第6章第2讲

人工神经网络

Artificial Neural Networks

向 世 明

smxiang@nlpr.ia.ac.cn

中科院自动化研究所　模式识别国家重点实验室

助教： 何文浩 (wenhao.he@nlpr.ia.ac.cn)
杨红明 (hongming.yang@nlpr.ia.ac.cn)

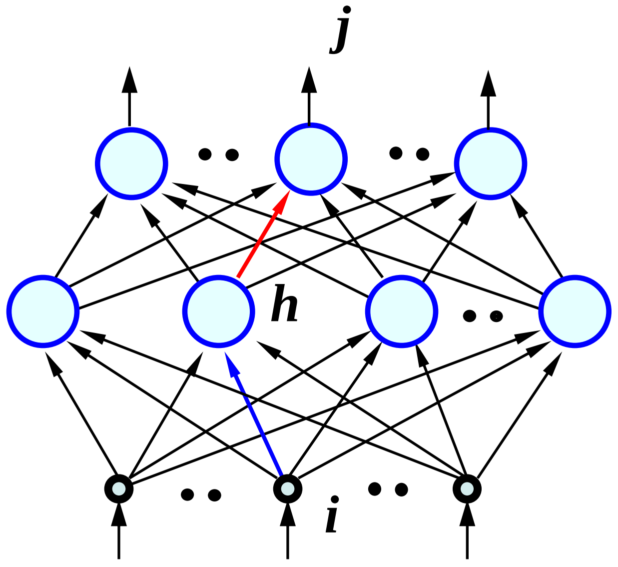
内容提要

- 介绍
  - 发展历史
  - 网络结构
- 基本模型
  - 单层感知器、多层感知器、RBF网络
- 扩展模型
  - Hopfield网络、RBM、DNN、CNN、Autoencoder、RNN、LSTM等

第四节　多层感知器

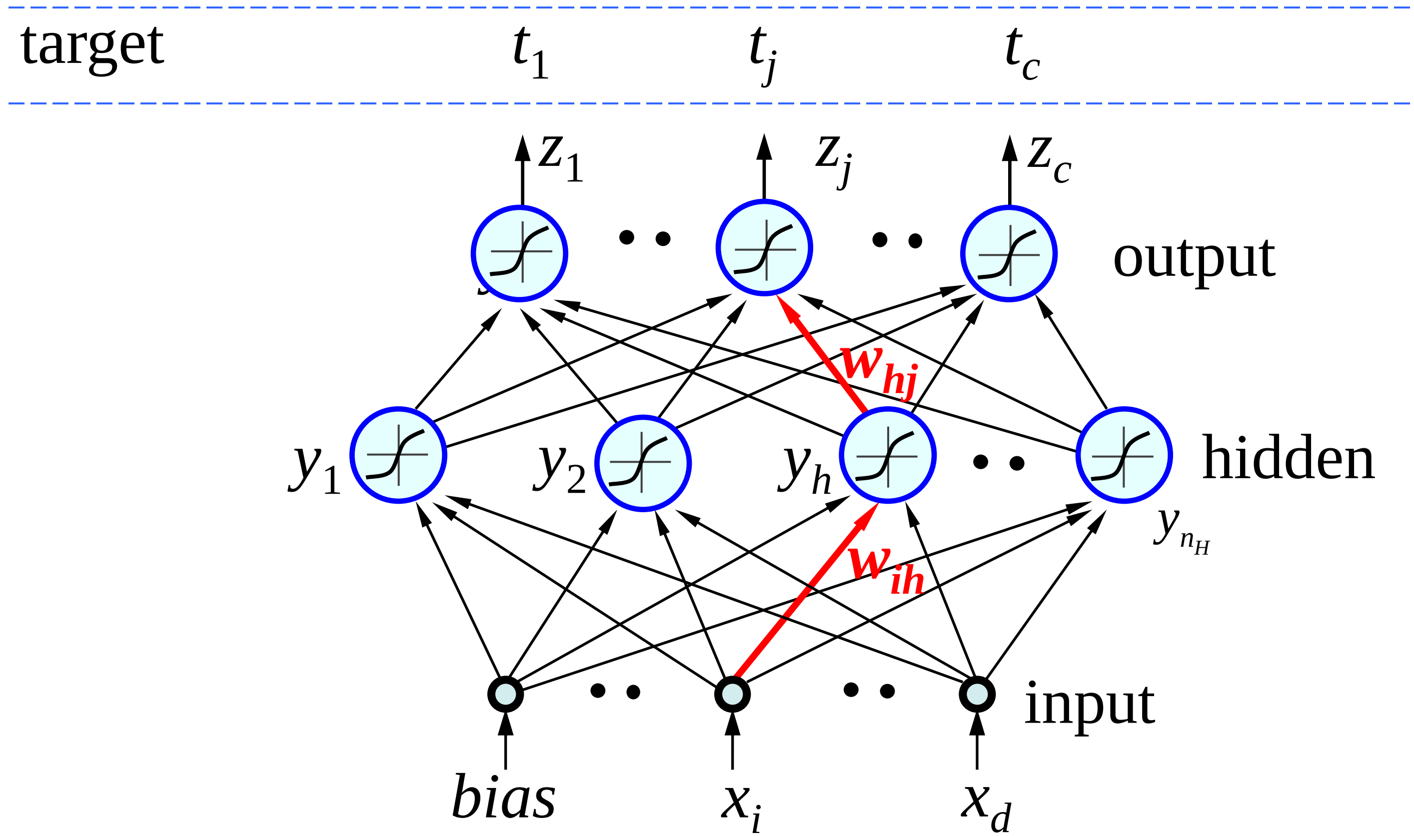
6.4.1 多层感知器

- 三层网络的描述
  - 训练数据输入输出对：$\{x_i^k, t_j^k\}$
  - 输出层结点的输出：z_j^k
  - 隐含层结点的输出：y_h^k
  - 输入信号：x_i^k
  - 输入端点数目：$d+1$
  - 输入层结点 i 至隐含层结点 h 的权重：w_{ih}
  - 隐含层结点 h 至输出层结点 j 的加权表示：w_{hj}
  - 上标 k 表示训练对的序号，$k=1, 2, \ldots, n$



6.4.1 多层感知器

- **Hope:** $z_1 \approx t_1, \cdots, z_c \approx t_c$, **for all samples:** $J(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{c}(t_j - z_j)^2 \approx 0$



- 网络描述-每个样本所经历的计算

上标 $\boldsymbol{k}$ 联系
第 $\boldsymbol{k}$ 个样本

对第 k 个样本，隐含层 h 结点的输入加权和为：

$$net_h^k = \sum_i w_{ih} x_i^k$$

经过激励，隐含层 h 结点的输出：

$$y_h^k = f(net_h^k) = f\left(\sum_i w_{ih} x_i^k\right)$$

输出层 j 结点的输入加权和为：

$$net_j^k = \sum_h w_{hj} y_h^k = \sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)$$

经过激励，输出层 j 结点的输出：

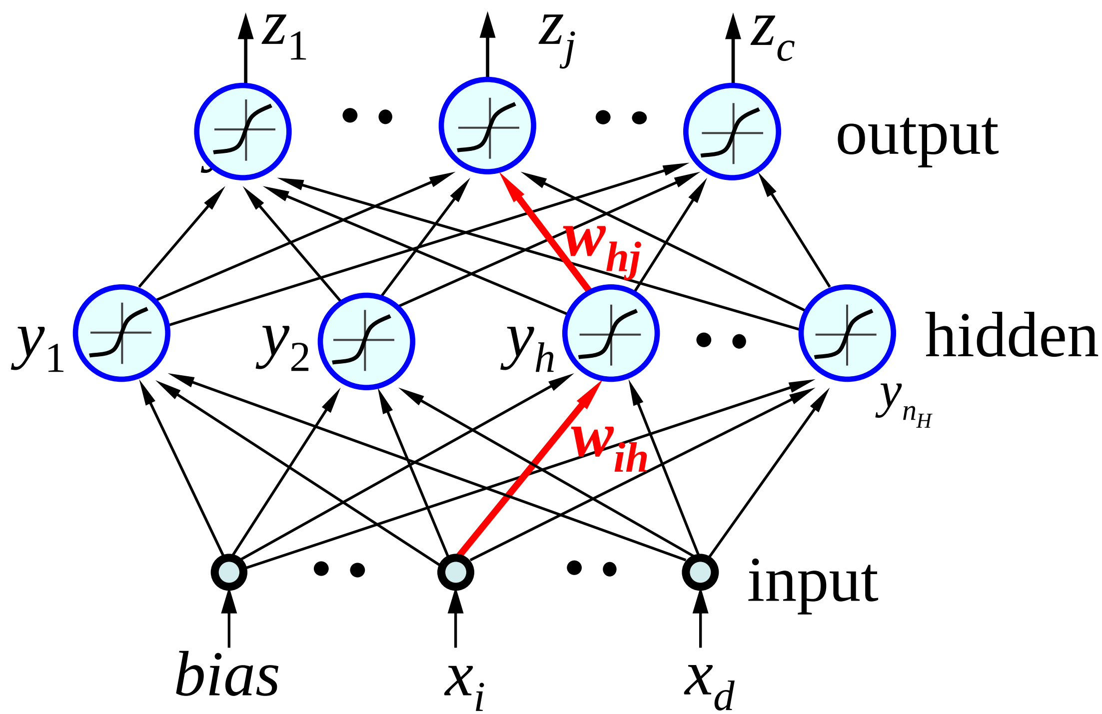
$$z_j^k = f(net_j^k) = f\left(\sum_h w_{hj} y_h^k\right) = f\left(\sum_h w_{hj} f(\sum_i w_{ih} x_i^k)\right)$$

- 网络描述–每个样本所经历的计算

上标k：第 k 个样本

$$z_j^k = f\left(\sum_h w_{hj} y_h^k\right)$$

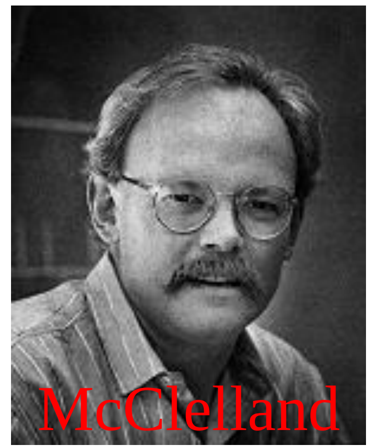
$$y_h^k = f\left(\sum_i w_{ih} x_i^k\right)$$



$$z_j^k = f(net_j^k) = f\left(\sum_h w_{hj} y_h^k\right) = f\left(\sum_h w_{hj} f(\sum_i w_{ih} x_i^k)\right)$$

6.4.2 误差反向传播(BP)算法

- D. Rumelhart，J. McClelland于1985年提出了误差反向传播(Back Propagation, BP)学习算法



- 基本原理
  - 利用输出后的误差来估计**输出层的前一层**的误差，再用这个误差估计更前一层的误差，如此一层一层地反传下去，从而获得所有其它各层的误差估计

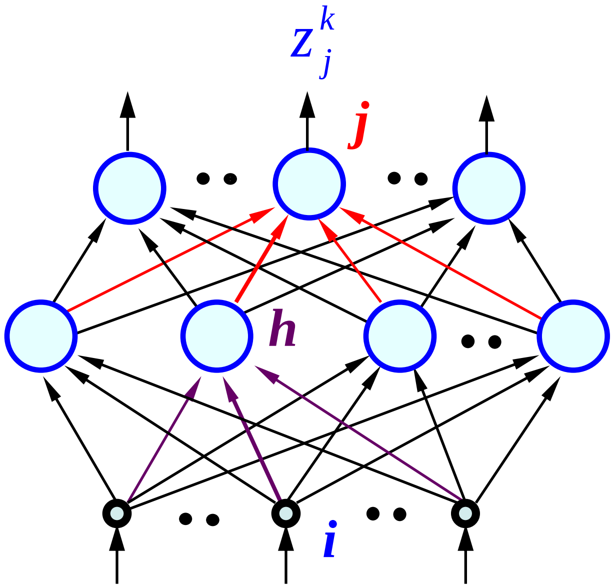
6.4.2 BP算法

- 误差反向传播训练算法
  - 属于监督学习算法，**通过调节各层的权重，使网络学会**由“输入-输出对”组成的训练组。
  - BP算法核心是梯度下降法。
  - 权重先从输出层开始修正，再依次修正各层权重
    - 首先修正：**“输出层至最后一个隐含层”**的连接权重
    - 再修正：**“最后一个隐含层至倒数第二个隐含层”**的连接权重，….
    - 最后修正：**“第一隐含层至输入层”**的连接权重。

学习的本质：对网络各连接权重作动态调整！

- 误差函数一单个样本

上标 k 联系
第 k 个样本



$$z_j^k = f\left(net_j^k\right) = f\left(\sum_h w_{hj} y_h^k\right)$$

$$y_h^k = f\left(net_h^k\right) = f\left(\sum_i w_{ih} x_i^k\right)$$

$$E(\mathbf{w})^k = J(\mathbf{w})^k = \frac{1}{2}\sum_j (t_j^k - z_j^k)^2$$

$$= \frac{1}{2}\sum_j \left(t_j^k - f\left(net_j^k\right)\right)^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} y_h^k\right)\right\}^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} f\left(net_h^k\right)\right)\right\}^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)\right\}^2$$

