《机器学习》
神经网络

主讲人：王宗威

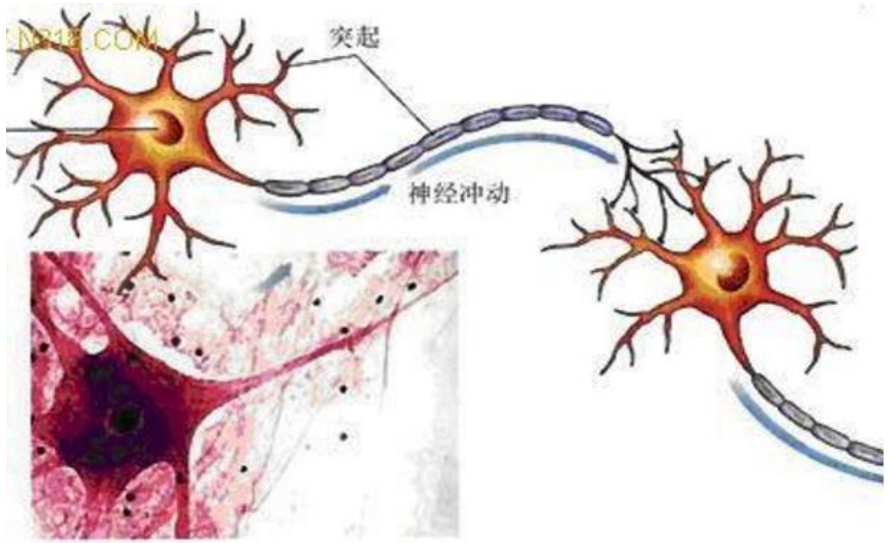
- 神经网络是什么？
- 神经网络是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应。

- 那这个是我们要学习的神经网络吗？

不全面！

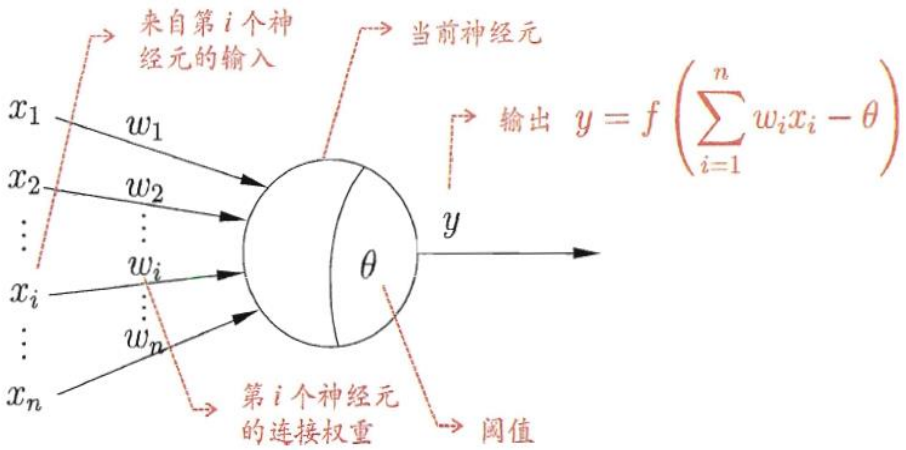
我们机器学习里的神经网络指的是“神经网络学习”。

神经元模型

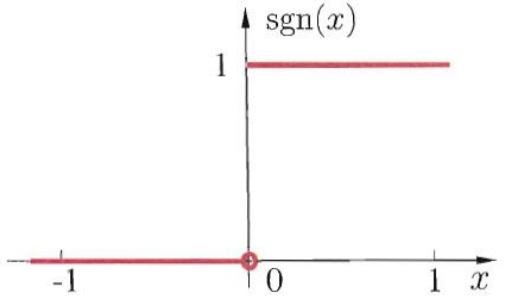


神经元是构成神经系统结构和功能的基本单位。每个神经元可以有一或多个树突，可以接受刺激并将兴奋传入细胞体。

神经元模型

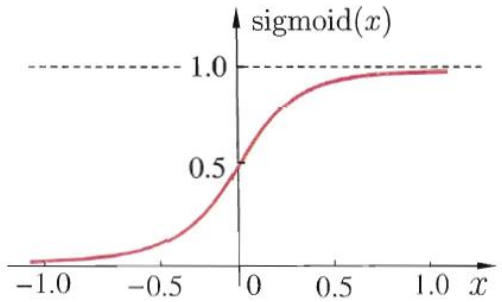


一个神经元模型是否能输出，是由其他神经元的输入乘以相应的权值与阈值作比较判断的。这里提出一点就是激活函数，常采用sigmoid函数。前者阶跃函数有着不连续不光滑的特点，如图:



