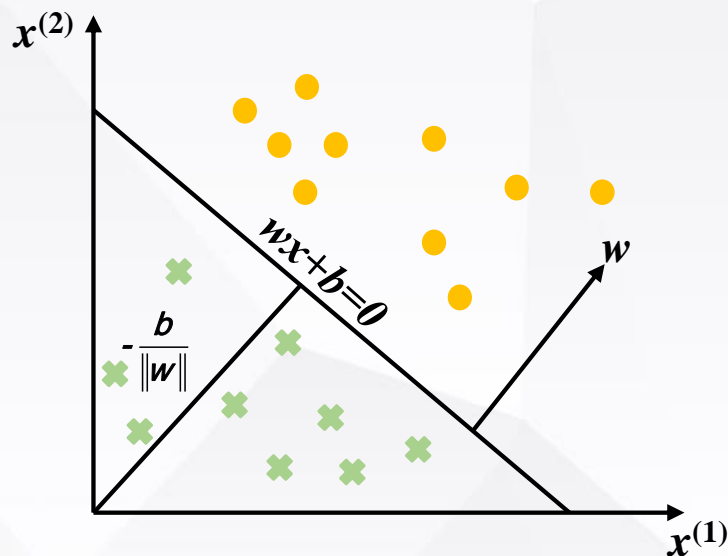


7.2 感知机

7.2.2 感知机模型求解

- 假设训练数据集线性可分，感知机学习的目标是获得一个能够将训练数据集中正、负实例完全分开的分离超平面 S 。

以二维为例：



w : 超平面的法向量;
 b : 超平面的截距。

- 感知机预测：利用学习得到的感知机模型，对于新的输入实例给出其对应的输出类别为1或者-1（阶跃激活函数）。

7.2 感知机

7.2.3 感知机参数学习基本原理

- 定义输入空间 R^n 中任一点 x_0 到分离超平面 S 的距离为：

$$\frac{1}{\|w\|} |wx_0 + b|$$

$\|w\|$ ： w 的 L_2 范数

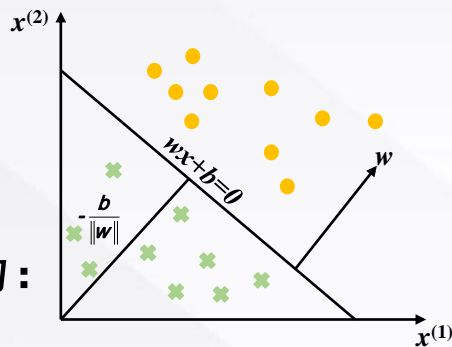
- 对于误分类的数据 (x_i, y_i) ，如下成立：

$$-y_i (wx_i + b) > 0$$

思考为什么？

误分类点 x_i 到分离超平面 S 的距离为：

$$\frac{1}{\|w\|} |wx_i + b|$$



7.2 感知器

- 给定 N 个样本的训练数据集：

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\} \quad y_i \in \{+1, -1\}$$

- 假设分离超平面 S 的**误分类点集合为 M** ，则所有误分类点到超平面 S 的总距离为：

$$\frac{1}{\|w\|} \sum_{x_i \in M} |wx_i + b|$$

- 如果不考虑 $\|w\|$ ，感知机学习的算法是通过学习找到一组参数 (w^*, b^*) ，使得以下损失函数 $L(w, b)$ 最小：

$$\min_{w, b} L(w, b) = - \sum_{x_i \in M} y_i (w_i x_i + b)$$

误分类的数据：

$$-y_i (wx_i + b) > 0$$

- 感知机学习算法是**误分类驱动的在线学习算法**，采用**随机梯度下降法**。任意选取一个具有参数 (w_0, b_0) 的分离超平面，然后利用梯度下降法不断极小化损失函数。经过多次迭代，训练得到感知器参数 (w^*, b^*)

7.2 感知机

7.2.4 感知机参数学习详细过程

算法：两类感知器参数学习算法

输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ，最大迭代次数 T ，学习率 η

输出： (w^*, b^*)

步骤 1：将参数 (w, b) 均初始化为 0；

步骤 2：训练数据集中选择数据 (x_i, y_i) ；

如果 $y_i(w x_i + b) \leq 0$ ，则 (x_i, y_i) 为误分类样本。利用误分类样本，进行参数更新：

随机梯度下降法

$$w \leftarrow w + \eta y_i x_i$$

$$b \leftarrow b + \eta y_i$$

步骤 3：重复步骤 2，直到达到最大迭代次数 T 。

$$\min_{w, b} L(w, b) = - \sum_{x_i \in M} y_i (w x_i + b)$$

误分类的数据：

$$-y_i (w x_i + b) > 0$$

η ：学习步长，也称为学习率。用于控制每一步调整权值的幅度

L 对 w 的偏导数： $-y_i x_i$

L 对 b 的偏导数： $-y_i$

7.2 感知机

7.2.5 感知机算例

- 某个训练数据集，其正实例点为：

$$x_1 = [3 \ 3]^T \quad y_1 = 1$$

$$x_2 = [4 \ 3]^T \quad y_2 = 1$$

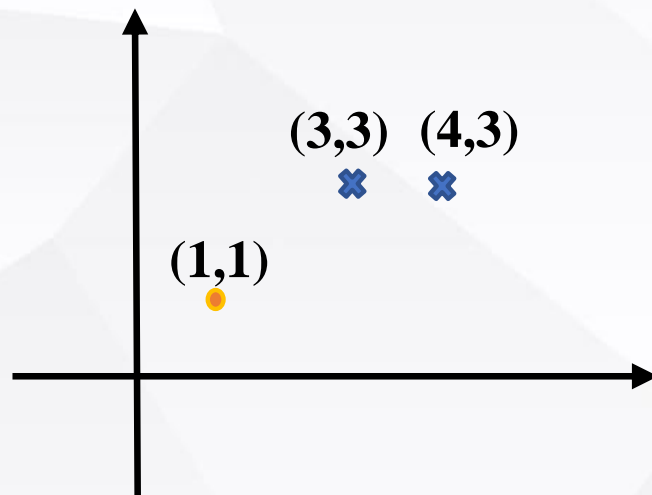
- 负实例点为：

$$x_3 = [1 \ 1]^T \quad y_3 = -1$$

- 试用感知机学习算法求感知机模型：

$$g(x) = f(w \cdot x + b)$$

其中： $w = [w^{(1)} \ w^{(2)}]^T$

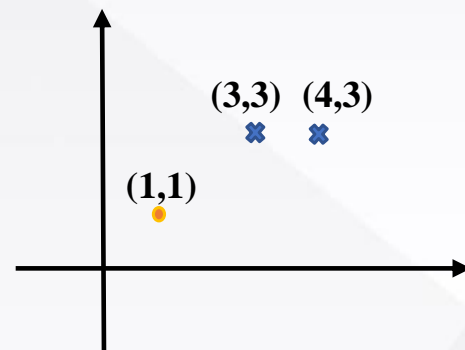


7.2 感知机

【求解过程】

➤ 构建最优化问题： $\min_{w,b} L(w,b) = - \sum_{x_i \in M} y_i (wx + b)$

➤ 根据感知机学习算法求解 w, b 。设学习率 $\eta = 1$




① 给出权值的初始值 $w_0 = [0 \ 0]^T, b_0 = 0$

② 对于 $x_1 = [3 \ 3]^T$ ，得到 $y_1 (w_0 \cdot x_1 + b_0) = 0$ 。则判定 x_1 未被正确分类，利用该点更新 w, b

$$w_1 = w_0 + y_1 x_1 = [3 \ 3]^T$$

$$b_1 = b_0 + y_1 = 1$$

$$\begin{aligned} w &\leftarrow w + \eta y_i x_i \\ b &\leftarrow b + \eta y_i \end{aligned}$$



得到线性模型：

$$w_1 \cdot x + b_1 = 3x^{(1)} + 3x^{(2)} + 1$$

7.2 感知机

【求解过程】

- ③ 对于 x_1 和 x_2 ，得到 $y_1(w_1 \cdot x_1 + b_1) > 0, y_2(w_2 \cdot x_2 + b_2) > 0$ 。则判定 x_1 和 x_2 被正确分类，此时不更新 w, b
- ④ 对于 $x_3 = [1 \ 1]^T$ ，得到 $y_3(w_1 \cdot x_3 + b_1) < 0$ 。则判定 x_3 被错误分类，利用该点更新 w, b

$$w_2 = w_1 + y_3 x_3 = [2 \ 2]^T \quad b_2 = b_1 + y_3 = 0$$

得到线性模型： $w_2 \cdot x + b_2 = 2x^{(1)} + 2x^{(2)}$

- ⑤ 不断输入实例点，判定该实例点是否为误分类点，若是，则利用该点更新权值，直至：

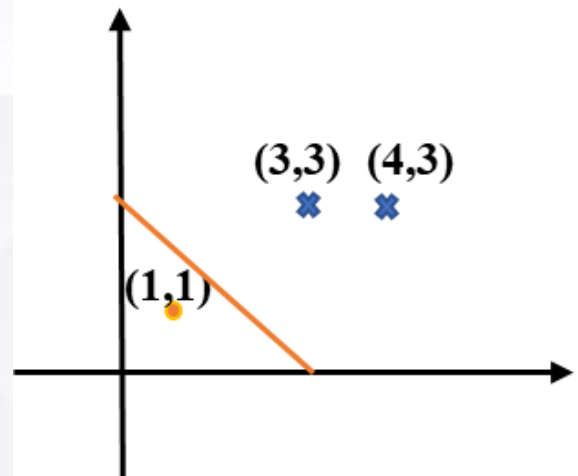
$$w_7 = [1 \ 1]^T \quad b_7 = -3$$

对所有数据点 $y_i(w_7 \cdot x_i + b_7) > 0$ ，没有误分类点，损失函数达到极小。

7.2 感知机

此时得到的分离超平面为: $x^{(1)} + x^{(2)} - 3 = 0$

感知机模型为: $g(x) = f(x^{(1)} + x^{(2)} - 3)$



迭代次数	误分类点	$wx + b$
0		0
1	x_1	$3x^{(1)} + 3x^{(2)} + 1$
2	x_3	$2x^{(1)} + 2x^{(2)}$
3	x_3	$x^{(1)} + x^{(2)} - 1$
4	x_3	-2
5	x_1	$3x^{(1)} + 3x^{(2)} - 1$
6	x_3	$2x^{(1)} + 2x^{(2)} - 2$
7	x_3	$x^{(1)} + x^{(2)} - 3$
8	0	$x^{(1)} + x^{(2)} - 3$

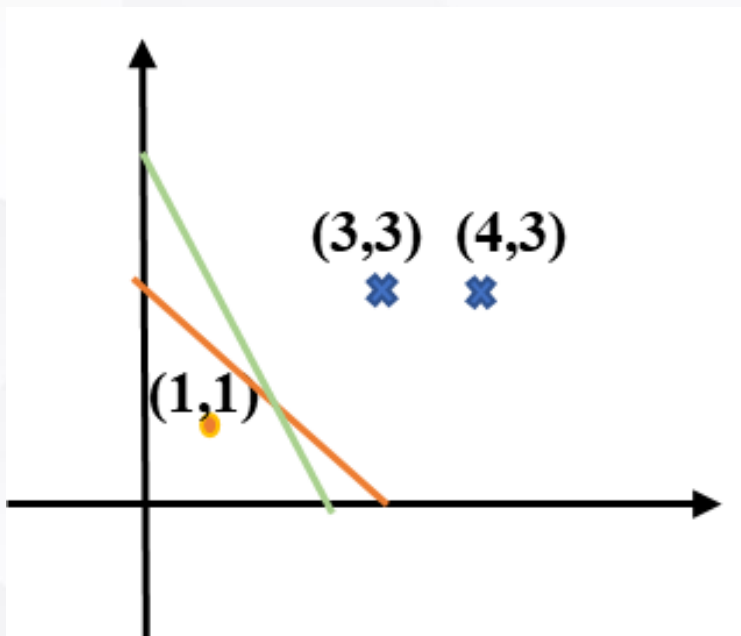
7.2 感知机

- 在计算中误分类点先后取 $x_1, x_3, x_3, x_3, x_1, x_3, x_3$, 分离超平面（红色实线）为：

$$x^{(1)} + x^{(2)} - 3 = 0$$

- 在计算中误分类点先后取 $x_1, x_3, x_3, x_3, x_2, x_3, x_3, x_3, x_1, x_3, x_3$, 分离超平面（绿色实线）为：

