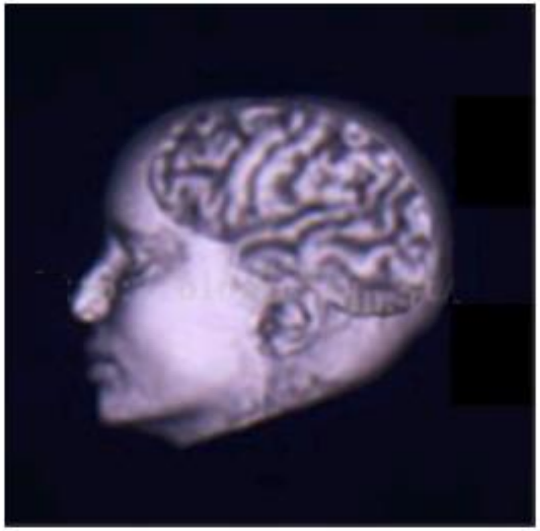
人工神经网络

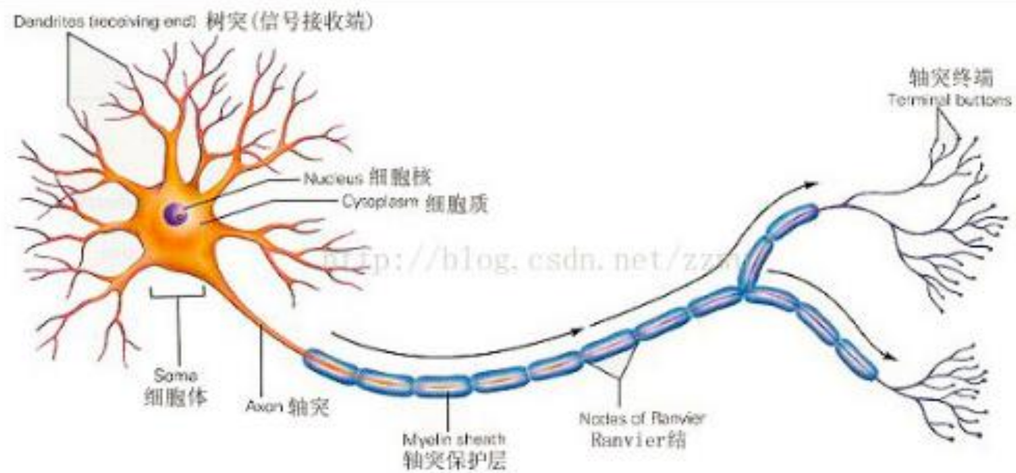
黄晟
huangsheng@cqu.edu.cn
办公室：信息大楼B701

Motivation——How the brain works?