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

(a) 阶跃函数

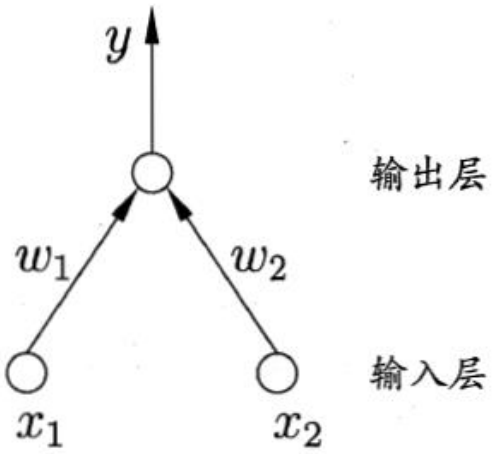


$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

(b) Sigmoid 函数

感知机

- 感知机是由两层神经网络组成，输入层和输出层，也称“阈值逻辑单元”。



感知机

- 从上面的例子可以看出，在整个训练过程中，阈值和权重是我们最关心的两个变量。
- 　　这里，我们有个办法将二者联系起来。就是规定，阈值可看作一个固定输入为-1.0的“哑结点”所对应的权重。这样我们就可以统一为权重的学习。

学习规则非常简单, 对训练样例 $(\boldsymbol{x}, y)$, 若当前感知机的输出为 $\hat{y}$, 则感知机权重将这样调整:

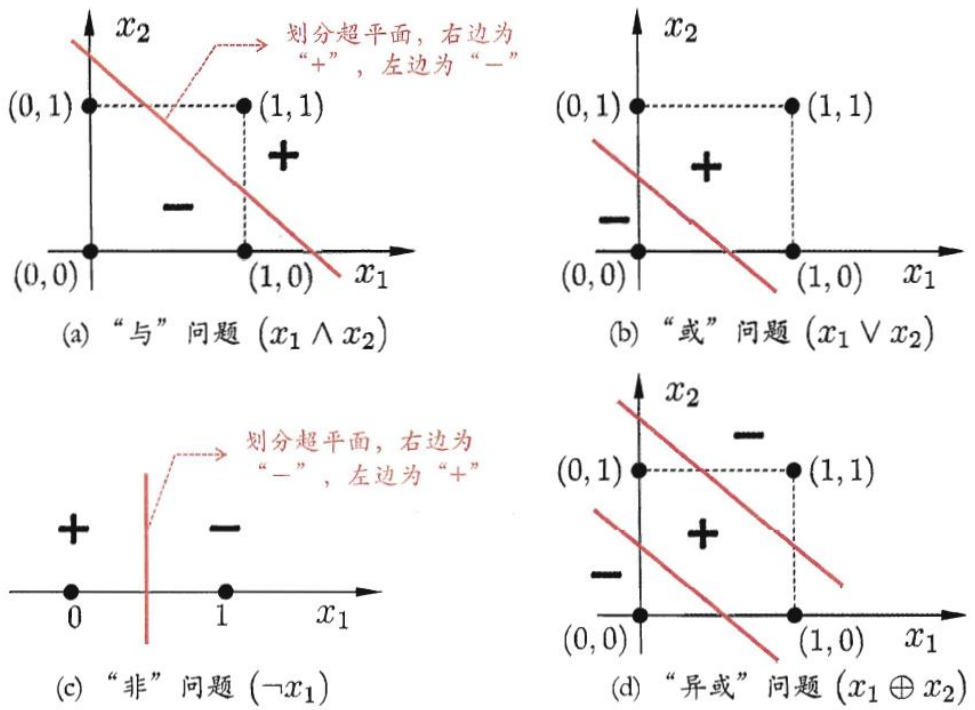
$$w_i \leftarrow w_i + \Delta w_i \ , \tag{5.1}$$

