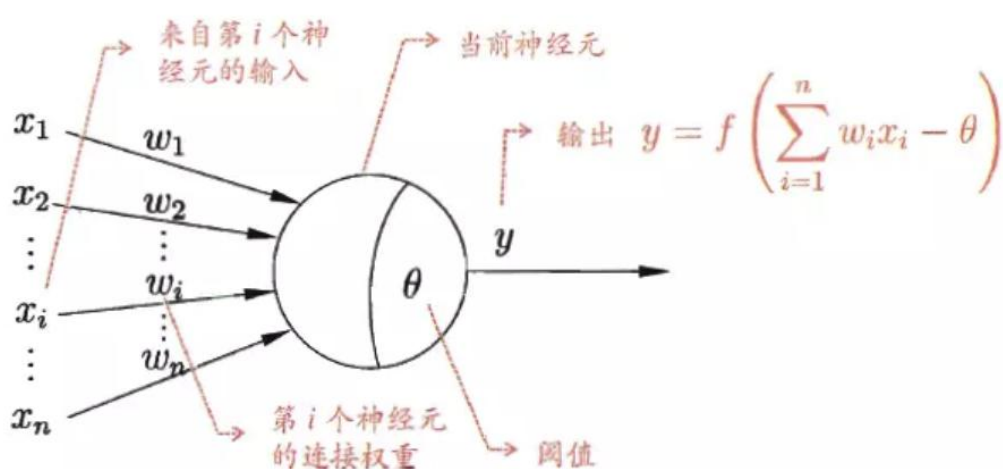


神经网络

一、神经元模型

神经网络最基本的成分是神经元模型（neuron）。在生物神经网络中，当神经元内的电位超过某个阈值，则会被激活，兴奋起来向其他神经元发送化学物质。

最早将这一神经元模型抽象起来的是 1943 年提出的 M-P 神经元，该神经元将接受 n 个来自其他神经元的信号，并将其带权组合并与阈值比较，并通过激活函数处理。



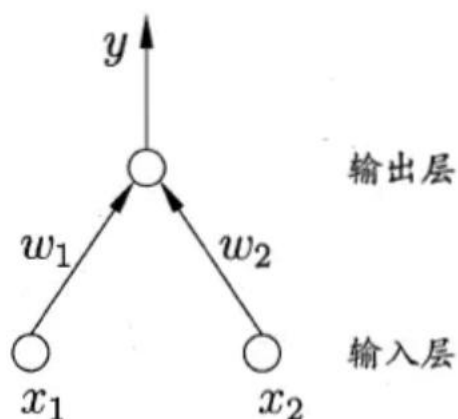
二、感知机模型与多层神经网络

感知机：两层神经元组成。

输入层：接受外界信号，传递给输出层

输出层：M-P 神经元，亦称“阈值逻辑单元”

可以证明：如果两类模型是线性可分的，那感知机的学习过程一定会收敛

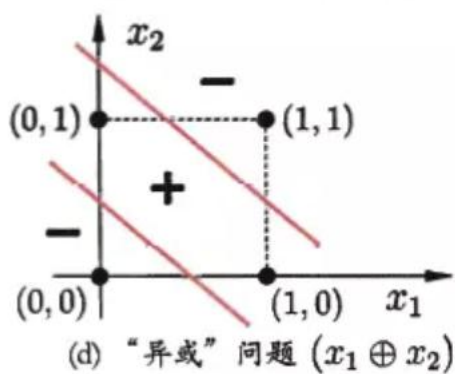


$$w_i \leftarrow w_i + \Delta w_i ,$$

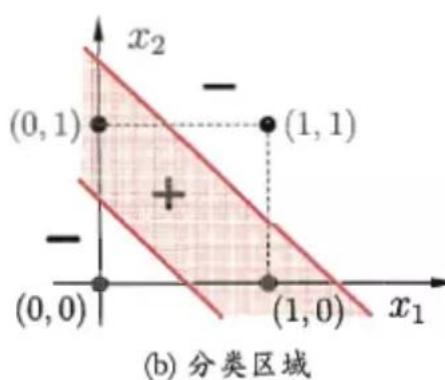
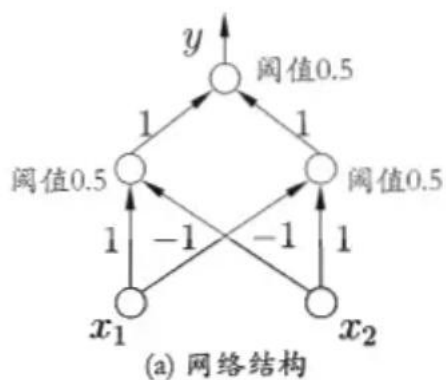
$$\Delta w_i = \eta(y - \hat{y})x_i ,$$

