五、神经网络

什么是神经网络?

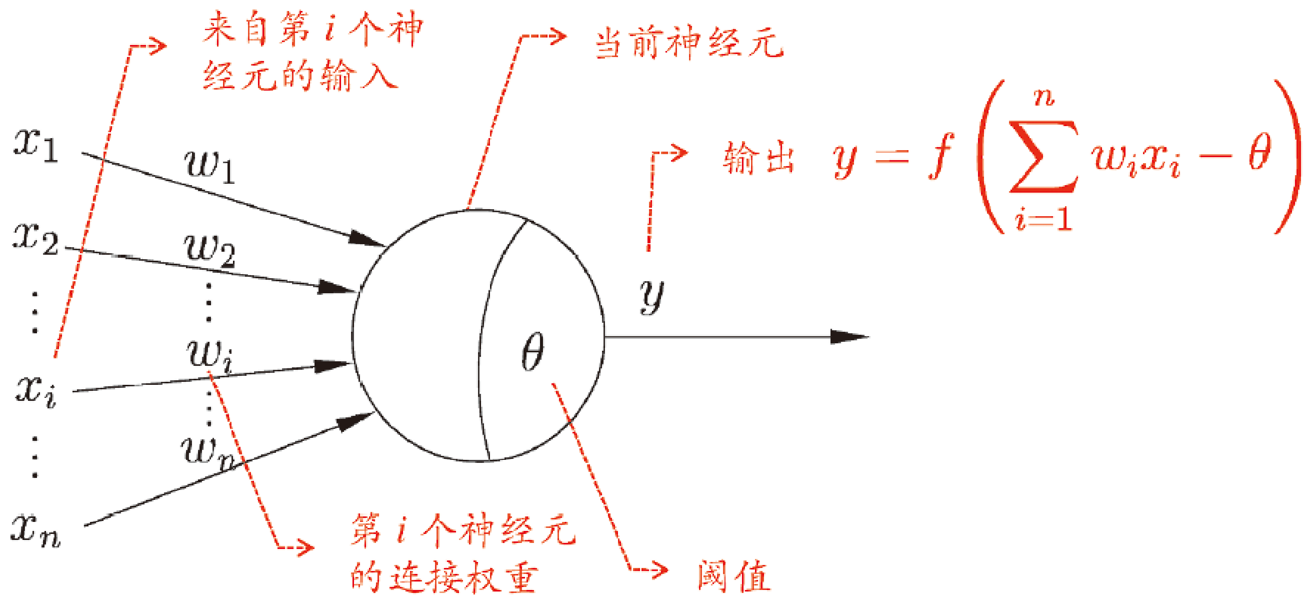
- ”神经网络是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应”

  [T. Kohonen, 1988，Neural Networks 创刊号]

- 神经网络是一个很大的学科领域，本课程仅讨论神经网络与机器学习的交集，即“神经网络学习”亦称“连接主义(connectionism)” 学习

“简单单元”：神经元模型

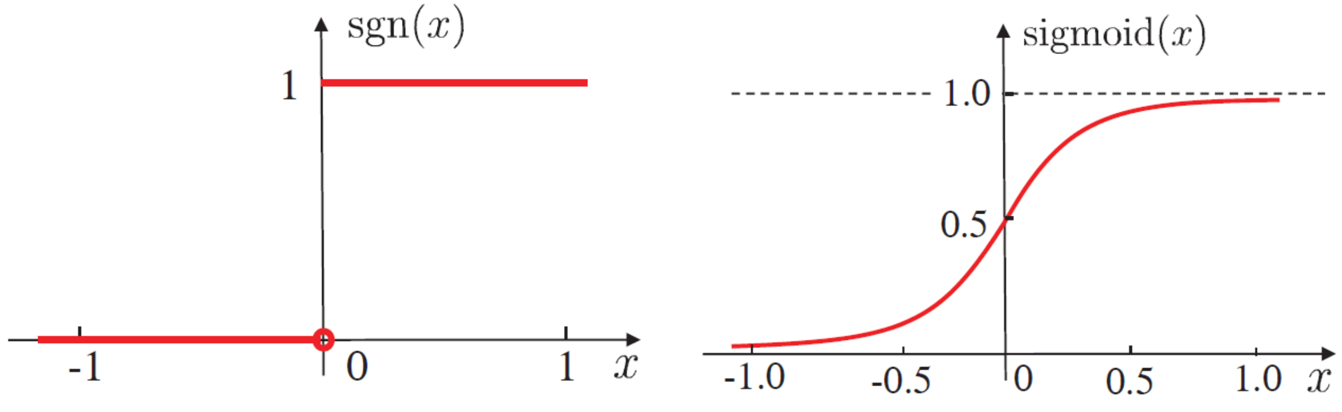
M-P 神经元模型 [McCulloch and Pitts，1943]