- # 复合函数求导数

设 $f(x)=g_1(g_2(g_3(\cdots g_n(x))))$,

令 $h_2=g_2(g_3(\cdots g_n(x)))$,

$h_3=g_3(g_4(\cdots g_n(x))),\cdots$

$$\frac{\partial f(x)}{\partial x}=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2}{\partial x}=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2(h_3)}{\partial h_3}\frac{\partial h_3}{\partial x}=\cdots\cdots=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2(h_3)}{\partial h_3}\cdots\frac{\partial g_n(x)}{\partial x}$$

设 $f(x)=g_1(g_2(g_3(\cdots g_n(x)))+t_2(t_3(\cdots t_m(y))))$,

令 $H_2=g_2(g_3(\cdots g_n(x)))+t_2(t_3(\cdots t_m(y)))$,

$h_2=g_2(g_3(\cdots g_n(x)))$,

$h_3=g_3(g_4(\cdots g_n(x))),\cdots$

$$\frac{\partial f(x)}{\partial x}=\frac{\partial g_1(H_2)}{\partial H_2}\frac{\partial H_2}{\partial x}$$

$$=\frac{\partial g_1(H_2)}{\partial H_2}\left(\frac{\partial h_2(h_3)}{\partial h_3}\frac{\partial h_3}{\partial x}+0\right)$$

$$=\cdots\cdots$$

$$=\frac{\partial g_1(H_2)}{\partial H_2}\frac{\partial h_2(h_3)}{\partial h_3}\cdots\frac{\partial g_n(x)}{\partial x}$$

- 网络训练：隐含层—输出层

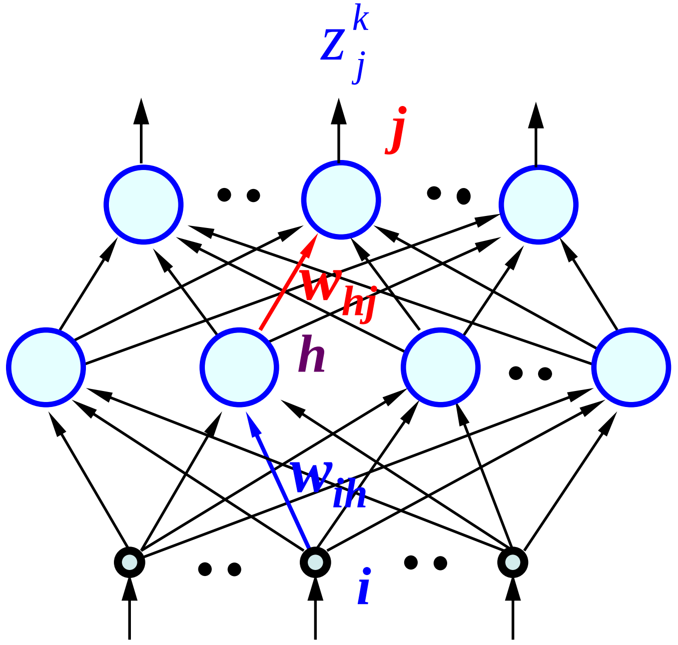
隐含层到输出层的连接权重调节量：

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}} = -\eta \sum_k \frac{\partial E}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{hj}}$$

$$= \eta \sum_k (t_j^k - z_j^k) f'(net_j^k) y_h^k$$

$$= \eta \sum_k \delta_j^k y_h^k$$

(δ规则)



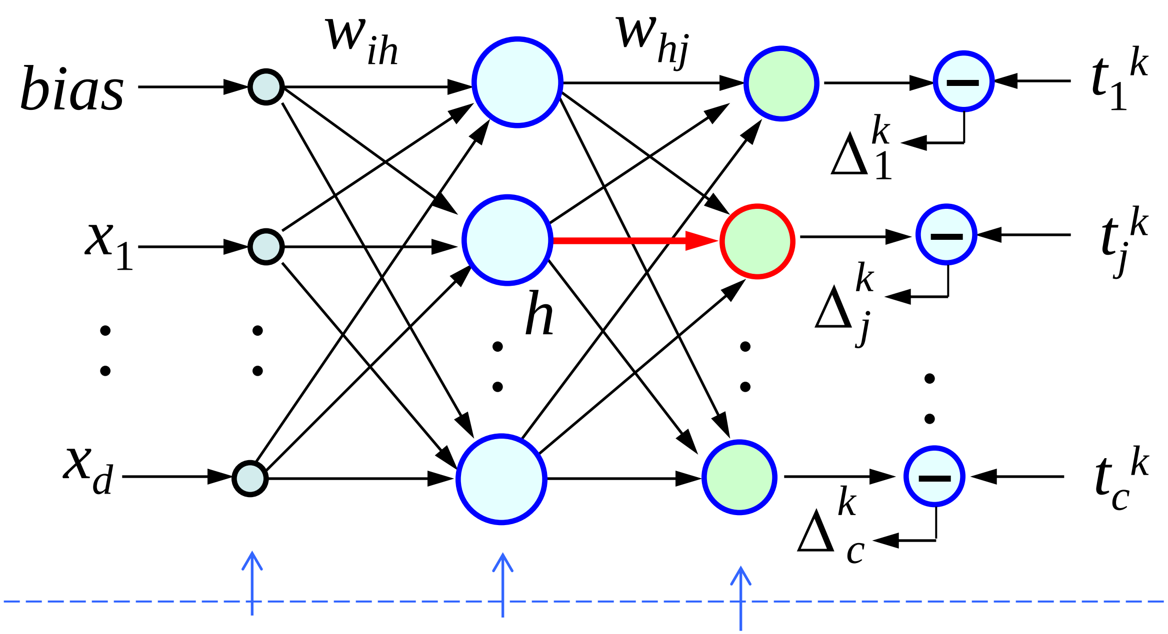
$$\delta_j^k = \frac{\partial E}{\partial net_j^k} = f'(net_j^k)(t_j^k - z_j^k) = f'(net_j^k)\Delta_j^k,$$

边的指向结点的误差信号(局部梯度)

$$\boxed{\Delta_j^k = t_j^k - z_j^k}$$

- 隐含层—输出层，**准备好输出层的误差：** $\Delta_j^k = t_j^k - z_j^k$

样本→：　x_i^k　→ y_h^k　→ z_j^k　−　t_j^k



下标联系的结点　***i***　***h***　***j***

注：上标 k 联系第 k 个样本

- 隐含层—输出层：第 k 个训练样本对权重 w_{hj} 的贡献

$h \rightarrow j$, for sample k:

δ规则：

$$\Delta w_{hj} \,|_{\text{sample } k} = \eta \delta_j^k y_h^k$$

权重所联边的起始结点（隐含结点 h）的输出

权重所联边的指向结点（输出结点 j）收集到的误差信号

$$\delta_j^k = f'(net_j^k)\Delta_j^k, \quad \Delta_j^k = t_j^k - z_j^k$$

误差**在权重所联边的指向结点处**计算。

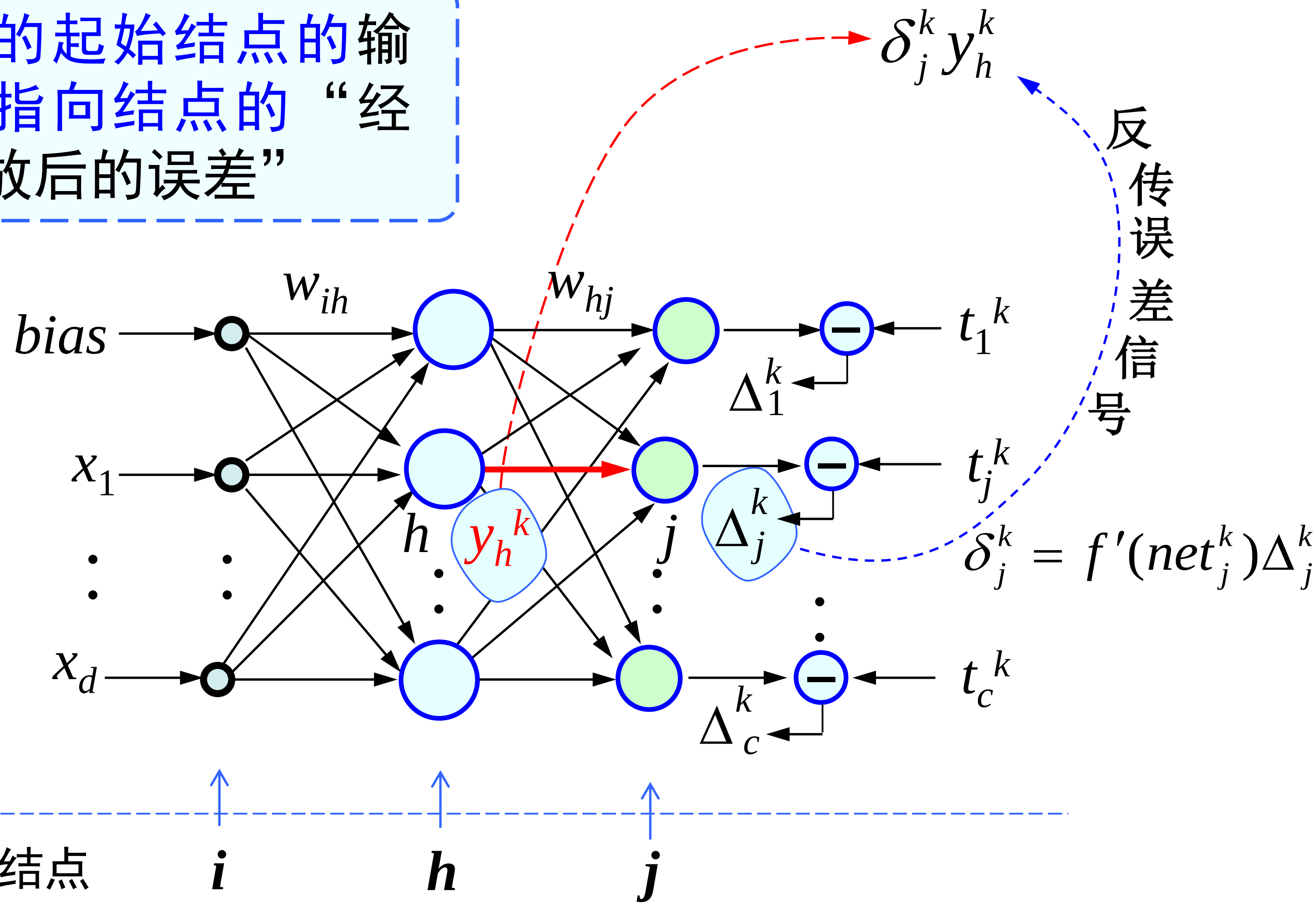
误差大小等于：该结点收集到的误差乘以激励函数对“该结点加权和”的导数。

中国科学院大学
University of Chinese Academy of Sciences

误差反传与权重更新：

第 k 个样本对权重更新的贡献

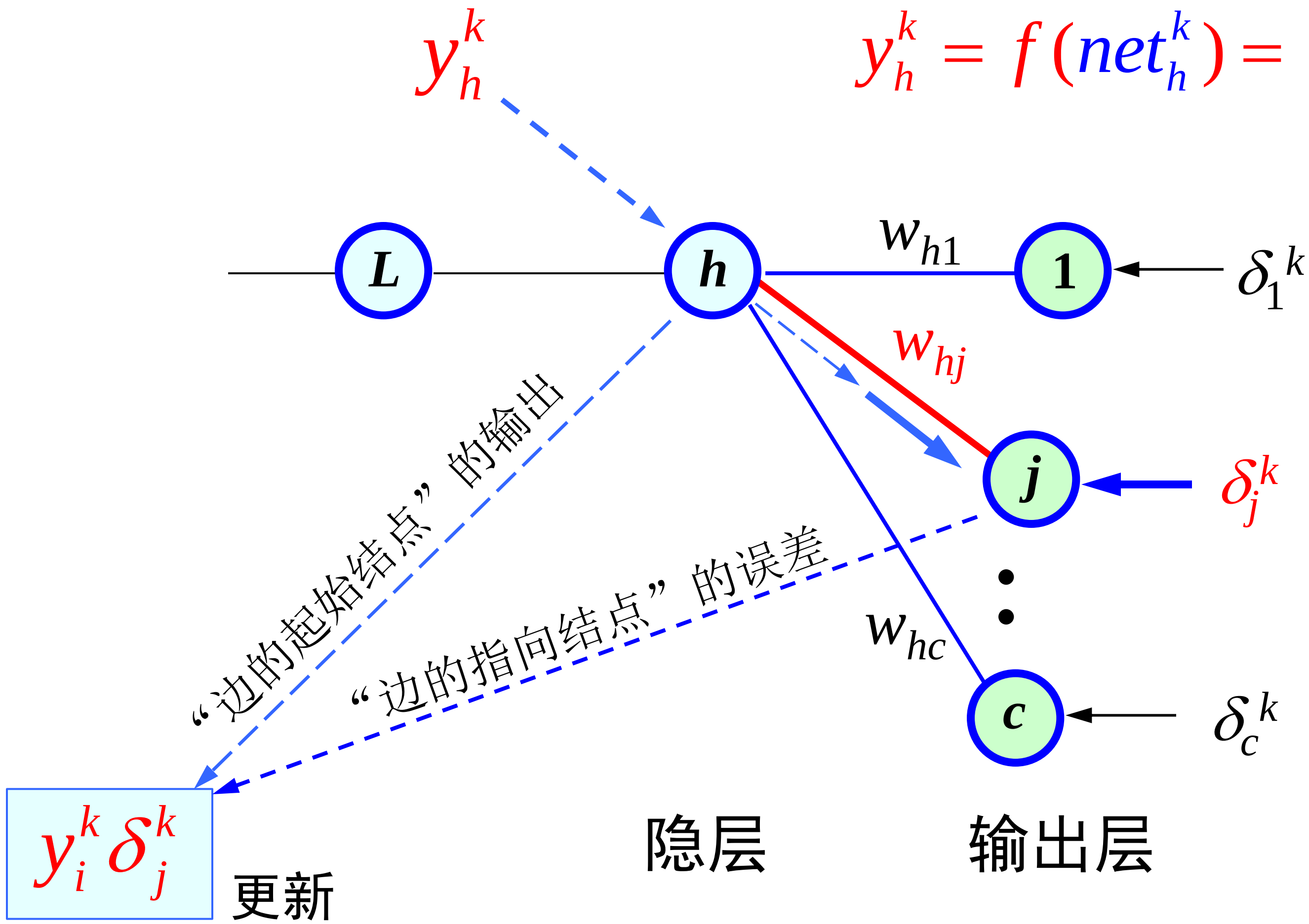
所联边的起始结点的输出乘以指向结点的“经导数缩放后的误差”