$$\Delta w_i = \eta(y - \hat{y})x_i \ , \tag{5.2}$$

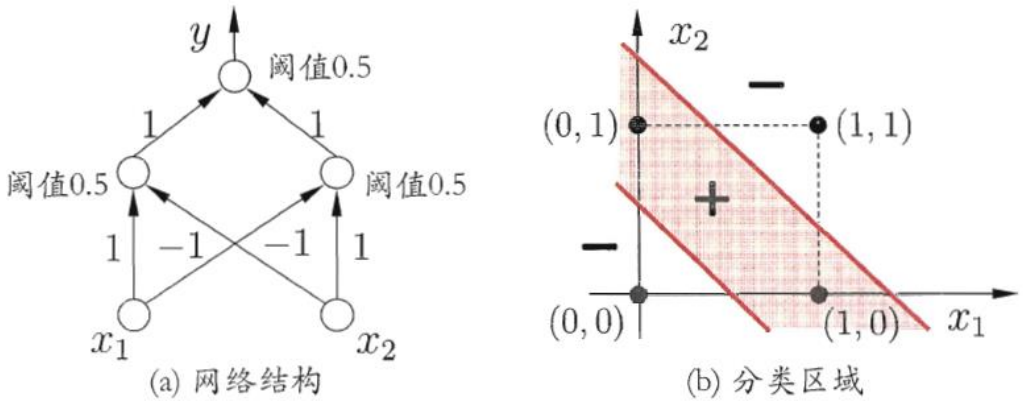
其中 $\eta \in (0, 1)$ 称为学习率(learning rate). 从式(5.1) 可看出, 若感知机对训练样例 $(\boldsymbol{x}, y)$ 预测正确, 即 $\hat{y} = y$, 则感知机不发生变化, 否则将根据错误的程度进行权重调整.

问题?
这个权重更新公式
怎么来的

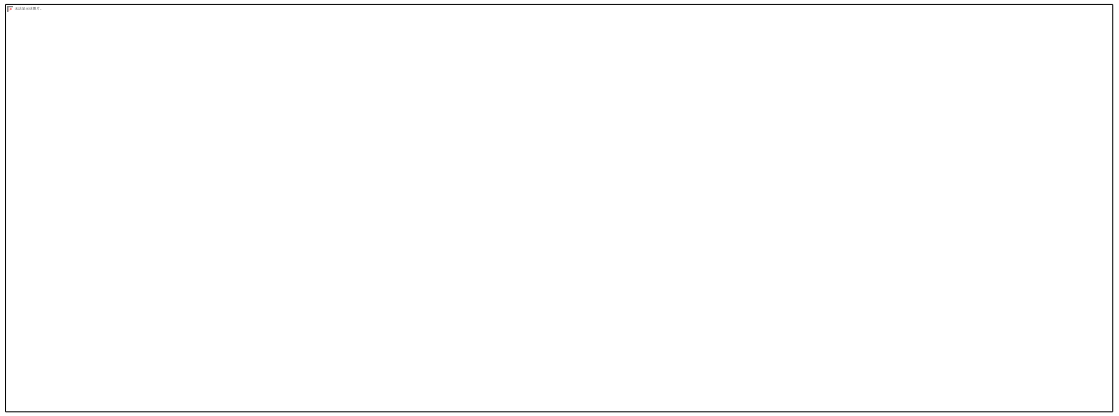
- 感知机学习结构太简单了！对于一些复杂的情况，感知机的能力就达不到我们的需求。比如如图，我们发现异或情况就是感知机解决不了的：



