智能控制
第五章 神经控制

刘山
浙江大学控制科学与工程学院
2022/12/11

人工神经网络（ANN）

- 人工神经网络（artificial neural network, ANN）是由大量简单的处理单元（神经元，neurons）互联而成的并行分布式系统。它不是人脑的真实描述，是对人脑简单的抽象和模拟，反映人脑的基本特征。

- ANN建立在神经科学、数学、物理学、计算机科学的基础上，是一种新的计算机制和模型。可解决一些传统算法所难以解决的问题。

- ANN可以用电子线路来实现，也可以用计算机程序来模拟。用硬件实现后，系统达到稳定即为计算完毕，计算效率极大地提高。

建模问题

- 模型评价
  - 模型精度
    - 通常根据对学习样本和测试样本的输出误差来评价。
  - 模型结构的复杂度
    - 取决于实际应用。
  - 模型的自适应性
    - 对变化的环境，可方便地调整模型的结构和参数，且新的调整不会破坏或完全丢失原来学习已获得的结果。
- 神经网络
  - 神经网络具有很强的学习性能和自适应性。
  - 基于模型的神经控制方法不是基于对象的数理数学模型，而是基于对象的神经元网络模型。

神经网络在控制中的作用

- 在传统的控制系统中用于动态系统建模，充当对象模型。
- 在反馈控制系统中直接充当控制器的作用。
- 在传统控制系统中起优化计算作用。
- 与其他智能控制方法如模糊逻辑、遗传算法、专家控制等相融合。

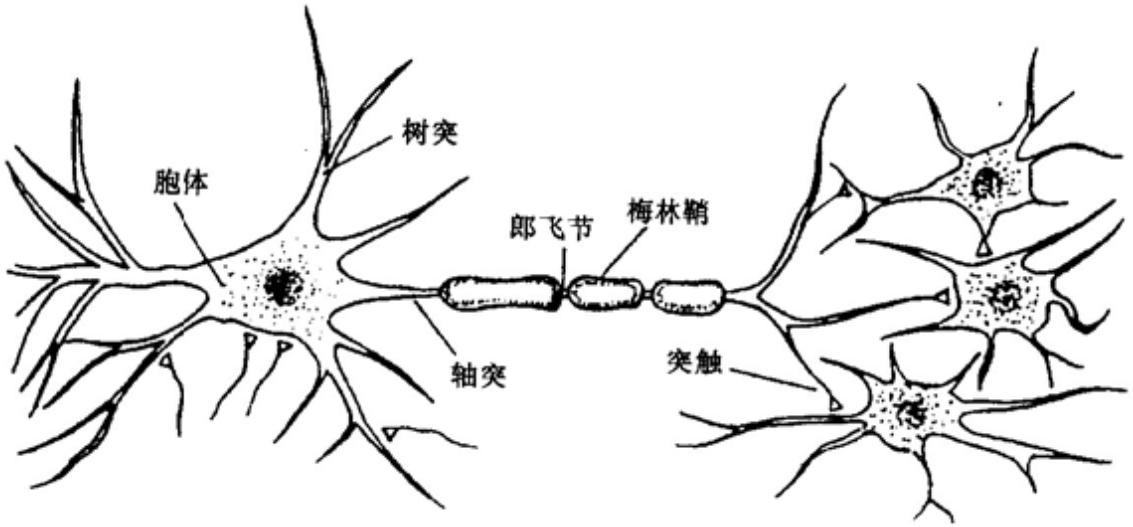
内容

- 1、神经网络模型
- 2、神经网络类型
- 3、基于神经网络的系统建模与辨识
- 4、神经网络PID控制
- 5、神经网络控制结构方案
- 6、模糊神经网络
- 7、神经控制的应用

一、神经网络模型

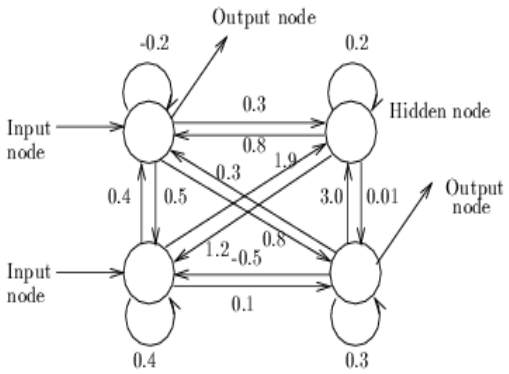
神经网络

- 神经网络是由大量人工神经元（处理单元）广泛互联而成的网络，它是在现代神经生物学和认识科学对人类信息处理研究的基础上提出来的，具有很强的自适应性和学习能力、非线性映射能力、鲁棒性和容错能力。
- 生物神经元
  - 输入与输出
  - 兴奋与抑制



