

# 机器学习

## Machine Learning

北京航空航天大学计算机学院

School of Computer Science and Engineering, Beihang University

黄迪 张永飞 陈佳鑫

2023年秋季学期

Fall 2023

# 神经元结构

## ● 工作机制

- 一个神经元有两种状态：兴奋和抑制；
- 平时处于抑制状态的神经元，其树突和胞体接收其他神经元由突触传来的兴奋电位，多个输入在神经元中以代数和的方式叠加；
- 如果输入兴奋电位总量超过某个阈值，神经元会被激发进入兴奋状态，发出输出脉冲，并由突触传递给其他神经元。
- 神经元被触发后进入不应期，在不应期不能被触发，然后阈值逐渐下降，恢复兴奋性。

# 神经元结构

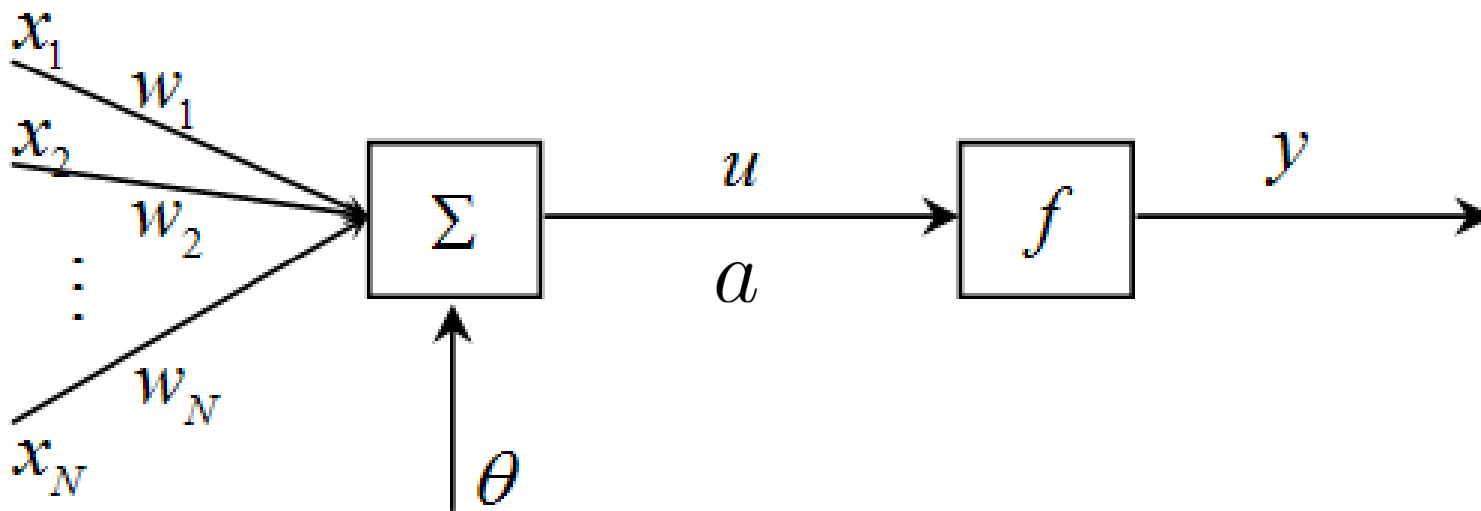
## ● 基本特征

- 神经元及其联接；
- 神经元之间的联接强度决定信号传递的强弱；
- 神经元之间的联接强度是可以随训练改变的；
- 信号可以是起刺激作用的，也可以是起抑制作用的；
- 一个神经元接受信号的累积效果决定该神经元的状态；
- 每个神经元可以有一个“阈值”。