- 我们为了应付更复杂的情况，提出在输出层和输入层之间添一层神经网络，称为隐层或者隐含层，如图：

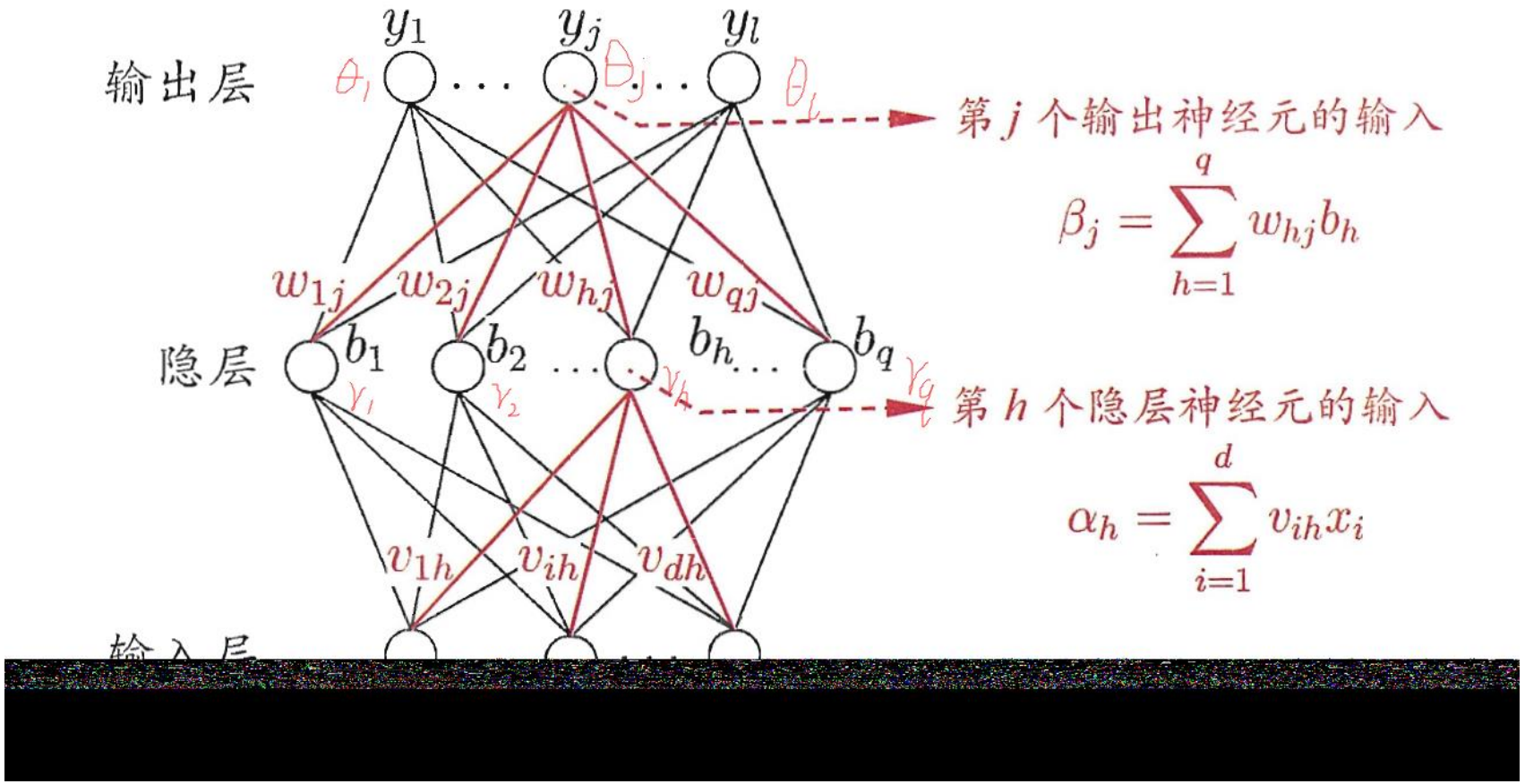


- 我们继续扩展，产生这样一张图。这样的一张图，每一层神经元与下一层神经元完全项链，神经元之间不存在同层连接，也不存在跨层连接，这样的网路称为“多层前馈网络”。这里的前馈不是指信号不能向后传，而是指在网络拓扑结构上不能出现环或者回路。



整个神经网络的学习，其实就是调整权值和阈值的过程

误差逆传播算法（error BackPropagation，简称BP）



对训练例 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, 假定神经网络的输出为 $\hat{\boldsymbol{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$, 即

$$\hat{y}_j^k = f(\beta_j - \theta_j) \ , \tag{5.3}$$

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

$$E_k = \frac{1}{2} \sum_{j=1}^{l} (\hat{y}_j^k - y_j^k)^2 \ . \tag{5.4}$$

- BP是一个迭代学习的算法，基于梯度下降对参数w，j，v，r进行更新。

$$\Delta w_{hj} = \eta g_j b_h \ .$$

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

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

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

BP算法伪代码

输入: 训练集 $D = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^{m}$;
学习率 η.

过程:

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

图 5.8 误差逆传播算法

标准BP算法和累积BP算法

- BP算法的目标是最小化训练集D上的累积误差：

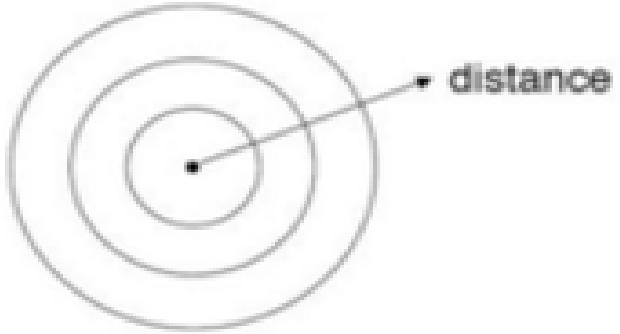
$$E = \frac{1}{m}\sum_{k=1}^{m} E_k \ ,$$

- 标准BP算法是针对一个训练样例更新权重和阈值，而累积BP算法是将整个训练集D读取一遍后才更新一次。
- 标准BP算法一般来说会获得更好的解。但是累积BP算法更快。