1 大脑半球像半个核桃

2 大脑皮层由灰质白质组成

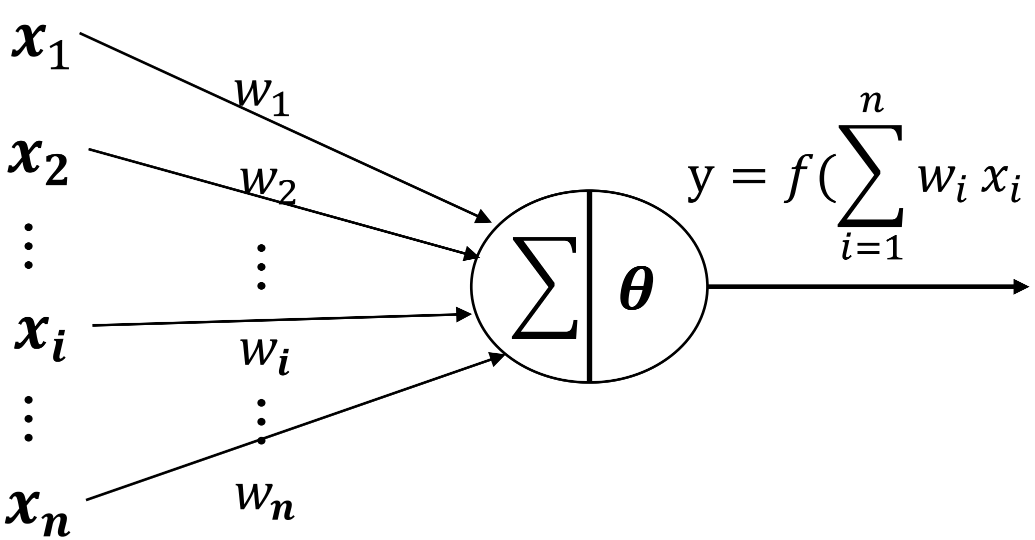


神经网络

神经网络（Neural Network）：

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做的交互反应。

神经元（Neuron）：



$$y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

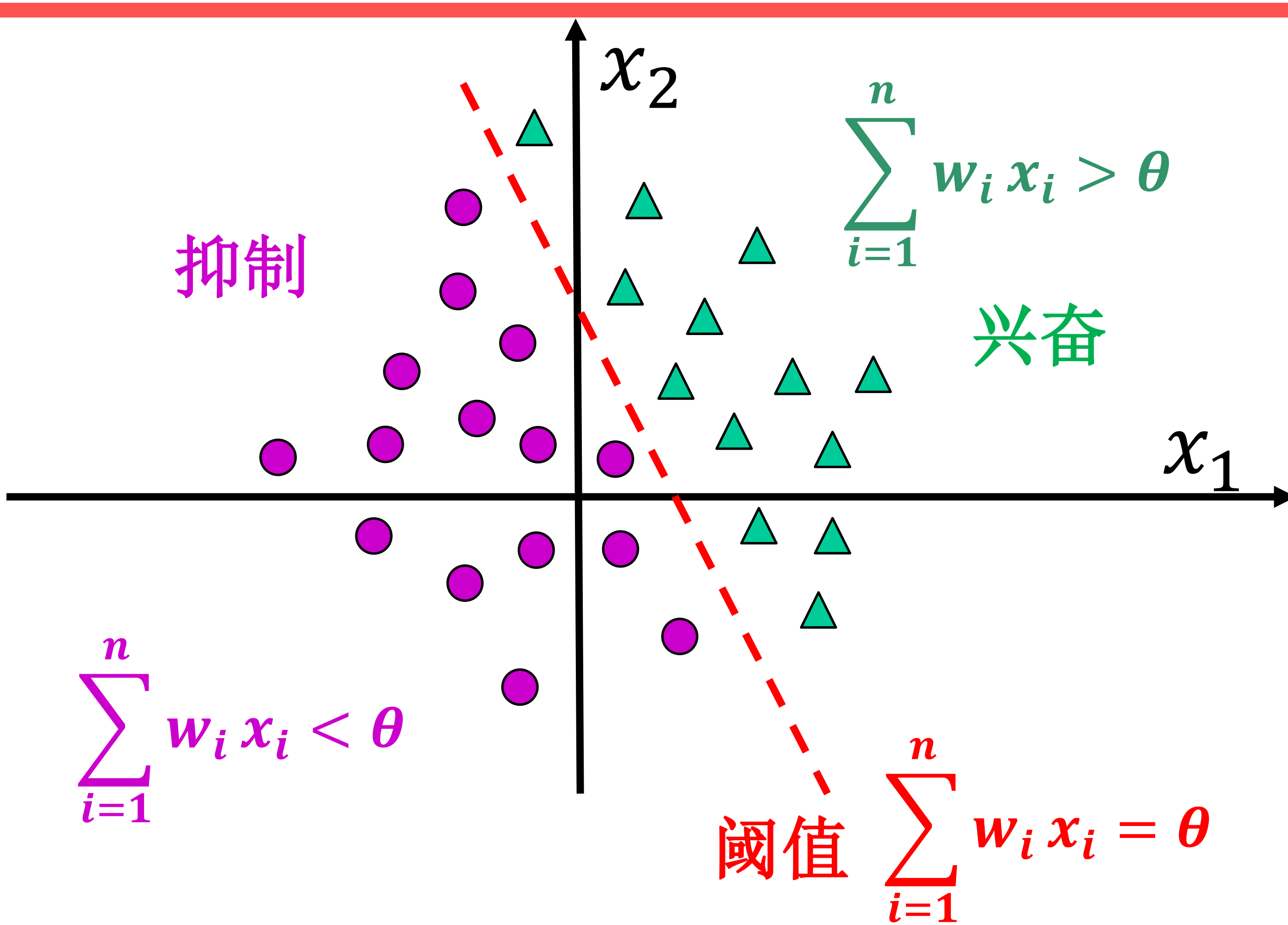
① $\boldsymbol{x_i}$ 来自第i个神经元的输入

② $\boldsymbol{w_i}$ 第i个神经元的连接权重

③ $\boldsymbol{\theta}$ 阈值(threshold)或称为偏置（bias）

M-P神经元模型[McCulloch and Pitts, 1943]

神经元工作机制



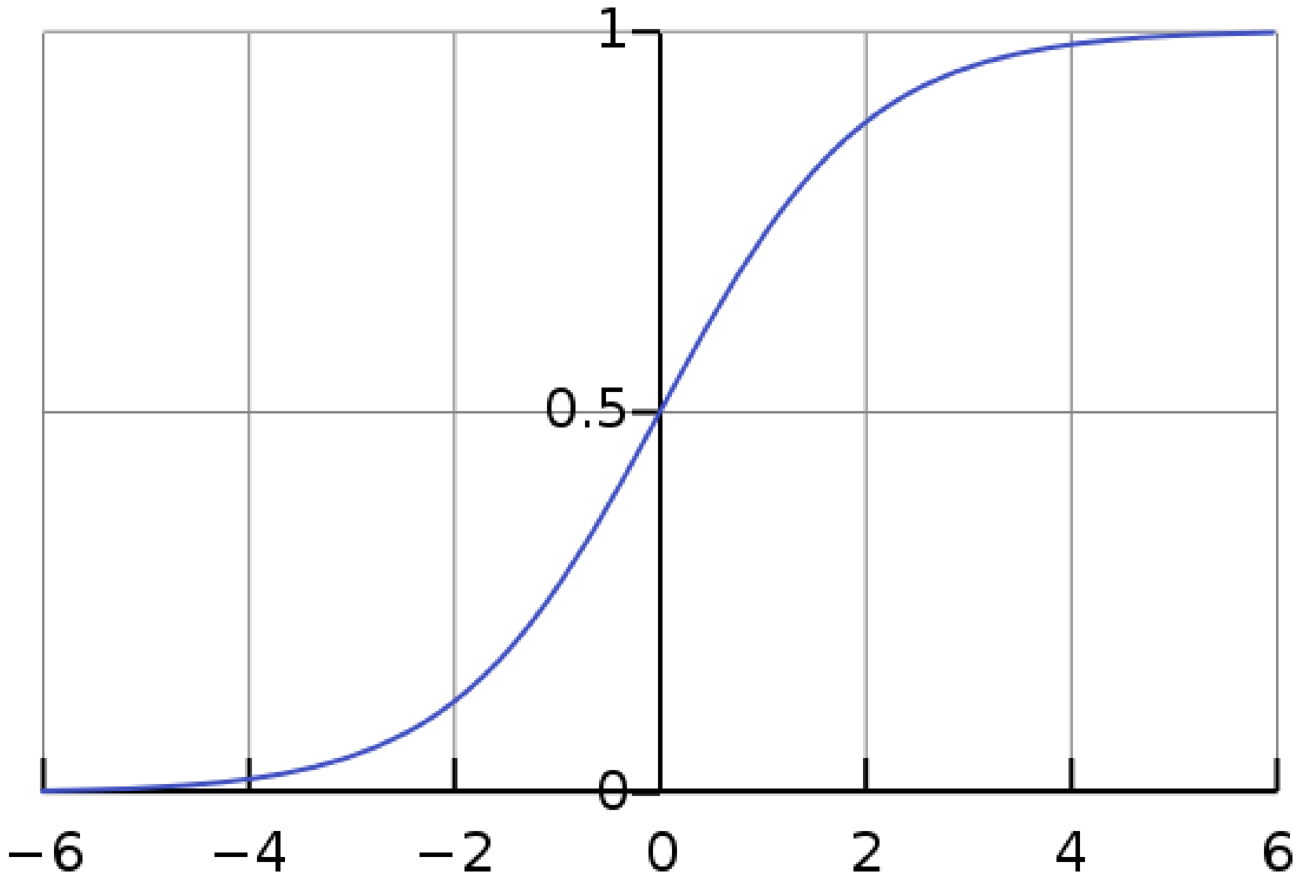
$$\sum_{i=1}^{n} w_i x_i - \theta = \hat{w}^T \hat{x}$$