# 人工神经元的MP模型

## ● MP模型

一种人工神经元的数学模型，它最早由美国的McCulloch和Pitts提出的神经元模型，是大多数神经网络模型的基础。



# 人工神经元的MP模型

人工神经元是仿照生物神经元提出的，神经元可以有 $N$ 个**输入**：

$$x_1, x_2, \dots, x_N$$

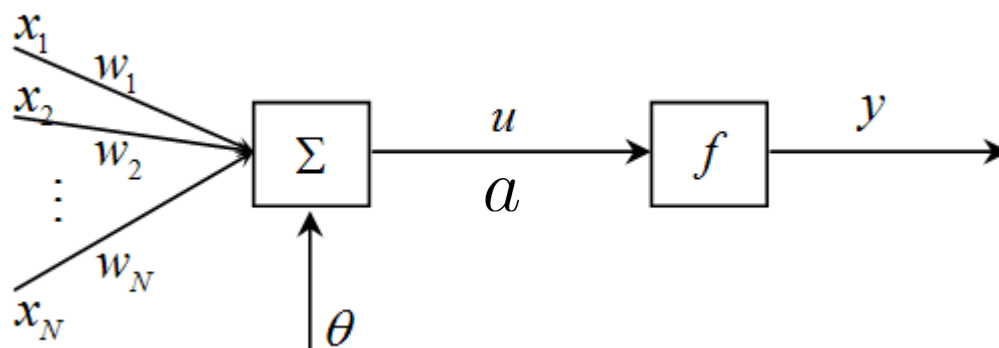
每个输入端与神经元之间有一定的**联接权值**：

$$w_1, w_2, \dots, w_N$$

神经元总输入为对每个输入的加权求和，同时减去**阈值** $\theta$ 。 $a$ 代表神经元的活跃值，即**神经元状态**：

$$a = \sum_{i=1}^N w_i x_i - \theta = \sum_{i=0}^N w_i x_i$$

# 人工神经元的MP模型



神经元的输出 $y$ 是对  $a$  的映射：

$$y = f(a) = f \left( \sum_{i=0}^N w_i x_i \right)$$

**$f$  称为激活函数**(激励函数，输出函数)。

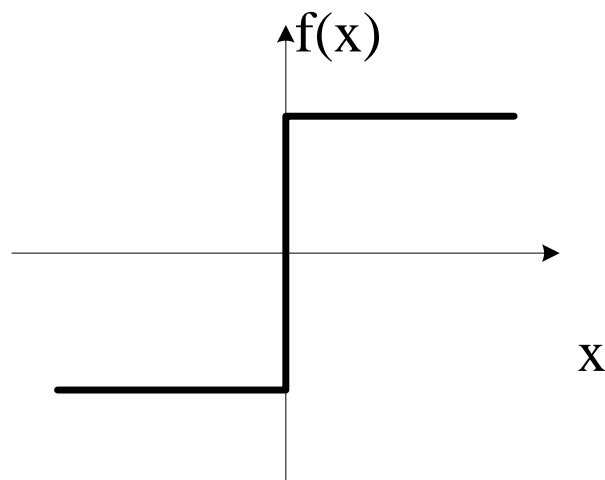
# 人工神经元的MP模型

MP模型采用**阶跃函数**作为激活函数

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



$$y = \text{sgn} \left( \sum_{i=0}^N w_i x_i \right)$$

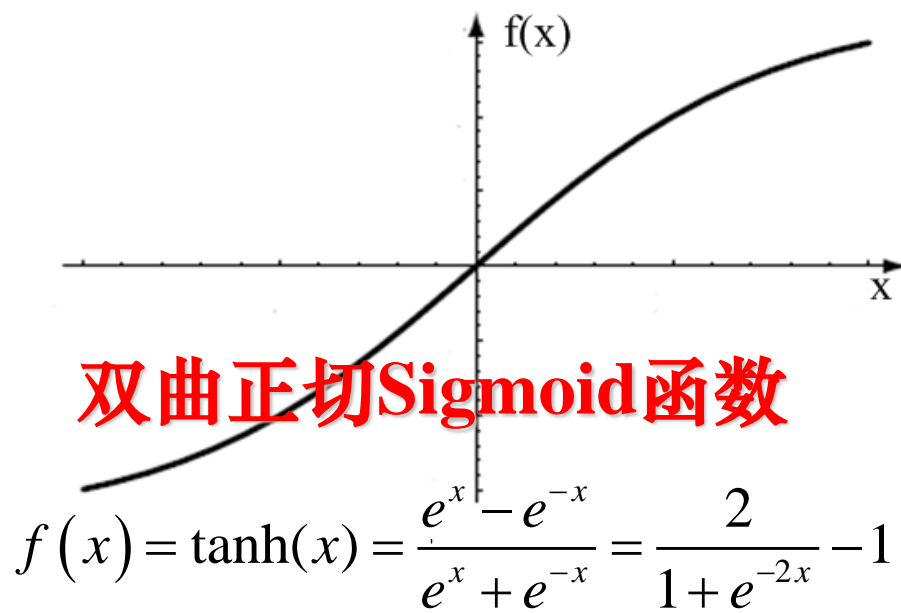
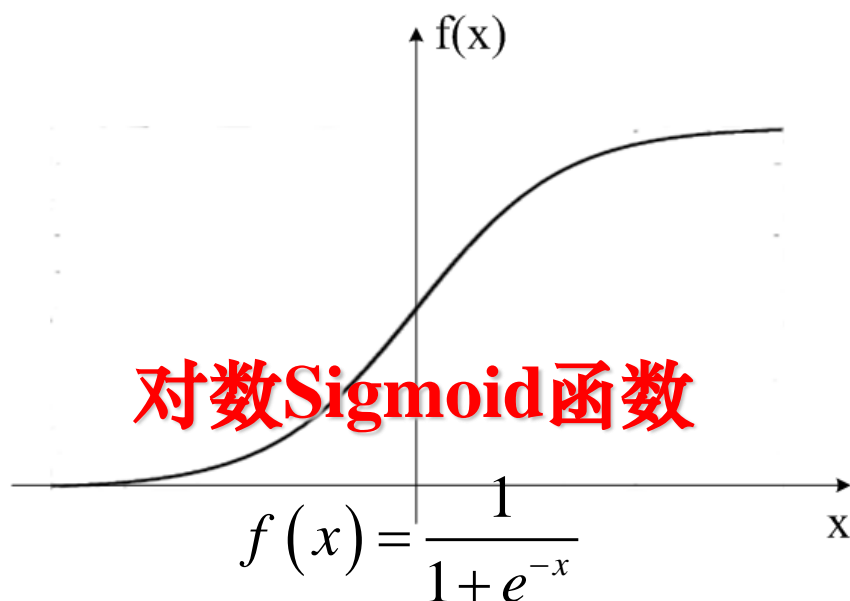


当激活函数  $f$  为阈值(阶跃)函数时，神经元就可以看作是一个**线性分类器**。

# 人工神经元的MP模型

## ● 激活函数

- 线性函数
- 非线性斜面函数(Ramp Function)
- Sigmoid输出函数(S型函数)
- .....



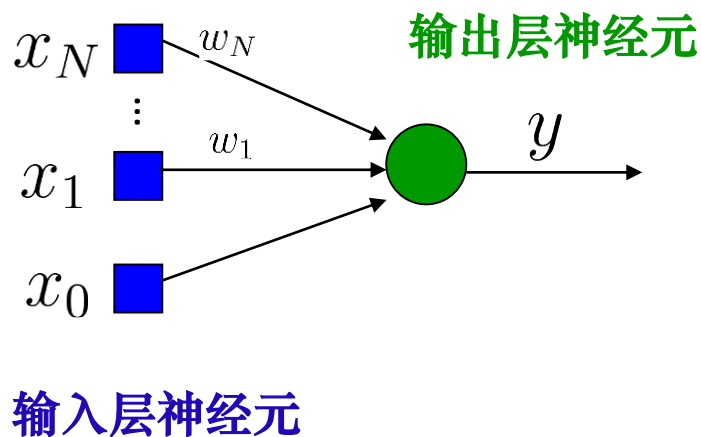


# 感知器

- 感知器是一种早期的神经网络模型，由美国学者F. Rosenblatt于1957年提出。
- 感知器中第一次引入了学习的概念，使人脑所具备的学习功能在基于符号处理的数学到了一定程度的模拟，所以引起了广泛的关注。
- 简单感知器模型实际上仍然是MP模型的结构。它是一种双层神经网络模型，一层为输入层(有时不把输入层计入网络层数之中)，另一层具有计算单元，可以通过监督学习来逐步增强模式划分的能力，达到学习的目的。

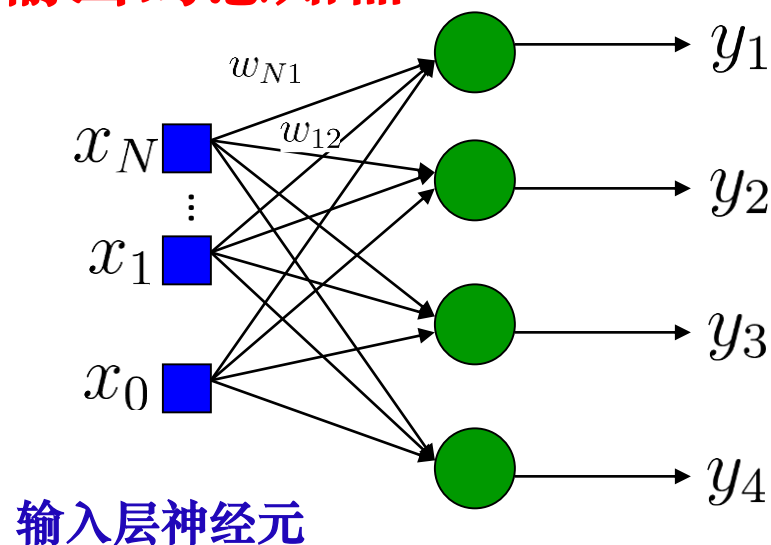
# 感知器

单输出的感知器(M-P模型)



$$y = f(a) = f\left(\sum_{i=0}^N w_i x_i\right)$$

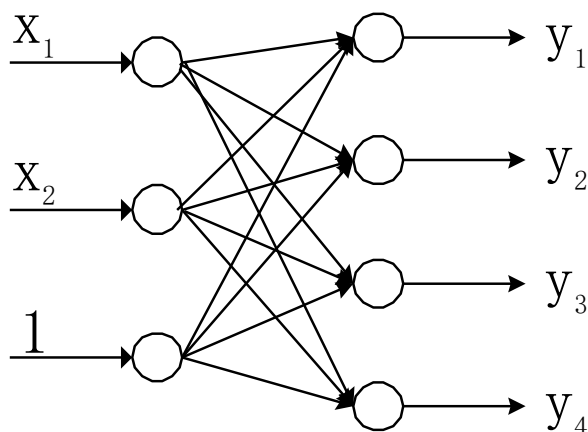
多输出的感知器



$$y_j = f(a_j) = f\left(\sum_{i=0}^N w_{ij} x_i\right)$$

# 感知器

- 单个神经元可以实现两类问题的线性分类，多个感知器则可以实现多类别问题的线性分类。



例子中的网络就可以实现四类问题的分类。

在训练时，理想输出为：

第1类训练样本：(1, 0, 0, 0)      (1, 0)