隐层-输出权重更新示意：

w_{hj}所连接的边的起始结点向外传递的信号值

$$y_h^k = f(net_h^k) = f\left(\sum_i w_{ih} x_i^k\right)$$



w_{hj}所连接的边的指向结点的误差

6.4.2 BP算法

- 激励函数采用sigmoid函数(最常用):

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

$$f'(s)=\frac{e^{-s}}{(1+e^{-s})^2}=\frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right)=z(1-z),\quad z=f(s)$$

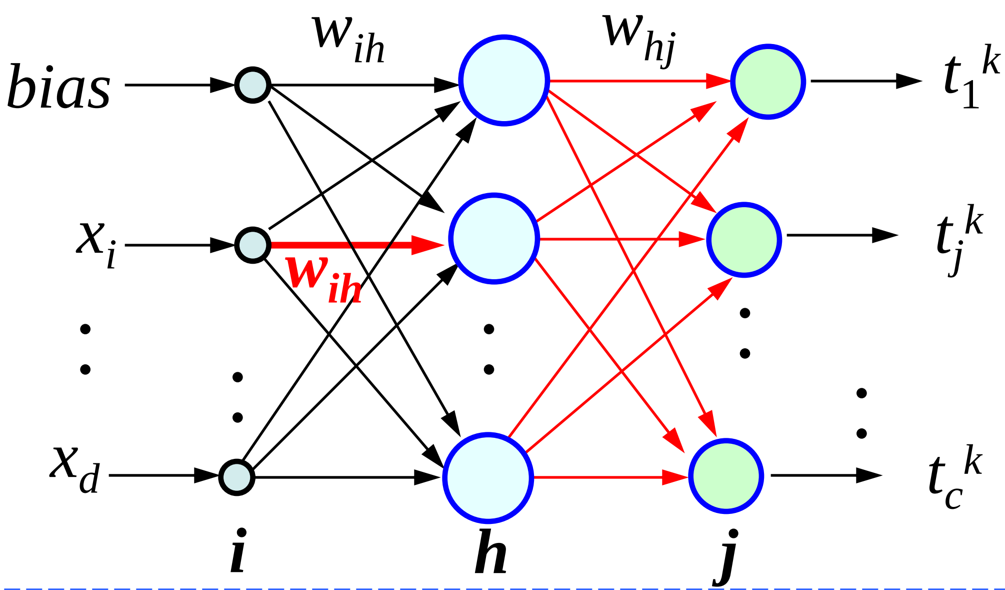
对输出层结点 j, 我们有:

$$\delta_j^k=\frac{\partial E}{\partial net_j^k}=f'\left(net_j^k\right)\left(t_j^k-z_j^k\right)=z_j^k\left(1-z_j^k\right)\left(t_j^k-z_j^k\right)$$

输入层至隐含层权重更新：

对于**输入层到隐含层**结点连接的边的权重修正量 Δw_{ih}，必须考虑将 $E(\mathbf{w})$ 对 w_{ih} 求导，需利用分层链路法。

$$E(\mathbf{w})=\sum_{k}J(\mathbf{w})=\frac{1}{2}\sum_{k,j}(t_j^k-z_j^k)^2=\frac{1}{2}\sum_{k,j}\left\{t_j^k-f\left(\sum_{h}w_{hj}f\left(\sum_{i}\boxed{w_{ih}}x_i^k\right)\right)\right\}^2$$



样本→： x_i^k → net_h^k → y_h^k → net_j^k → z_j^k t_j^k

输入层至隐含层权重更新：

用别名$\tilde{h}$代替h

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k,j}(t_j^k - z_j^k)^2 = \frac{1}{2}\sum_{k,j}\left\{t_j^k - f\left(\sum_{\tilde{h}} w_{\tilde{h}j} f\left(\sum_i \boxed{w_{i\tilde{h}}} x_i^k\right)\right)\right\}^2$$

z_j^k　net_j^k　$y_{\tilde{h}}^k$　$net_{\tilde{h}}^k$

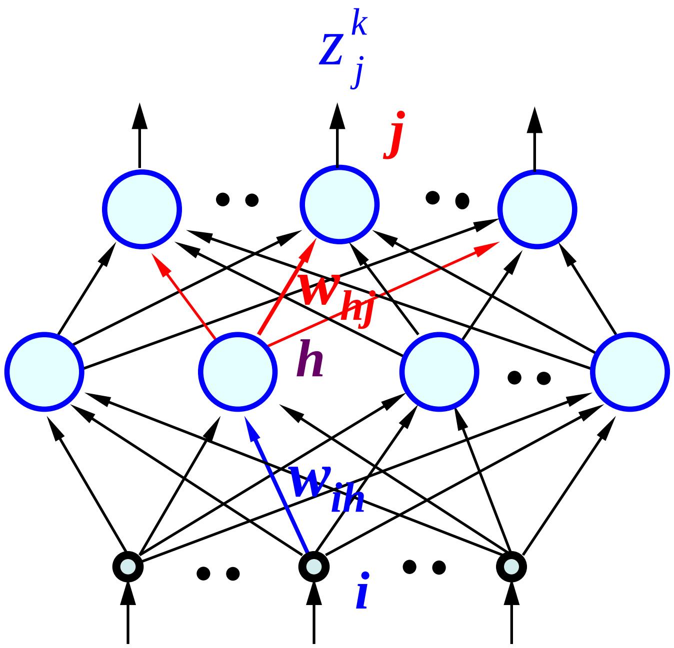
包含权重w_{ih}

$$net_{\tilde{h}}^k = \sum_i w_{i\tilde{h}} x_i^k, \quad y_{\tilde{h}}^k = f(net_{\tilde{h}}^k)$$

$$net_j^k = \sum_{\tilde{h}} w_{\tilde{h}j} y_{\tilde{h}}^k, \quad z_j^k = f(net_j^k)$$

待更新权重w_{ih}的增量：

$$\Delta w_{ih} = -\eta \frac{\partial E}{\partial w_{ih}} = -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial w_{ih}}$$



$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{ih}}$$

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

(链式法则) $$= -\eta \sum_{k,j} \frac{\partial E}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

待更新权重的增量（具体化）

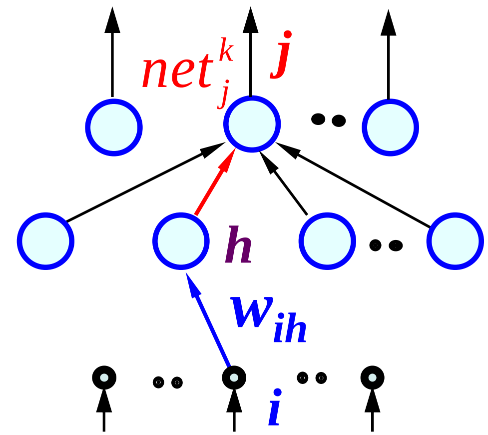
$$\Delta w_{ih} = -\eta \frac{\partial E}{\partial w_{ih}} = -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \cdot \frac{\partial z_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) \frac{\partial z_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) \frac{\partial z_j^k}{\partial net_j^k} \boxed{\frac{\partial net_j^k}{\partial w_{ih}}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) \boxed{\frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial w_{ih}}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) w_{hj} \frac{\partial y_h^k}{\partial w_{ih}}$$



$$net_j^k = \sum_{\tilde{h}} w_{\tilde{h}j} y_{\tilde{h}}^k$$

（只有当 $\tilde{h} = h$ 时 y_h^k 才包含 w_{ih}）

（接前一页）

$$\Delta w_{ih} = \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) w_{hj} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) w_{hj} \frac{\partial y_h^k}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) w_{hj} f'\left(net_h^k\right) x_i^k$$

$$= \eta \sum_{k,j} \delta_j^k w_{hj} f'(net_h^k) x_i^k$$

$$\boxed{\delta_j^k = f'\left(net_j^k\right)\left(t_j^k - z_j^k\right)}$$

$$= \eta \sum_{k} \left(f'(net_h^k) \sum_j \delta_j^k w_{hj} \right) x_i^k$$

$$= \eta \sum_{k} \delta_h^k x_i^k$$

$$\downarrow$$

$$\boxed{\delta_h^k = \frac{\partial E}{\partial net_h^k} = f'(net_h^k) \sum_j w_{hj} \delta_j^k = f'(net_h^k) \Delta_h^k, \quad \Delta_h^k = \sum_j w_{hj} \delta_j^k}$$

输入-隐层：第 k 个训练样本对权重 w_{ih} 的贡献

$i \rightarrow h$, for sample k:

δ规则：

$$\boxed{\Delta w_{ih}\,|_{sample\ k} = \eta \delta_h^k x_i^k}$$

w_{ih}所连接的边的起始结点（输入层结点 i）的输出(此时即为样本第 i 个分量)

w_{ih}所连接的边的指向结点（隐含结点 h）收集到的误差信号

输入-隐层：第 k 个训练样本对权重 w_{ih} 的贡献

$$\delta_h^k = \frac{\partial E}{\partial net_h^k} = \boxed{f'(net_h^k)}\boxed{\sum_j w_{hj}\delta_j^k}$$