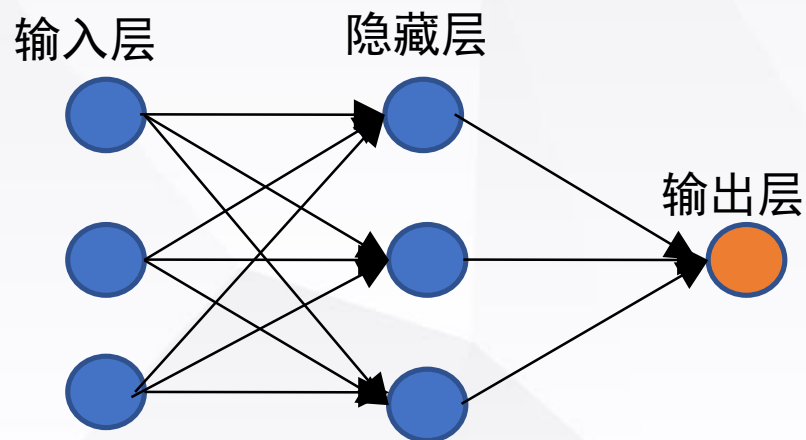
$$2x^{(1)} + x^{(2)} - 5 = 0$$



7.2 感知机

7.2.6 多层感知机

- 单层感知机可以很好的解决线性分类问题，不适用于解决非线性问题，例如简单的异或问题。通过将多个神经元分层组合，即构建多层感知机可以很好的解决非线性可分问题，通常将多层感知机称之为神经网络。



多个感知机组成的两层结构(层与层之间全连接)

- 20世纪60年代，学者发现感知机学习算法不能直接应用到这种多层感知机模型参数学习中。

7.2 感知机

➤ 【原因解释】

感知机学习算法以训练样本被错分的程度作为目标函数，训练过程中每次出现错分，就对权值进行修改，使其朝着目标函数相对于权值的负梯度方向更新，直到目标函数取得极小值，此时没有训练样本被错分。

$$\begin{array}{c} \boxed{\nabla_w L(w, b) = - \sum_{x_i \in M} y_i x_i \quad \nabla_b L(w, b) = - \sum_{x_i \in M} y_i} \\ \downarrow \\ \boxed{w \leftarrow w + \eta y_i x_i \quad b \leftarrow b + \eta y_i} \\ \downarrow \\ \min_{w, b} L(w, b) = - \sum_{x_i \in M} y_i (w_i x_i + b) \end{array}$$

感知机学习算法是一种单层网络的学习算法。在多层网络中，它只能改变最后层的权值（通过判断训练样本被错分的情况，代入如上权值更新公式进行权值更新）。

7.3 BP网络

7.3.1 BP网络结构

7.3.2 BP网络参数设置

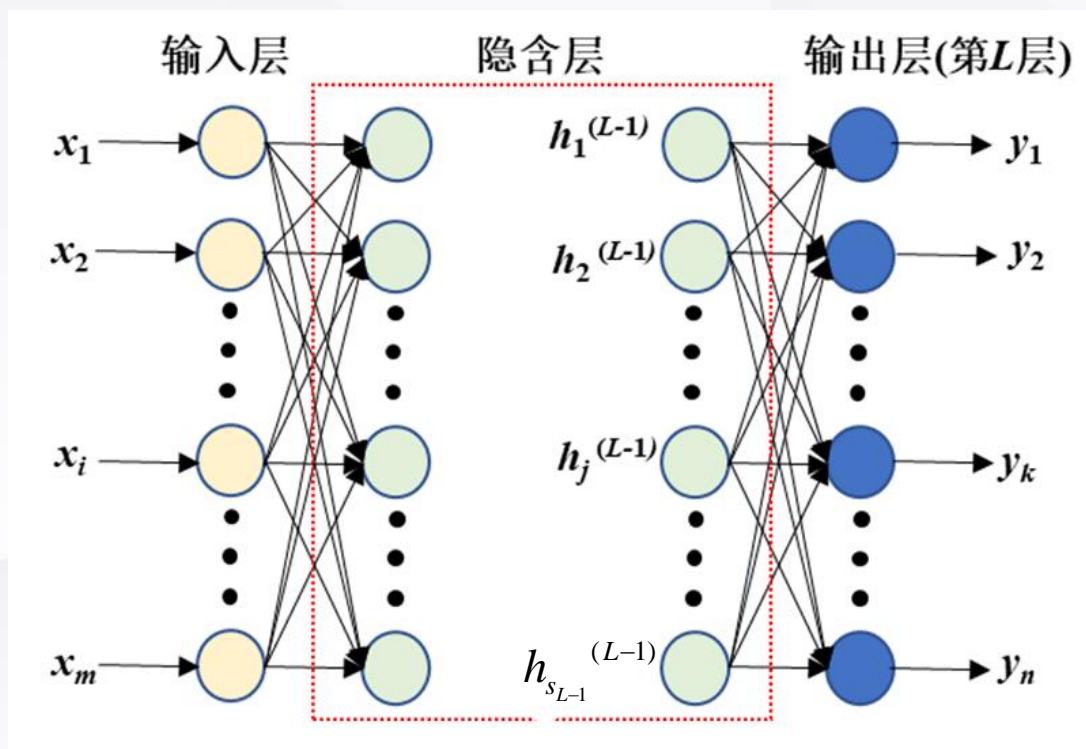
7.3.3 BP推导

7.3.4 BP具体实现步骤

7.3 BP网络

7.3.1 BP网络结构

- BP算法：误差反向传播算法（Error Back Propagation Training），系统解决了多层神经网络隐含层神经元连接权值学习的问题。
- BP网络：基于BP算法进行误差校正的多层前馈网络。用Sigmoid激活函数替代了阶跃函数。



信号前向传播

误差反向传播

7.3 BP网络

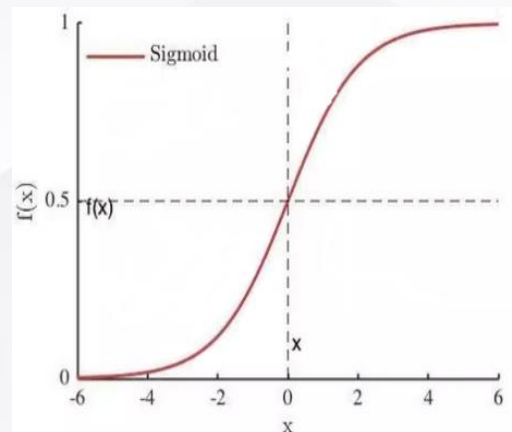
7.3.1 BP网络结构

➤ Sigmoid激活函数

① 具有指数函数形式：

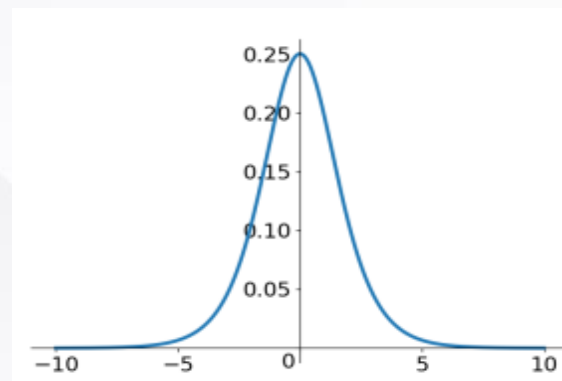
$$f(x) = \frac{1}{1 + e^{-x}}$$

将一个实数输入映射到 $[0,1]$ 范围。



② 该激活函数导数为：

$$\begin{aligned} f'(x) &= \frac{e^{-x}}{(1 + e^{-x})^2} = \frac{1}{1 + e^{-x}} \left(1 - \frac{1}{1 + e^{-x}}\right) \\ &= f(x)(1 - f(x)) \end{aligned}$$



在定义域内处处可导，且两侧导数逐渐趋近于0

7.3 BP网络

7.3.2 BP网络参数设置

➤ 输入向量为：

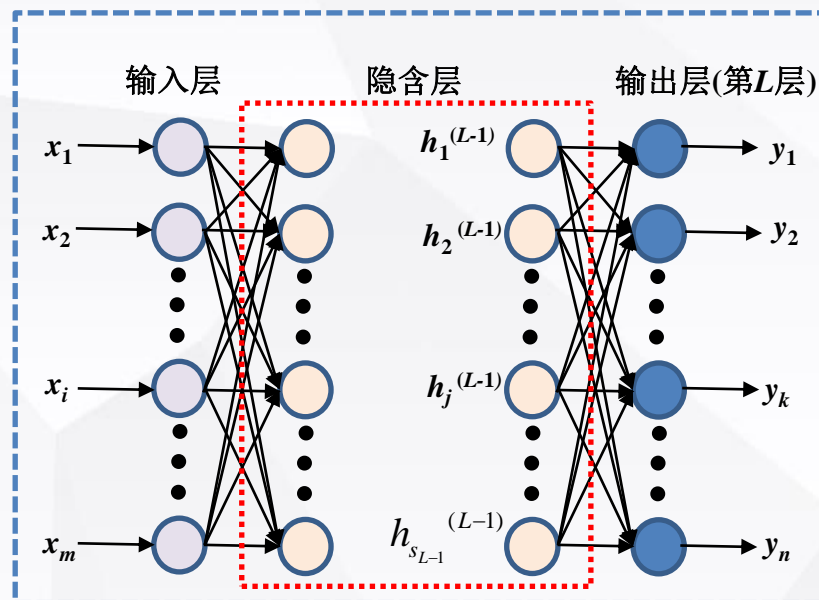
$$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_i \ \dots \ x_m], i = 1, 2, \dots, m$$

➤ 输出向量为：

$$\mathbf{y} = [y_1 \ y_2 \ \dots \ y_k \ \dots \ y_n], k = 1, 2, \dots, n$$

➤ 第 l 隐含层各神经元的输出为：

$$\mathbf{h}^{(l)} = [h_1^{(l)} \ h_2^{(l)} \ \dots \ h_j^{(l)} \ \dots \ h_{s_l}^{(l)}], j = 1, 2, \dots, s_l$$



s_l : 第 l 层神经元的个数

7.3 BP网络

7.3.2 BP网络参数设置

➤ 第 l 隐含层各神经元的输出：

$$h_i^{(l)} = f\left(\text{net}_i^{(l)}\right)$$



$$\text{net}_i^{(l)} = \sum_{j=1}^{s_{l-1}} W_{ij}^{(l)} h_j^{(l-1)} + b_i^{(l)}$$