第2类训练样本：(0, 1, 0, 0)      (0, 1)

第3类训练样本：(0, 0, 1, 0)      (0, 0)

第4类训练样本：(0, 0, 0, 1)      (1, 1)

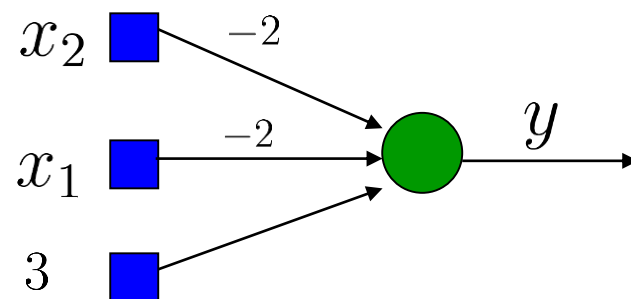
也就是每个神经元输出为1代表某一类别。

# 感知器

## ● 示例：与非门

真值表

		$y$	
		0	1
$x$	0	1	1
	1	1	0



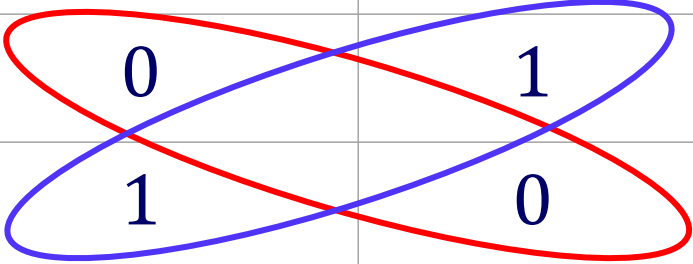
$x_1$	$x_2$	$a$	$y$
1	1	-1	0
1	0	1	1
0	1	1	1
0	0	3	1