人工神经网络

- 构成：
  - 由许多神经元模型组成的信息处理网络；
  - 并行分布结构；
  - 每个神经元具有单一输出，并且能够与其它神经元连接；
  - 存在许多（多重）输出连接方法，每种连接方法对应一个连接权系数。



- 性质：
  - 由基于生物神经元特性的互连模型来构造一个表示大脑基本组分的神经元模型，对逻辑操作系统表现出通用性。
  - 连接机制结构的基本处理单元与神经生理学类比，为神经元。
  - 每个构造网络的神经元模型模拟一个生物神经元。
  - 神经网络的结构是由基本处理单元及其互连方法决定的。
  - 本质上是大规模非线性动力学系统。可逼近任意复杂的非线性关系。

人工神经网络

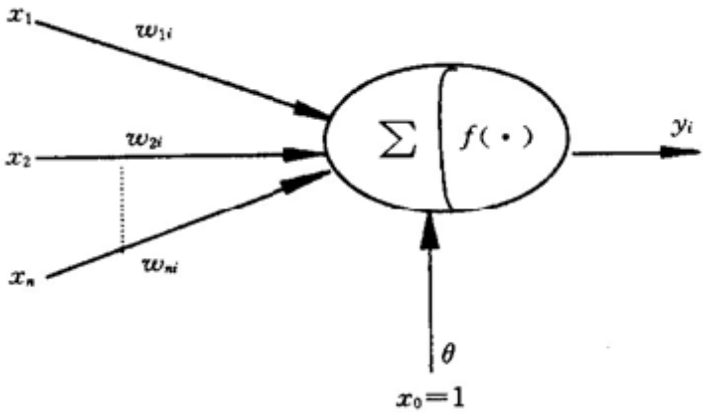
- 存储的知识体现在神经元以及神经元之间的连接权上。
- 具有很强的鲁棒性，适于对不完全信息的处理。
- 具有很强的容错性，部分结构变化后，能够进行自我恢复。
- 具有自学习和自适应能力，从环境中获取知识和经验。
- 若硬件实现后，实现模拟计算和不精确计算，不同于编程实现的计算机的离散计算和精确计算。
- 具有联想能力。

人工神经元模型

- 人工神经元是对生物神经元的一种模拟与简化，它是神经网络的基本处理单元
- 输入输出关系为

$$I_i = \sum_{j=1}^{n} w_{ij} x_j - \theta_i$$

$$y_i = f(I_i)$$



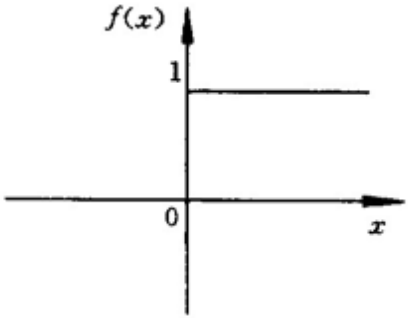
w_{ij}表示从神经元j到神经元i的连接权值；
θ_i为阈值；
$f(\cdot)$ 称为激发函数或作用函数。

人工神经元是一个多输入、单输出的非线性元件。

常用激发函数

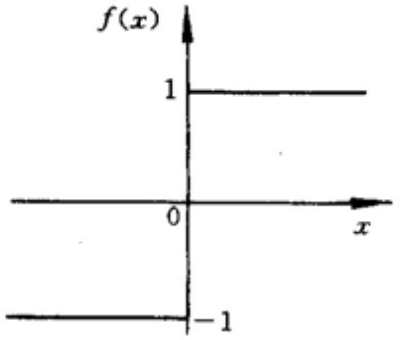
- 阈值型函数
  - 阶跃函数

$$f(x)=\begin{cases}1, & x\geq 0\\ 0, & x<0\end{cases}$$



  - 符号函数

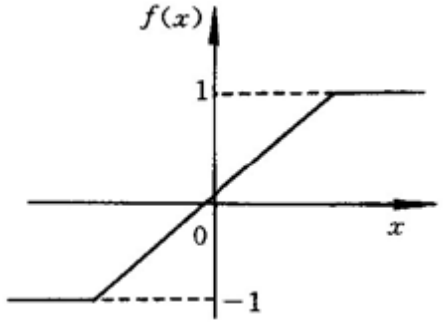
$$\mathrm{sgn}(x)=f(x)=\begin{cases}1, & x\geq 0\\ -1, & x<0\end{cases}$$