$\text{net}_i^{(l)}$ ：第 l 层第 i 个神经元的输入

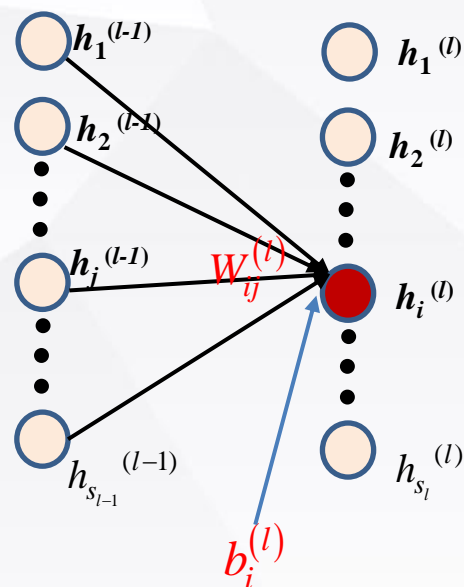
$f(\bullet)$ ：神经元激活函数，BP网络通常采用Sigmoid激活函数

$W_{ij}^{(l)}$ ：从第 $l-1$ 层第 j 个神经元与第 l 层第 i 个神经元之间的连接权值

$b_i^{(l)}$ ：第 l 层第 i 个神经元的偏置

第 $l-1$ 隐含层

第 l 隐含层



7.3 BP网络

7.3.3 BP推导

- 假设有 M 个训练样本

$$\{(x(1), d(1)), (x(2), d(2)), \dots, (x(M), d(M))\}$$

$d(i)$ 是对应样本 $x(i)$ 的期望输出

- **BP算法**：通过最优化各层神经元的输入权值以及偏置量，使得BP网络输出尽可能接近期望输出。采用**批量样本更新算法**，即对于给定的 M 个训练样本，定义**误差函数**为所有样本的误差均值：

$$E = \frac{1}{M} \sum_{i=1}^M E(i)$$

$E(i)$ 为样本 $x(i)$ 的训练误差，采用均方误差损失函数：

$$E(i) = \frac{1}{2} \sum_{k=1}^n (d_k(i) - y_k(i))^2 \quad n \text{代表输出的维度}$$

7.3 BP网络

7.3.3 BP推导

- 将 $E(i)$ 代入 E ，得到给定 M 个样本的误差均值：

$$E = \frac{1}{2M} \sum_{i=1}^M \sum_{k=1}^n (d_k(i) - y_k(i))^2$$

- 在采用BP算法每次迭代时，按照如下方式对权值和偏置量进行更新：

$$W_{ij}^{(l)} \leftarrow W_{ij}^{(l)} - \eta \frac{\partial E}{\partial W_{ij}^{(l)}} \quad b_i^{(l)} \leftarrow b_i^{(l)} - \eta \frac{\partial E}{\partial b_i^{(l)}}$$

η ：为学习率，取值范围(0,1)

BP网络权值和偏置更新的关键是计算误差损失函数 E 关于权值和偏置的偏导数。

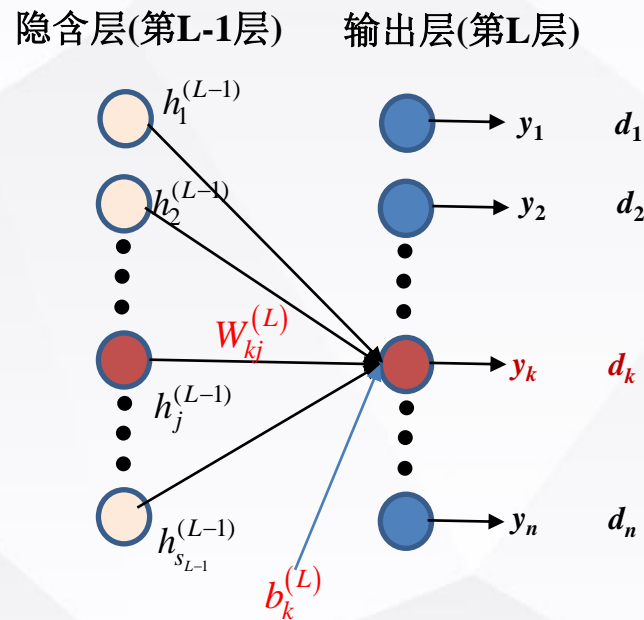
7.3 BP网络

$$E(i) = \frac{1}{2} \sum_{k=1}^n (d_k(i) - y_k(i))^2$$

7.3.3 BP推导

➤ 对于单个训练样本 $x(i)$ ，隐含层-》输出层 $E(i)$ 关于权值的偏导数为：

$$\begin{aligned} \frac{\partial E(i)}{\partial W_{kj}^{(L)}} &= \frac{\partial}{\partial W_{kj}^{(L)}} \left(\frac{1}{2} \sum_{k_1=1}^n (d_{k_1}(i) - y_{k_1}(i))^2 \right) \\ &= \frac{\partial}{\partial W_{kj}^{(L)}} \left(\frac{1}{2} (d_k(i) - y_k(i))^2 \right) \quad \text{仅与输出层第 } k \text{ 个神经元有关} \\ &= -(d_k(i) - y_k(i)) \frac{\partial y_k(i)}{\partial W_{kj}^{(L)}} \\ &= -(d_k(i) - y_k(i)) \frac{\partial y_k(i)}{\partial net_k^{(L)}} \frac{\partial net_k^{(L)}}{\partial W_{kj}^{(L)}} \quad \text{链式法则} \\ &= -(d_k(i) - y_k(i)) f'(x) \Big|_{x=net_k^{(L)}} \frac{\partial net_k^{(L)}}{\partial W_{kj}^{(L)}} \\ &= -(d_k(i) - y_k(i)) f'(x) \Big|_{x=net_k^{(L)}} h_j^{(L-1)} \end{aligned}$$



$$y_k^{(L)} = f(net_k^{(L)})$$

$$net_k^{(L)} = \sum_{j=1}^{s_{L-1}} W_{kj}^{(L)} h_j^{(L-1)} + b_k^{(L)}$$

7.3 BP网络

7.3.3 BP推导

- 同理，对于单个训练样本 $x(i)$ ，隐含层-》输出层 $E(i)$ 关于偏置量的偏导数为：

$$\frac{\partial E(i)}{\partial b_k^{(L)}} = -\left(d_k(i) - y_k(i)\right) f'(x) \Big|_{x=net_k^{(L)}}$$

令 $\delta_k^{(L)} = -\left(d_k(i) - y_k(i)\right) f'(x) \Big|_{x=net_k^{(L)}}$

若 $f(x)'=1$ ，则 $\delta_k^{(L)} = -\left(d_k(i) - y_k(i)\right)$ 为网络输出与期望输出之差

此时，隐含层-》输出层 $E(i)$ 关于权值和偏置量的偏导数可简化为：

$$\frac{\partial E(i)}{\partial W_{kj}^{(L)}} = \delta_k^{(L)} h_j^{(L-1)} \qquad \frac{\partial E(i)}{\partial b_k^{(L)}} = \delta_k^{(L)}$$

7.3 BP网络

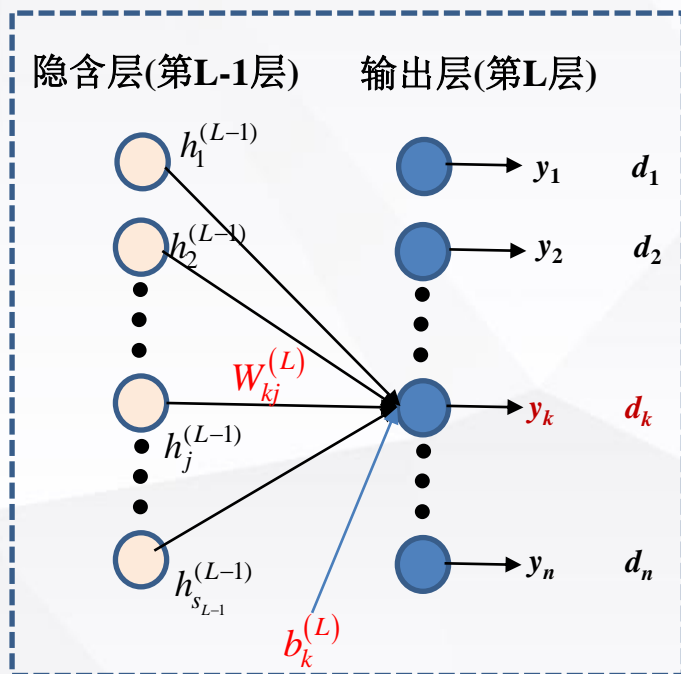
7.3.3 BP推导

- 对于单个训练样本 $x(i)$ ，隐含层→输出层 $E(i)$ 关于权值和偏置量的偏导数的直观解释：

$$\frac{\partial E(i)}{\partial W_{kj}^{(L)}} = \underbrace{\delta_k^{(L)}}_{\text{误差}} \underbrace{h_j^{(L-1)}}_{\text{输入}}$$

$$\frac{\partial E(i)}{\partial b_k^{(L)}} = \delta_k^{(L)}$$

误差反向传播



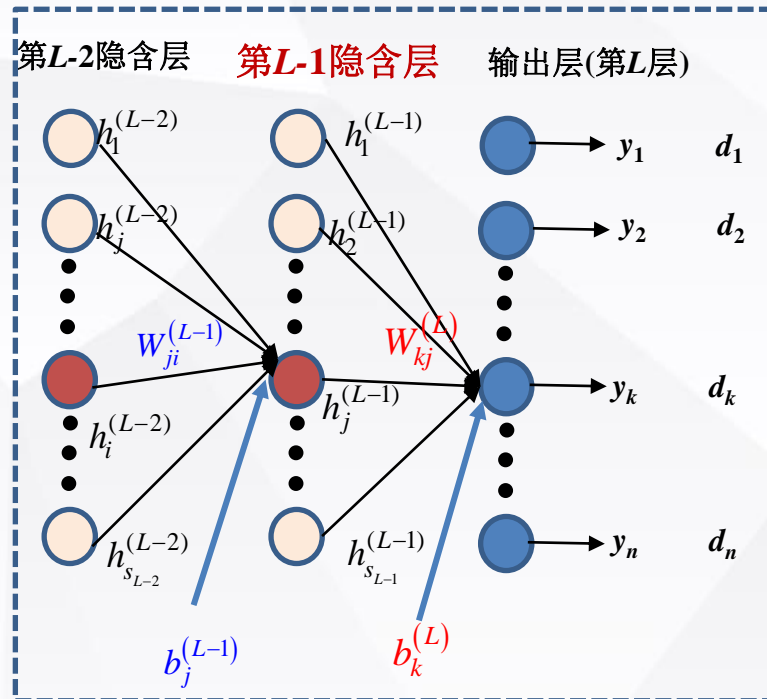
输出层（第L层）中第 k 个神经元关于与其连接的第L-1隐含层中第 j 个神经元的权值的偏导数为：**误差***第L-1隐含层中第 j 个神经元到第L层输出层中第 k 个神经元的**输入**