# 感知器

- **线性不可分**问题：感知器模型的局限

例：异或(XOR)问题

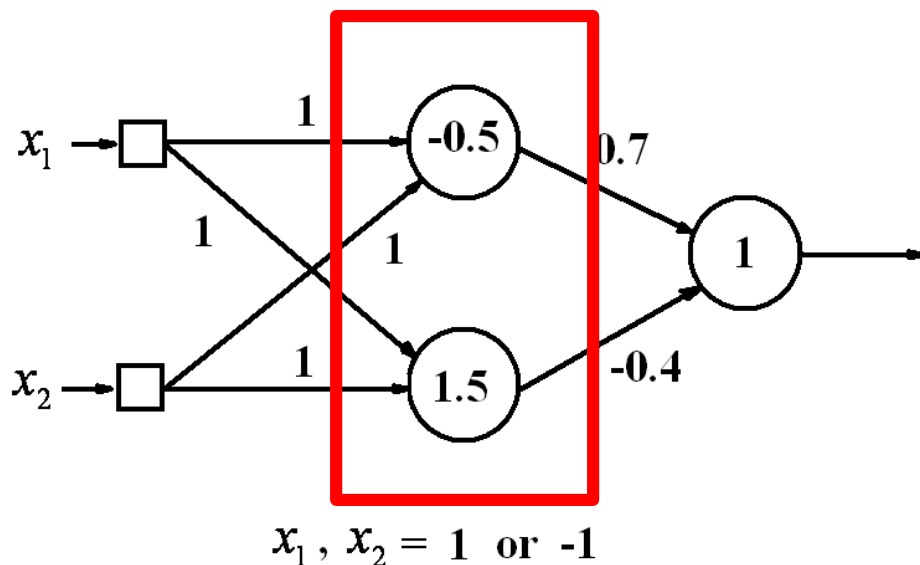
		$y$	
		0	1
$x$	0	0	1
	1	1	0



**解决方法：三层感知器**

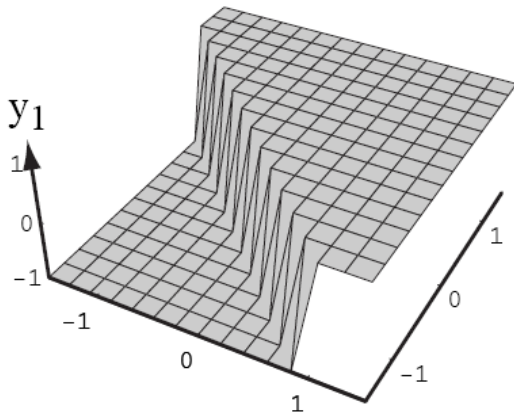
# 多层感知器

- 对于异或问题，用一个简单的三层感知器就可得到解决。

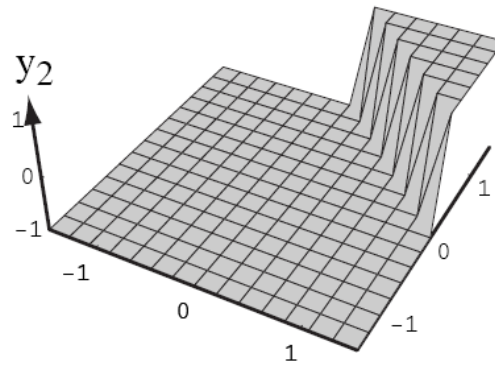


$x_1$	$x_2$	$y$
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

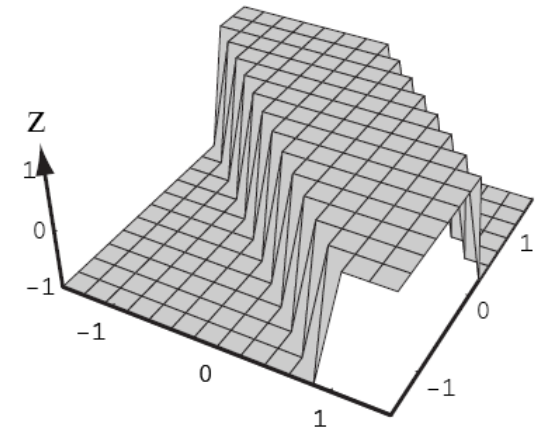
# 多层感知器



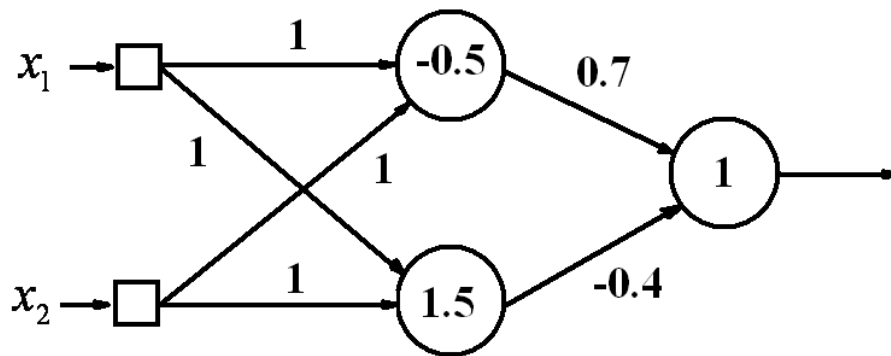
$$x_1 + x_2 + 0.5 = 0$$



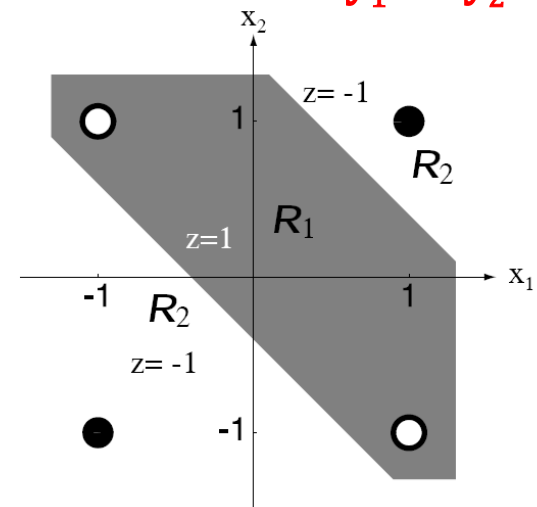
$$x_1 + x_2 - 1.5 = 0$$



$$0.7y_1 - 0.4y_2 - 1 = 0$$



$$x_1, x_2 = 1 \text{ or } -1$$

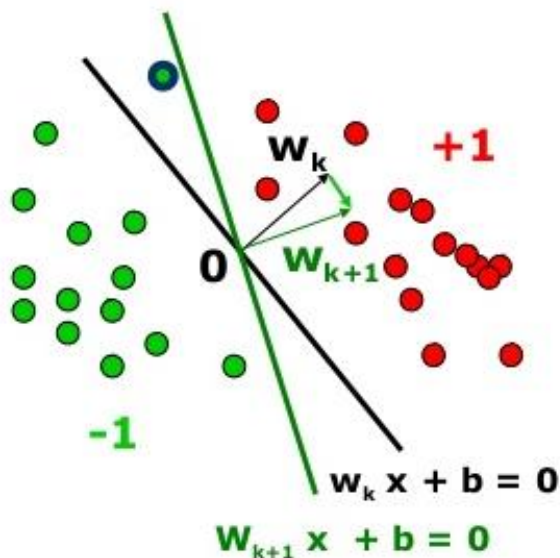


# 感知器

- 感知器训练算法的基本原理来源于著名的Hebb学习律(1949年提出)

- 基本思想

逐步地将样本集中的样本输入到网络中，根据输出结果和理想输出之间的差别来调整网络中的权重。





# 感知器

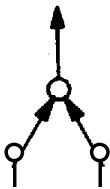
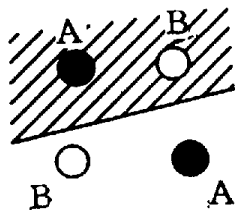
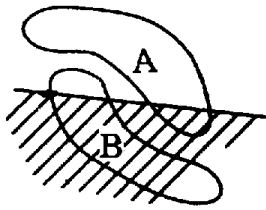

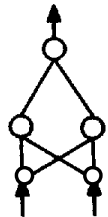
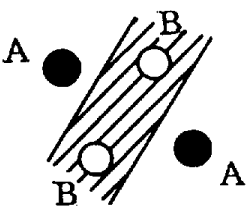
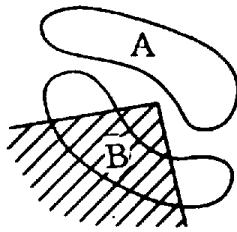

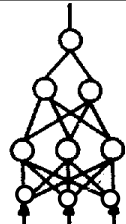
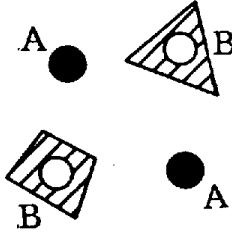
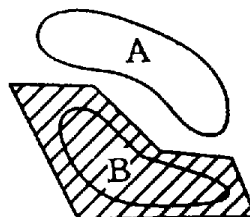
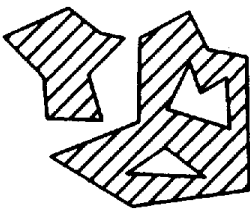
## ● Hebb学习规则

- 神经网络具有学习功能。对于人工神经网络而言，这种学习归结为神经元连接权的变化。
- Hebb学习规则调整 $w_{ij}$ 的原则：若第 $i$ 个神经元 $u_i$ 和第 $j$ 个神经元 $u_j$ 同时处于兴奋状态，则它们之间的连接 $w_{ij}$ 应当加强，即：

$$\Delta w_{ij} = \eta x_i y_j$$

这一规则与“条件反射”学说一致，并已得到神经细胞学说的证实。 $\eta$ 是表示学习速率的比例常数(学习率)。

# 感知器小结

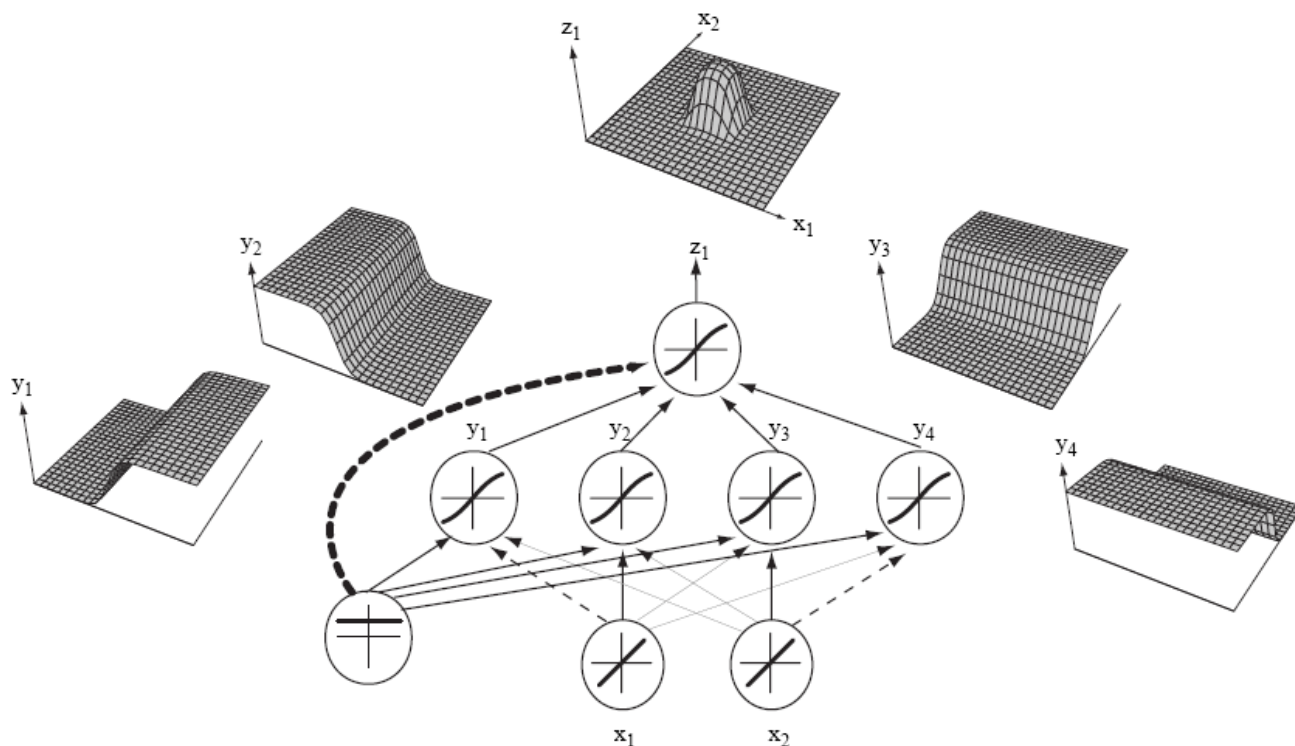
感知器结构	异或问题	复杂问题	判决域形状	判决域
无隐层 				半平面
单隐层 				凸 域
双隐层 				任意复杂 形状域