从前一层收集误差：加权和

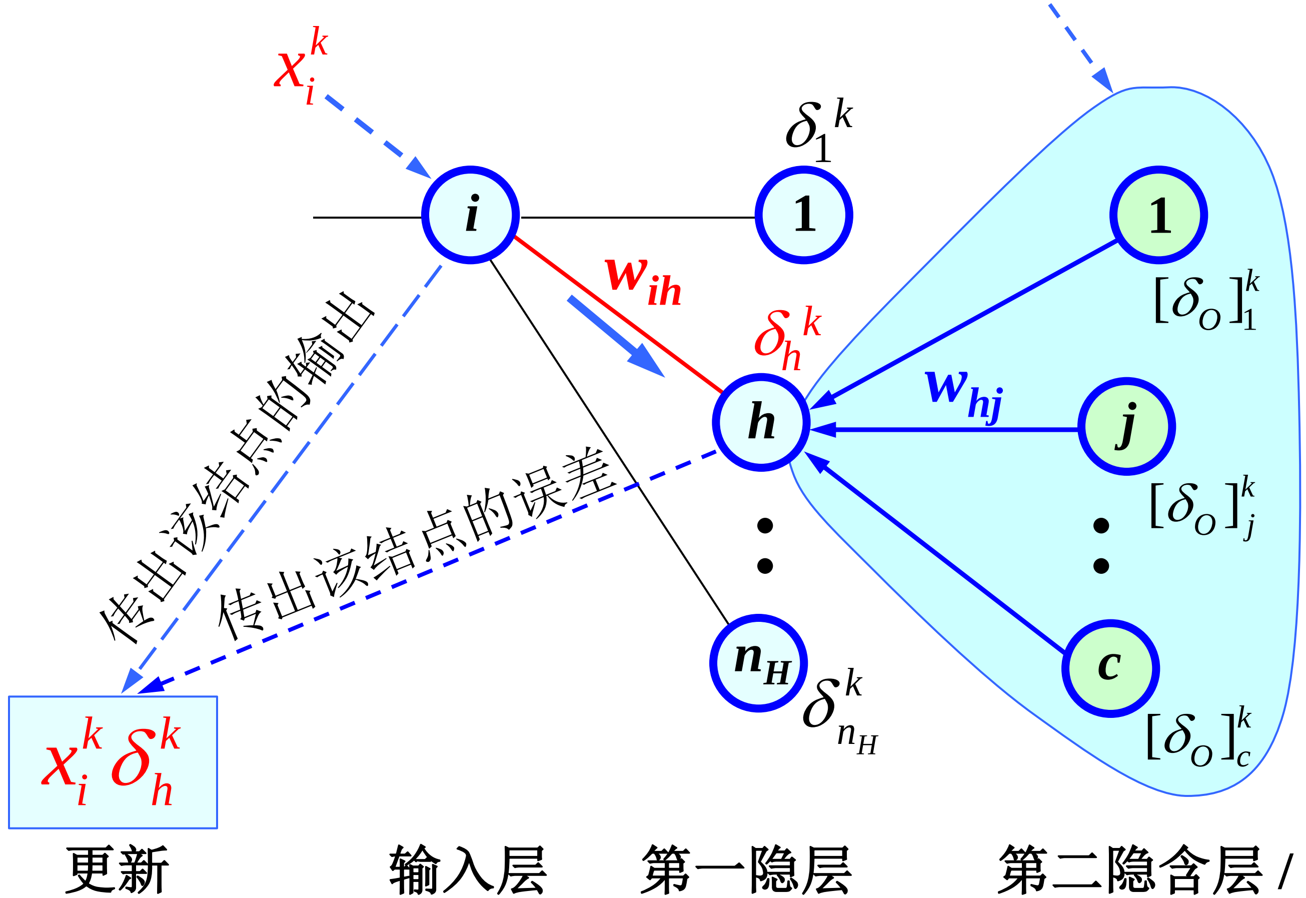
误差**在权重所联边的指向结点处**计算。

误差大小等于：该结点收集到的误差乘以激励函数对“该结点加权和”的导数。

输入—隐层权重更新示意：

误差收集：$\delta_h^k = f'(net_h^k)\sum_j w_{hj}[\delta o]_j^k$

注：已将上页中的 δ 更为 δ_O（对输出层）



x_i^k：边i-h起点的输出，即向外传递的信号值

6.4.2 BP算法

- 网络训练
  - BP算法对任意层的加权修正量的一般形式：

$$\Delta w_{in\to o} = \eta \sum_{all\ samples} \delta_o y_{in}$$

  - 单个训练样本的贡献：

从后一层各结点h收集误差

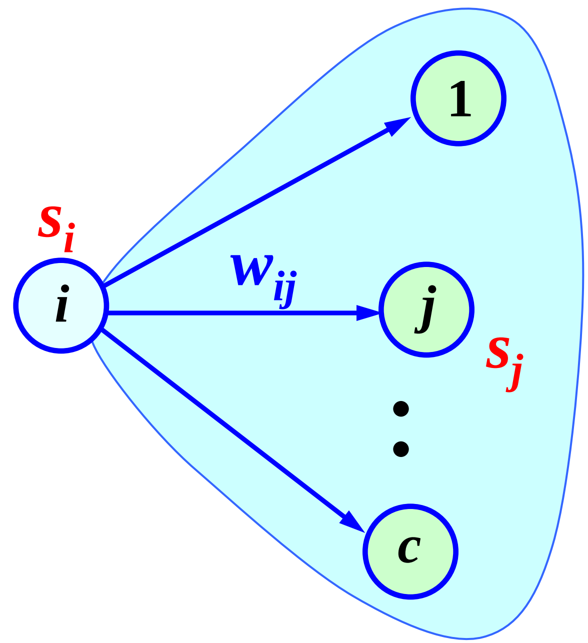
$$\Delta w_{in\to o} = \eta \cdot \delta_o \cdot y_{in} = \eta \cdot \left(\sum_h w_{o\to h} [\delta_o]_h \right) \cdot y_{in}$$

下标 in 和 o 分别指**“待更新权重”**所连边的起始结点和指向结点，y_{in}代表起始结点的实际输出，δ_o表示指向结点的误差 (由后一层收集得到)。

中国科学院大学
University of Chinese Academy of Sciences

- 网络训练
  - 从更一般的角度来认识网络
    - 目标函数不是误差平方损失，比如交叉熵、softmax、hinge loss等，对于权重更新是否有上述同样的文字表述？
  - 目标函数对某一层结点 i 的输出 s_i 的梯度（见右上）。
  - 目标函数对权重的梯度(导数)：

$$\frac{\partial E}{\partial s_i} = \sum_{j \in N(i)} \frac{\partial E}{\partial s_j} \frac{\partial s_j}{s_i}$$



$$\nabla_{w_{ij}} E = \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}}$$

$j \in \{$结点 i 指向的所有结点$\}$

中国科学院大学 University of Chinese Academy of Sciences

6.4.2 BP算法

- 随机更新

Stochastic Backpropagation

begin initialize: n_H , **w**, η , criterion θ, k=0

  do $k \leftarrow k+1 \pmod{n}$

    $\mathbf{x}^k$, randomly chosen a sample (pattern)

    $w_{hj} \leftarrow w_{hj} + \eta \delta_j^k y_h^k, \quad w_{ih} \leftarrow w_{ih} + \eta \delta_h^k x_i^k$

  until $\| \nabla J(\mathbf{w}) \| < \theta$

  return **w**

end

适用：超大规模数据

- # 批量更新算法

Batch Backpropagation

begin initialize: n_H, $\mathbf{w}$, η, criterion θ, $r=0$

  do $r \leftarrow r+1$ (increment epoch)

    $k=0$, $\Delta w_{ih}=0$, $\Delta w_{hj} \neq 0$

    do $k \leftarrow k+1 \pmod{n}$

      $\mathbf{x}^k$, selected a sample (pattern)

      $\Delta w_{hj} \leftarrow \Delta w_{hj} + \eta \delta_j^k y_h^k$, $\Delta w_{ih} \leftarrow \Delta w_{ih} + \eta \delta_h^k x_i^k$

    until $k=n$

    $\boxed{w_{hj} \leftarrow w_{hj} + \Delta w_{hj}, \quad w_{ih} \leftarrow w_{ih} + \Delta w_{ih}}$

  until $\|\nabla J(\mathbf{w})\| < \theta$ // 所有样本完成之后再更新

  return $\mathbf{w}$

end

第五节　BP算法讨论

6.5.1 准则函数

- 预测问题（回归问题）
- 分类问题
  - 对于模式分类问题，假定其类别数为 c，通常输出层的结点个数为 c 。
  - 对于训练样本 $\mathbf{x}$，如果它属于第 i 类，则其目标值可以定义为一个 c 维向量，该向量只有第 i 个元素为1，其余元素的值均为 0 (或-1)。这些值分别按序分配给 c 个输出结点。
  - 在人工神经网络中，通常称为 one-hot vectors: A one-hot vector is a vector which is 0 in most dimensions, and 1 in a single dimension.

6.5.1 准则函数

- 常用的准则函数

平方误差准则 (最常用)：	$E(\mathbf{w}) = \sum_{k,j} \left(t_j^k - z_j^k \right)^2$
交叉熵准则：	$E_{ce}(\mathbf{w}) = \sum_{k,j} t_j^k \ln\left(t_j^k / z_j^k \right)$
Minkowski 误差准则：	$E_{Mink}(\mathbf{w}) = \sum_{k,j} \left| t_j^k - z_j^k \right|^R, \quad 1 \le R < 2$

6.5.2 激励函数

- 激励函数
  - 在BP算法中，任何**连续可导函数**都可以作为激励函数。
  - 激励函数应是**非线性**的。
  - 激励函数是**有界连续可导**的。
  - 激励函数**最好是单调**的。否则，误差函数会包含更多的局部极小值点，从而增加训练难度。
  - Sigmoid函数（以及双曲正切函数）满足上述性质。

$$\boxed{\delta_j^k = f'\left(net_j^k\right)\left(t_j^k - z_j^k\right)}$$

6.5.3 隐含层数

- 隐含层数设定
  - Heche-Nielsen证明，**当各结点具有不同的阈值时，具有一个隐含层的网络可以表示任意函数**，但由于该条件很难满足，该结论意义不大。
  - Cybenko指出，当各结点均采用S型函数时，一个隐含层就足以实现任意判别分类问题，两个隐含层则足以实现输入向量的任意输出函数。
  - 对于分类问题，隐含层的个数决定了网络的表达能力，决定决策面的复杂程度。
  - 网络层次的选取依经验和情况而定。

6.5.4 结点个数

- 模式分类问题
  - 各层结点数的选择对网络的性能影响较大
    - 隐含层结点数太少，网络难以建立复杂的判别界面；取得太多，判决界面仅包含训练样本点而失去推广能力。
  - 通常隐含层结点的个数设置得较大。待网络训练之后，考察有无需要减少结点数的可能。
    - 压缩神经网络
    - 稀疏连接的神经网络
    - dropout技术

6.5.5 初始权重

- 初始权重
  - 在批处理权重更新算法中，初始权重的更新值$\Delta w_{hj} \neq 0$，否则不会产生学习。（最后一个隐含层至输出层）
  - **权重可以为正，也可以为负。**
  - 通常从一个**均匀分布中随机选择初始值**：$-w_0 < w < w_0$。
    - 如果 w_0太小，隐含层的网络加权和就会很小，网络则类似于线性网络。

6.5.6 正则化技术

- 目标函数正则化
  - 防止网络出现 overfitting 的一种有效方法是采用一些正则化技术。
    - 权重2范数正则化技术修正函数：

$$E_{new}(\mathbf{w}) = E(\mathbf{w}) + \boxed{\frac{2\varepsilon}{\eta}\mathbf{w}^T\mathbf{w}}$$

    - 权重启发式目标函数修正策略：

$$E_{new}(\mathbf{w}) = E(\mathbf{w}) + \frac{2\varepsilon}{\eta}\sum_{i,j}\frac{w_{ij}^2/\mathbf{w}^T\mathbf{w}}{1+w_{ij}^2/\mathbf{w}^T\mathbf{w}}$$

中国科学院大学 University of Chinese Academy of Sciences

6.5.6 学习率

- 学习率（梯度更新步长）
  - 学习率 η 太小，则收敛较慢；过大则不稳定。
  - 调节参数的准则是检查某特定**权重修正**是否确实降低了误差函数的值：
    - 如果不是，则 η 应该减小。
    - 如果连续几次迭代均降低了误差函数，则表明所选的η 值可能太保守了，应将 η 增加一个量。
  - 最优的学习率最好是经过一次学习就能得到最小值点。对实际问题这是不可能的。通常情况下，并不要求一定精确地收敛至全局最小值点。

6.5.7 附加冲量项

- 附加冲量项
  - **在最优点附近，误差表面可能会较平坦，梯度下降法收敛较慢。可以考虑更多的历史迭代信息。**
  - 权重更新可以参考以下公式来进行（$0 \le \alpha < 1$）：

$$\mathbf{w}(t+1) = \mathbf{w}(t) + (1-\alpha)\boxed{\Delta_{bp}\mathbf{w}(t)} + \alpha\,\boxed{\Delta(t-1)}$$

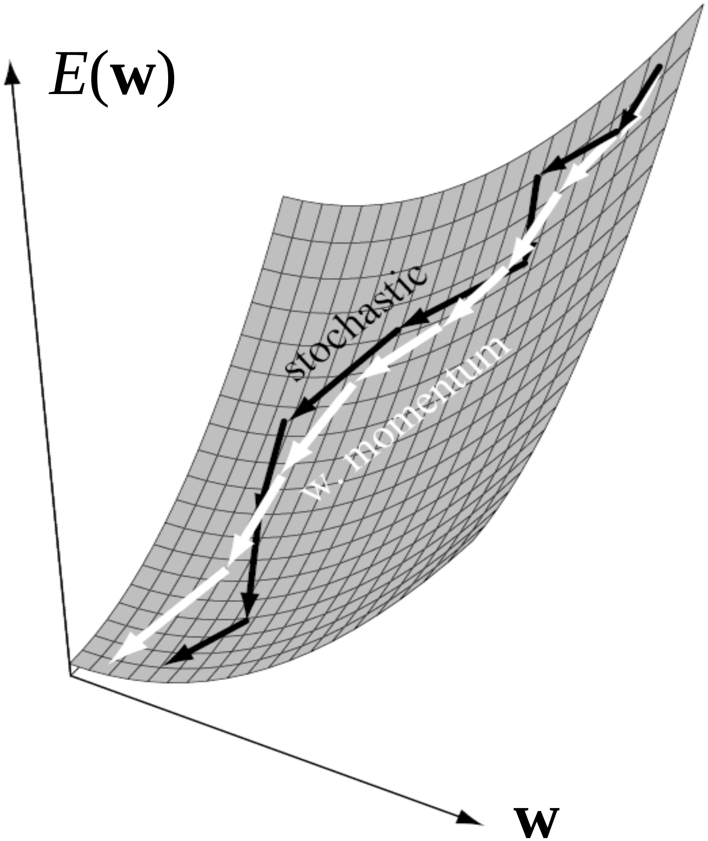
按反向传播算法获取梯权重更新量

$$\Delta(t-1) = \mathbf{w}(t) - \mathbf{w}(t-1)$$

  - 采用动量技术后迭代轨迹会更平滑一些

6.5.7 附加冲量项

- 迭代序列更平滑
- 通过调整α，尽快逃离饱和区



6.5.7 附加冲量项

- 算法

Stochastic Backpropagation with Momentum

begin initialize: n_H, $\mathbf{w}, \eta, \theta, k=0, \alpha < 1, b_{hj} = 0, b_{ih} = 0$

  do $k \leftarrow k+1 \pmod{n}$

    $\mathbf{x}^k$, randomly chosen a sample (pattern)

    $b_{hj} = \eta(1-\alpha)\delta_j^k y_h^k + \alpha b_{hj}$, $b_{ih} = \eta(1-\alpha)\delta_h^k x_i^k + \alpha b_{ih}$