7.3 BP网络

7.3.3 BP推导

- 对于隐含层 $L-1$ 层，该层 $E(i)$ 关于权值的偏导数为：

$$\begin{aligned}
 \frac{\partial E(i)}{\partial W_{ji}^{(L-1)}} &= \frac{\partial}{\partial W_{ji}^{(L-1)}} \left(\frac{1}{2} \sum_{k=1}^n (d_k(i) - y_k(i))^2 \right) \\
 &= - \sum_{k=1}^n (d_k(i) - y_k(i)) \frac{\partial y_k(i)}{\partial W_{ji}^{(L-1)}} \quad \text{链式法则} \\
 &= - \sum_{k=1}^n (d_k(i) - y_k(i)) \frac{\partial f(net_k^L)}{\partial W_{ji}^{(L-1)}} \quad \text{链式法则} \\
 &= - \sum_{k=1}^n (d_k(i) - y_k(i)) \frac{\partial f(net_k^L)}{\partial net_k^{(L)}} \frac{\partial net_k^{(L)}}{\partial W_{ji}^{(L-1)}} \\
 &= - \sum_{k=1}^n (d_k(i) - y_k(i)) f'(x) \Big|_{x=net_k^{(L)}} \frac{\partial net_k^{(L)}}{\partial W_{ji}^{(L-1)}}
 \end{aligned}$$



$$\begin{aligned}
 net_k^{(L)} &= \sum_{j=1}^{s_{L-1}} W_{kj}^{(L)} h_j^{(L-1)} + b_k^{(L)} \\
 &= \sum_{j=1}^{s_{L-1}} W_{kj}^{(L)} f \left(\sum_{i=1}^{s_{L-2}} W_{ji}^{(L-1)} h_i^{(L-2)} + b_j^{(L-1)} \right) + b_k^{(L)} \\
 &= \sum_{j=1}^{s_{L-1}} W_{kj}^{(L)} f \left(net_j^{(L-1)} \right) + b_k^{(L)}
 \end{aligned}$$

进一步得到

7.3 BP网络

7.3.3 BP推导

➤ 隐含层 $L-1$ 层 $E(i)$ 关于权值的偏导数进一步表示为：

$$\begin{aligned}\frac{\partial E(i)}{\partial W_{ji}^{(L-1)}} &= -\sum_{k=1}^n (d_k(i) - y_k(i)) f(x)' \Big|_{x=net_k^{(L)}} \frac{\partial net_k^{(L)}}{\partial W_{ji}^{(L-1)}} \quad \text{链式法则} \quad net_k^{(L)} = \sum_{j=1}^{S_{L-1}} W_{kj}^{(L)} f(net_j^{(L-1)}) + b_k^{(L)} \\&= -\sum_{k=1}^n (d_k(i) - y_k(i)) f(x)' \Big|_{x=net_k^{(L)}} \frac{\partial net_k^{(L)}}{\partial f(net_j^{(L-1)})} \frac{\partial f(net_j^{(L-1)})}{\partial net_j^{(L-1)}} \frac{\partial net_j^{(L-1)}}{\partial W_{ji}^{(L-1)}} \\&= -\sum_{k=1}^n (d_k(i) - y_k(i)) f(x)' \Big|_{x=net_k^{(L)}} W_{kj}^{(L)} f(x)' \Big|_{x=net_j^{(L-1)}} h_i^{(L-2)}\end{aligned}$$

其中：

$$\frac{\partial net_k^{(L)}}{\partial f(net_j^{(L-1)})} = W_{kj}^{(L)}$$

$$\frac{\partial f(net_j^{(L-1)})}{\partial net_j^{(L-1)}} = f(x)' \Big|_{x=net_j^{(L-1)}}$$

$$\frac{\partial net_j^{(L-1)}}{\partial W_{ji}^{(L-1)}} = h_i^{(L-2)}$$

结合
定义

$$net_j^{(L-1)} = \sum_{i=1}^{S_{L-2}} W_{ji}^{(L-1)} h_i^{(L-2)} + b_j^{(L-1)}$$

7.3 BP网络

$$\delta_k^{(L)} = -\left(d_k(i) - y_k(i)\right) f'(x) \Big|_{x=net_k^{(L)}}$$

7.3.3 BP推导

➤ 同理，得到隐含层L-1层E(i)关于偏置的偏导数：

$$\frac{\partial E(i)}{\partial b_j^{(L-1)}} = -\sum_{k=1}^n \left(d_k(i) - y_k(i)\right) f'(x) \Big|_{x=net_k^{(L)}} W_{kj}^{(L)} f'(x) \Big|_{x=net_j^{(L-1)}}$$

$$\begin{aligned} \text{令 } \delta_j^{(L-1)} &= -\sum_{k=1}^n \left(d_k(i) - y_k(i)\right) f'(x) \Big|_{x=net_k^{(L)}} W_{kj}^{(L)} f'(x) \Big|_{x=net_j^{(L-1)}} \\ &= \sum_{k=1}^n \delta_k^{(L)} W_{kj}^{(L)} f'(x) \Big|_{x=net_j^{(L-1)}} \end{aligned}$$

第L-1层的误差是第L层误差的反向传播

➤ 隐含层L-1层E(i)关于权值和偏置的偏导数可进一步简化为：

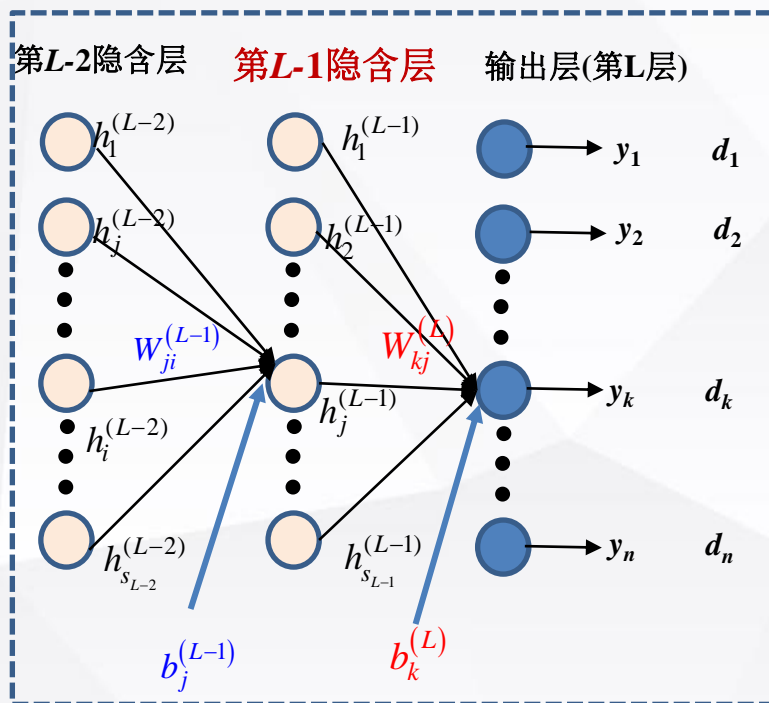
$$\frac{\partial E(i)}{\partial W_{ji}^{(L-1)}} = \delta_j^{(L-1)} h_i^{(L-2)} \qquad \frac{\partial E(i)}{\partial b_j^{(L-1)}} = \delta_j^{(L-1)}$$

7.3 BP网络

7.3.3 BP推导

➤ 隐含层 $L-1$ 层 $E(i)$ 关于权值和偏置的偏导数的直观解释:

$$\frac{\partial E(i)}{\partial W_{ji}^{(L-1)}} = \underbrace{\delta_j^{(L-1)}}_{\text{误差}} \underbrace{h_i^{(L-2)}}_{\text{输入}} \quad \frac{\partial E(i)}{\partial b_j^{(L-1)}} = \delta_j^{(L-1)} \quad \text{误差反向传播}$$



• 第 $L-1$ 层的误差是第 L 层误差的反向传播。

$$\begin{aligned} \delta_j^{(L-1)} &= - \sum_{k=1}^n \underbrace{(d_k(i) - y_k(i)) f'(x)}_{\text{第L层误差}} \bigg|_{x=net_k^{(L)}} W_{kj}^{(L)} f'(x) \bigg|_{x=net_j^{(L-1)}} \\ &= \sum_{k=1}^n \underbrace{\delta_k^{(L)}}_{\text{第L层误差}} W_{kj}^{(L)} f'(x) \bigg|_{x=net_j^{(L-1)}} \end{aligned}$$

7.3 BP网络

7.3.3 BP推导

- 依次类推，得到第 l 层（ $2 \leq l \leq L-1$ ） $E(i)$ 关于权值和偏置的偏导数：

$$\frac{\partial E(i)}{\partial W_{ji}^{(l)}} = \delta_j^{(l)} h_i^{(l-1)} \quad \frac{\partial E(i)}{\partial b_j^{(l)}} = \delta_j^{(l)} \quad \text{误差反向传播}$$

$$\text{其中, } \delta_j^{(l)} = \sum_{k=1}^{s_{l+1}} W_{kj}^{(l+1)} \delta_k^{(l+1)} f'(x) \Big|_{x=net_j^{(l)}}$$

输出误差通过隐含层向输入层逐层反向传播，并将误差分摊给各层所有节点（获得所有层的误差估计），按误差函数的负梯度方向修改各节点连接权重；

$$W_{ij}^{(l)} \leftarrow W_{ij}^{(l)} - \eta \frac{\partial E}{\partial W_{ij}^{(l)}} \quad b_i^{(l)} \leftarrow b_i^{(l)} - \eta \frac{\partial E}{\partial b_i^{(l)}}$$

7.3 BP网络

7.3.4 BP具体实现步骤

若采用批量样本更新算法对BP神经网络的权值和偏置进行更新，则BP算法步骤为：

➤ **步骤1：**网络初始化。初始化各连接权值，给定学习率。

➤ **步骤2：**输入 M 个样本

$$\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_i \quad \dots \quad x_m], i = 1, 2, \dots, m \quad \text{每个样本为 } m \text{ 维向量}$$

$$\mathbf{y} = [y_1 \quad y_2 \quad \dots \quad y_k \quad \dots \quad y_n], k = 1, 2, \dots, n \quad \text{标签为 } n \text{ 维向量}$$

对所有层 $2 \leq l \leq L$

$$\Delta W^{(l)} = 0, \quad \Delta b^{(l)} = 0$$

➤ **步骤3：**遍历所有样本训练执行如下操作：

7.3 BP网络

7.3.4 BP具体实现步骤

➤ **步骤3:** 遍历所有样本训练执行如下操作:

- a) 对样本 x_i , 计算误差损失函数 $E(i)$;
- b) 利用BP算法, 计算每个样本误差损失函数 $E(i)$ 关于各层权值和偏置的偏导数

$$\frac{\partial E(i)}{\partial W_{ji}^{(l)}} \quad \frac{\partial E(i)}{\partial b_j^{(l)}} \quad l \text{ 为网络的第 } l \text{ 层}$$

得到梯度矩阵 $\nabla W^{(l)}(i)$ 和向量 $\nabla b^{(l)}(i)$;

- c) 计算所有样本误差损失函数 E 关于各层权值和偏置的偏导数:

$$\Delta W^{(l)} = \Delta W^{(l)} + \nabla W^{(l)}(i) \quad \Delta b^{(l)} = \Delta b^{(l)} + \nabla b^{(l)}(i)$$

7.3 BP网络

7.3.4 BP具体实现步骤

- **步骤4：** 进行权值和偏置的更新

$$W^{(l)} = W^{(l)} - \eta \left[\frac{1}{M} \Delta W^{(l)} \right] \quad b^{(l)} = b^{(l)} - \eta \left[\frac{1}{M} \Delta b^{(l)} \right]$$

- **步骤5：** 利用更新后的权值和偏置，重新进行网络前向传输计算，计算网络输出。
- **步骤6：** 重复步骤3、步骤4和步骤5，直至网络输出与期望输出损失函数小于某个设定的阈值或者迭代到某次数，则算法结束。