常用激发函数

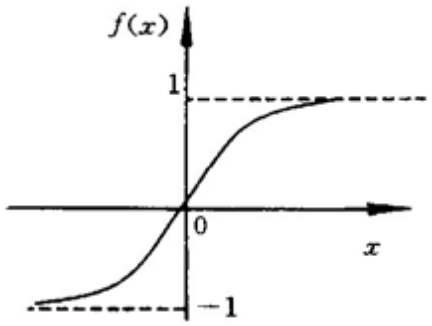
- 饱和型函数

$$f(x)=\begin{cases}1 & x\geq 1/k\\ kx & -1/k\leq x<1/k\\ -1 & x<1/k\end{cases}$$



- 双曲函数

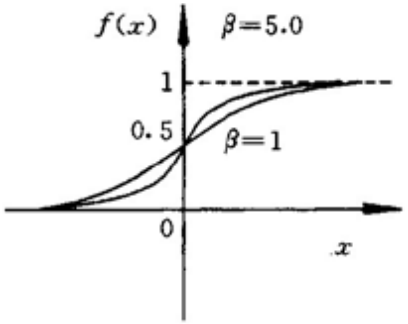
$$f(x)=\tanh(x)$$



常用激发函数

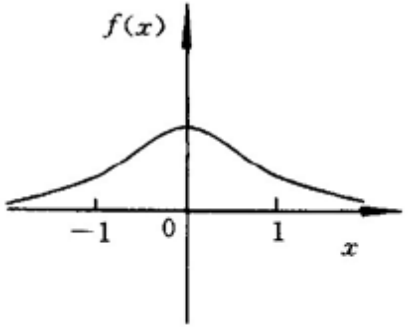
- **S 型函数（Sigmoid）**

$$f(x)=\frac{1}{1+\exp(-\beta x)},\qquad \beta>0$$



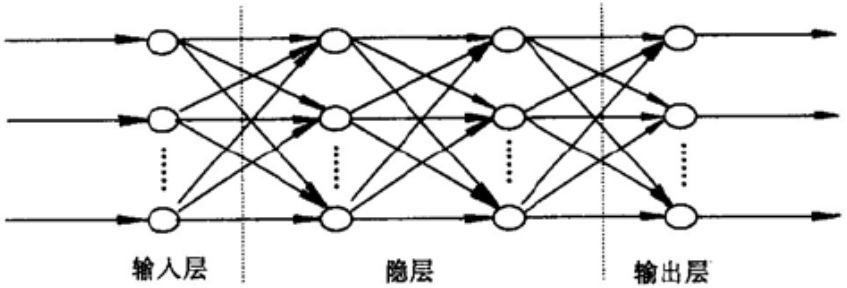
- **高斯函数（Radial Basis Function，RBF）**

$$f(x)=e^{-x^2/\delta^2}$$

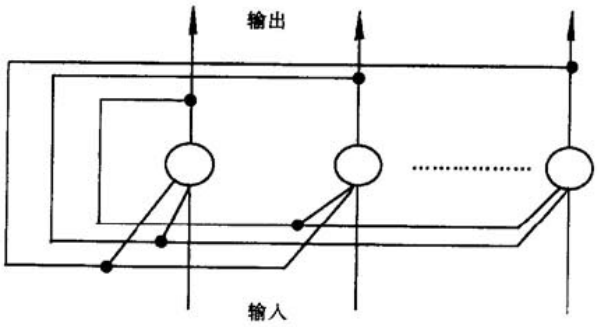


神经网络结构模型

- 前馈型神经网络（Feedforward NNs）



- 反馈型神经网络（Feedback NNs）



- 自组织网络（Self-organizing NNs）

神经网络的学习

- 通过一定的学习算法实现对突触结合强度（权值）的调整，使其达到具有记忆、识别、分类、信息处理和问题优化求解等功能。

- 要使人工神经网络具有学习能力，就是使神经网络的知识结构变化，使神经元间的结合模式变化，这同把连接权向量用什么方法变化是等价的。

神经网络的学习方法

- 有师学习（**supervised learning** 监督学习）
  - 网络输出和期望输出（即教师信号）进行比较，然后根据两者之间的差异调整网络的权值，最终使差异变小。
- 无师学习（**unsupervised learning** 无监督学习）
  - 输入模式进入网络后，网络按照预先设定的规则（如竞争规则）自动调整权值，使网络最终具有模式分类等功能。
- 强化学习（**reinforcement learning**，再励学习，评价学习）
  - 有师学习的特例，介于有师学习与无师学习之间。
  - 是一个试探评价过程。它不需要教师给出目标输出，而是通过环境受到试探动作的影响后，给出一个评价信号作为强化信号。
  - 强化学习算法采用一个“评论员”来评价与给定输入相对应的神经网络输出的优度。

常用的神经网络训练方法

- δ 学习规则
  - 有师学习。
  - 梯度下降法。
- 模拟退火算法
  - 有师学习。
  - 概率式学习。
  - 基于模拟退火的统计优化方法。
  - 网络处于某一状态的概率主要取决于在此状态下的能量，能量越低，概率越大。同时，此概率还取决于温度参数T。
- **Hebb 学习规则**
  - 无师学习。
  - 联想式学习方法。
  - 两个神经元同时处于激发状态时，它们之间的连接强度将得到加强。
- 竞争式学习
  - 无师学习