**理论证明，三层感知器可以实现任意的逻辑运算，在激活函数为Sigmoid函数的情况下，可以逼近任何非线性多元函数。**

# 感知器小结

## ● 输出函数：S型函数

若给出足够多隐单元，任何从输入到输出的连续函数都可用三层神经网络以任意精度近似。



# 感知器

## ● 小结

- 感知器的学习过程与求解线性判决函数(单样本感知器算法)的过程等价；
- 两层感知器(一个输入层和一个输出层)只能解决线性可分问题；
- 只要隐层和隐层单元数足够多，多层感知器网络可实现任何模式分类；
- 确定多层网络的权值，即网络如何进行学习，在感知器上没有得到解决。

# 人工神经网络典型结构

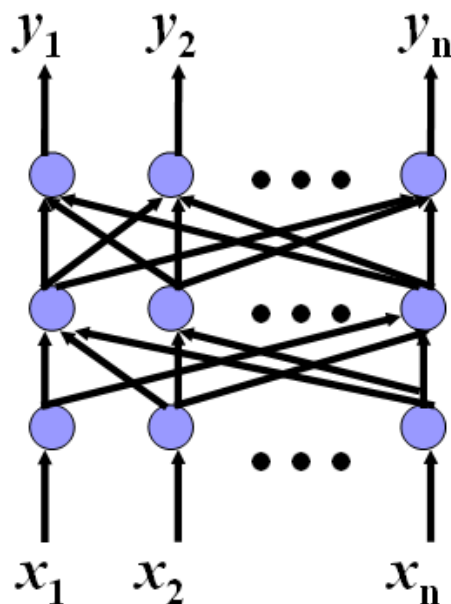
## ● 人工神经网络连接的几种基本形式

- 前向网络
- 从输出到输入有反馈的前向网络
  - 用来存储某种模式序列
- 层内互连前向网络
  - 限制层内同时动作的神经元
- 反馈型全互联网络和反馈型局部互联网络

# 人工神经网络典型结构

## ● 前向网络

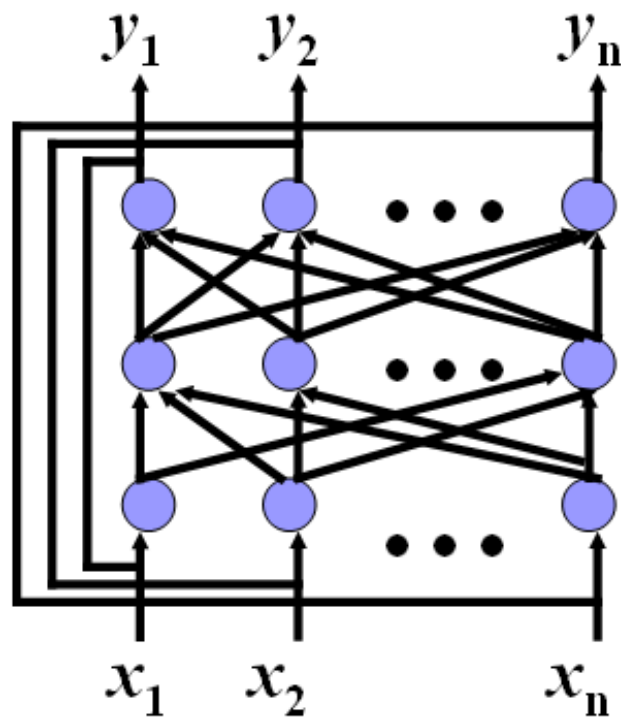
- 最初称之为感知器。应用最广泛，最主要原因是具有BP学习方法。
- 前向网络结构是分层的，信息只能从前一层单元传递到当前层单元。