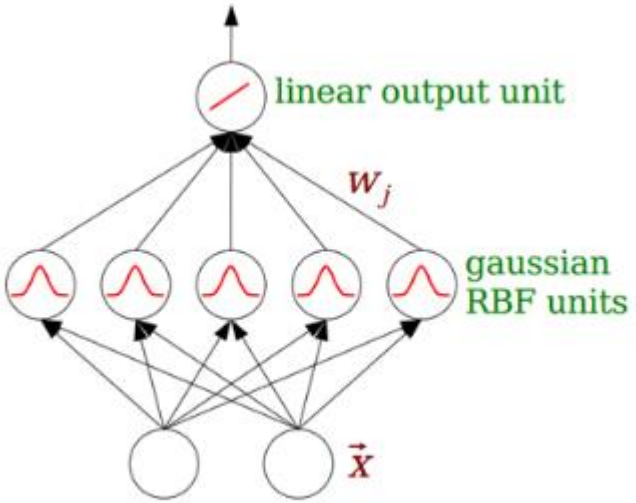
常见的神经网络

- RBF网络（径向基函数网络）
- 径向基函数：所谓径向基函数，其实就是某种沿径向对称的标量函数。通常定义为空间中任一点x到某一中心c之间欧氏距离的单调函数，可记作k(||x-c||)，其作用往往是局部的，即当x远离c时函数取值很小。例如高斯径向基函数：

$$\rho(\boldsymbol{x}, \boldsymbol{c}_i) = e^{-\beta_i \|\boldsymbol{x}-\boldsymbol{c}_i\|^2} .$$



RBF网络表示



$$\varphi(\boldsymbol{x}) = \sum_{i=1}^{q} w_i \rho(\boldsymbol{x}, \boldsymbol{c}_i) \ ,$$

第一步：确定隐层神经元中心ci，常采用随机采样、聚类的方法。
第二步：利用BP算法确定参数。

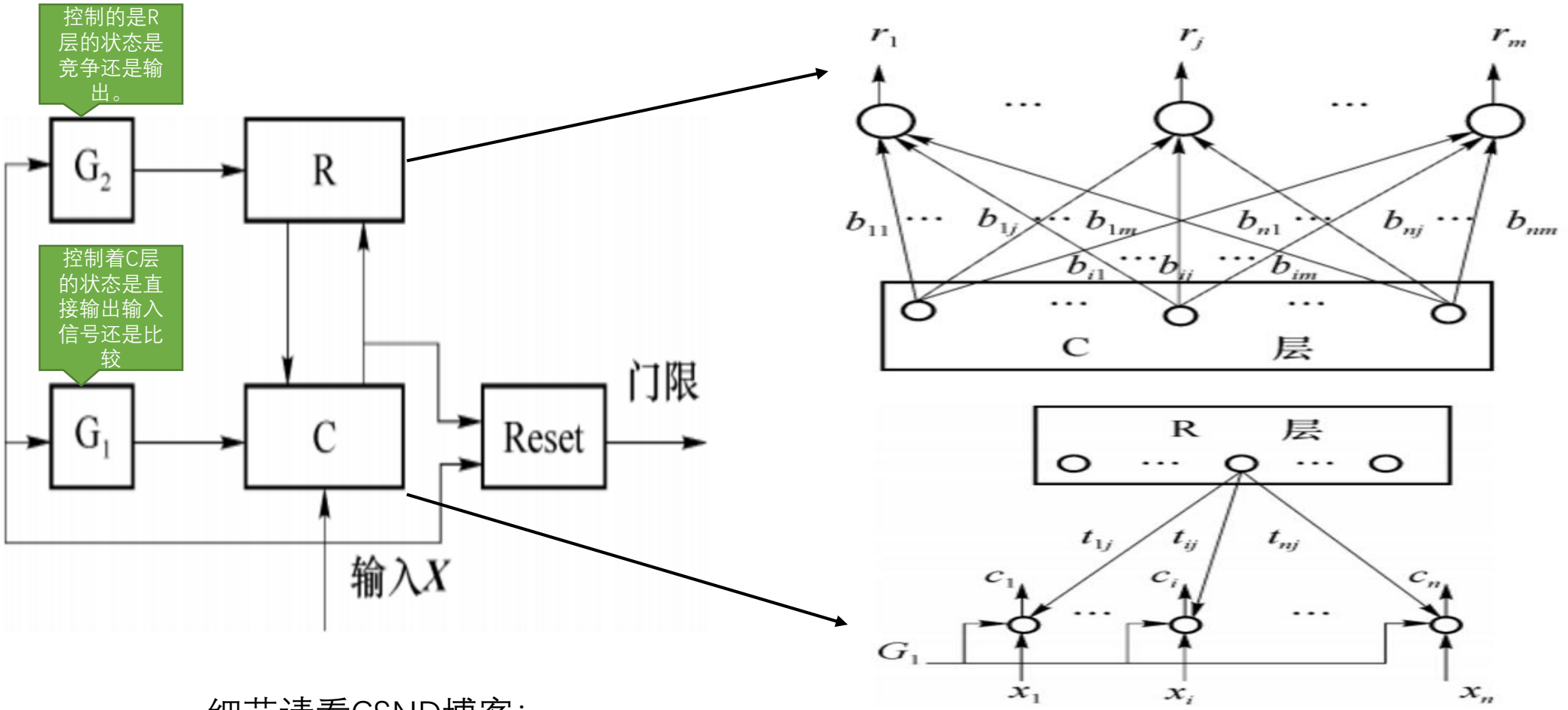
优点：逼近能力，分类能力和学习速度等方面都优于BP神经网络，结构简单、训练简洁、学习收敛速度快、能够逼近任意复杂度的连续函数，克服局部极小值问题。原因在于其参数初始化具有一定的方法，并非随机初始化。