  - 神经网络中高层次的神经元对低层次神经元的输入模式进行竞争识别。

Delta（δ）学习规则

- 误差准则函数 $E=\frac{1}{2}\sum_{p=1}^{p}(d_p-y_p)^2=\sum_{p=1}^{p}E_p$

  d_p代表期望的输出（教师信号）；$y_p=f(WX_p)$为网络的实际输出；W是网络的所有权值组成的向量。
- 用梯度下降法来调整权值W，使准则函数最小。求解基本思想是沿着E的负梯度方向不断修正W值，直到E达到最小。

$$\nabla W=\eta(-\frac{\partial E}{\partial W_i})$$

$$\frac{\partial E}{\partial W_i}=\sum_{p=1}^{p}\frac{\partial E_p}{\partial W_i}$$

其中 $E_p=\frac{1}{2}(d_p-y_p)^2$

Delta（δ）学习规则

用 θ_p 表示 WX_p，则有

$$\frac{\partial E_p}{\partial W_i}=\frac{\partial E_p}{\partial \theta_p}\frac{\partial \theta_p}{\partial W_i}=\frac{\partial E_p}{\partial y_p}\frac{\partial y_p}{\partial \theta_p}X_{ip}=-(d_p-y_p)f'(\theta_p)X_{ip}$$

W的修正规则为

$$\Delta W_i=\eta\sum_{p=1}^{p}(d_p-y_p)f'(\theta_p)X_{ip}$$

上式称为δ学习规则，又称误差修正规则。

定义误差传播函数δ为

$$\delta=\frac{\partial E_p}{\partial \theta_p}=-\frac{\partial E_p}{\partial y_p}\frac{\partial y_p}{\partial \theta_p}$$

Delta（δ）学习规则

- δ规则实现了E中的梯度下降，使误差函数达到最小值。
- δ学习规则只适用于线性可分函数，无法用于多层网络。
- **BP网络的学习算法称为BP算法，是在δ规则基础上发展起来的，可在多层网络上有效地学习。**

概率式学习

- 概率式学习：从统计力学、分子热力学和概率论中关于系统稳态能量的标准出发，进行神经网络学习的方式。
- 特点：
  - 神经网络处于某一状态的概率主要取决于在此状态下的能量，能量越低，概率越大。
  - 同时，此概率还取决于温度参数***T***。
  - ***T***越大，不同状态出现概率的差异便越小，较容易跳出能量的局部极小点而到全局的极小点；
  - ***T***越小时，情形正相反。
- 操作：
  - 热静力学操作：用于安排降温过程；
  - 随机状态转移：用于搜索特定温度下的平衡态。
- 概率式学习的典型代表是**Boltzmann**机学习规则。它是基于模拟退火的统计优化方法，因此又称模拟退火算法。