# 人工神经网络典型结构

## ● 从输出到输入有反馈的前向网络

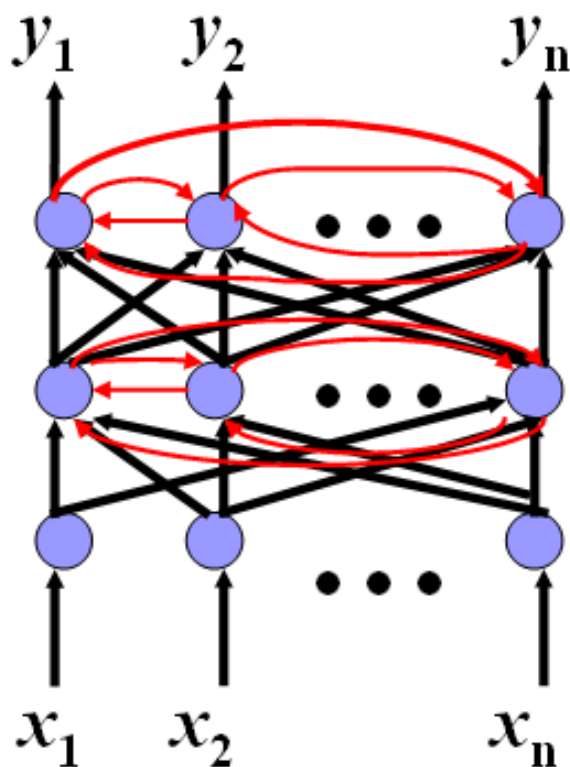
- 输出层上存在一个反馈回路，将信号反馈到输入层。而网络本身还是前馈型的。



# 人工神经网络典型结构

## ● 层内互连前向网络

- 外部看还是一个前向网络，内部有很多自组织网络在层内互联。





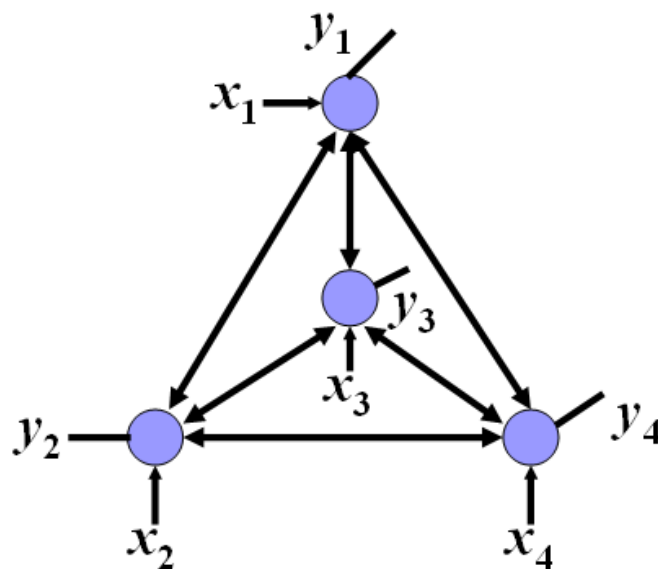
# 人工神经网络典型结构

## ● 反馈型全互联网络

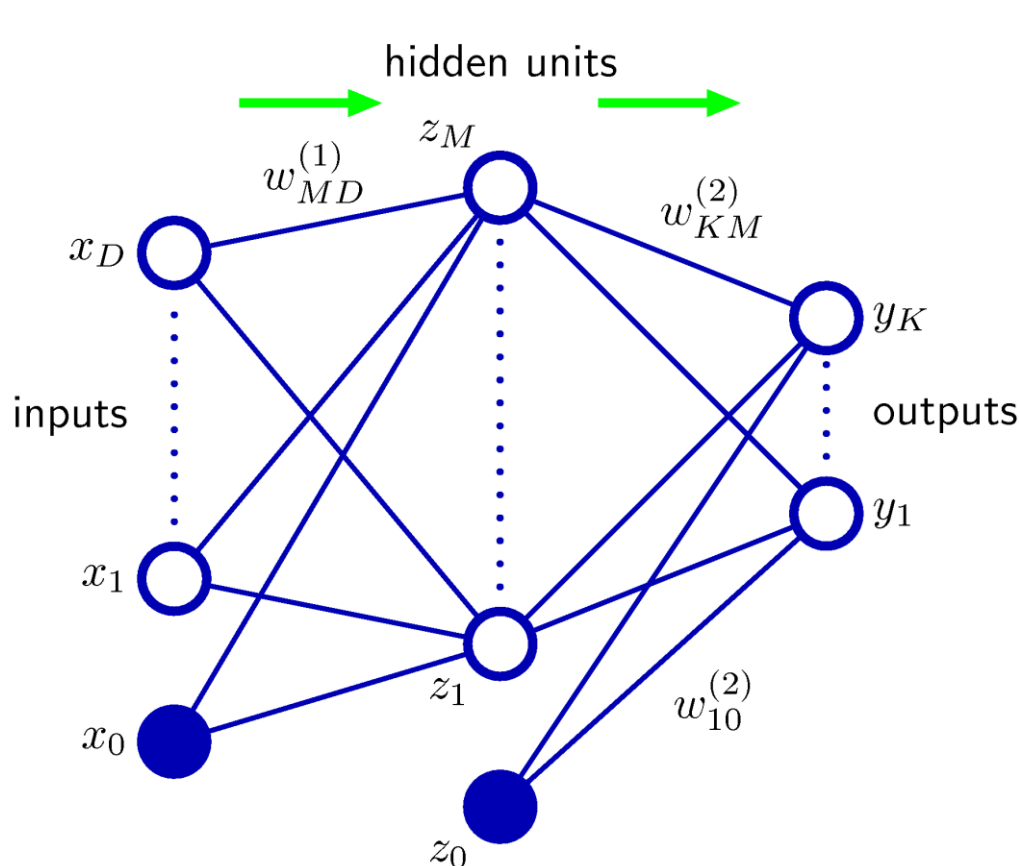
- 所有计算单元之间都有联接，如：Hopfield网络。

## ● 反馈型局部联接网络

- 特例，每个神经元的输出只与其周围的神经元相连，形成反馈网络。



# 前馈神经网络



$$a_j = \sum_{i=0}^D w_{ji}^{(1)} x_i$$

$$z_j = h(a_j)$$

$$a_k = \sum_{j=0}^M w_{kj}^{(2)} z_j$$

$$y_k = \sigma(a_k)$$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left( \sum_{j=0}^M w_{kj}^{(2)} h \left( \sum_{i=0}^D w_{ji}^{(1)} x_i \right) \right)$$

# 如何训练神经网络？

- 定义准则函数

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2$$

寻找一个  $\mathbf{w}$ ，使得  $E(\mathbf{w})$  最小。

$$\nabla E(\mathbf{w}) = 0$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

# 反向传播(BP)算法

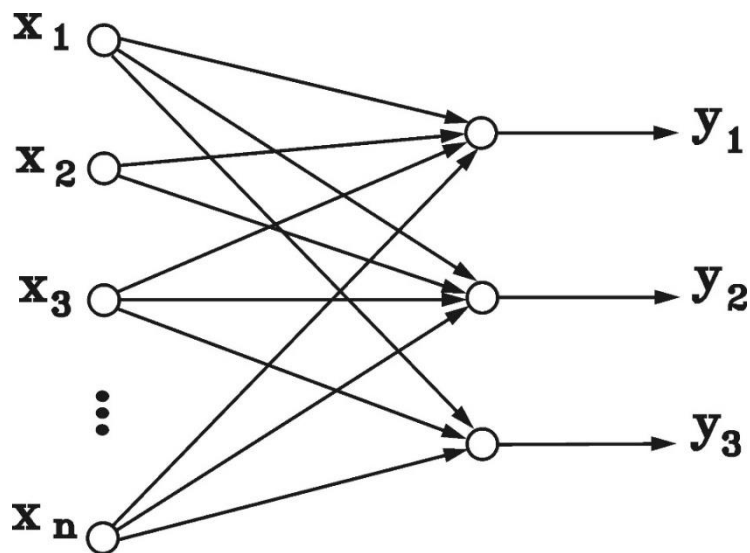
感知器算法实际上是在利用**理想输出与实际输出之间的误差作为增量来修正权值**，然而多层感知器只能计算出输出层的误差，中间隐层由于不直接与外界连接，其误差无法估计。

$$w_{ij}(t+1) = w_{ij}(t) + \eta(y_j - \tilde{y}_j)x_i$$

**反向传播算法思想**：从后向前反向逐层传播输出层的误差，以**间接计算隐层的误差**。算法可以分为两个阶段：

- **前馈(正向过程)**：从输入层经隐层逐层正向计算各单元的输出；
- **学习(反向过程)**：由输出误差逐层反向计算隐层各单元的误差，并用此误差修正前层的权值。