7.3 BP网络

7.3.5 BP网络问题分析

- 在层次深的情况下性能变得很不理想：传播时容易出现梯度弥散，陷入局部最优，且这种情况随着网络层数的增加而更加严重，所以只能处理浅层结构从而限制了性能。

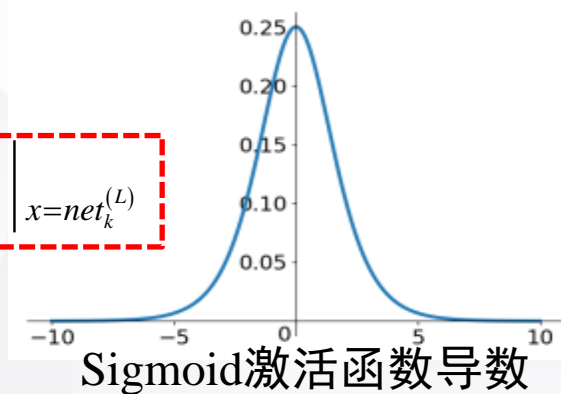
第 l 层（ $2 \leq l \leq L-1$ ） $E(i)$ 关于权值和偏置的偏导数：

$$\frac{\partial E(i)}{\partial W_{ji}^{(l)}} = \delta_j^{(l)} h_i^{(l-1)} \quad \frac{\partial E(i)}{\partial b_j^{(l)}} = \delta_j^{(l)}$$

其中， $\delta_j^{(l)} = \sum_{k=1}^{s_{l+1}} W_{kj}^{(l+1)} \delta_k^{(l+1)} f'(x) \Big|_{x=net_j^{(l)}}$

激活函数的导数连乘

$$\delta_k^{(L)} = -(d_k(i) - y_k(i)) f'(x) \Big|_{x=net_k^{(L)}}$$



7.4 自组织神经网络

- **感知器和BP神经网络**：有监督学习方式，样本带标签
- **自组织神经网络**：早期无监督学习网络模型的代表。样本不带标签

基本思想：模拟生物神经系统功能的人工神经网络。

在生物神经系统中，存在着一种**侧抑制现象**，即一个神经细胞兴奋以后，会对周围其他神经细胞产生抑制作用。这种**抑制作用会使神经细胞之间出现竞争，竞争获胜的神经细胞兴奋，失败的神经细胞被抑制。**

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

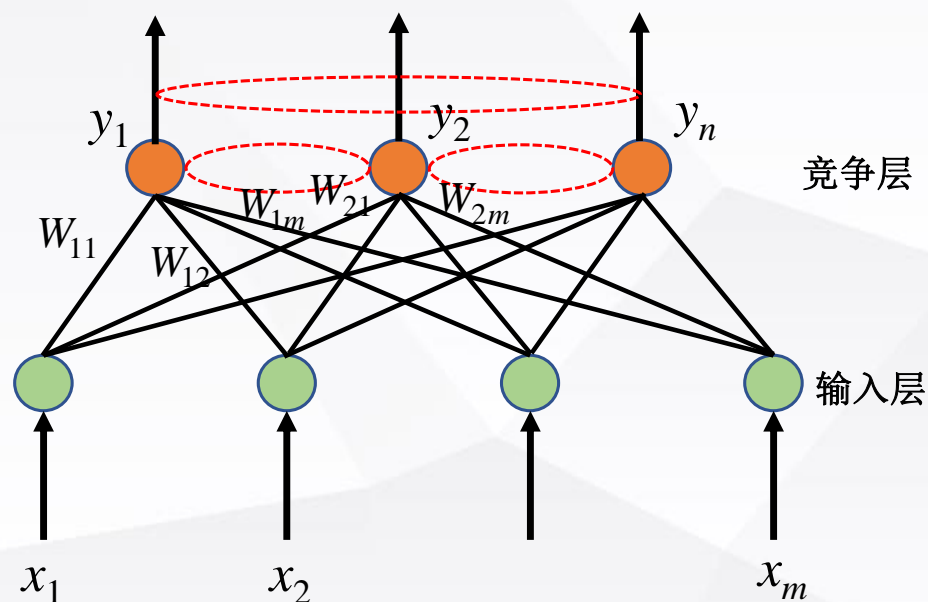
7.4.2 自组织映射网络（SOM网络）

7.4.3 小结

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

一种基于竞争性学习的网络模型，输出神经元之间相互竞争激活，在任意时刻只有一个神经元被激活。**被激活的神经元被称为胜利者神经元（winner-takes-all neuron）**，而其它神经元的状态被抑制，故称为Winner Take All（简称**WTA**）。



竞争层：对输入模式进行分析比较，寻找规律，并进行归类处理。

输入层：接收外界信息，将输入模式向竞争层传递

7.4 自组织神经网络

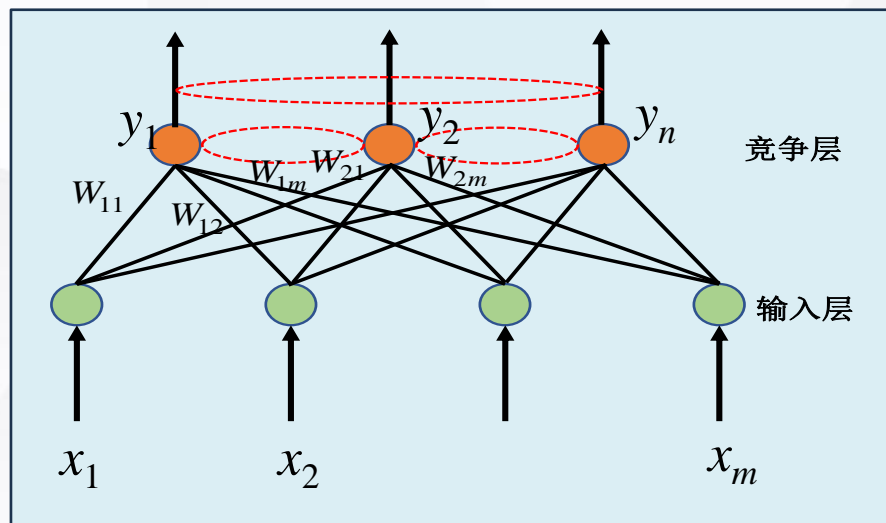
➤ WTA竞争学习过程

步骤1：向量归一化

对网络当前输入向量 X (m 维列向量) 和竞争层中各神经元对应的权重向量 W_j (对应第 j 个神经元, 为 m 维列向量 $W_{j1}, W_{j2}, \dots, W_{jm}$) 全部进行归一化, 使得 X 和 W_j 的模为1, 得到:

$$\hat{X} = \frac{X}{\|X\|}$$

$$\hat{W}_j = \frac{W_j}{\|W_j\|}$$



7.4 自组织神经网络

➤ WTA竞争学习过程

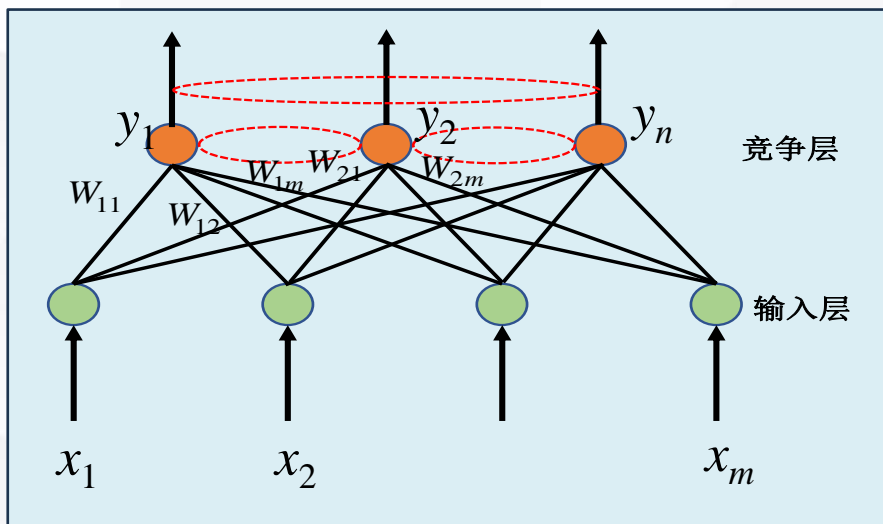
步骤2：寻找获胜神经元

n 为竞争层神经元数量

将 \hat{X} 与竞争层所有神经元对应的权向量 \hat{W}_j ($j = 1, 2, \dots, n$) 进行相似性比较。

将最相似的神经元判为竞争获胜神经元：

$$\|\hat{X} - \hat{W}_{j^*}\| = \min_{j \in \{1, 2, \dots, n\}} \{\|\hat{X} - \hat{W}_j\|\}$$



7.4 自组织神经网络

➤ WTA竞争学习过程

步骤3：网络输出与权值调整

按WTA学习规则，获胜神经元输出为1，其他神经元输出为0：

$$y_j(t+1) = \begin{cases} 1 & j = j^* \\ 0 & j \neq j^* \end{cases}$$

只有获胜神经元的权值才能进行调整更新：

$$\begin{cases} W_{j^*}(t+1) = \hat{W}_{j^*}(t) + \Delta W_{j^*} = \hat{W}_{j^*}(t) + \eta(\hat{X} - \hat{W}_{j^*}) \\ W_{j^*}(t+1) = \hat{W}_{j^*}(t) & j \neq j^* \end{cases}$$

其中学习率 $0 < \eta \leq 1$ ，一般随着学习的进展而减小，即调整的程度越来越小，神经元（权重）趋于聚类中心。

7.4 自组织神经网络

➤ WTA竞争学习过程

步骤4：重新归一化处理

归一化后的权向量经过调整后，得到的新权向量 W_{j^*} 不再是单位向量，因此需要对学习调整后的权向量 W_{j^*} 重新进行归一化。

$$\hat{W}_{j^*} = \frac{W_{j^*}}{\|W_{j^*}\|}$$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析

利用竞争学习算法将下列模式分为两类：

$$X_1 = [0.8 \quad 0.6]^T$$

$$X_2 = [0.1736 \quad -0.9848]^T$$

$$X_3 = [0.707 \quad 0.707]^T$$

$$X_4 = [0.342 \quad -0.9397]^T$$

$$X_5 = [0.6 \quad 0.8]^T$$

其中学习率： $\eta = 0.5$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

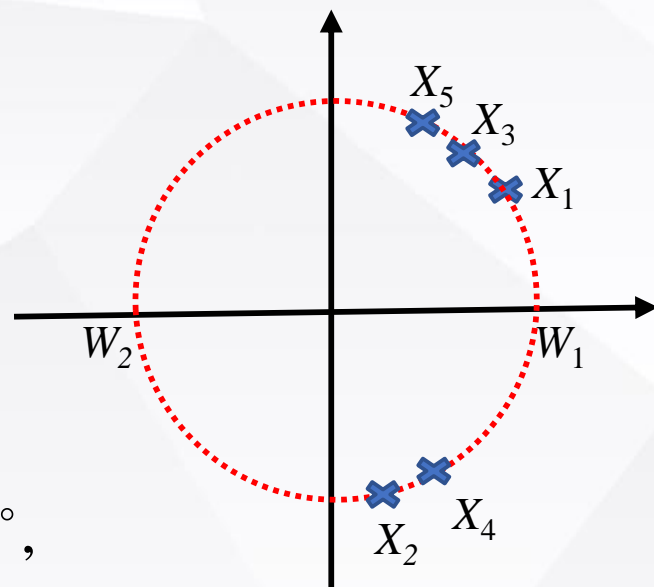
➤ 算例分析

【求解】

将上述输入量转换为极坐标形式，

$$X_1 = 1\angle 36.89^\circ, X_2 = 1\angle -80^\circ, X_3 = 1\angle 45^\circ,$$

$$X_4 = 1\angle -70^\circ, X_5 = 1\angle 53.13^\circ$$



要求将如上模式分为两类，则竞争层为两个神经元（对应两个权向量）。两个权向量被随机初始化为单位向量：

$$W_1(0) = [1 \ 0]^T = 1\angle 0^\circ \quad W_2(0) = [-1 \ 0]^T = 1\angle -180^\circ$$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析