概率式学习

- 模拟退火算法（Simulated Annealing）
  - Metropolis提出原始的SA算法（1953年），Kirkpatrick提出现代的SA算法（1982年），是基于Monte-Carlo迭代求解策略的一种随机寻优算法。
  - 从某一较高初温出发，伴随温度参数的不断下降，结合概率突跳特性在解空间中随机寻找目标函数的全局最优解。
  - 能够在局部最优解处概率性跳出，最终趋于全局最优。

- METROPOLIS准则
  - 能量降低，接受为新的状态，否则按一定概率接受。

概率式学习

- **METROPOLIS准则**
  - 粒子在温度T时趋于平衡的概率为$\exp(-\Delta E/(kT))$，其中E为温度T时的内能，ΔE为其改变量，k为Boltzmann常数。
  - 用固体退火模拟优化问题，将内能E模拟为目标函数值f，温度T演化成控制参数t，得到解优化问题的模拟退火算法。
  - 由初始解i和控制参数初值t开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减t值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。
  - 退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的初值t及其衰减因子Δt、每个t值时的迭代次数L和停止条件S。

概率式学习

- 模拟退火算法的步骤：
  - 1、初始化。任选初始解S_0, 给定初始温度T_0, 终止温度T_f. 令迭代指标$k=0$, $S_k=S_0$, $T_k=T_0$.
  - 2、令$i=0$，$S_i=S_k$;（同一温度下，寻找平衡态）
  - 3、随机产生S_i的一个邻域解S_n, 计算目标值增量$\Delta f=f(S_n)-f(S_i)$
  - 4、若$\Delta f<0$, 令$S_{i+1}=S_n$, 转步骤5， (S_n比S_i好, 无条件接受) ； 否则按概率$\exp(-\Delta f/T_k)$接受$S_{i+1}=S_n$ ，(S_n比S_i差, 有条件接受)。
  - 5、若达到热平衡(由某个给定的收敛准则决定内循环算法是否结束)，转步骤6，否则，$i=i+1$ 转步骤3.
  - 6、$k=k+1$, $S_k=S_i$, 降低T_k, 若$T_k<T_f$, 停止，否则转步骤2.
- 注意：
  - 两层循环。
  - 选择T_0时，要足够高。
  - T_k高时，广域搜索； T_k低时，局域搜索。
  - 避免了局部极小点。

Hebb学习规则

- **Hebb学习规则：两个神经元同时处于激发状态时，它们之间的连接强度将得到加强。**

$$w_{ij}(k+1) = w_{ij}(k) + I_i I_j$$

$w_{ij}(k)$ 为连接从神经元i到神经元j的当前权值；I_i，I_j 为神经元的激活水平。

- **一种无师的学习方法，只根据神经元连接间的激活水平改变权值，为相关学习或并联学习。**

Hebb学习规则

- 神经元描述

$$I_i = \sum w_{ij} x_j - \theta_j$$

$$y_i = f(I_i) = 1/(1+\exp(-I_i))$$

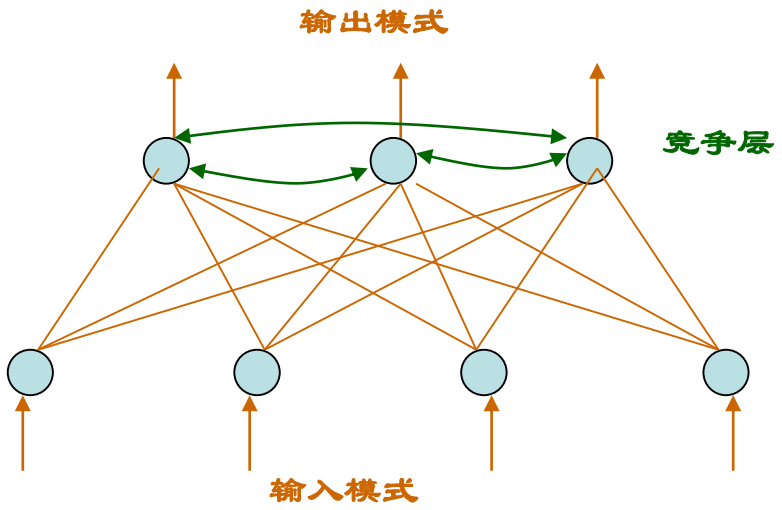
- **Hebb 学习规则：** $w_{ij}(k+1) = w_{ij}(k) + y_i y_j$