神经网络学得的知识蕴含在连接权与阈值中

神经元的“激活函数”

- 理想激活函数是阶跃函数, 0表示抑制神经元而1表示激活神经元
- 阶跃函数具有不连续、不光滑等不好的性质, 常用的是 Sigmoid 函数



$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

(a) 阶跃函数

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

(b) Sigmoid 函数

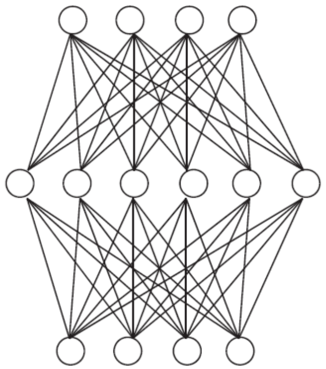
图 5.2　典型的神经元激活函数

多层前馈网络结构

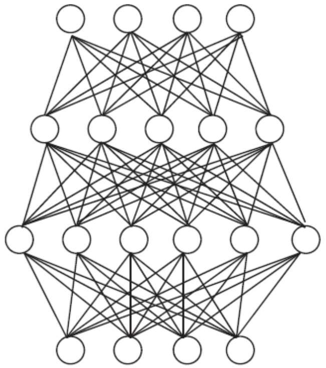
多层网络：包含隐层的网络

前馈网络：神经元之间不存在同层连接也不存在跨层连接

隐层和输出层神经元亦称“功能单元”(Functional Unit)



(a) 单隐层前馈网络



(b) 双隐层前馈网络

多层前馈网络有强大的表示能力（”万有逼近性”）

仅需一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数 [Hornik et al., 1989]

但是，如何设置隐层神经元数是未决问题(Open Problem). 实际常用“试错法”

BP (BackPropagation: 误差逆传播算法)

迄今最成功、最常用的神经网络算法，可用于多种任务（不仅限于分类）

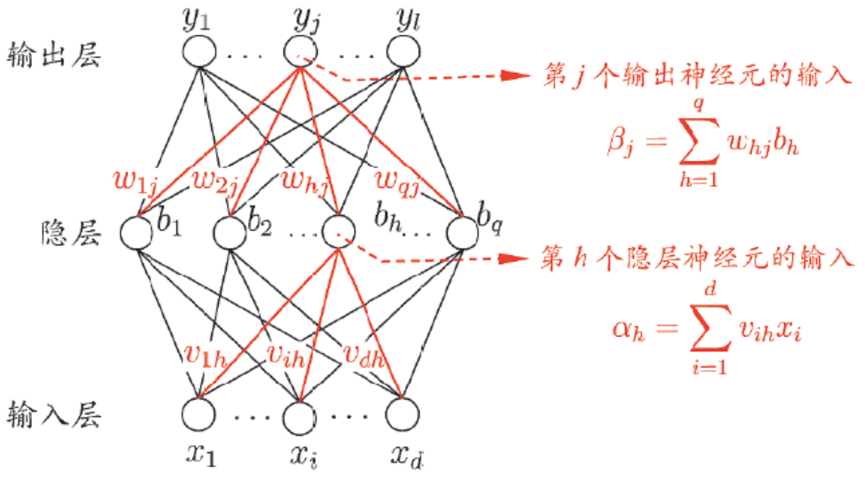
P. Werbos在博士学位论文中正式完整描述:P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral science. Ph.D dissertation, Harvard University, 1974

给定训练集 $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}, \boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y}_i \in \mathbb{R}^l$