细节请看CSND博客：
https://blog.csdn.net/ecnu18918079120/article/details/53365341

ART（Adaptive Resonance Theory，自适应谐振理论）网络

- 这种神经网络属于竞争型学习，其根本区别在于仅有一个竞争获胜的神经元被激活，其他神经元状态被抑制。

该网络由比较层、识别层、识别阈值和重置模块构成。如图



细节请看CSND博客：
https://blog.csdn.net/lg1259156776/article/details/47780695

SOM（Self-Organizing Map，自组织映射）网络

- 它是一种无监督的竞争学习网络，学习过程中不需要任何监督信息。SOM网络将高维数据映射到低维空间中，一般是一维或者两维，并且映射过程中保持数据的拓扑结构不变，即高维空间中相似的数据在低维空间中接近。

- SOM由两层神经元组成：输入层和输出层。输入层的每个神经元和输出层的所有神经元连接。输入层的神经元数量由输入空间决定。输出层神经元的数量由用户定义，每个输出神经元对应一个位置信息（可以是一维空间的坐标或者二维空间中的坐标），并且每个输出神经元拥有一个权重向量，权重向量的维度等于输入神经元数。

- 对每个输入实例，计算其和所有输出神经元权重向量之间的距离，距离最小的输出神经元称为获胜神经元。该实例在低维空间中的位置就是获胜神经元所处的位置。SOM训练的目的就是找到一组权重向量，使得输入数据在低维空间中拓扑结构不变。

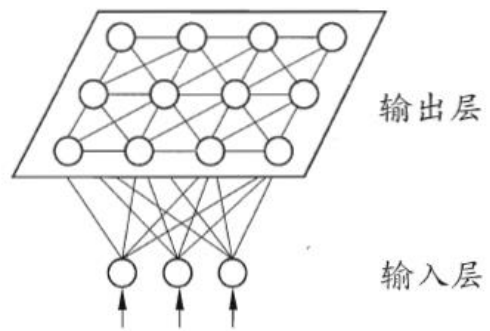


图 5.11　SOM 网络结构

SOM网络的学习过程包含6个步骤：

1. 初始化所有的权重向量；
2. 从训练数据集中随机选择一个实例作为网络的输入；
3. 计算每个权重向量和输入向量之间的距离，取距离最小的权重向量对应的输出神经元作为当前输入的BMU；
4. 为BMU计算邻域半径：邻域半径开始比较大，随时间逐渐减小；
5. 对位于BMU邻域内的所有输出神经元，更新其权重向量；
6. 重复2-5直至N次。

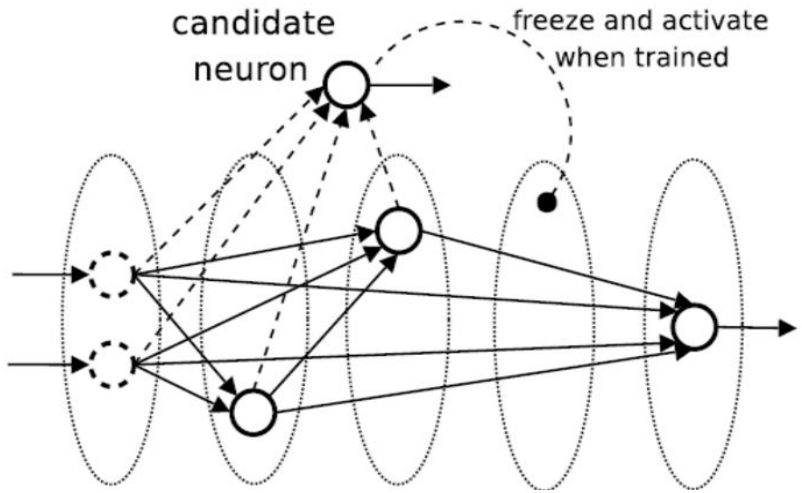
距离最小的输出神经元称为获胜神经元，也可以称为Best Match Unit，简写为BMU