神经元

- 神经元状态：
  - 当$\sum_{i=1}^{n} w_i\, x_i \geq \theta$时，神经元被激活，处在兴奋状态，假设其对应输出应为$\mathrm{y}=1$。
  - 当$\sum_{i=1}^{n} w_i\, x_i < \theta$时，神经元未被激活，处在抑制状态，假设其对应输出应为$\mathrm{y}=0$。
- $f(\cdot)$函数称为激活函数(activation function):
  - 化学物质与阈值差值➔神经元状态
  - 连续空间[$-\infty$，$+\infty$]➔离散空间{0,1}
  - 理想情况：单位跃阶函数（性质不好）
  - 常见情况：使用Sigmoid函数(又称挤压函数Squashing function)

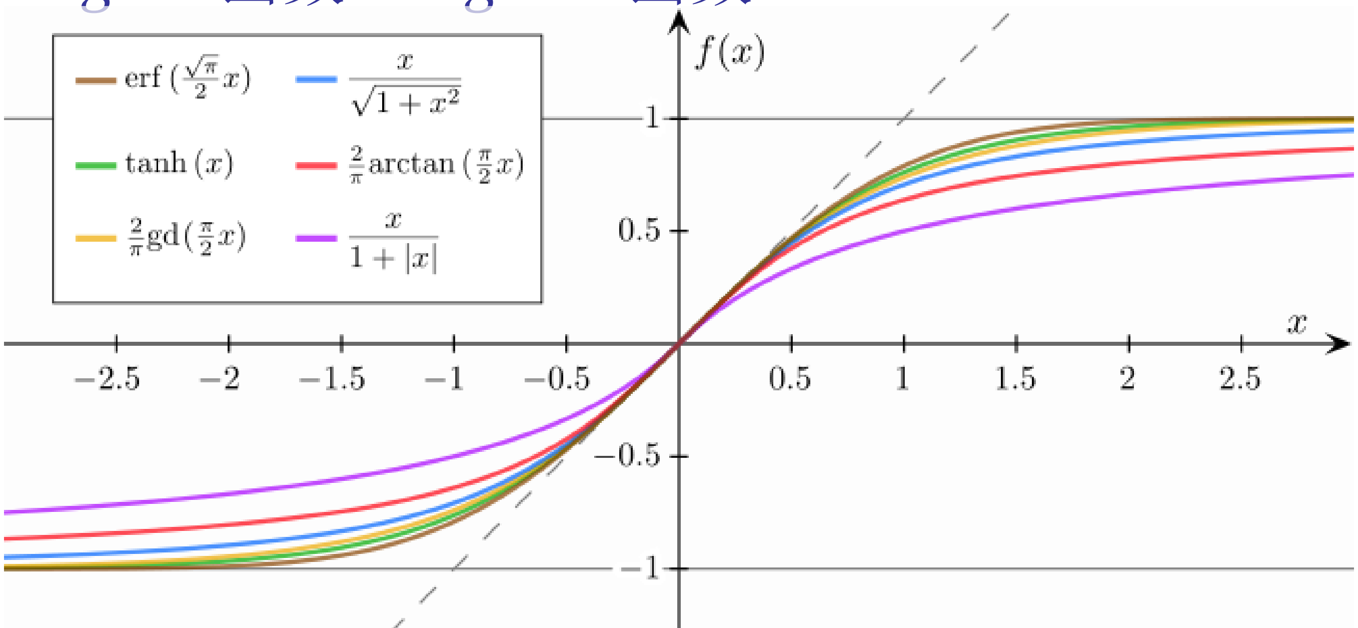
Sigmoid函数

- 典型Sigmoid函数：$y = \frac{1}{1+e^{-z}}$



Connect with Logistic Regression

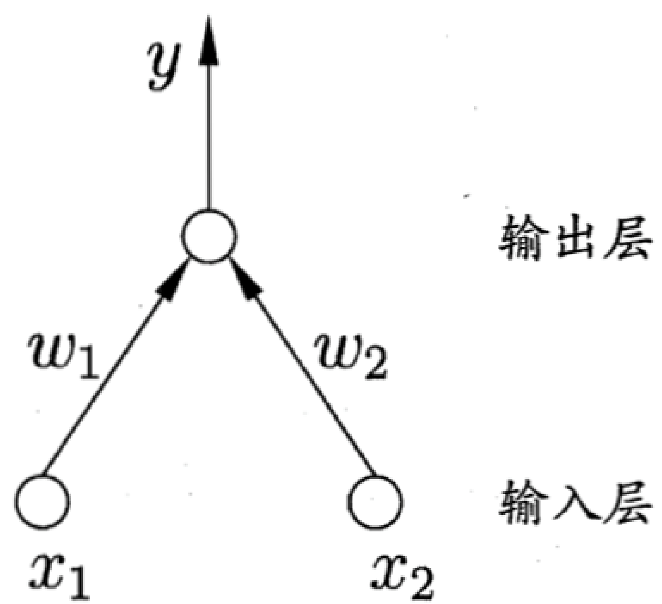
- 回归逻辑回归，同学们发现了什么？
  - 神经元模型与逻辑回归模型求解的优化问题是一致的，都是线性二分类问题。
- Sigmoid函数=Logistic函数？
  - Sigmoid函数≠Logistic函数
  - Logistic函数⊂ Sigmoid函数