输入：d 维特征向量
输出：l 个输出值

隐层：假定使用 q 个隐层神经元

假定功能单元均使用 Sigmoid 函数



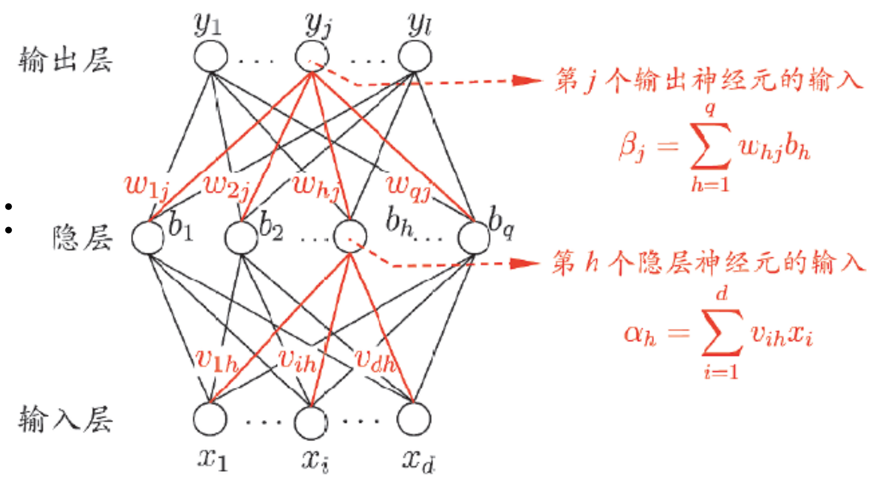
BP 算法推导

对于训练例 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$，假定网络的实际输出为 $\hat{\boldsymbol{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$