# 反向传播(BP)算法



$$y_k = \sum_i w_{ki} x_i$$

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w})$$

误差




$$\frac{\partial E}{\partial w_{ki}} = (y_k - t_k) x_i$$

# 反向传播(BP)算法

## ● 练习:

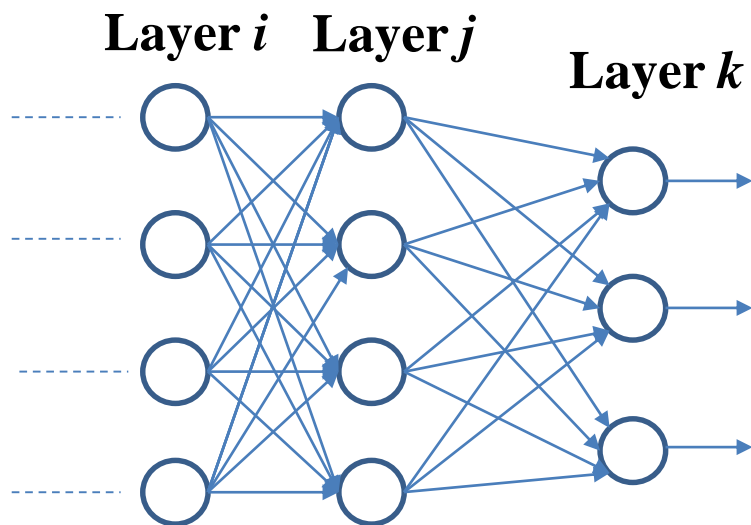
对非线性  $y_k = h(a_k) = h(\sum_i w_{ki}x_i)$

计算  $\frac{\partial E}{\partial w_{ki}}$       $E = \frac{1}{2} \sum_k (y_k - t_k)^2$

 
$$\begin{aligned}\frac{\partial E}{\partial w_{ki}} &= (y_k - t_k) \frac{\partial y_k}{\partial w_{ki}} \\ &= (y_k - t_k) \frac{\partial y_k}{\partial a_k} \frac{\partial a_k}{\partial w_{ki}} \\ &= (y_k - t_k) h'(a_k) x_i\end{aligned}$$

# 反向传播(BP)算法

$$a_j = \sum_i w_{ji} z_i \quad z_j = h(a_j)$$



$$\frac{\partial E}{\partial w_{ji}} = \boxed{\frac{\partial E}{\partial a_j}} \frac{\partial a_j}{\partial w_{ji}}$$

$$\delta_j \equiv \frac{\partial E}{\partial a_j} \quad \frac{\partial a_j}{\partial w_{ji}} = z_i$$

$$\rightarrow \frac{\partial E}{\partial w_{ji}} = \delta_j z_i$$

# 反向传播(BP)算法

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2 \quad a_j = \sum_i w_{ji} z_i \quad z_j = h(a_j)$$

$$\delta_j \equiv \frac{\partial E}{\partial a_j} \quad \frac{\partial a_j}{\partial w_{ji}} = z_i \quad \frac{\partial E}{\partial w_{ji}} = \delta_j z_i$$

对于输出层而言:  $\delta_k \equiv \frac{\partial E}{\partial a_k} = y_k - t_k$

对于隐藏层而言:  $\delta_j \equiv \frac{\partial E}{\partial a_j} = \sum_k \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial a_j}$



$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$




# 反向传播(BP)算法

● 练习:  $a_k = \sum_j w_{kj} z_j \quad z_j = h(a_j)$

推导  $\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$

$$\delta_j = \sum_k \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j}$$


$$= \sum_k \delta_k \frac{\partial a_k}{\partial z_j} \frac{\partial z_j}{\partial a_j} = \sum_k \delta_k w_{kj} \frac{\partial h(a_j)}{\partial a_j}$$

$$= h'(a_j) \sum_k \delta_k w_{kj}$$

# 反向传播(BP)算法

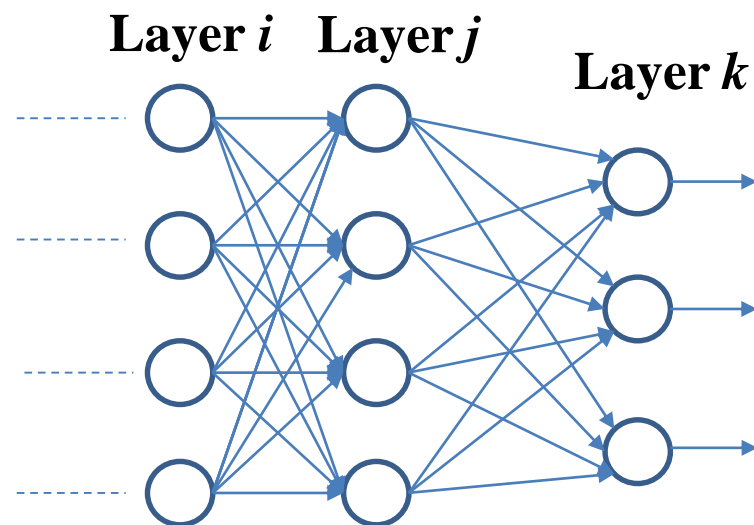
## ● 示例流程

- Step1: 选定权系数初值
- Step2: 重复下述过程直至收敛 (对各个样本依次计算)
  - Step2.1: 从前向后各层计算各单元输入值

$$a_j = \sum_i w_{ij} x_j \quad z_j = h(a_j) = \frac{1}{1 + e^{-a_j}}$$

- Step2.2: 对输出层计算  $\delta_k$

$$\delta_k = y_k - t_k$$



# 反向传播(BP)算法

## ● 示例流程

- Step2.3: 从后向前计算各隐层  $\delta_j$

$$\delta_j = z_j(1 - z_j) \sum_k w_{jk} \delta_k$$

- Step2.4: 计算并保存各个权值修正量

$$\Delta w_{ij} = \delta_j z_i$$

- Step2.5: 修正权值  $w^{\tau+1} = w^{\tau} + \eta \Delta w_{ij}$

以上算法是对每个样本做权值修正(单样本)

也可对各样本计算  $\delta$  后求和, 按总误差修正权值(批处理)