感知机（Perceptron）

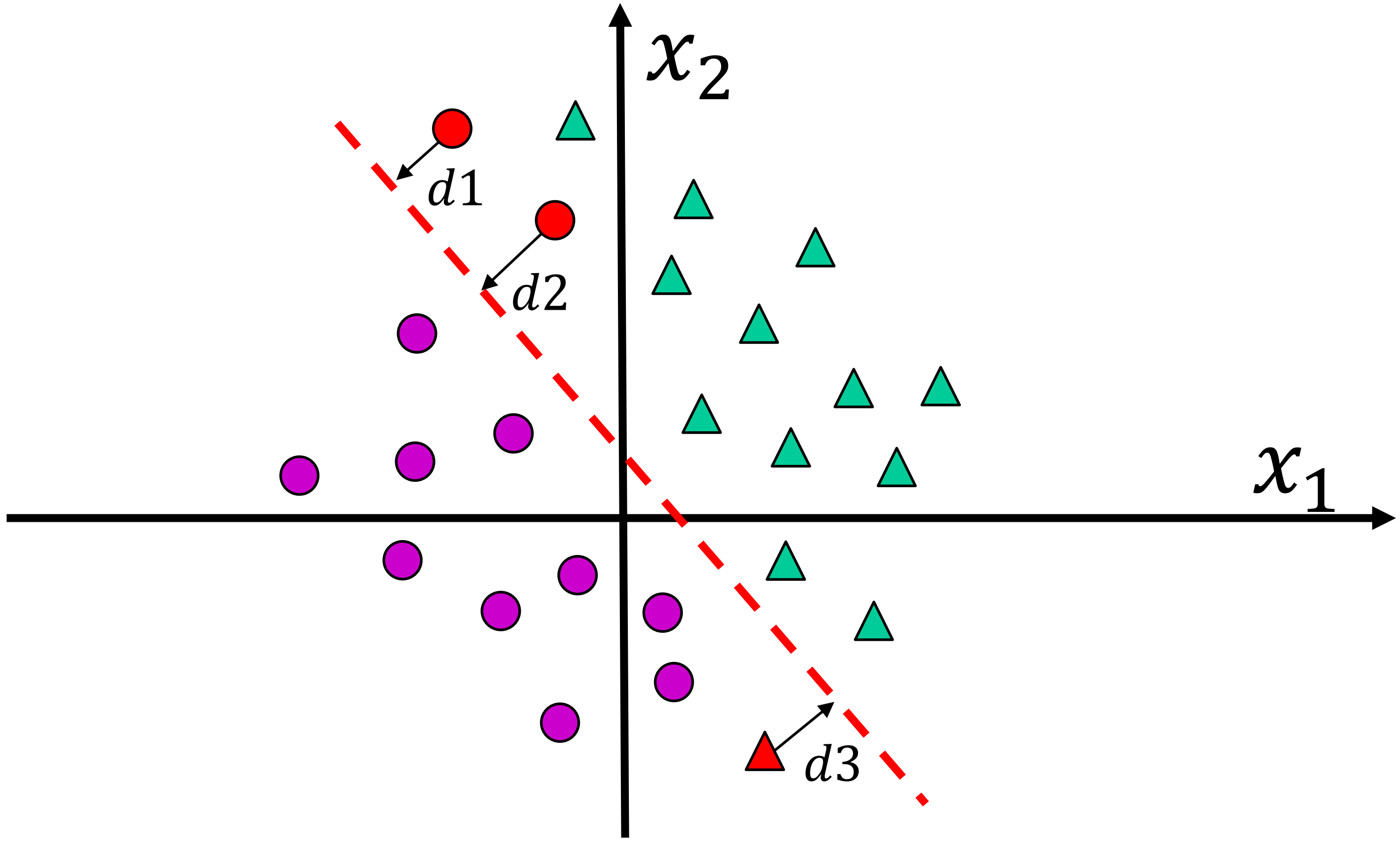
- 本质上, M-P神经元=线性二分类器
- 感知机（Perceptron）：
  - The simplest example of NN
  - Two layers= input layer + one M-P neuron
  - M-P neuron also known as threshold logic unit(阈值逻辑单元)

两个输入神经元的感知机网络。



优化目标

- Same problem with different solution



Objective：错分点到超平面的距离和最短，即$(d1 + d2 + d3) \downarrow$

点到超平面距离

- 为了便于讨论，数据表示向量化：
  - Let $\boldsymbol{w_0} = \boldsymbol{\theta}$, then $\sum_{i=1}^{n} \boldsymbol{w_i}\,\boldsymbol{x_i} - \boldsymbol{\theta} = \sum_{i=1}^{n} \boldsymbol{w_i}\,\boldsymbol{x_i} - \boldsymbol{w_0} = \boldsymbol{\hat{w}}^T\boldsymbol{\hat{x}}$
  - Where $\boldsymbol{\hat{w}} = [\boldsymbol{w_0}; \boldsymbol{w_1}; \cdots; \boldsymbol{w_n}]$and $\boldsymbol{\hat{x}} = [-\mathbf{1}; \boldsymbol{x_1}; \cdots; \boldsymbol{x_n}]$.
  - The ground truth label of $\boldsymbol{\hat{x}}$ is y while its predicted label via NN is $\boldsymbol{\hat{y}} = \boldsymbol{f}(\boldsymbol{\hat{w}}^T\boldsymbol{\hat{x}})$.
- 点到超平面(Hyperplane)的距离:

$$dist(\boldsymbol{\hat{x}}, \boldsymbol{\hat{w}}) = \frac{|\boldsymbol{\hat{w}}^T\boldsymbol{\hat{x}}|}{||\boldsymbol{\hat{w}}||_2}$$

  - 错误分类点到超平面距离：
    - 0被分为1，距离为$(\boldsymbol{\hat{y}} - \boldsymbol{y})\frac{\boldsymbol{\hat{w}}^T\boldsymbol{\hat{x}}}{||\boldsymbol{\hat{w}}||_2}$
    - 1被分为0，距离也为$(\boldsymbol{\hat{y}} - \boldsymbol{y})\frac{\boldsymbol{\hat{w}}^T\boldsymbol{\hat{x}}}{||\boldsymbol{\hat{w}}||_2}$

损失函数

- Cost function (Mean loss)

$$\min \frac{1}{|M|}\sum_{t\in M}(\hat{y}_t - \mathrm{y}_t)\frac{\hat{\boldsymbol{w}}^{\boldsymbol{T}}\hat{x}_{\boldsymbol{t}}}{||\hat{\boldsymbol{w}}||_2} := \frac{1}{|M|}\sum_{t}(\hat{y}_t - \mathrm{y}_t)\frac{\hat{\boldsymbol{w}}^{\boldsymbol{T}}\hat{x}_{\boldsymbol{t}}}{||\hat{\boldsymbol{w}}||_2}$$

Where $M \subset D$ is the subscript collection of misclassified samples, and D is the subscript collection of training samples.