    $w_{hj} = w_{hj} + b_{hj}$, $w_{ih} = w_{ih} + b_{ih}$

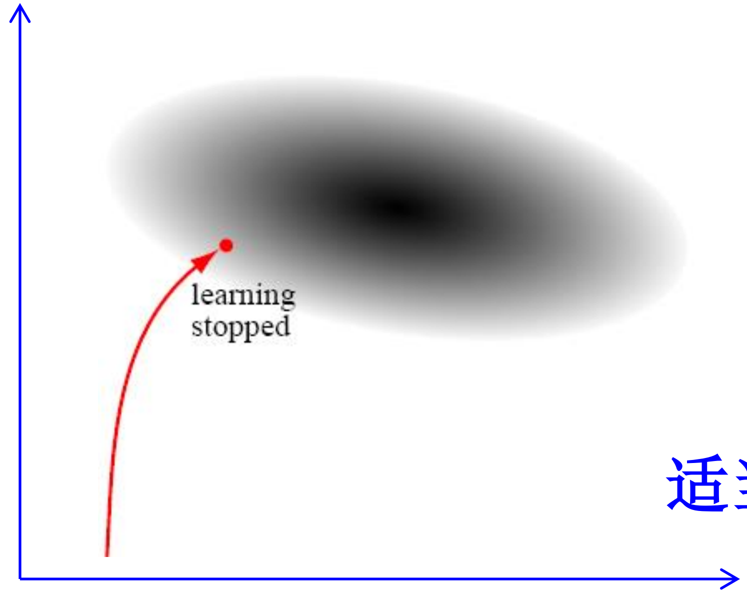
  until $\| \nabla J(\mathbf{w}) \| < \theta$

  return $\mathbf{w}$

end

6.5.8 训练停止准则

- 训练停止－没有固定准则
  - 如果过度训练网络（即使训练样本都能正确的分类）有可能会产生一个 overfitting 问题：
    - **训练样本的分类正确率很高，但对新样本的分类能力不高**。因此，网络的泛化能力不强

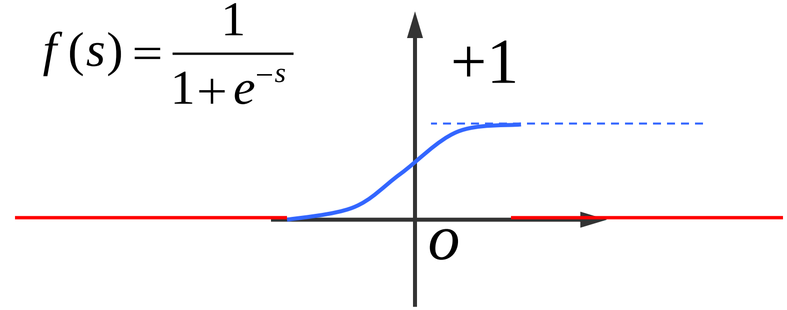


适当时停止！

6.5.9 典型问题

- BP训练算法存在的问题
  - 尽管BP训练算法应用得很广泛，但其训练过程存在不确定性：
    - **完全难以训练**
      - 网络的麻痹现象
      - 梯度消失
      - 局部最小
    - **训练时间过长**
      - 尤其对复杂问题需要很长时间训练；
      - 可能选取了不恰当的训练速率η。

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



网络的麻痹现象

- 在计算权重修正量时，误差 δ 正比于 $f'(\mathrm{net})$ 。
- 当 $f'(\mathrm{net})\to 0$ 时， $\delta\to 0$， 从而 $\Delta w_{ij}\to 0$，相当于调节过程几乎停顿下来。
  - 在训练过程中（如采用Sigmoid函数），权重调得较大，可能使所有或部分加权和 net_j 较大，梯度更新将在S型函数的**饱和区域**进行，即处在其导数 $f'(\mathrm{net})$ 非常小的区域内（**平坦区域**）。
- 改进目标准则函数，比如：

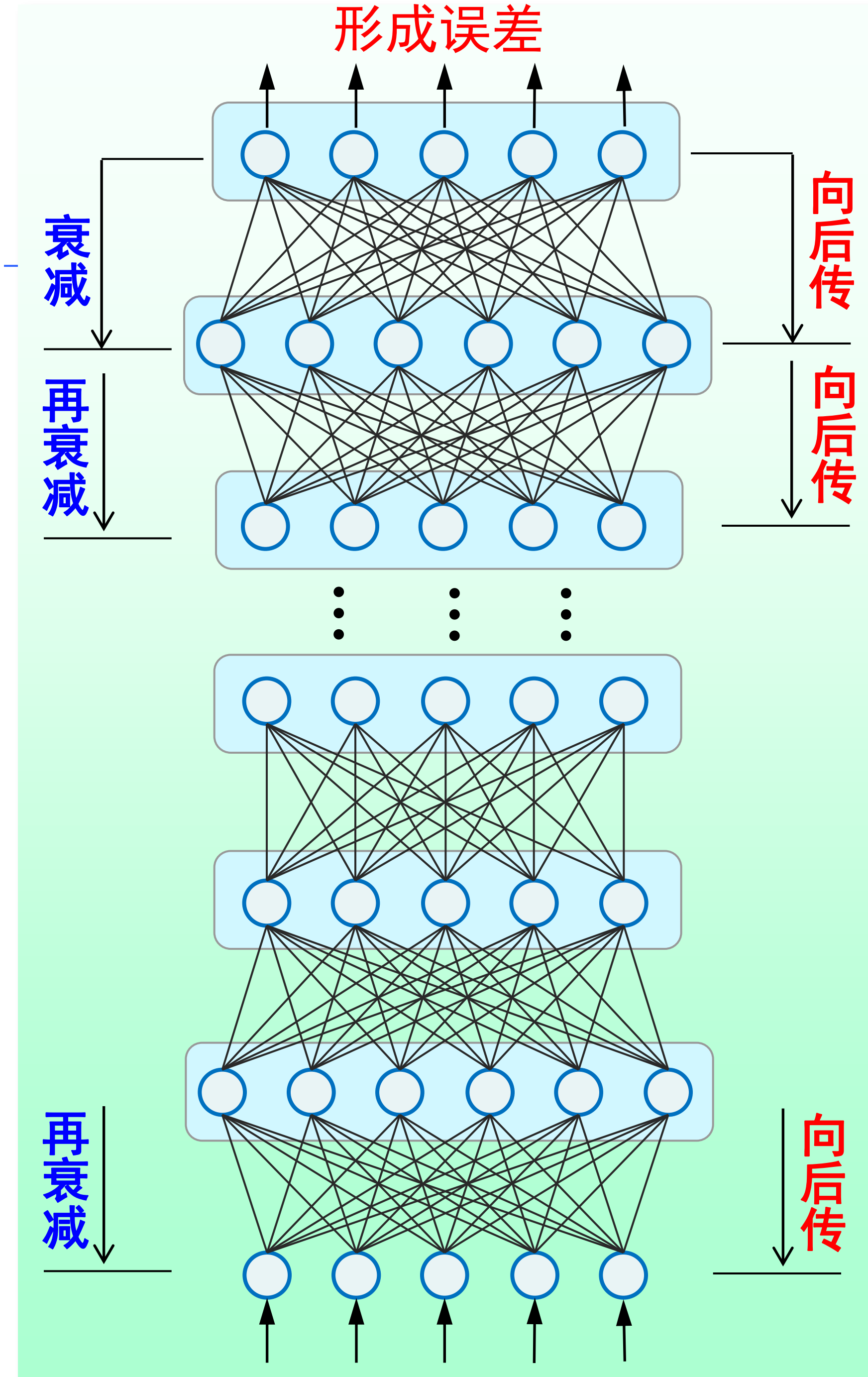
$$E(\mathbf{w})=\frac{1}{2}\sum_{k,j}(1+t_j^k)\log\frac{1+t_j^k}{1+z_j^k}+(1-t_j^k)\log\frac{1-t_j^k}{1-z_j^k}$$

梯度消失

- 在多层神经前向神经网络中，越靠近输入层越容易出现此问题。
- 该问题是由于如下局部梯度在传播过程引起的：

$$\delta_h^k = f'(net_h^k)\sum_j w_{hj}[\delta o]_j^k$$

该导数通常小于1



局部极小

- BP训练算法实际上采用梯度下降法，训练过程从某一起始点沿误差函数的斜面最陡方向逐渐达到最小点 $E \to 0$。
  - 对于复杂的网络，其误差函曲面在多维空间中表面可能凹凸不平，因而在训练过程中可能会陷入某一小的峡谷区，即局部最小点。
  - 如果训练过程中网络处于S型函数的饱和区，也可能陷入局部最小。
- 在网络训练中引入随机因素。

6.6 径向基函数网络

- 径向基函数网络
  - **对模式分类或函数近似任务，径向基函数是一个较好的网络模型**
  - 与多层神经网络相似，**RBF可以对任意连续的非线性函数进行近似**，可以处理系统内的难以解析的规律性
  - 收敛速度比通常的多层神经网络更快

D. Hush and B. Horne, Progress in Supervised Neural Networks, IEEE Signal Processing Magazine, 8-39, 1993

6.6 径向基函数网络

- 解决的典型问题
  - 设有 d 维空间中的 n 个样本点 $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$。各样本点经过一个未知的函数被映射为一个 p 维空间的目标点$\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$。采用径向基函数的组合来近似原未知函数（广义线性判别函数）：

$$g_j(\mathbf{x}) = \sum_{k=1}^{n} w_{kj}\phi_k(\mathbf{x}), \quad j = 1, 2, \ldots, p$$

每一维均对应一个非线性映射

其中，$\phi_k(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_k\|^2}{2\sigma^2}\right)$

6.6 径向基函数网络

- 矩阵形式

第一个样本点的目标向量的各分量值

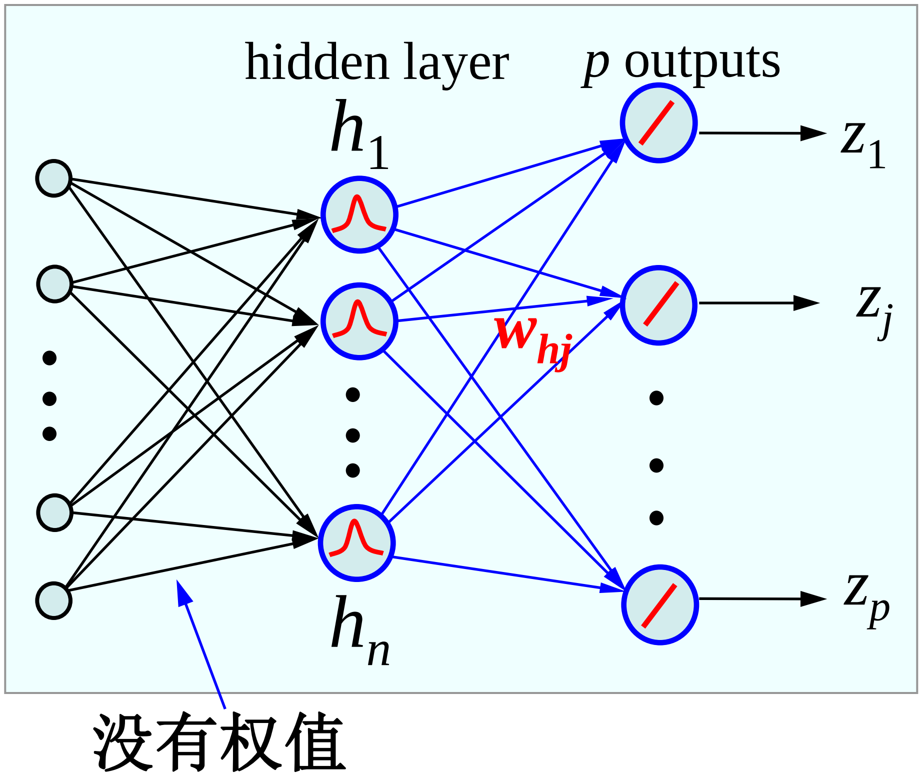
第一个函数的待求权重

$$
\underbrace{\begin{pmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_n(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_n(\mathbf{x}_2) \\ \vdots & \vdots & & \vdots \\ \phi_1(\mathbf{x}_n) & \phi_2(\mathbf{x}_n) & \cdots & \phi_2(\mathbf{x}_n) \end{pmatrix}}_{\mathbf{\Phi}}
\underbrace{\begin{pmatrix} \boxed{w_{11}} & w_{12} & \cdots & w_{1p} \\ w_{21} & w_{22} & \cdots & w_{2p} \\ \vdots & \vdots & & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{np} \end{pmatrix}}_{\mathbf{W}}
\underset{=}{=}
\underbrace{\begin{pmatrix} t_1^1 & t_2^1 & \cdots & t_p^1 \\ t_1^2 & t_2^2 & \cdots & t_p^2 \\ \vdots & \vdots & & \vdots \\ t_1^n & t_2^n & \vdots & t_p^n \end{pmatrix}}_{\mathbf{T}}
$$

$$
\mathbf{\Phi W} = \mathbf{T}, \quad \text{or} \quad \mathbf{\Phi}^T \mathbf{\Phi W} = \mathbf{\Phi}^T \mathbf{T}
$$