感知机的训练过程：

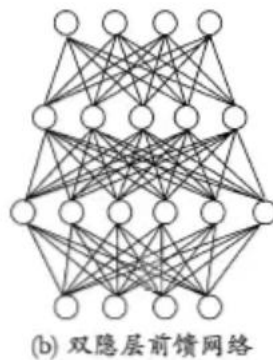
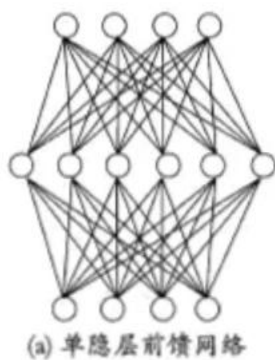
但是，对于线性不可分的问题，例如：异或问题，感知机模型是无法解决的。



解决这种线性不可分问题，我们应该使用多层神经网络。如：



一般的多层神经网络结构如下：



三、神经网络训练算法：BP 算法

神经网络的权重和偏置可以通过不断的训练进行优化，优化方法一般采用误差的反向传播，然后进行梯度下降法。

具体推导过程如下：

θ_j : 输出层第 j 个神经元的阈值 $\beta_j = \sum_{h=1}^H w_{hj} b_h$ (第 j 个输出神经元的输入)
 v_h : 隐层第 h 个神经元的阈值 $\alpha_h = \sum_{i=1}^I v_{hi} x_i$ (第 h 个隐层神经元的输入)
假设隐层和输出层都使用 Sigmoid 函数。
对训练集实例 (x_k, y_k) , 假定神经网络的输出为 $\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \dots, \hat{y}_L^k)$, 即
 $\hat{y}_j^k = f(\beta_j - \theta_j)$
神经网络在 (x_k, y_k) 上的均方误差为: $E_k = \frac{1}{2} \sum_{j=1}^L (\hat{y}_j^k - y_j^k)^2$
梯度下降 (gradient descent) 策略:
 η : 学习率
 $\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$
 $\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} = \left(\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \right) \cdot b_h$
 $g_j = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} = -(\hat{y}_j^k - y_j^k) \cdot f'(\beta_j - \theta_j)$
 $= \hat{y}_j^k (1 - \hat{y}_j^k) (y_j^k - \hat{y}_j^k)$
 $\therefore \Delta w_{hj} = \eta \cdot (-\frac{\partial E_k}{\partial w_{hj}}) = \eta \cdot g_j \cdot b_h = \eta g_j b_h$
类似可得 $\Delta \theta_j = -\eta \frac{\partial E_k}{\partial \theta_j} = -\eta g_j$
 $\Delta v_{hi} = -\eta \cdot \frac{\partial E_k}{\partial v_{hi}} = -\eta \cdot \frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial v_{hi}} = \left(-\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \right) \cdot \eta x_i$
 ~~$\frac{\partial E_k}{\partial b_h} = \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h}$~~ $b_h = f(\alpha_h - v_h)$
 $\therefore e_h = -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} = -\sum_{j=1}^L \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot f'(\alpha_h - v_h) = \sum_{j=1}^L w_{hj} g_j f'(\alpha_h - v_h)$
 $= b_h (1 - b_h) \sum_{j=1}^L w_{hj} g_j$
 $\therefore \Delta v_{hi} = \eta e_h x_i$

Sigmoid 函数的一个性质:
 $f(x) = f(x)(1 - f(x))$

BP 算法流程如下：

输入: 训练集 $D = \{(x_k, y_k)\}_{k=1}^m$;
学习率 η .

过程:

- 1: 在 $(0, 1)$ 范围内随机初始化网络中所有连接权和阈值
- 2: **repeat**
- 3: **for all** $(x_k, y_k) \in D$ **do**
- 4: 根据当前参数和式(5.3) 计算当前样本的输出 \hat{y}_k ;
- 5: 根据式(5.10) 计算输出层神经元的梯度项 g_j ;
- 6: 根据式(5.15) 计算隐层神经元的梯度项 e_h ;
- 7: 根据式(5.11)-(5.14) 更新连接权 w_{hj} , v_{ih} 与阈值 θ_j , γ_h
- 8: **end for**
- 9: **until** 达到停止条件

输出: 连接权与阈值确定的多层前馈神经网络