  – 等价于:

$$\min J(\hat{\boldsymbol{w}}) := \frac{1}{|M|}\sum_{t}(\hat{y}_t - \mathrm{y}_t)\hat{\boldsymbol{w}}^{\boldsymbol{T}}\hat{x}_{\boldsymbol{t}}$$

模型求解

- 上述模型无解析解，采用梯度下降迭代求解。
- 梯度下降法：

$$w_{t+1} = w_t - \eta \Delta J(w)$$

- 权重更新法则

$$\hat{\boldsymbol{w}} \leftarrow \hat{\boldsymbol{w}} - \eta \sum_t (\hat{y}_t - \mathrm{y}_t)\, \hat{x}_{\boldsymbol{t}}$$

等价于：

$$\hat{\boldsymbol{w}} \leftarrow \hat{\boldsymbol{w}} + \eta \sum_t (\mathrm{y}_t - \hat{y}_t)\, \hat{x}_{\boldsymbol{t}}$$

标量化：

$\boldsymbol{w_i} \leftarrow \boldsymbol{w_i} + \eta \sum_t (\mathrm{y}_t - \hat{y}_t)\, \boldsymbol{x_t^{(i)}}, \boldsymbol{x_t^{(i)}}$ is the ith element of $\hat{\boldsymbol{x}}_{\boldsymbol{t}}$

思考

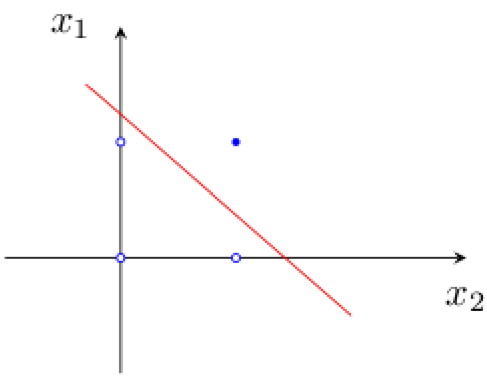
- 对比本权重更新法则与周志华《机器学习》99页公式（5.1）与（5.2）？
- 更新法则不同，我们采用的标准梯度下降法，教材采用了随机梯度下降法(Stochastic gradient descent, SGD)。
  - 标准梯度下降法：所有样本的平均梯度方向下降
  - SGD：随机选取一个样本，沿其梯度下降方向更新：
    - $\hat{\boldsymbol{w}} \leftarrow \hat{\boldsymbol{w}} + \eta(y - \hat{y})\hat{\boldsymbol{x}}$
    - $\boldsymbol{w_i} \leftarrow \boldsymbol{w_i} + \eta(y - \hat{y})x_i$

单层感知器特性

- 线性可分情况：可以解决 （可证明收敛）

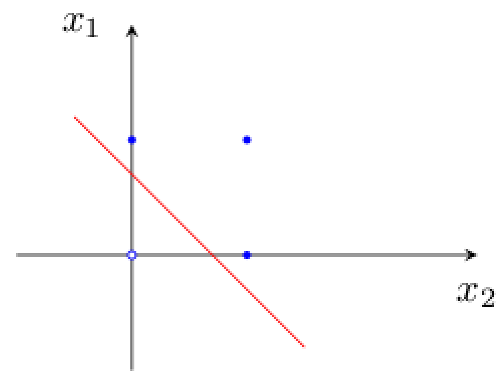
“与”

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



“或”

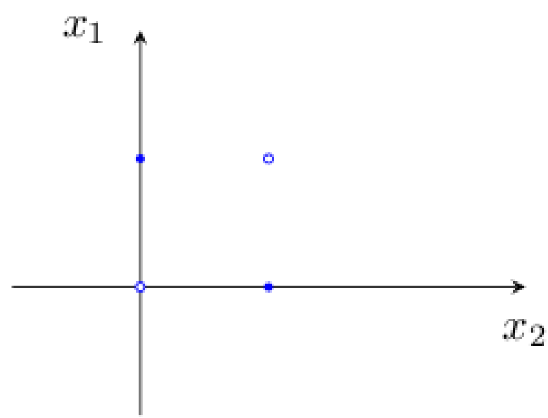
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



- 线性不可分情况：不能解决

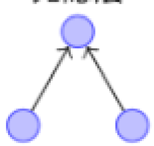
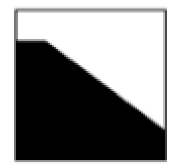
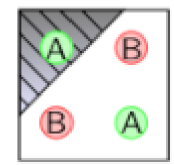
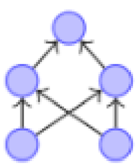
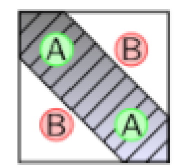
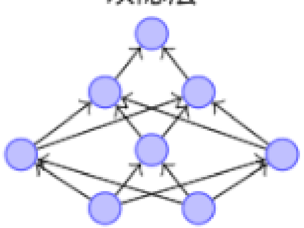
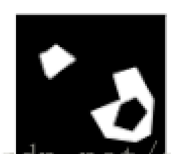
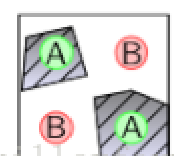
“异或”

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



多层网络

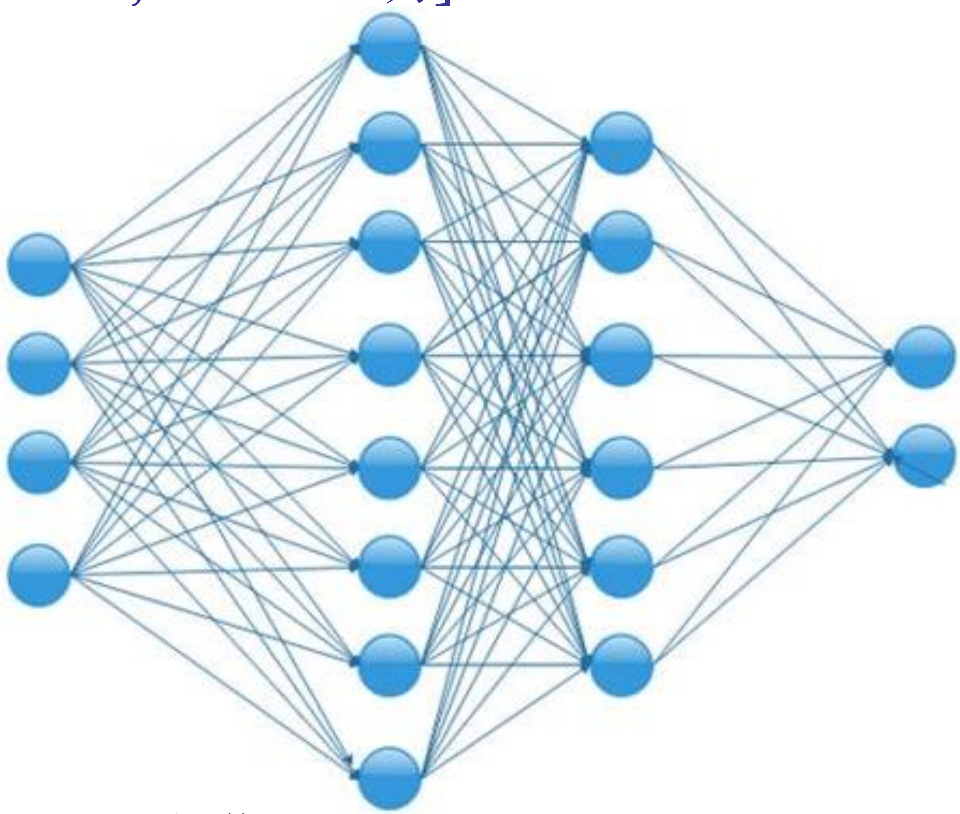
- 线性不可分：一个超平面没法解决问题，就用两个超平面来解决，什么？还不行！那就再增加超平面直到解决问题为止。——多层神经网络。