步骤1: 对输入 X_1 ，与两个神经元所对应的权向量 $W_1(0)$ 和 $W_2(0)$ 进行相似性比较：

$$d_1 = \|X_1 - W_1(0)\| = 1 \angle 36.89^\circ$$

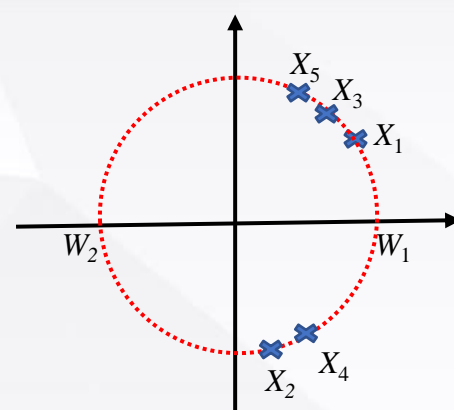
$$d_2 = \|X_1 - W_2(0)\| = 1 \angle 216.89^\circ$$

因为 $d_1 < d_2$ ，所以神经元1为获胜神经元，对其权值 W_1 进行调整：

$$W_1(1) = W_1(0) + \eta(X_1 - W_1(0)) = 0 + 0.5(36.89 - 0) = 1 \angle 18.43^\circ$$

神经元2对应的权值 W_2 不变，即：

$$W_2(1) = W_2(0) = 1 \angle -180^\circ$$



7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析

步骤2: 对输入 X_2 ，与两个神经元所对应的权向量 $W_1(1)$ 和 $W_2(1)$ 进行相似性比较：

$$d_1 = \|X_2 - W_1(1)\| = 1 \angle 98.43^\circ$$

$$d_2 = \|X_2 - W_2(1)\| = 1 \angle 100^\circ$$

因为 $d_1 < d_2$ ，神经元1为获胜神经元，对其权值 W_1 进行调整：

$$W_1(2) = W_1(1) + \eta(X_2 - W_1(1)) = 18.43 + 0.5(-80 - 18.43) = 1 \angle -30.8^\circ$$

神经元2对应的权值 W_2 不变，即：

$$W_2(2) = W_2(1) = 1 \angle -180^\circ$$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析

步骤3: 对输入 X_3 ，与两个神经元所对应的权向量 $W_1(2)$ 和 $W_2(2)$ 进行相似性比较：

$$d_1 = \|X_3 - W_1(2)\| = 1 \angle 75.8^\circ$$

$$d_2 = \|X_3 - W_2(2)\| = 1 \angle 225^\circ$$

因为 $d_1 < d_2$ ，神经元1为获胜神经元，对其权值 W_1 进行调整：

$$W_1(3) = W_1(2) + \eta(X_3 - W_1(2)) = -30.8 + 0.5(45 + 30.8) = 1 \angle 7^\circ$$

神经元2对应的权值 W_2 不变，即：

$$W_2(3) = W_2(2) = 1 \angle -180^\circ$$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析

步骤4: 对输入 X_4 ，与两个神经元所对应的权向量 $W_1(3)$ 和 $W_2(3)$ 进行相似性比较：

$$d_1 = \|X_4 - W_1(3)\| = 1 \angle 77^\circ$$

$$d_2 = \|X_4 - W_2(3)\| = 1 \angle 110^\circ$$

因为 $d_1 < d_2$ ，神经元1为获胜神经元，对其权值 W_1 进行调整：

$$W_1(4) = W_1(3) + \eta(X_4 - W_1(3)) = 7 + 0.5(-70 - 7) = 1 \angle -31.5^\circ$$

神经元2对应的权值 W_2 不变，即：

$$W_2(4) = W_2(3) = 1 \angle -180^\circ$$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析

步骤5：对输入 X_5 ，与两个神经元所对应的权向量 $W_1(4)$ 和 $W_2(4)$ 进行相似性比较：

$$d_1 = \|X_5 - W_1(4)\| = 1 \angle 84.63^\circ$$

$$d_2 = \|X_5 - W_2(4)\| = 1 \angle 126.87^\circ$$

因为 $d_1 < d_2$ ，神经元1为获胜神经元，对其权值 W_1 进行调整：

$$W_1(5) = W_1(4) + \eta(X_5 - W_1(4)) = -31.5 + 0.5(53.13 + 31.5) \approx 1 \angle 11^\circ$$

神经元2对应的权值 W_2 不变，即：

$$W_2(5) = W_2(4) = 1 \angle -180^\circ$$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析

至此形成一个循环。重复循环直至收敛或迭代至特定循环次数。

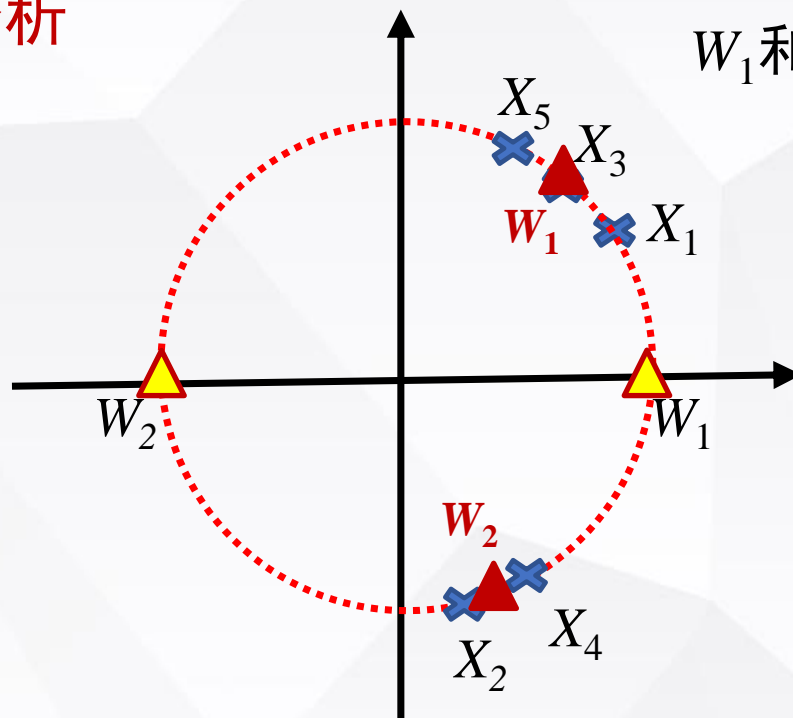
学习次数	W_1	W_2	学习次数	W_1	W_2
1	18.43°	-180°	11	40.5°	-100°
2	-30.8°	-180°	12	40.5°	-90°
3	7°	-180°	13	43°	-90°
4	-32°	-180°	14	43°	-81°
5	11°	-180°	15	47.5°	-81°
6	24°	-180°	16	42°	-81°
7	24°	-130°	17	42°	-80.5°
8	34°	-130°	18	43.5°	-80.5°
9	34°	-100°	19	43.5°	-75°
10	44°	-100°	20	48.5°	-75°

W_1 和 W_2 逐渐稳定于两类聚类中心 $\frac{1}{3}(X_1 + X_3 + X_5) = 1\angle 45^\circ$ $\frac{1}{2}(X_2 + X_4) = 1\angle -75^\circ$

7.4 自组织神经网络

7.4.1 自组织竞争神经网络

➤ 算例分析



W_1 和 W_2 逐渐稳定于两类聚类中心

首先给定训练数据（蓝色叉号）和神经元权重初始值（黄色三角）

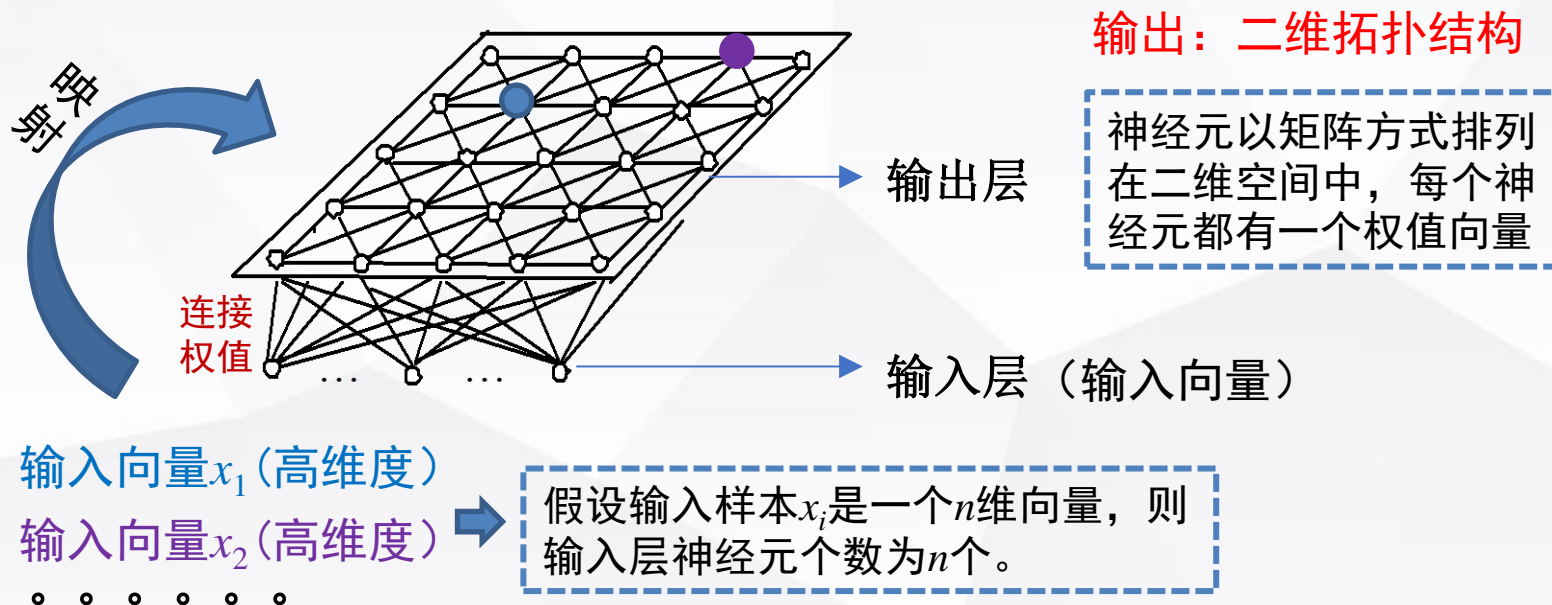
迭代训练之后，神经元权重趋向于聚类中心（红色三角）；

Test阶段，给定数据点，基于WTA策略，直接算出和哪个神经元权向量最相似就分到哪个类。

7.4 自组织神经网络

7.4.2 自组织映射网络

- Self-Organizing Map, 简称SOM网。
- 一种竞争学习型的无监督神经网络。
- 能将高维度输入数据映射到低维空间（通常为二维），同时保持输入数据在高维空间的拓扑结构，即将高维空间中相似的样本点映射到网络输出层中的邻近神经元。