6.6 径向基函数网络

- 网络结构
  - 一个三层神经网络，**隐含层激励函数为高斯函数**，输出层转移函数为线性函数，隐含层结点数为样本个数。

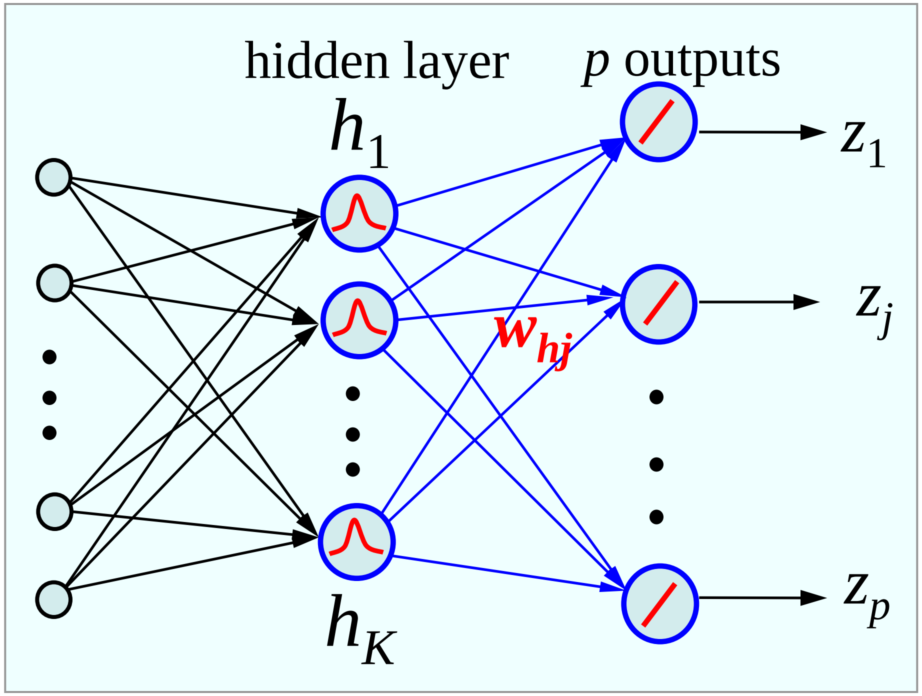


6.6 径向基函数网络

- 网络结构简化与普遍化
  - 对于大规模数据，隐含层的结点个数会很大。采用聚类技术对数据进行聚类，每个隐含层结点代表一个聚类中心。这样简化了网络规模，提高计算效率。
  - 经过聚类处理，还会防止 overfitting，增强网络的泛化能力，提高精度。
  - **当然输出层结点也可以采用非线性转移函数**

6.6 径向基函数网络

- **以聚类中心来代替原来的样本点**



聚类中心

$$\phi_h(\mathbf{x}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{m}_h\|^2}{2\sigma_h^2}\right), h = 1,\ldots K$$

$$net_j(\mathbf{x}) = \sum_{h=1}^{K} w_{hj}\phi_h(\mathbf{x}) + w_{h0}$$

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

6.6 径向基函数网络

- 径向基函数
  - 径向基函数通常采用高斯函数，也可采用其它函数。

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{(Gaussian)}$$

$$\phi(r) = \frac{1}{1+\exp\left(\frac{r^2}{2\sigma^2}\right)} \quad \text{(Reflected Sigmoidal)}$$

$$\phi(r) = \frac{1}{\sqrt{r^2+\sigma^2}} \quad \text{(Inverse multi-quadrics)}$$

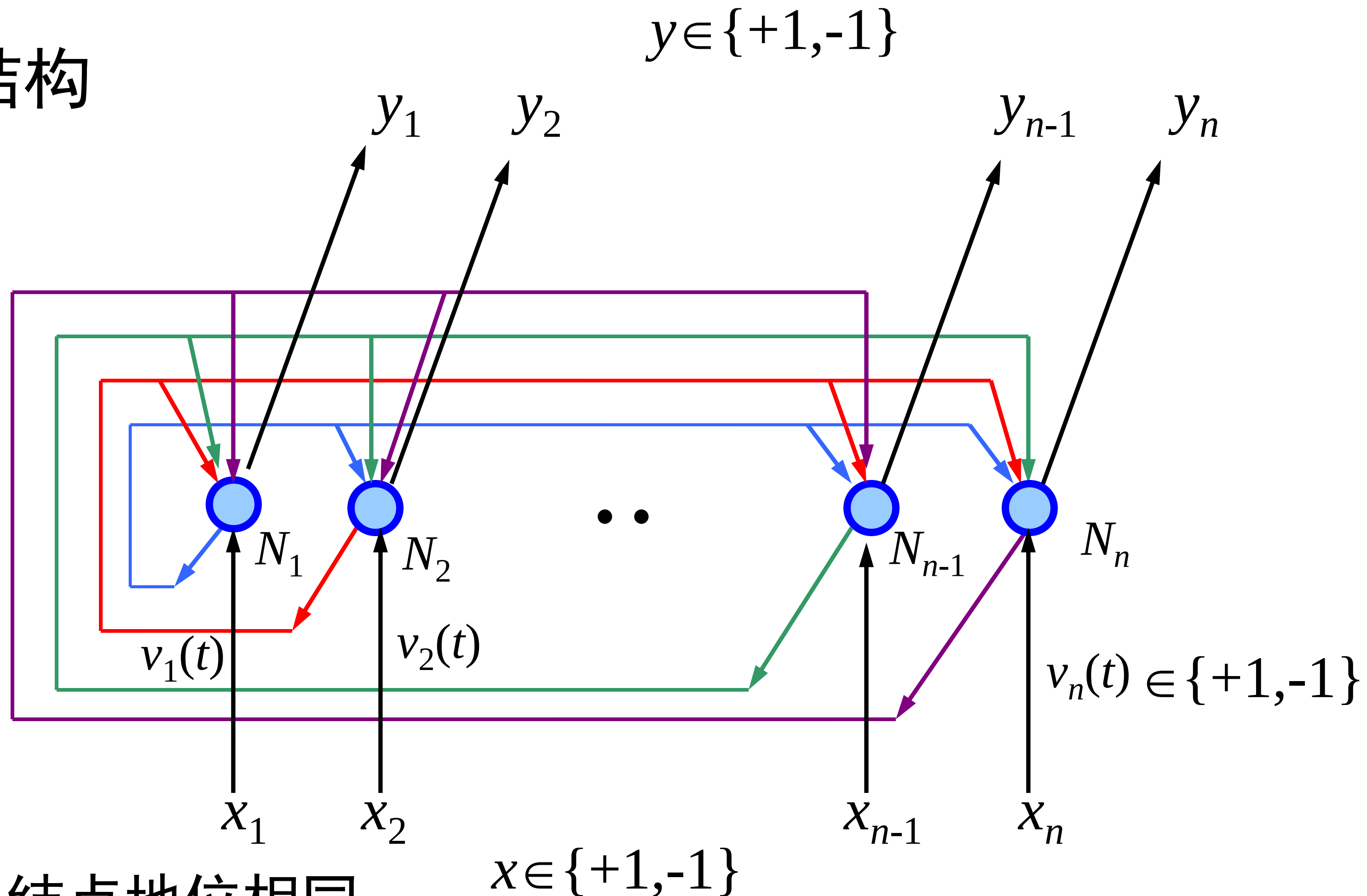
6.7 反馈神经网络

- 按照神经网络运行过程中的信息流向分类：
  - 前馈网络
    - 通过许多具有简单处理能力的神经元的复合作用，使整个网络具有复杂的非线性映射能力。
  - 反馈网络
    - 通过网络神经元状态的变迁，最终稳定于某一状态，得到联想存储或神经计算的结果。
    - 典型的（应用广泛的）反馈神经网络
      - Hopfield网络、受限Boltzman机(RBM)
    - 反馈网络具有一般**非线性系统的许多性质**，如**稳定性、各种类型的吸引子以及混沌现象**，等等。

中国科学院大学
University of Chinese Academy of Sciences

6.7 反馈神经网络

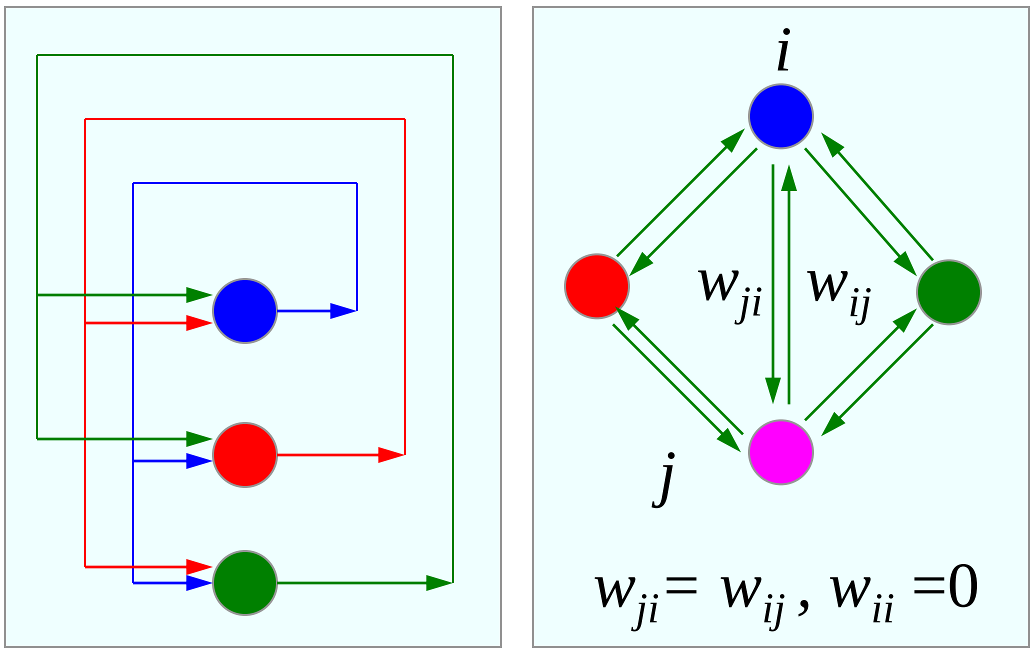
- 网络结构



单层，且各结点地位相同

6.7.1 Hopfield网络

- 网络结构



常见的两种形式

Hopfield网络按动力学方式运行，其工作过程为状态的演化过程，即从初始状态按能量减小的方向进行演化，直到达到稳定状态。稳定状态即为网络的输出

网络演化特点

设表示网络有d个神经元，其转移特性函数为f_1，f_2，…，f_d，阈值为w_1，w_2，…，w_d。对于离散型Hopfield网络，各结点一般取相同的转移函数，且选符号函数，即：$f_1(x) = f_2(x) = \ldots = f_d(x) = \mathrm{sgn}(x)$。为方便起见，可令所有阈值相等且为0，即：$w_1 = w_2 = \ldots = w_d = 0$

- **网络的输入：** $x = (x_1, x_2, \cdots, x_d),\ x \in \{-1, +1\}^d$

  x具有d个分量，每个分量可能为-1或+1。

- **网络的输出：** $y = (y_1, y_2, \cdots, y_d),\ y \in \{-1, +1\}^d$

- **网络在时刻 t 的状态：** $v(t) = (v_1(t), v_2(t), \cdots, v_d(t)),\ v(t) \in \{-1, +1\}^d$

  其中，t为离散时间变量。

- **连接权 w_{ij}：** 从结点i到结点j的连接权重，Hopfield网络是对称的：

$$w_{ij} = w_{ji}, \quad w_{ii} = 0, \quad i, j \in \{1, 2, \cdots, d\}$$

这个网络所有n个结点之间的连接权用矩阵 $\mathbf{W} = (w_{ij})_{dxd}$来表示。

6.7.1 Hopfield网络

- Hopfield网络（离散型）
  - 网络运行方式
    - Hopfield网络为一层结构的反馈网络，可处理双极型离散数据（即输入 $x \in \{-1, +1\}$）或二进制数据（即输入 $x \in \{0, +1\}$）。
    - 给定初始输入**x**，网络处于特定的初始状态。由此初始状态开始运行，可得到网络的输出（下一状态）。
    - 输出状态通过反馈连接送到网络的输入端，作为网络下一阶段运行的输入信号。

6.7.1 Hopfield网络

- Hopfield网络（离散型）
  - 网络运行方式：**信息在网络中循环往复传递。**
    - 如果网络是稳定的，则随着多次反馈运行，网络状态的变化逐渐减少，最后不再变化，达到稳定态。此时由输出端可得到网络的稳定输出。
    - 运行过程：

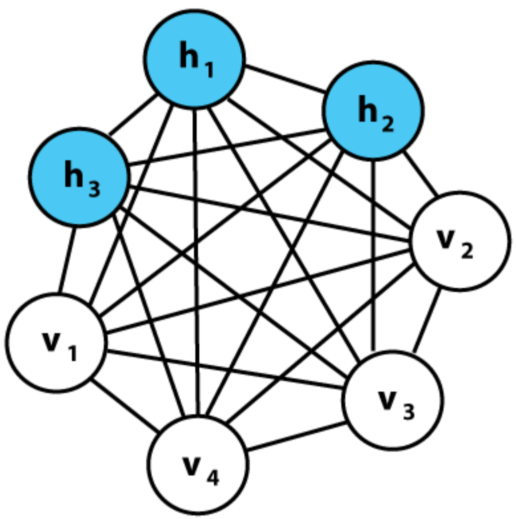
输入信号**x**的第j个分量

$$\begin{cases} v_j(0) = x_j \\ v_j(t+1) = f_j\left(\sum_{i=1}^{n} w_{ij} v_i(t) + w_j\right), \quad f_j = \operatorname{sgn}(x), \quad w_j = 0 \end{cases}$$