结构	决策区域类型	区域形状	异或问题
无隐层 	由一超平面分成两个		
单隐层 	开凸区域或闭凸区域		
双隐层 	任意形状（其复杂度由单元数目确定）		

多层神经网络

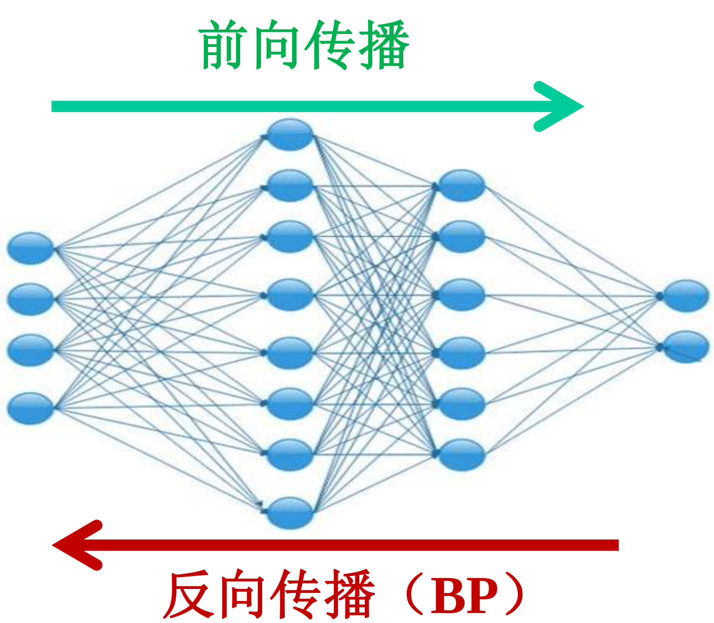
- 多层前馈神经网络(Multi-layer feedforward neural networks)

  Powerful：可以以任意精度逼近任意复杂度的连续函数。[Hornik et al, 1989 证明]



多层神经网络的优化更新

- 1. 初始多层神经网络各层参数（如w,θ）.
- 2. 给定一组训练样本（x,y），进行一次前向传播，计算预测误差。
- 3. 根据预测误差，进行一次方向传播，利用BP算法（链式求导法则+随机梯度下降法）完成各层神经网络参数更新（修正）。
- 重复2,3直至收敛。



误差逆传播算法

- 多层网络的权重优化法则——误差逆传播（BackPropagations，简称BP）算法。

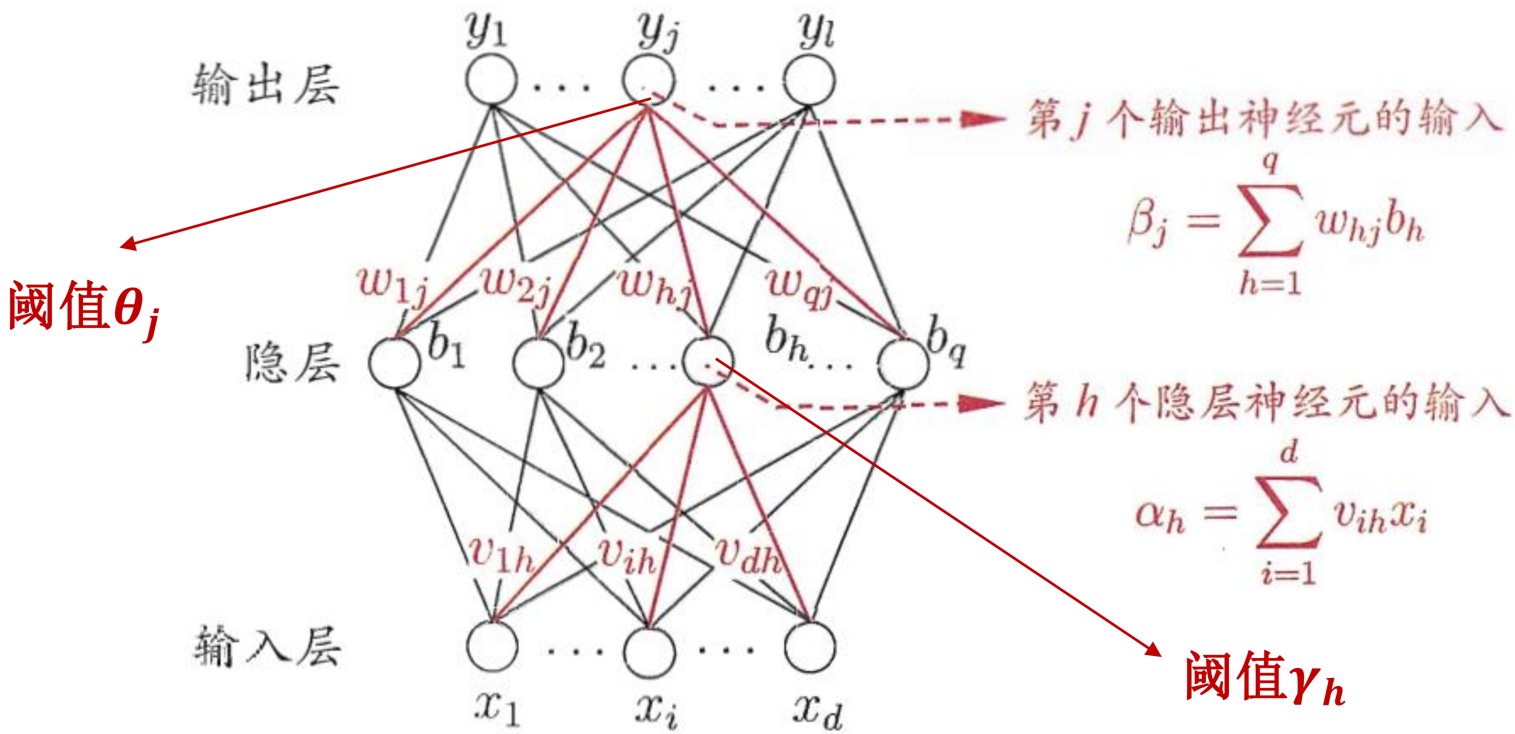


图 5.7 BP 网络及算法中的变量符号

误差逆传播算法

- BP算法中的待求参数：

$$w_{hj},\ \theta_j, v_{ih}, \gamma_h$$

- Other notations：
  - 给定训练样本(x_k, y_k),假定神经网络的输出为$\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \cdots, \hat{y}_l^k)$, 即$\hat{y}_j^k = f(\beta_j - \theta_j)$
  - 关于训练样本(x_k, y_k)在所有属性上的误差为
  - 最小化目标函数（代价函数）$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$