级联相关网络

- 一般的神经网络是固定好拓扑结构，然后训练权重和阈值。级联相关神经网络是从一个小网络开始，自动训练和添加隐含单元，最终形成一个多层的结构。

- 首先，**候选神经元**连结到所有的**输入**和**隐含**神经元(也就是图中的虚线)，并且候选神经元的输出不连结到网络上；
- 然后固定住图中的实线部分，只训练候选神经元的权重(也就是图中的虚线)；
- 当权重训练好之后，就将候选神经元安装到图中空白的层上，也就是第四个区域，这时候选项的连接权就不能再改变了；
- **接着**，

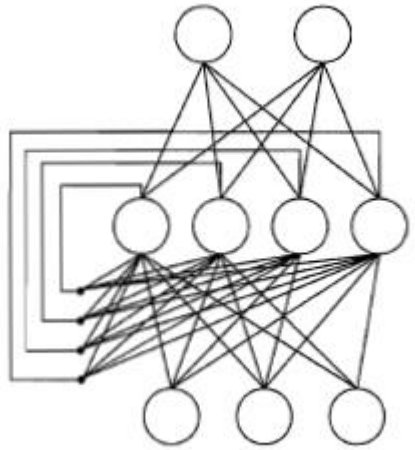
  将候选神经元连结到网络的输出上，这时候选神经元被**激活**，开始训练网络的所有输出连接权；
- 重复以上步骤；



细节请看CSND博客：
https://blog.csdn.net/xc_xc_xc/article/details/53163478

Elman网络

- 让一些神经元的输出反馈回来作为输入信号，使得网络在t时刻的输出状态不仅与t时刻的输入有关，还与t-1时刻的网络状态有关。



Boltzmann机

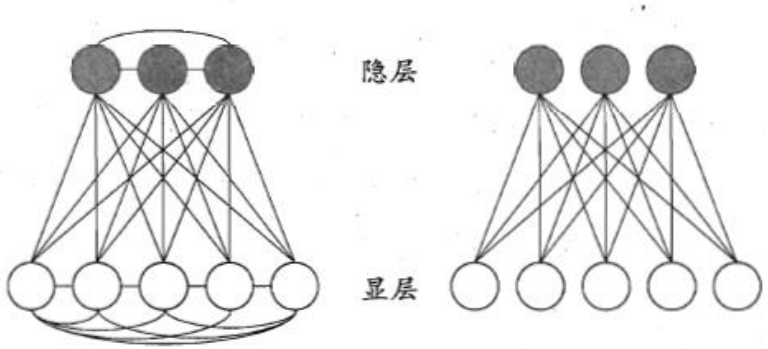
- 基于能量的模型
- 神经元分为两层：显层和隐层。显层用于表示数据的输出和输入，隐层则理解为数据的内在表达。

Boltzmann 机中的神经元都是布尔型的, 即只能取 0、1 两种状态, 状态 1 表示激活, 状态 0 表示抑制. 令向量 $\boldsymbol{s} \in \{0,1\}^n$ 表示 n 个神经元的状态, w_{ij} 表示神经元 i 与 j 之间的连接权, θ_i 表示神经元 i 的阈值, 则状态向量 $\boldsymbol{s}$ 所对应的 Boltzmann 机能量定义为

$$E(\boldsymbol{s}) = -\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}s_i s_j - \sum_{i=1}^{n}\theta_i s_i \ . \tag{5.20}$$

- 状态向量出现在概率为 $P(\boldsymbol{s}) = \dfrac{e^{-E(\boldsymbol{s})}}{\sum_{\boldsymbol{t}} e^{-E(\boldsymbol{t})}}$.
- Boltzmann机的训练过程就是将训练样本视为一个状态向量，使其出现的概率尽可能大。