若有某个时刻 t，从此之后网络状态不再改变，既$v(t+1) = v(t)$，则有输出$y = v(t)$。

6.7.2 玻尔兹曼机

- 玻尔兹曼机 (Boltzman Machine, BM)
  - 是一种随机的Hopefield网络，是具有隐单元的反馈互联网络



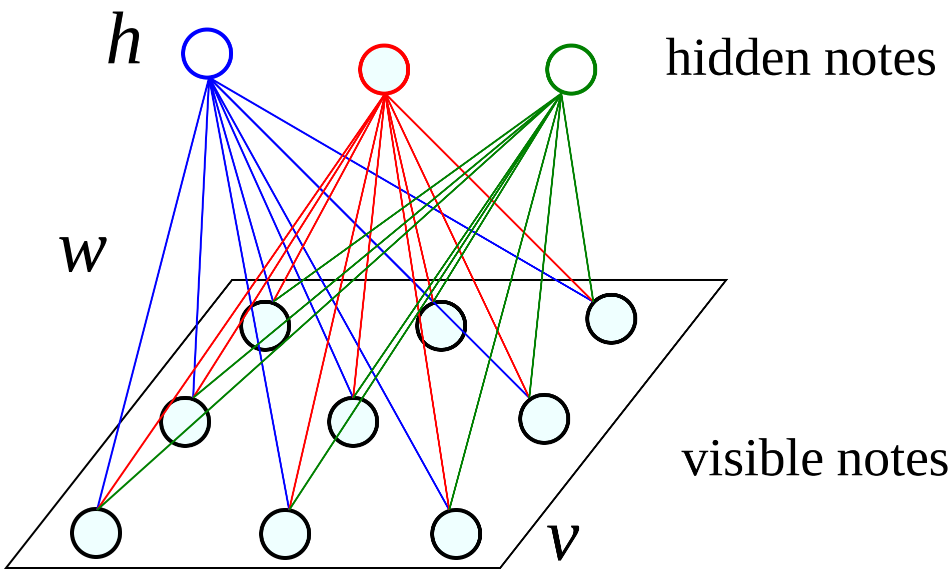
$w_{ji} = w_{ij}$, $w_{ii} = 0$

1. Hopfield网络的神经元的结构功能及其在网络中的地位是一样的。**但BM中一部分神经元与外部相连,可以起到网络的输入、输出功能,或者严格地说可以受到外部条件的约束。**另一部分神经元则不与外部相连，因而属于隐单元
2. 神经元的状态为0或1的概率取决于相应的输入。

网络结构复杂、训练代价大、局部极小

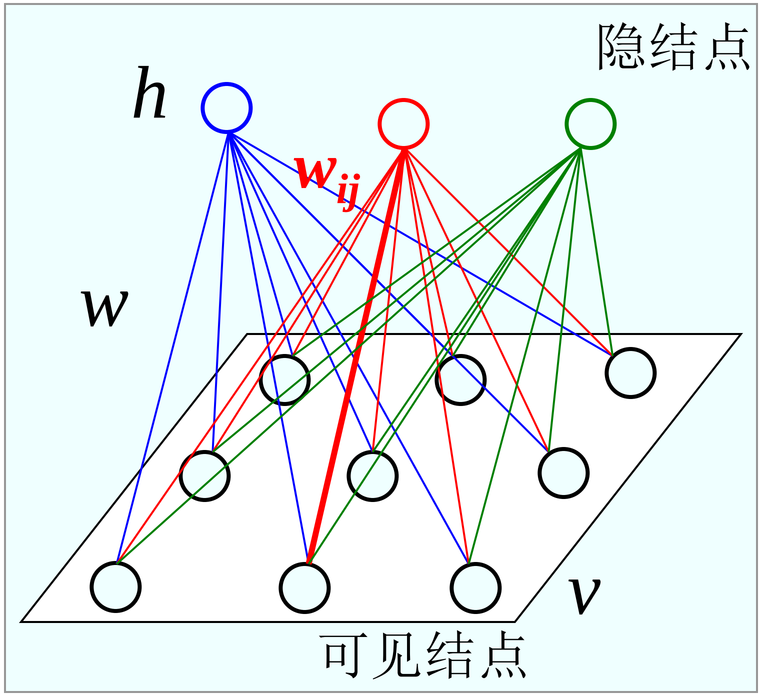
6.7.3 受限玻尔兹曼机

- Restricted BM, RBM
  - 具有两层结构，层内结点不相连，**信息可双向流动**
  - 包含可视结点层（与外界相连）和隐含层（状态层）



University of Chinese Academy of Sciences

RBM



Classical structure

Stochastic binary visible variables $\mathbf{v} \in \{0,1\}^d$ are connected to stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^m$.

网络的能量函数:

$$E(\mathbf{v},\mathbf{h};\theta) = -\sum_{ij} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j$$

θ={w,a,b}—模型参数

可见状态和隐含状态的联合概率分布:

$$p_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{z(\theta)} \exp(-E(\mathbf{v},\mathbf{h};\theta)) = \frac{1}{z(\theta)} \prod_{ij} e^{w_{ij} v_i h_j} \prod_i e^{b_i v_i} \prod_j e^{a_j h_j}$$

$$z(\theta) = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h};\theta))$$

玻尔兹曼分布

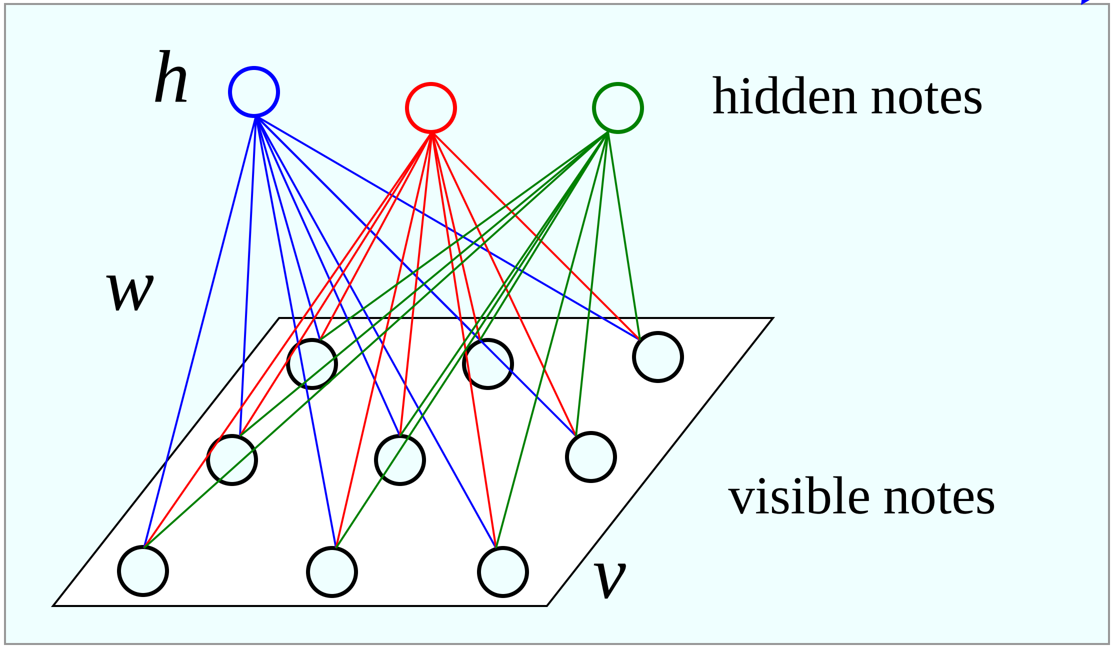
6.7.3 受限玻尔兹曼机

- **目标**

$$\log(p_\theta(\mathbf{v})) = \log\left(\prod_i \exp(b_i v_i)\prod_j\left(1+\exp(a_j+\sum_i w_{ij}v_i\right)\right) - \log z(\theta)$$

给定N个样本： $\max \sum_{i=1}^{N} \log p(\mathbf{v}_i)$

借助该模型



6.8 自组织映射

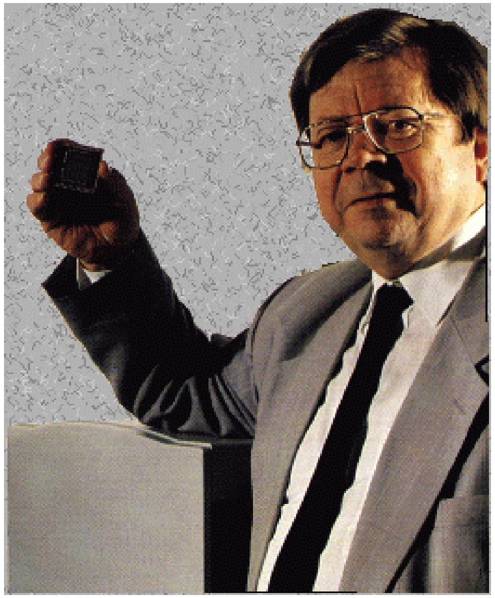
- 自组织竞争神经网络类型
  - **自组织特征映射(Self-Organizing Map，SOM)网络**
  - 自适应共振理论(Adaptive Resonance Theory，ART)网络
  - 对传(Counter Propagation，CP)网络
  - 协同神经网络(Synergetic Neural Network，SNN)

6.8 自组织映射

- 描述

T. Kohonen认为：神经网络中邻近的各个神经元通过**侧向交互作用彼此竞争，自适应地发展成检测不同信号的特殊检测器**。

Kohonen的思想在本质上是希望解决有关**外界信息在人脑中自组织地形成概念的问题**。



芬兰：T. Kohonen (1981)

6.8 自组织映射

- Kohonen认为大脑有如下特点：
  - 大脑的神经元虽然在结构上相同，但**它们的排序不同**。
    - 排序不是指神经元位置的移动，而是指神经元有关参数在神经网络受外部输入刺激而识别事物的过程中产生变动。
  - 神经元参数在变动后形成特定的**参数组织**；具有这种特定参数组织的神经网络对外界的特定事物特别敏感。
  - 生物学和神经生理学：大脑皮层分成不同的**局部区域**，分别管理某种专门功能，如听觉、视觉、思维等。
  - 大脑中神经元的排序受遗传决定，但会在外界信息的刺激下，不断接受传感信号，不断执行聚类过程，形成经验信息，对大脑皮层的功能产生**自组织作用，形成新功能。**

6.8 自组织映射

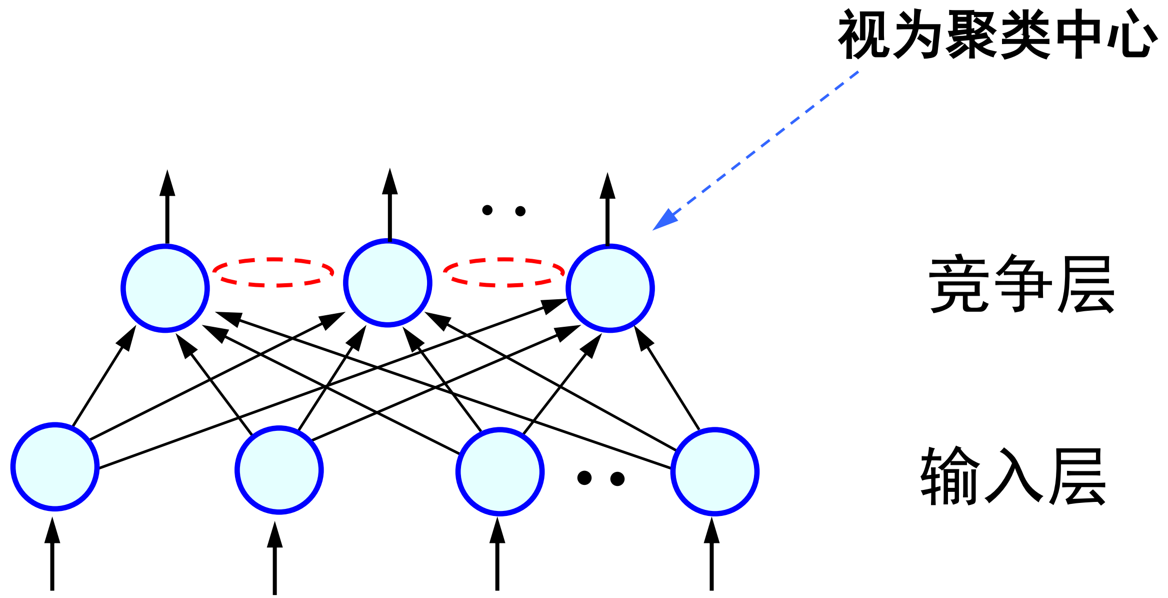
- 生物学基础

生物学研究的事实表明，在人脑的感觉通道上，**神经元的组织原理是有序排列**。因此当人脑通过感官接受外界的特定时空信息时，**大脑皮层的特定区域兴奋**，而且类似的外界信息在对应区域是连续映象的。