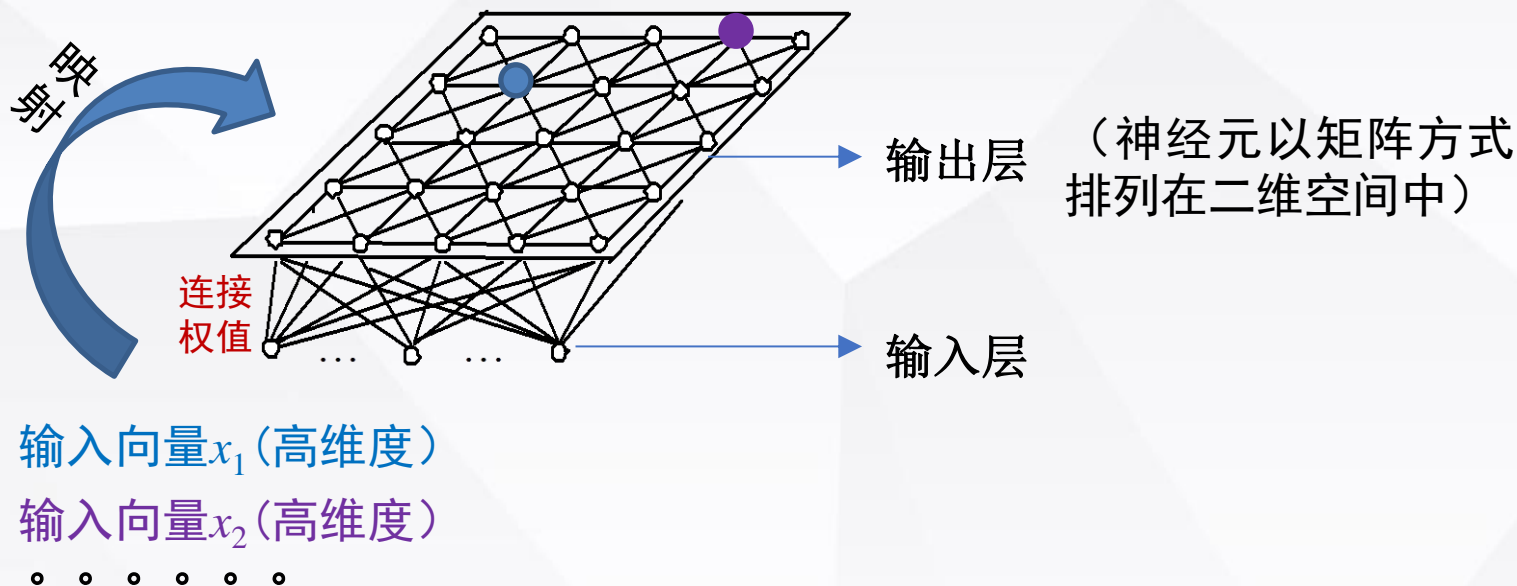


7.4 自组织神经网络

7.4.2 自组织映射网络

- **基本思想**：通过学习输入空间中的数据，生成一个低维、离散的映射(Map)，从某种程度上也可看成一种降维算法。
- **无监督人工神经网络**：不同于一般神经网络基于损失函数的反向传递来训练，运用竞争学习策略，依靠神经元之间互相竞争逐步优化网络。利用近邻关系函数来维持输入空间的拓扑结构。
- **维持输入空间的拓扑结构**：二维映射包含了数据点之间的相对距离。输入空间中相邻的样本会被映射到相邻的输出神经元。
- 无需样本标签，可用于聚类。

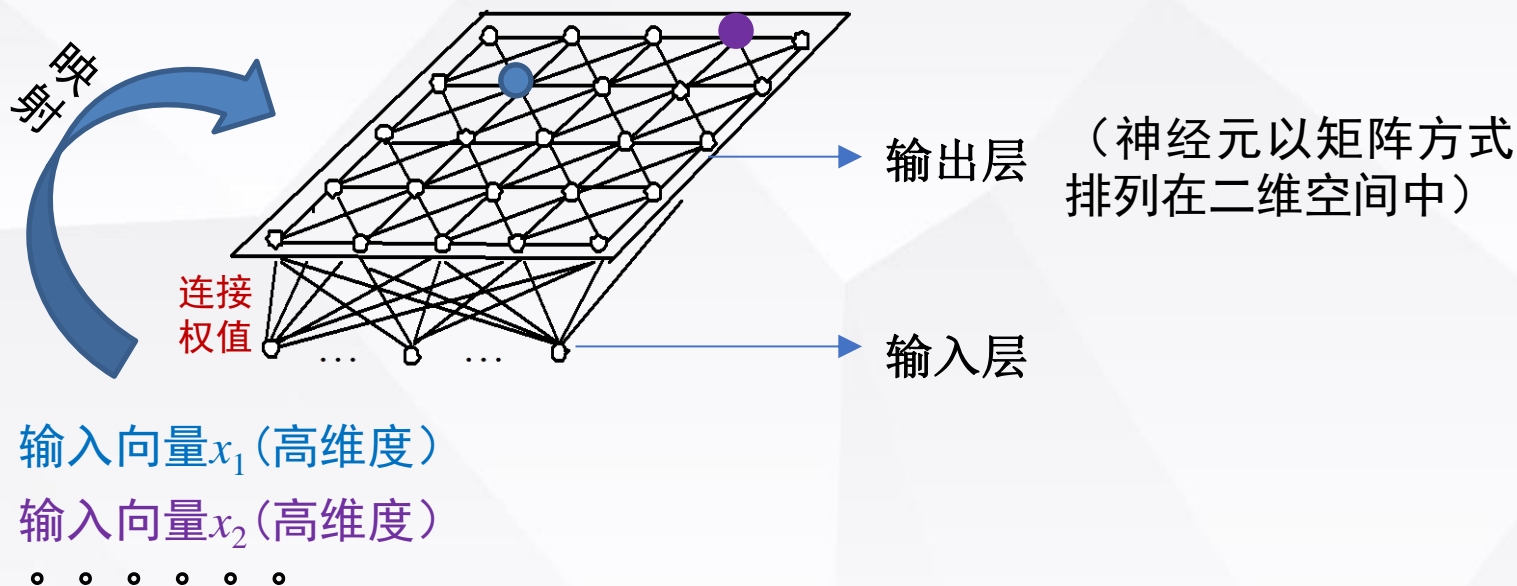
7.4 自组织神经网络



➤ SOM训练目标

为输出层每个神经元找到合适的权向量，达到保持输入空间拓扑结构的目的。

7.4 自组织神经网络



➤ SOM训练过程

- 首先，输入一个训练样本，计算输出层每个神经元的权向量与该输入样本之间的距离，**距离最近的神经元成为竞争获胜者**（最佳匹配单元）。
- 然后，**调整最佳匹配单元及其近邻神经元（优胜邻域）的权向量**，使得这些权向量与当前输入样本的距离缩小；
- 该过程不断迭代，直至收敛。

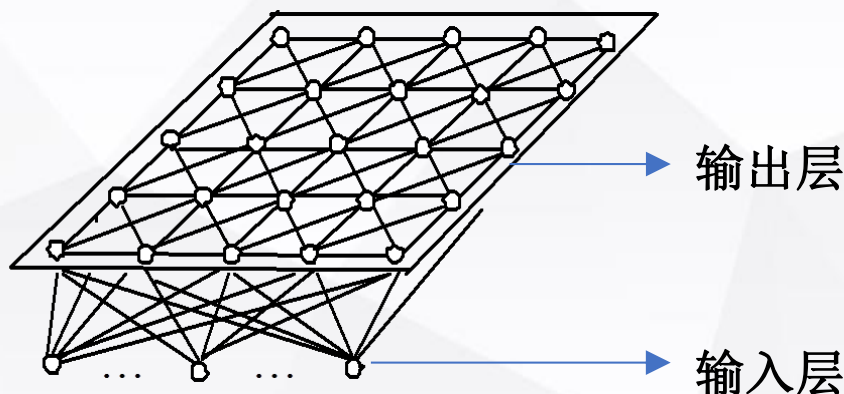
7.4 自组织神经网络

➤ SOM算法学习步骤

步骤1： 确定SOM网络拓扑结构，根据样本类型和经验确定网络竞争层维度、神经元数量和神经元的拓扑结构

a) 设定输出层神经元的数量：

输出层神经元的数量和训练集样本的类别数相关。若不清楚类别数，尽可能地设定较多的节点数，以便较好地映射样本的拓扑结构，如果分类过细再适当减少输出节点。



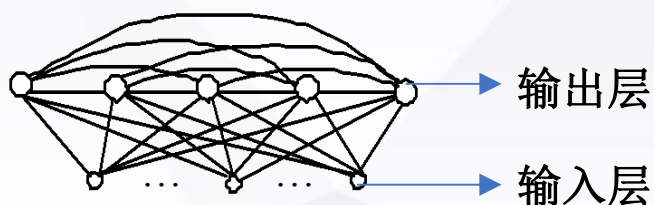
7.4 自组织神经网络

➤ SOM算法学习步骤

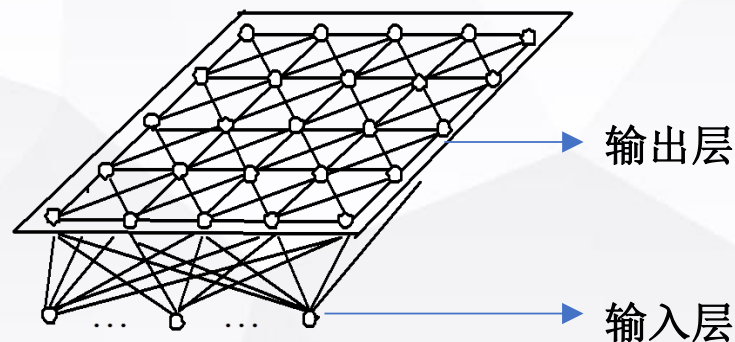
步骤1： 确定SOM网络拓扑结构，根据样本类型和经验确定网络竞争层维度、神经元数量和神经元的拓扑结构

b) 设计输出层节点的排列：

与实际应用需求相关。一般根据预估的聚类簇数进行设置。比如要分成4类，那可以将输出层（竞争层）设计为 2×2 ， 1×4 ， 4×1 。



一维线阵



二维平面阵

7.4 自组织神经网络

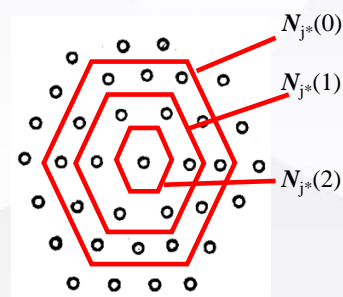
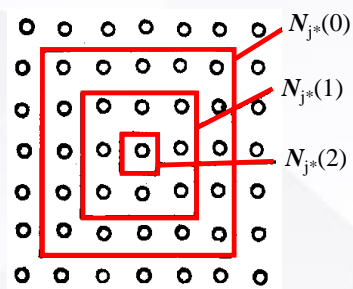
➤ SOM算法学习步骤

步骤2：初始化

a) 对竞争层（也是输出层）各神经元的权值赋小的随机数初值，该操作可使权向量充分分散在样本空间中。对其进行归一化处理，得到 \hat{W}_j ($j = 1, 2, \dots, n$, n 为输出层神经元数量)

b) 设定初始优胜邻域 $N_{j^*}(0)$

邻域形状可以是正方形、六边形等。



c) 设定学习率初始值 η

在训练开始时，学习率可以选取较大的值，之后以较快的速度下降

7.4 自组织神经网络

➤ SOM算法学习步骤

步骤3：从训练数据中随机选取一个输入数据，并对其进行归一化，得到 \hat{X}

步骤4：寻找获胜神经元

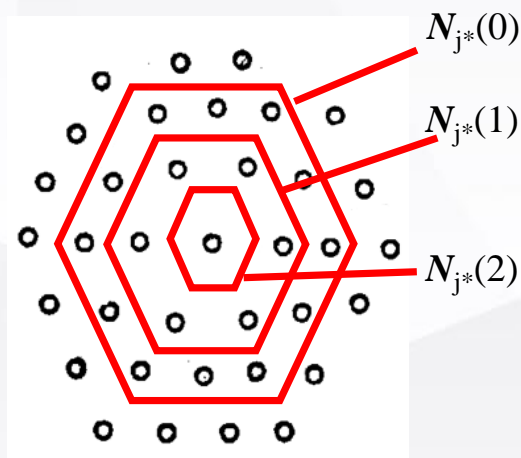
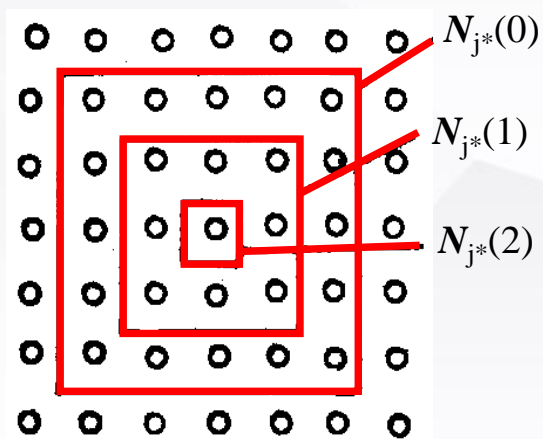
计算 \hat{X} 与竞争层（输出层）每个神经元归一化权向量 \hat{W}_j 的内积，找到内积最大的获胜神经元 j^*