- 在现实情况中，很难解决现实问题。所以提出了



- 受限Boltzmann机常用对比散度算法来训练。

假定网络中有 d 个显层神经元和 q 个隐层神经元, 令 $\boldsymbol{v}$ 和 $\boldsymbol{h}$ 分别表示显层与隐层的状态向量, 则由于同一层内不存在连接, 有

$$P(\boldsymbol{v}|\boldsymbol{h}) = \prod_{i=1}^{d} P(v_i \mid \boldsymbol{h}) \ , \tag{5.22}$$

$$P(\boldsymbol{h}|\boldsymbol{v}) = \prod_{j=1}^{q} P(h_j \mid \boldsymbol{v}) \ . \tag{5.23}$$

CD 算法对每个训练样本 $\boldsymbol{v}$, 先根据式(5.23)计算出隐层神经元状态的概率分布, 然后根据这个概率分布采样得到 $\boldsymbol{h}$; 此后, 类似地根据式(5.22)从 $\boldsymbol{h}$ 产生 $\boldsymbol{v}'$, 再从 $\boldsymbol{v}'$ 产生 $\boldsymbol{h}'$; 连接权的更新公式为

$$\Delta w = \eta \left(\boldsymbol{v}\boldsymbol{h}^{\top} - \boldsymbol{v}'\boldsymbol{h}'^{\top} \right) \ . \tag{5.24}$$

细节请看CSND博客:
https://blog.csdn.net/itplus/article/details/19408143

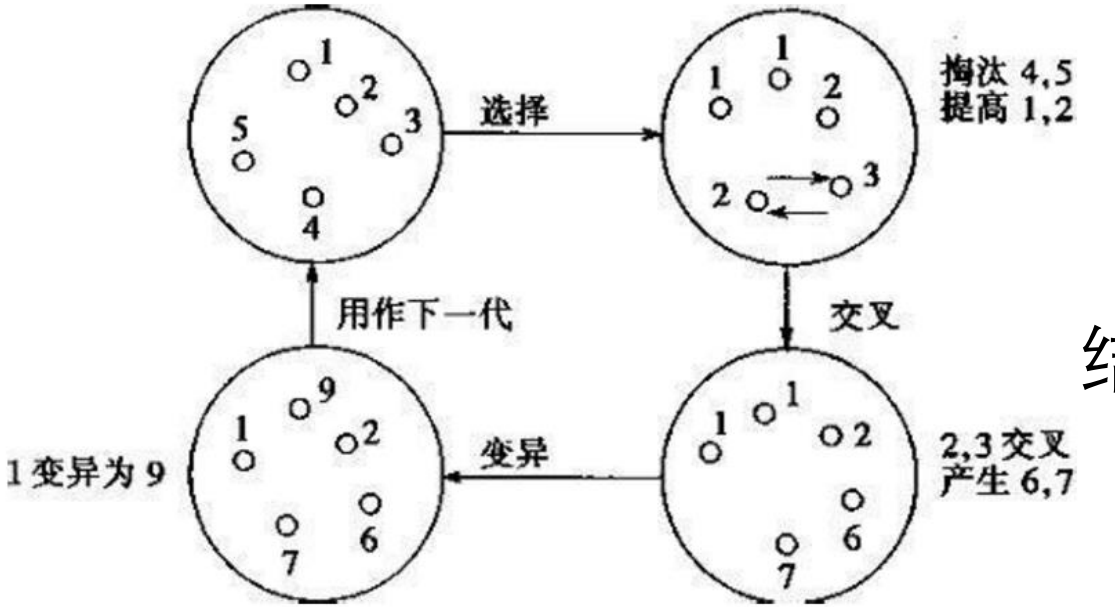
设置隐层的神经元个数?

- 利用试错法:
- 早停: 将数据分为训练集和验证集，训练集来计算梯度、更新连接权和阈值。验证集来估计误差，当训练集误差降低且验证集误差升高时，返回最小验证集误差的连接权和阈值。
- 正则化: 在误差目标函数里加一个描述网络复杂度的成分。

$$E = \lambda \frac{1}{m} \sum_{k=1}^{m} E_k + (1-\lambda) \sum_i w_i^2 , \qquad (5.17)$$

加入连接权和阈值的平方和可以是训练结果偏好比较小的连接权和阈值

全局最小和局部最小



提出如何跳出局部最

- 以多组不同参数值
- 模拟退火：每一步 结果。迭代过程中接受“次优解
- 随机梯度下降
- 遗传算法：利用生物中优胜劣汰的原则，对数据集进行训练。利用遗传算法就是在交叉和变异两步过程的随机性跳出局部最小。

深度学习

- 深度学习中会遇到的一个问题：发散不能收敛。
- 解决方法为：
- 预训练+微调
- 权共享：无论是卷积层还是采样层，都是用的相同的连接权，减少了大量的参数数目。