- 微分**Hebb 学习方法：根据神经元状态变化来调整权值。**

$$w_{ij}(k+1) = w_{ij}(k) + [y_i(k) - y_i(k-1)][y_j(k) - y_j(k-1)]$$

竞争式学习

- 竞争式学习是在联接机制中引入竞争机制的学习方式，属于无师学习方式。
- 竞争式机制的思想来源于人脑的自组织能力。大脑能够及时地调整自身结构，自动地向环境学习，完成所需执行的功能，而并不需要教师训练。
- 竞争式神经网络又称为自组织神经网络（自适应共振网络模型 **Adaptive Resonance Theory ART**）
- 竞争式神经网络由一组具有层次结构的分层神经元集合而成，其中的任一层与其上面的一层是兴奋性连接，而同一层中的神经元之间是抑制性连接。



竞争式学习

- 竞争式神经网络的同一层中的每个神经元接收来自下一层中的每个神经元的输出。

- 在同一层中的神经元被分成若干簇，在同一簇内神经元都抑制本簇内的其它神经元。因此，在每层同簇内所有神经元经过竞争学习响应下一层的模式，任何一个神经元对所接收的刺激响应越强，它对本簇内其它成员的抑制越强。

- 本质在于神经网络中高层次的神经元对低层次神经元的输入模式进行竞争识别。

竞争式学习

- 自组织神经网络要求识别与输入最匹配的节点。
- 定义距离d_j为接近距离测度，即

$$d_j = \sum_{i=0}^{N-1} (u_i - w_{ij})^2$$

其中，u为N维输入向量，具有最短距离的节点选作胜者，它的权向量经修正使该节点对输入u更敏感。

定义N_c，其半径逐渐减小至接近于零，权值的学习规则为

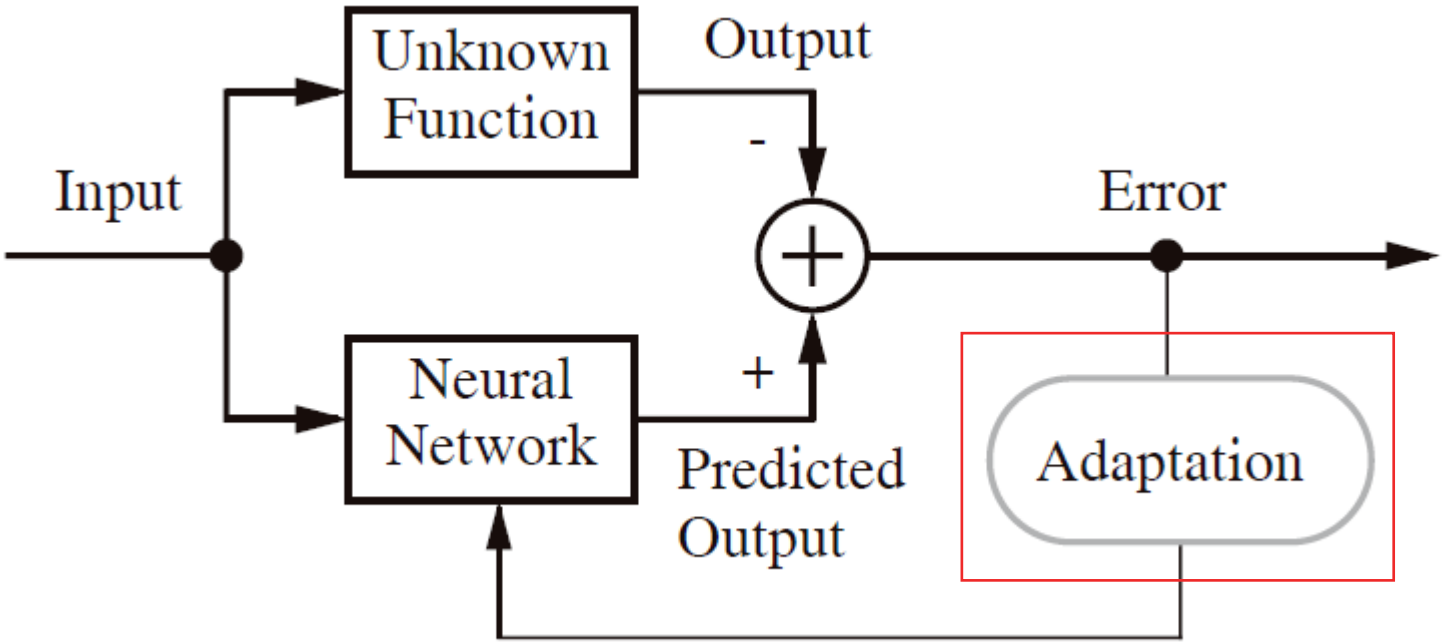
$$\Delta w_{ij} = \begin{cases} \alpha(u_i - w_{ij}) & i \in N_c \\ 0 & i \notin N_c \end{cases}$$