# 反向传播(BP)算法

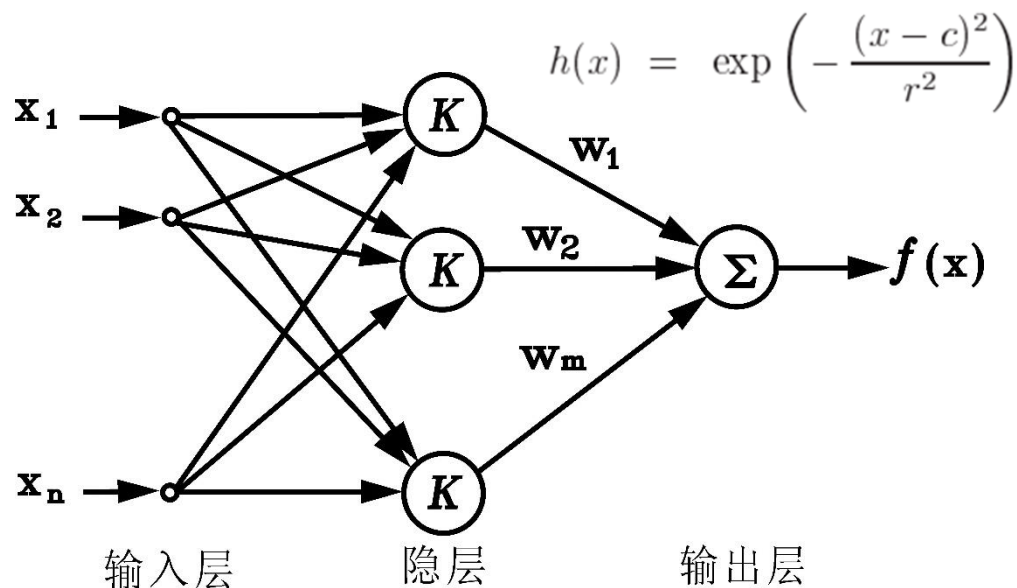
## ● 局限性

- 用梯度法求非线性函数极值，有可能陷入局部极小点，不能收敛到全局极小点。
- 如果权值初值都相同，隐层各单元无差异，运算不能正常进行。通常用较小随机数(例如在 $-0.3 \sim +0.3$ 之间的随机数)作为权值初值。初值对收敛有影响，当计算不收敛时，可以改变初始值调试。

# 其他神经网络

## ● 径向基函数神经网络(RBF网络)

- 只有一个隐层，隐层单元采用径向基函数作为其输出函数，输入层到隐层之间的权值均固定为1；输出节点为线性求和单元，隐层到输出节点之间的权值可调，因此，输出为隐层的加权求和。



# 其他神经网络

## ● 径向基函数神经网络(RBF网络)

- 在RBF网络中，从输入层到隐层的基函数输出是一种非线性映射，而输出则是线性的。这样，RBF网络可以看成是首先将原始的线性不可分特征空间变换到另一空间(通常是高维的)，通过此变换使样本在新空间中线性可分，然后用一个线性单元来解决。

把网络看成对未知函数 $f(x)$ 的逼近器。一般任何函数都可表示成一组基函数的加权和，这相当于用隐层单元的输出函数构成一组基函数来逼近 $f(x)$ 。

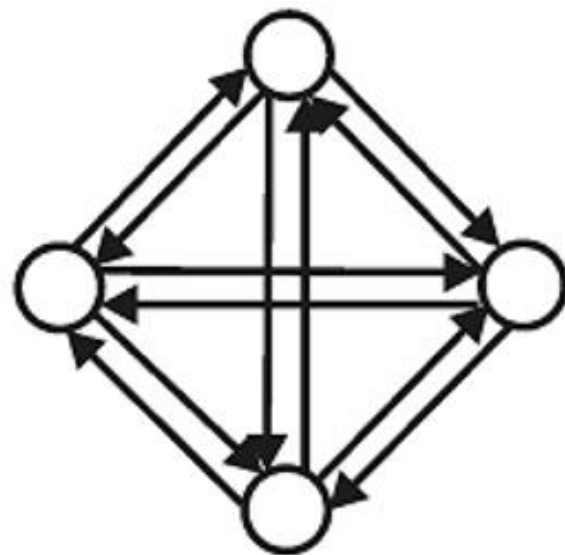
# 其他神经网络

## ● Hopfield网络

- 是一种反馈网络。反馈网络具有一般非线性系统的许多性质，如稳定性问题等，在某些情况下还有随机性、不可预测性。因此它比前馈网络的内容复杂。

除具有反馈网络的结构和性质外，还满足以下条件：

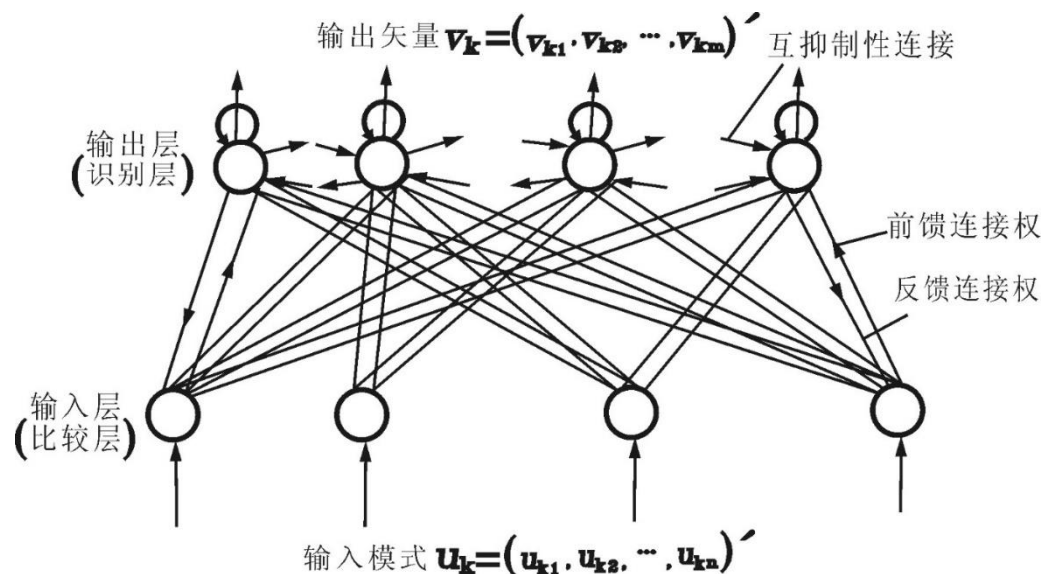
- (1) 权值对称，即  $\omega_{ij} = \omega_{ji}$ ，权矩阵  $W = W^T$ ，为对称阵。
- (2) 无自反馈，即  $\omega_{ii} = 0$ ，权矩阵  $W$  的对角线元素为0。



# 其他神经网络

## ● 自适应共振理论神经网络

- 通过反复将输入学习模式由输入向输出层自下而上短时记忆和由输出向输入层自上而下长期记忆和比较来实现。当记忆和比较达到共振时，输出矢量可正确反映输入学习模式的分类，且网络原有记忆不受影响。





# 其他神经网络

## ● 自组织特征映射神经网络

- 人脑的记忆不是神经元与记忆模式的一一对应，而是一群神经元对应一个模式。生理实验表明，某外界信息引起的兴奋刺激并不只针对一个神经细胞，而是对以某一神经元为中心的一个区域内各神经元的兴奋刺激，且刺激强度以区域中心为最大，随与中心距离的增大，强度逐渐减弱，远离区域中心的神经元受抑制。
- 自组织特征映射 (Self-Organizing Map, SOM) 网络可以很好地模拟人类大脑的功能区域性、自组织特性及神经元兴奋刺激规律。

# 其他神经网络

## ● 自组织特征映射神经网络

- 能将任意维输入模式在输出层映射成一或二维离散图形，并保持拓扑结构不变。
- 在输出层，获胜神经元邻域内的神经元在不同程度上都得到兴奋，而在邻域外的神经元都被抑制。
- 邻域可为任意形状，但一般是对称的。邻域是时间的函数，随时间增大而减小，最后可能剩下一个或一组神经元。最终得到的区域反映了一类输入模式的属性。