BP算法

- General update rules(SGD):η为学习率
  - $\boldsymbol{v} \leftarrow \boldsymbol{v} + \Delta\boldsymbol{v}, \Delta\boldsymbol{v} = -\eta \frac{\partial E}{\partial v}$
  - 例子： $\Delta \boldsymbol{w}_{hj} = -\eta \frac{\partial E_k}{\partial \boldsymbol{w}_{hj}}$
- 核心问题求解：$\frac{\partial E}{\partial v}$，例$\frac{\partial E_k}{\partial \boldsymbol{w}_{hj}}$
- 本质问题（复合函数求偏导）：
  - $E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\beta_j - \theta_j) - y_j^k)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} \boldsymbol{w}_{hj}\boldsymbol{b}_h - \theta_j) - y_j^k)^2$

BP算法

- 求解余项：$\Delta \boldsymbol{w}_{\boldsymbol{hj}} = -\eta \frac{\partial E_k}{\partial \boldsymbol{w}_{\boldsymbol{hj}}}$：

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

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

已知 $\beta_j = \sum_{h=1}^{q} \boldsymbol{w}_{\boldsymbol{hj}} \boldsymbol{b}_{\boldsymbol{h}}$ 则 $\frac{\partial \beta_j}{\partial \boldsymbol{w}_{\boldsymbol{hj}}} = \boldsymbol{b}_{\boldsymbol{h}}$,

BP算法

- 已知激活函数$f()$为logistic函数且$f\left(\beta_j-\theta_j\right)$=$\hat{y}_j^k$则有

$$\frac{\partial \hat{y}_j^k}{\partial \beta_j}=f'\left(\beta_j-\theta_j\right)=f\left(\beta_j-\theta_j\right)(1-f\left(\beta_j-\theta_j\right))=\hat{y}_j^k(1-\hat{y}_j^k)$$

可得$\frac{\partial E_k}{\partial \beta_j}=\frac{\partial E_k}{\partial \hat{y}_j^k}\cdot\frac{\partial \hat{y}_j^k}{\partial \beta_j}=(\hat{y}_j^k-y_j^k)\,\hat{y}_j^k(1-\hat{y}_j^k)$,

令$g_j=-\frac{\partial E_k}{\partial \beta_j}=\ \hat{y}_j^k(1-\hat{y}_j^k)(y_j^k-\hat{y}_j^k)$，则

$$\frac{\partial E_k}{\partial \boldsymbol{w}_{\boldsymbol{hj}}}=\frac{\partial E_k}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial \boldsymbol{w}_{\boldsymbol{hj}}}=-g_j\boldsymbol{b}_{\boldsymbol{h}}$$

从而$\Delta\boldsymbol{w}_{\boldsymbol{hj}}$=$-\eta\frac{\partial E_k}{\partial \boldsymbol{w}_{\boldsymbol{hj}}}=\eta g_j\boldsymbol{b}_{\boldsymbol{h}}$

BP算法

- 求解$\Delta\theta_j = -\eta\frac{\partial E_k}{\partial \theta_j} = -\eta g_j$：

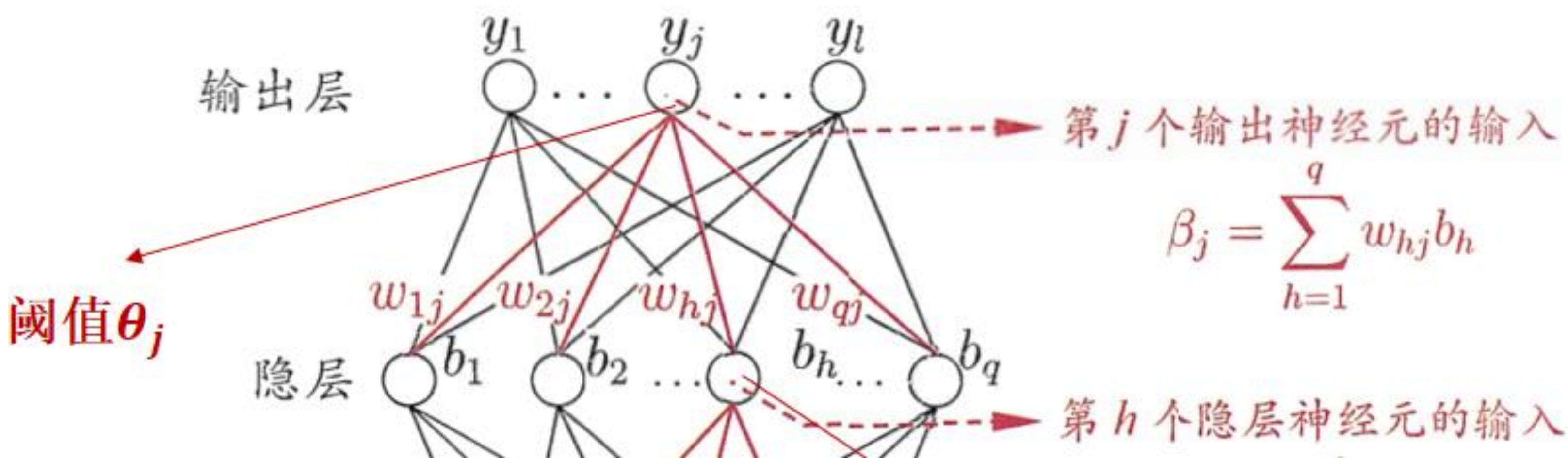
$$\frac{\partial E_k}{\partial \theta_j} = \frac{\partial E_k}{\partial \hat{y}_j^k}\cdot\frac{\partial \hat{y}_j^k}{\partial \theta_j} = -(\hat{y}_j^k - y_j^k)\hat{y}_j^k(1-\hat{y}_j^k) = g_j$$

$$\frac{\partial \hat{y}_j^k}{\partial \theta_j} = -f'(\beta_j - \theta_j) = -\hat{y}_j^k(1-\hat{y}_j^k)$$