- 在这类学习规则中，关键不在于实节点的输出怎样与外部的期望输出相一致，而在于调整权向量以反映观察事件的分布，提供基于检测特性空间的活动规律的性能描写。

神经网络要素

- 网络结构：一组单元和连接拓扑
  - 前向结构
  - 从输出到输入有反馈的前向网络
  - 层内互连前向网络
  - 互联网络
- 神经元模型：信息处理单元
  - 激发函数
- 学习方法（核心问题）：通过训练来调整权重，使神经网络学会解决问题的知识和经验
  - **Hebb**学习规则
  - δ学习规则
  - 相近学习规则

神经网络函数逼近


Unknown Function
Output
Input
-
Error
+
Neural Network
+
Predicted Output
Adaptation

神经网络在控制系统中所起的作用

- 为控制系统提供某种模型。
- 充当各类控制器。
- 作为计算部件实现某种优化计算。
- 应用方面
  - 模式识别器
  - 信号处理器
  - 系统模型
  - 系统优化器
  - 控制器

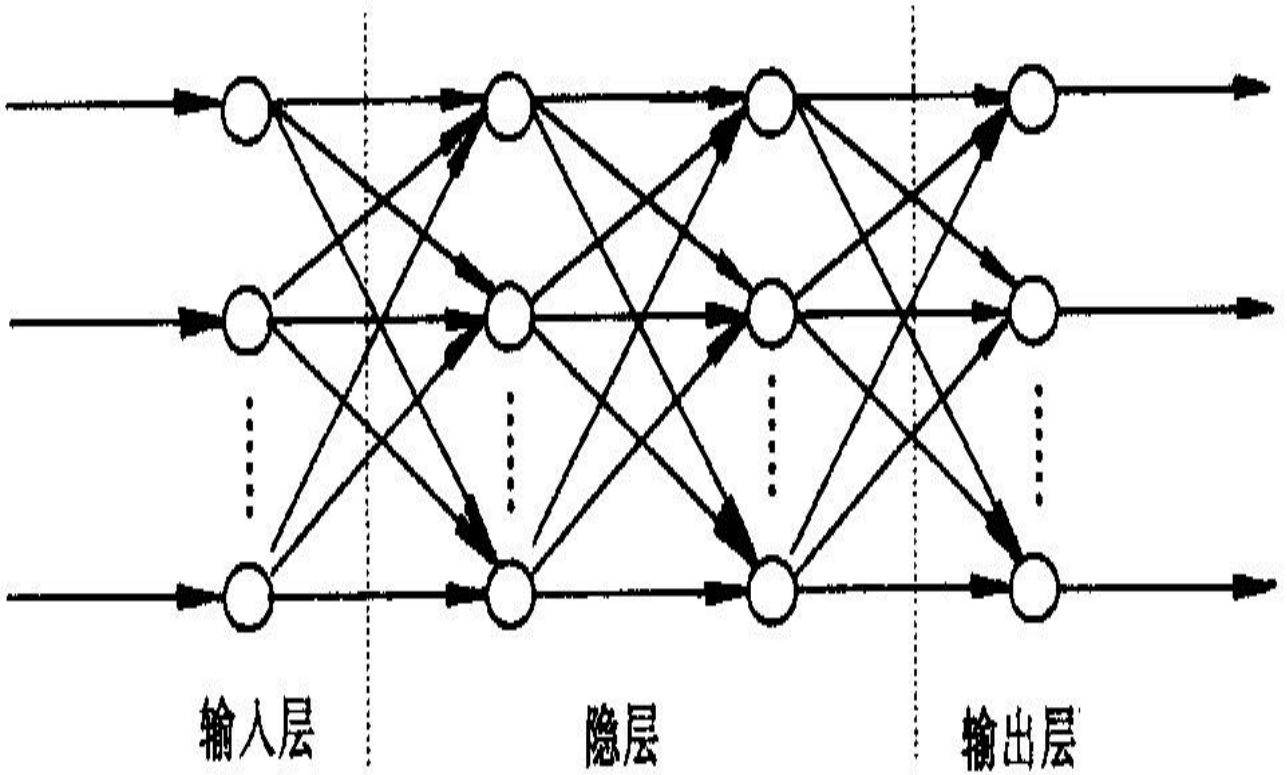
用于控制的人工神经网络的特性

- 并行分布处理。
- 非线性映射。