对于某一图形或某一频率的特定兴奋过程，神经元的有序排列以及对外界信息的连续映象是**自组织特征映射网中竞争机制的生物学基础**。

6.8 自组织映射

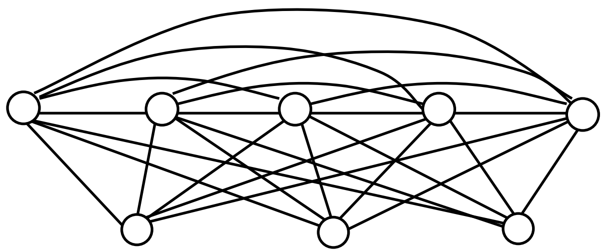
- 结构



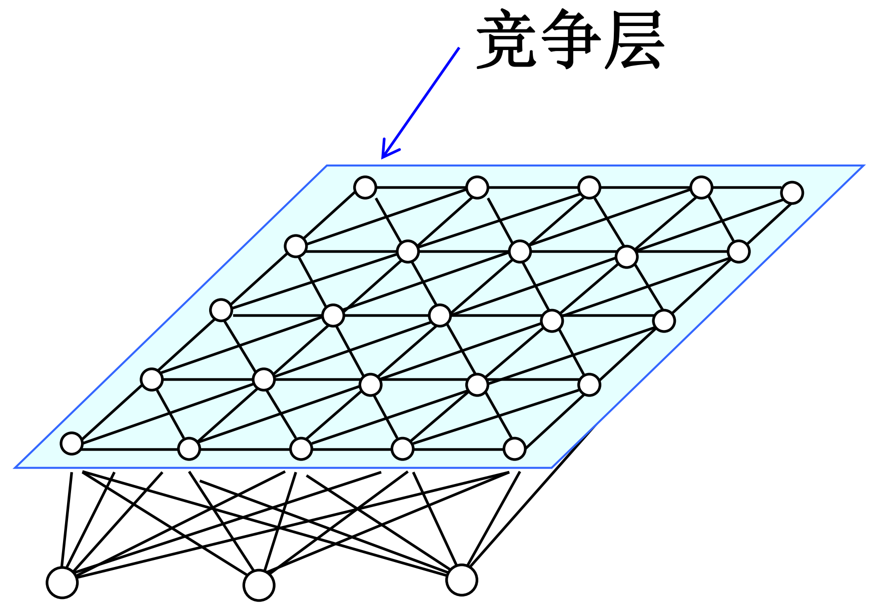
与两层前馈神经网络相似，输入层的每一个单元与输出层的每个单元相联。但输出层相邻神经元之间有相互作用。

6.8 自组织映射

- 常见的网络结构



一维线结构



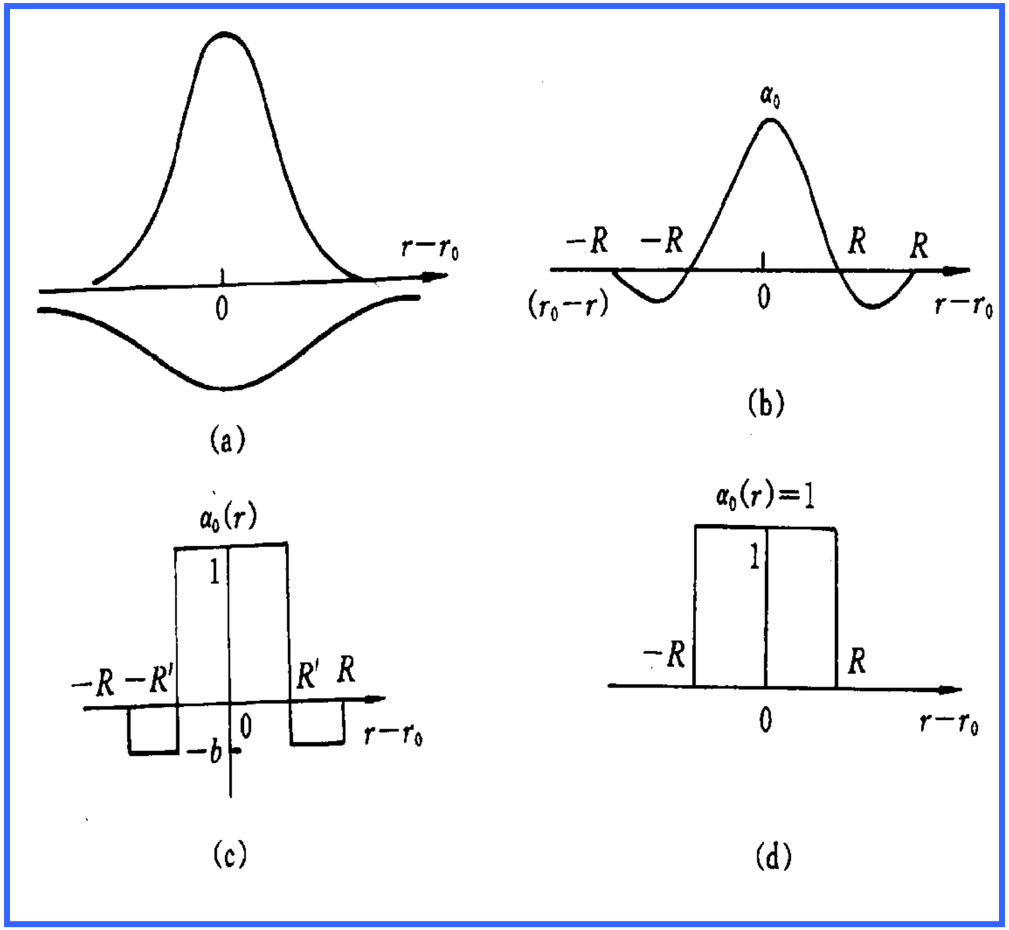
二维平面结构

注：输出层结点之间的连线代表相邻交互作用

6.8 自组织映射

SOM获胜神经元对其邻近神经元的影响是由近及远的，由兴奋逐渐转变为抑制。

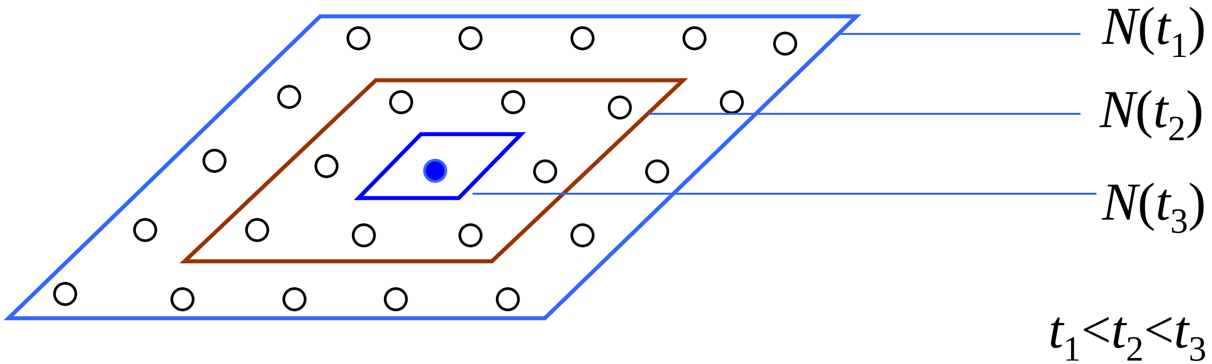
因此在学习算法中，不仅获胜神经元本身要调整权向量，它周围的神经元在其影响下也要不同程度地调整权重。

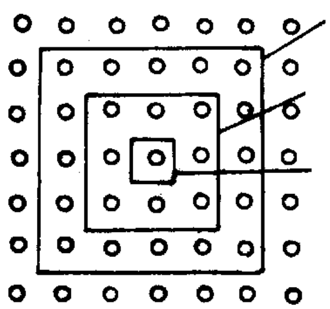


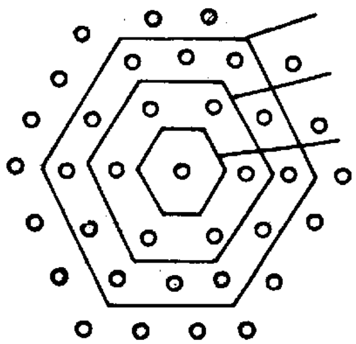
不同的权重影响函数

6.8 自组织映射

- 邻近结点相互作用
  - 权重作用的邻域大小可随时间增长而减小







6.8 自组织映射

- 邻近结点相互作用

✓ 以获胜神经元为中心设定一个邻域半径，该**半径圈定的范围称为优胜邻域**。

✓ 在SOM网学习算法中，优胜邻域内的所有神经元均按其与获胜神经元的距离远近不同程度地调整权重。

✓ 优胜邻域的大小通常随着训练次数的增加而不断收缩，最终收缩到半径为零。

6.8 自组织映射

- 功能描述

✓ 神经元结点的计算功能就是对输入样本给出响应。

✓ 输入向量连接到某个结点的权重组成该结点的**权重向量**。

✓ 一个结点对输入向量的响应强度，即该结点的**权重向量**与**输入向量**的匹配程度，可以用欧氏距离或内积来计算。

✓ 对于一个输入样本，在输出层的所有结点中，**响应最大的结点称为获胜结点**。

6.8 自组织映射

- **学习算法的原理**
  - 通过自动寻找样本中的内在规律和本质属性，自组织、自适应地改变网络参数与结构。
  - 其自组织功能是通过竞争学习来实现的。
  - **竞争学习规则—Winner-Take-All（胜者为王）**
    - 网络的输出神经元之间相互竞争并期望被激活；
    - 在每一时刻只有一个输出神经元被激活；
    - 被激活的神经元称为竞争获胜神经元，其它神经元的状态被抑制。

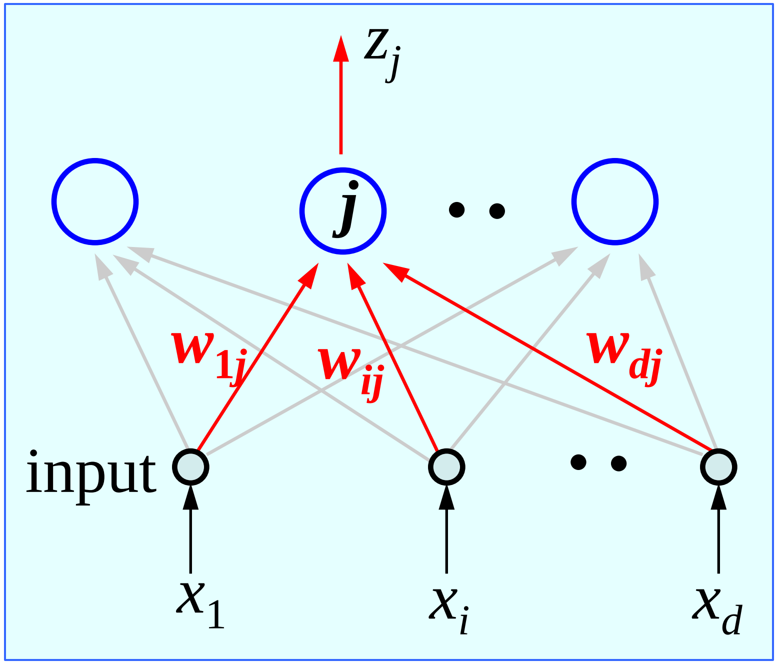
6.8 自组织映射

- **学习算法的原理**
  - Kohonen自组织特征映射算法，能够**自动找出输入样本之间的相似度**，将相似的输入在网络上就近配置。
  - 相似度准则：
    - 欧氏距离： $d_j = \sqrt{\sum_{i=1}^{d}(x_i - w_{ij})^2}$
  - 网络输出： $z_j = \begin{cases} 1 & j = j^* \\ 0 & j \neq j^* \end{cases},$ j^* 为当前输入的胜出单元

6.8 自组织映射

- **学习步骤**
  - S_1：网络初始化—通常采用随机初始化方法
  - S_2：输入向量
  - S_3：计算映射层的权重向量和输入向量的距离：

$$d_j = \sqrt{\sum_{i=1}^{d}(x_i - w_{ij})^2}$$



- **学习步骤**
  - S_4：选择与权重向量的距离最小的神经元**（确定胜者）**
    - 计算并选择使输入向量和权重向量的距离最小的神经元，将其作为胜出神经元 (j*) ，并给出其邻接神经元集合 $h(., j^*)$。
  - S_5：调整权重
    - 胜出神经元和其邻接神经元的权重，按下式更新：

$$\Delta w_{ij} = \eta h(j, j^*)(x_i - w_{ij})$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$

  - S_6：检查是否达到预先设定的要求。
    - 如达到要求则算法结束；否则返回S_2，进入下一轮学习。

6.8 自组织映射

- **邻域函数**

$$h(j, j^*) = \exp\left(-\left\| j - j^* \right\|^2 / \sigma^2\right)$$

由邻域函数可以看到，以获胜神经元为中心设定了一个邻域半径，称为**胜出邻域。**

学习初期，胜出神经元和其附近的神经元全部接近当前的输入向量，形成粗略的映射。

随着学习的进行而减小，**胜出邻域**变窄，胜出神经元附近的神经元数变少。因此，学习方法是一种从粗调整向微调整变化，最终达到预定目标的过程。

训练开始时，一般将近邻范围取得较大。随着训练的进行其近邻范围逐渐缩小。

SOM Training Algorithm

初始化、归一化权向量$\mathbf{w}_j$，建立初始优胜邻域N_j, j=1,2…, c, 学习率η

 do $k \leftarrow k+1 \pmod{n}$

  $\mathbf{x}^k$ randomly chosen a sample (normalized)

  calculate the dot-product $(\mathbf{w}_j)^T \mathbf{x}^k$, j=1, 2…,c,

  find the index with the maximum dot-product, denote j^*

  update the winner's neighborhood $N_j(j^*)$

  adjust the weights of the nodes in $N_j(j^*)$ as follows:

$$w_{ij}(t+1) = w_{ij}(t) + \eta_t [x_i^k - w_{ij}(t)], \; j \in N_j, i = 1, 2, .., d$$

 until $\eta(t) < \eta_{\min}$

 return $\mathbf{w}_j$, j=1,2…, c

end

Thank All of You!