$$\hat{y}_j^k = f(\beta_j - \theta_j)$$

则网络在 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ 上的均方误差为：

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$$



需通过学习确定的参数数目：$(d+l+1)q+l$

BP 是一个迭代学习算法，在迭代的每一轮中采用广义感知机学习规则

$$v \leftarrow v + \boxed{\Delta v.}$$

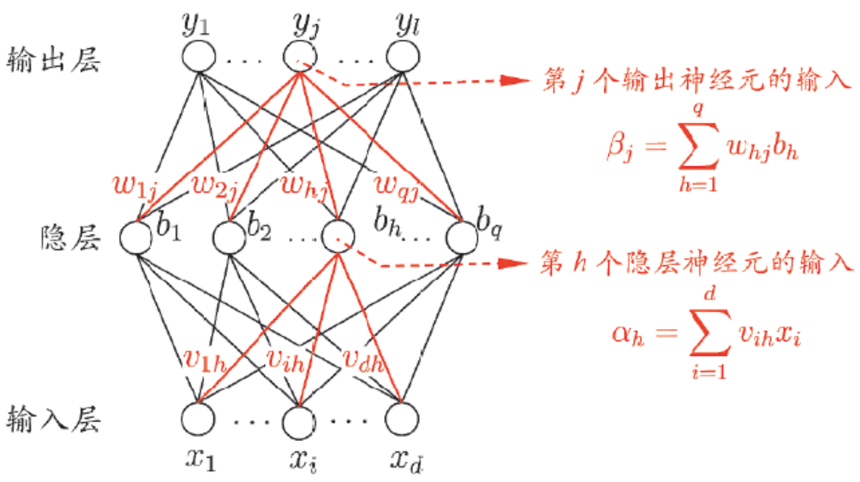
BP 算法推导

BP 算法基于**梯度下降**策略，以目标的负梯度方向对参数进行调整

以 w_{hj} 为例

对误差 E_k，给定学习率 η，有：

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$



注意到 w_{hj} 先影响到 β_j，再影响到 $\hat{y}_j^k$，然后才影响到 E_k，有：

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

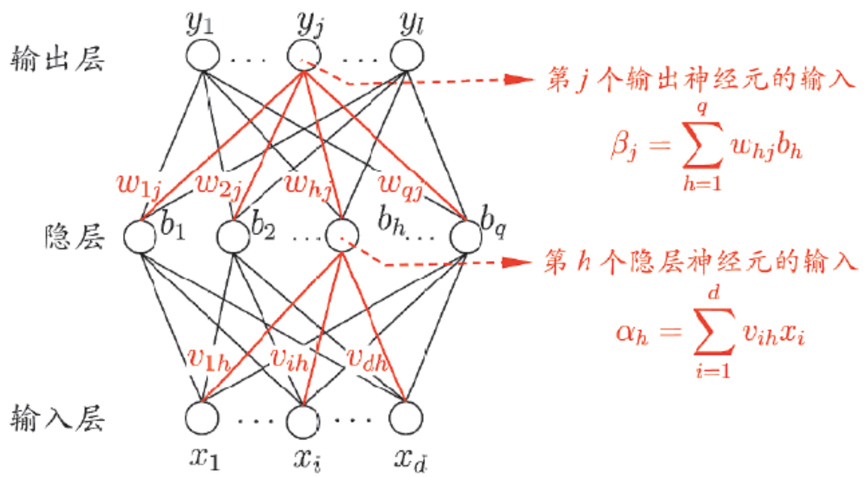
“链式法则”

BP 算法推导

$$\frac{\partial E_k}{\partial w_{hj}} = \boxed{\frac{\partial E_k}{\partial \hat{y}_j^k}} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \boxed{\frac{\partial \beta_j}{\partial w_{hj}}}$$

$$(\hat{y}_j^k - y_j^k)$$

$$= b_h$$



注意到

$$\hat{y}_j^k = f(\beta_j - \theta_j)$$

$$f'(\beta_j - \theta_j)$$

$$\hat{y}_j^k(1 - \hat{y}_j^k)$$

对 $\mathrm{sigmoid}(x) = \dfrac{1}{1+e^{-x}}$，有

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

令 $g_j = -\dfrac{\partial E_k}{\partial \hat{y}_j^k} \cdot \dfrac{\partial \hat{y}_j^k}{\partial \beta_j}$

$$= \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k)$$

于是，$\Delta w_{hj} = -\eta \dfrac{\partial E_k}{\partial w_{hj}} = \eta g_j b_h$

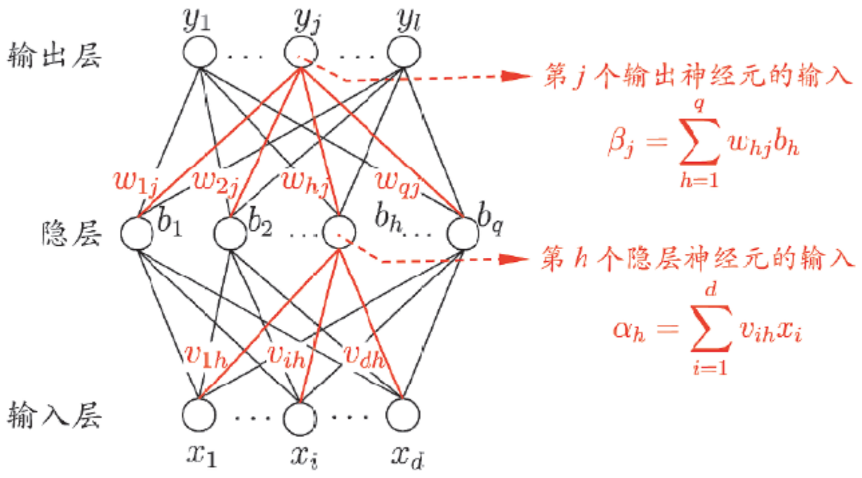
BP 算法推导

类似地，有：

$$\Delta\theta_j = -\eta g_j$$

$$\Delta v_{ih} = \eta e_h x_i$$

$$\Delta\gamma_h = -\eta e_h$$



其中：

$$e_h = -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h}$$

$$= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) \quad = \sum_{j=1}^{l} w_{hj} g_j f'(\alpha_h - \gamma_h)$$

$$= b_h(1 - b_h) \sum_{j=1}^{l} w_{hj} g_j$$

学习率 $\eta \in (0, 1)$ 不能太大、不能太小

缓解过拟合

主要策略：

- 早停(early stopping)
  - 若训练误差连续 a 轮的变化小于 b, 则停止训练
  - 使用验证集：若训练误差降低、验证误差升高，则停止训练
- 正则化 (regularization)
  - 在误差目标函数中增加一项描述网络复杂度

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

偏好比较小的连接权和阈值，
使网络输出更“光滑”