7.4 自组织神经网络

➤ SOM算法学习步骤

步骤5： 定义优胜邻域 $N_{j^*}(t)$

以 j^* 为中心确定 t 时刻的权值调整域，一般选择初始邻域 $N_{j^*}(0)$ 较大，训练时 $N_{j^*}(t)$ 随训练时间逐渐收缩。



7.4 自组织神经网络

➤ SOM算法学习步骤

步骤6： 调整权重。对优胜邻域 $N_{j^*}(t)$ 内的所有神经元进行权重调整

$$W_{ji}(t+1) = W_{ji}(t) + \eta(t, N) [x_i^p - W_{ji}(t)] \quad i = 1, 2, \dots, m, j \in N_{j^*}(t)$$



下标 i ：输入层神经元索引； $i=1, 2, \dots, m$ ， m 为输入层神经元总数量

下标 j^* ：输出层获胜神经元索引；

下标 j ：获胜神经元优胜邻域内的神经元索引； $j \in N_{j^*}(t)$

上标 p ：为从训练样本集中选取的第 p 个样本；

t ：训练时间或者训练次数；

N ：获胜神经元 j^* 邻域内第 j 个神经元与获胜神经元 j^* 之间的拓扑距离

$\eta(t, N)$ ：为训练时间为 t 、优胜邻域内第 j 个神经元与获胜神经元 j^* 之间拓扑距离为 N 情况下的学习率函数。

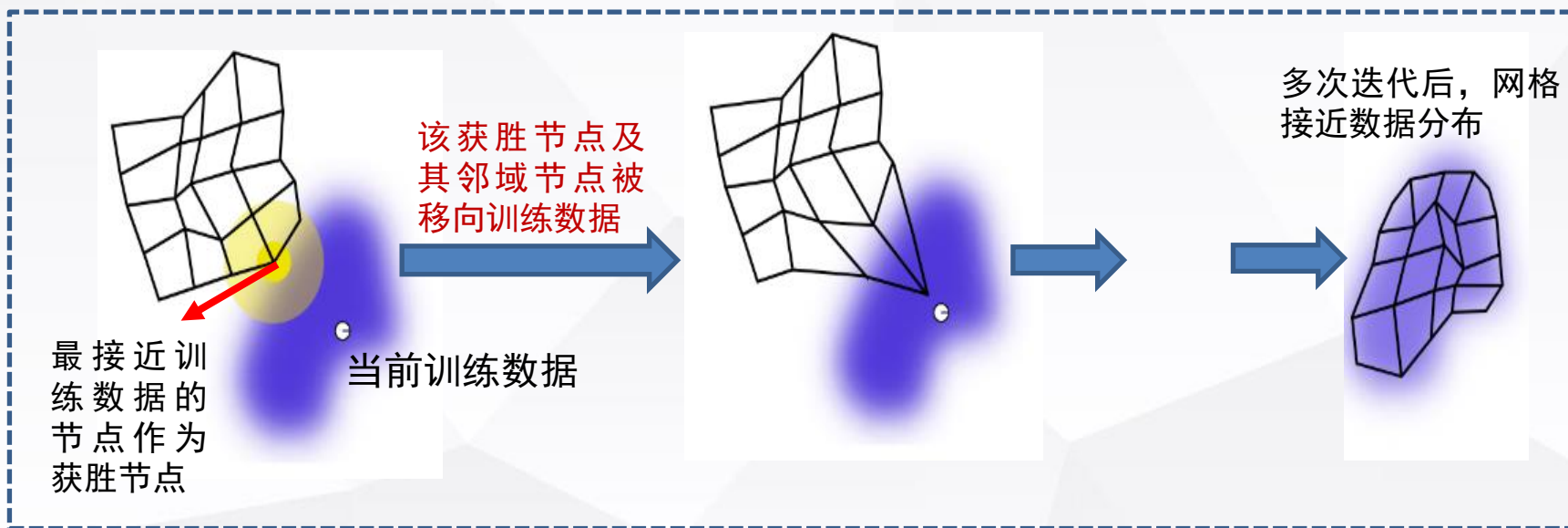
$$\eta(t, N) = \eta(t) e^{-N} \quad \Rightarrow \quad \eta(t) : t \text{ 的单调递减函数}$$

7.4 自组织神经网络

➤ SOM算法学习步骤

步骤7：结束判定

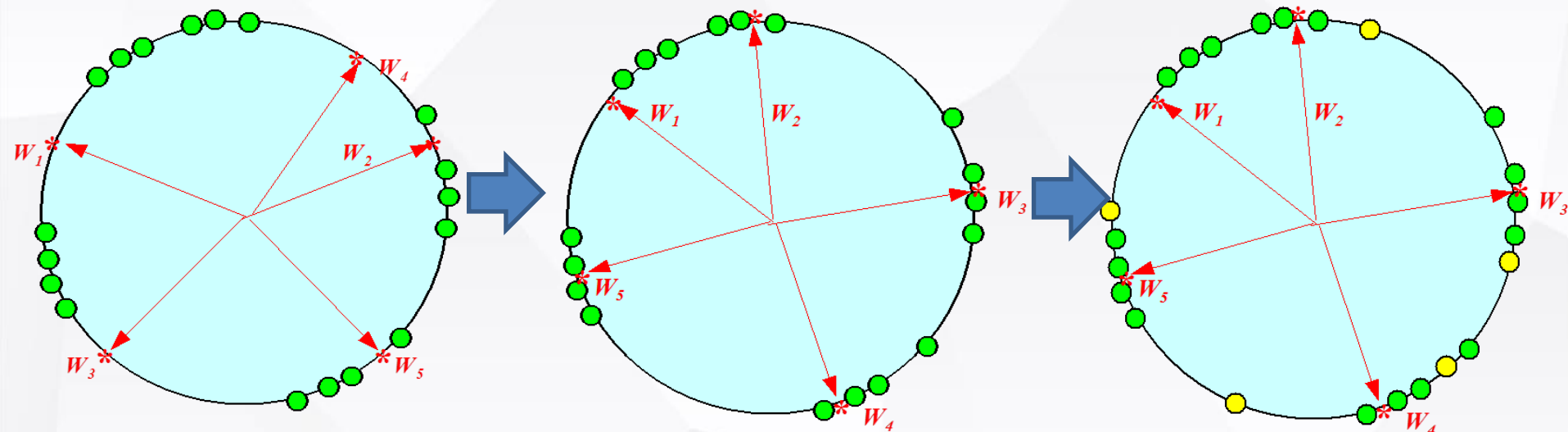
查看学习率是否减小到0，或者小于某个阈值。不满足条件时，转至步骤3，继续训练。



SOM 网络学习过程示意图

7.4 自组织神经网络

➤ SOM算法工作原理示意图



首先给定训练数据
(绿点)和神经元权重
初始值(红点)

迭代训练之后,神
经元权重趋向于聚
类中心;

Test阶段,给定数据点(黄
点),基于WTA策略,用内积
直接算出该数据和哪个神经
元权重最相似就分到哪个类。

7.4 自组织神经网络

➤ SOM网络与K均值对比

- a) K均值算法需要事先确定类的个数。而自组织映射神经网络则不用，利用WTA竞争策略，自动更新神经元权重，利用样本与最终更新后的权值内积大小，自动归类。
- b) 自组织映射神经网络可视化比较好，而且具有优雅的拓扑关系图。

7.4 自组织神经网络

7.4.3 小结

➤ 自组织神经网络特点

- a) 神经网络，竞争学习策略；
- b) 无监督学习，不需要额外标签；
- c) 非常适合高维数据的可视化，能够维持输入空间的拓扑结构；

本章小结

- 人工神经元
- 感知机
- BP神经网络
- 自组织神经网络

作业1

自组织竞争神经网络由输入层和竞争层组成，假设初始权值向量已被归一化为

$$W_1 = [1 \quad 0] \quad W_2 = [0 \quad -1]$$

4个输入模式，均为归一化后的单位向量：

$$X_1 = 1\angle 30^\circ \quad X_2 = 1\angle -160^\circ$$

$$X_3 = 1\angle 90^\circ \quad X_4 = 1\angle -180^\circ$$

学习率： $\eta = 0.5$

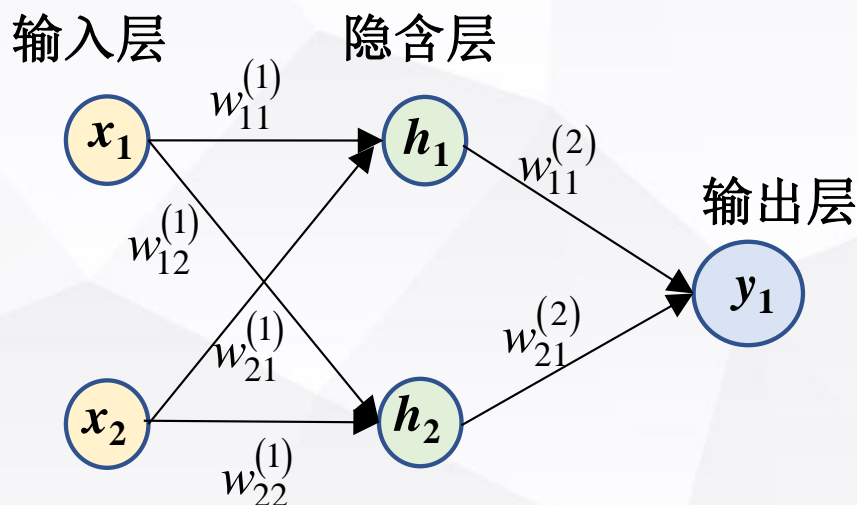
要求利用竞争学习算法将其分为两类。给出前5次的权值学习结果。

作业2

对如下BP神经网络，设定学习率 $\eta = 1$ ，各点偏置量均为0，激活函数采用Sigmoid函数。输入样本 $x_1 = 1$ ， $x_2 = 0$ ，输出节点 y 的期望输出为1。第 k 次学习得到的权值分别为

$$w_{11}^{(1)}(k) = 0 \quad w_{12}^{(1)}(k) = 2 \quad w_{21}^{(1)}(k) = 1 \quad w_{22}^{(1)}(k) = 1$$

$$w_{11}^{(2)}(k) = 1 \quad w_{21}^{(2)}(k) = 1$$



【求解】

a) 第 k 次学习得到的节点输出 $y(k)$

b) 第 $k+1$ 次学习得到的各点权值

$$w_{11}^{(1)}(k+1) \quad w_{12}^{(1)}(k+1) \quad w_{21}^{(1)}(k+1) \quad w_{22}^{(1)}(k+1)$$

$$w_{11}^{(2)}(k+1) \quad w_{21}^{(2)}(k+1)$$

c) 第 $k+1$ 次学习得到的节点输出 $y(k+1)$