BP算法

- 求解$\Delta v_{ih}=-\eta\frac{\partial E_k}{\partial v_{ih}}=$？：

$$\frac{\partial E_k}{\partial v_{ih}}=\frac{\partial E_k}{\partial b_h}\cdot\frac{\partial b_h}{\partial a_h}\cdot\frac{\partial a_h}{\partial v_{ih}}=\sum_{j=1}^{l}\frac{\partial E_k}{\partial \hat{y}_j^k}\cdot\frac{\partial \hat{y}_j^k}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial b_h}\cdot\frac{\partial b_h}{\partial a_h}\cdot\frac{\partial a_h}{\partial v_{ih}}$$



BP算法

$$T = \frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial a_h} = \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial a_h} = -b_h(1-b_h)\sum_{j=1}^{l} \boldsymbol{w_{hj}}\, g_j$$

根据前面推导已知: $\frac{\partial E_k}{\partial \beta_j} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} = -g_j$

有$\beta_j = \sum_{h=1}^{q} \boldsymbol{w_{hj} b_h}$, 则$\frac{\partial \beta_j}{\partial b_h} = \boldsymbol{w_{hj}}$

有$f(a_h - \boldsymbol{\gamma_h}) = b_h$

则 $\frac{\partial b_h}{\partial a_h} = f'(a_h - \boldsymbol{\gamma_h}) = f(a_h - \boldsymbol{\gamma_h})\big(1 - f(a_h - \boldsymbol{\gamma_h})\big) = b_h(1-b_h)$

令$e_h = -T = b_h(1-b_h)\sum_{j=1}^{l} \boldsymbol{w_{hj}}\, g_j$

BP算法

- 求解$\Delta v_{ih}=-\eta\frac{\partial E_k}{\partial v_{ih}}=$？：
- $\frac{\partial E_k}{\partial v_{ih}}=\frac{\partial E_k}{\partial b_h}\cdot\frac{\partial b_h}{\partial a_h}\cdot\frac{\partial a_h}{\partial v_{ih}}=T\cdot\frac{\partial a_h}{\partial v_{ih}}$
- 已知$T=-e_h$且$a_h=\sum_{i=1}^{d}v_{ih}x_i^k$则有

$$\frac{\partial a_h}{\partial v_{ih}}=x_i^k$$

$$\frac{\partial E_k}{\partial v_{ih}}=\frac{\partial E_k}{\partial b_h}\cdot\frac{\partial b_h}{\partial a_h}\cdot\frac{\partial a_h}{\partial v_{ih}}=-e_h x_i^k$$

得：$\Delta v_{ih}=-\eta\frac{\partial E_k}{\partial v_{ih}}=\eta e_h x_i$

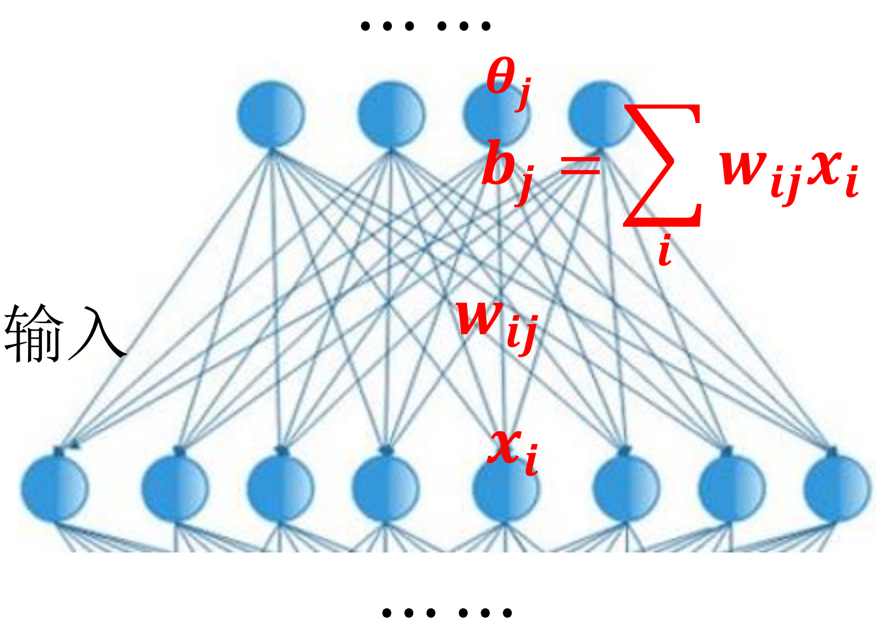
- 类似的可求解出：$\Delta\gamma_h=-\eta\frac{\partial E_k}{\partial\gamma_h}=-\eta e_h$

BP算法

- 根据教材公式(5.11)——(5.14)、(5.15)可归纳出参数更新通式：
- $\Delta w_{ij} = -\eta e_j x_i, \Delta\theta_j = \eta e_j,$
- $e_j = \frac{\delta E}{\delta b_j}$

$\frac{\delta E}{\delta b_j}$与$\frac{\delta E}{\delta a_t}$参见公式(5.15)，

a_t是上一层网络第t个神经元的输入



BP算法

- BP算法的核心思路：链式法则（复合函数求偏导）

  ① 利用前向传播，计算第n层输出值。

  ② 计算输出值和实际值的残差。

  ③ 将残差按影响逐步传递回第$n-1, n-2, \cdots, 2$层，以修正各层参数。（即所谓的误差逆传播）

- 累积误差逆传播(Accumulated Error Backpropagation)

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k$$

本质：采用的是标准梯度下降法。

BP算法局限性

- 容易过拟合！

  早停、正则化

- 容易陷入局部最优！

  选取多次初值、随机梯度下降法

- 难以设置隐层个数！

  试错法

BP算法

- How to tackle the overfitting?
  - Early stopping
  - Regularization
- How to find the optimizing neural network architecture?
  - Trial-by-error

Local & Global minimum

How to avoid the local minimum?

- Multiple initializations
- Simulated annealing technique
- Stochastic gradient descent