- 通过训练进行学习。
- 知识存储在连接权及连接关系上。
- 容错能力。
- 硬件实现。

二、神经网络类型

前馈型神经网络

- 神经元分层排列，有输入层、隐层（亦称中间层，可有若干层）和输出层，每一层的神经元只接受前一层神经元的输入。
- 学习的观点：前馈网络是一种强有力的学习系统，其结构简单而易于编程；
- 系统的观点：前馈网络是一静态非线性映射，通过简单非线性处理单元的复合映射，可获得复杂的非线性处理能力。
- 计算的观点：缺乏丰富的动力学行为。
- 大部分前馈网络都是学习网络，它们的分类能力和模式识别能力一般都强于反馈网络，典型的前馈网络有感知器网络、BP 网络等。


输入层
隐层
输出层

感知器

- 一个由线性阈值神经元组成的最简单的前向神经网络。
- 一般包括输入层、中间层和输出层。
- 激活函数：

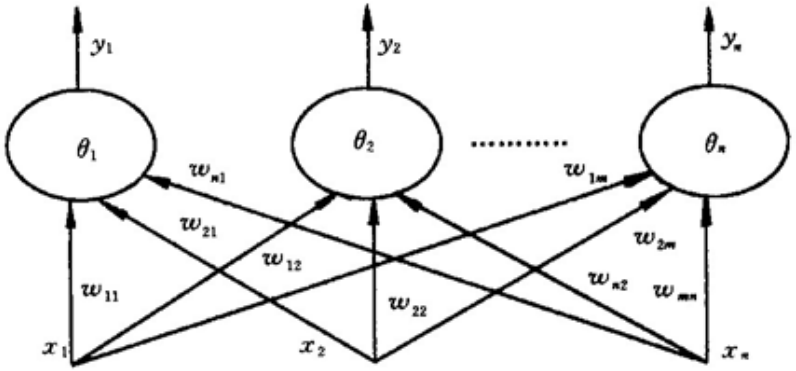
$$f(x)=\begin{cases}1 & x\geq 0\\ -1 & x<0\end{cases}$$

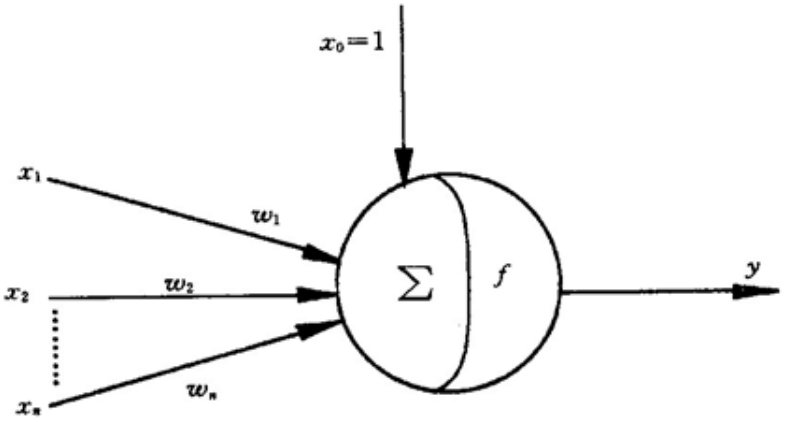
- 主要用于模式分类。

单层感知器

- 单层感知器的一个神经元的输入输出关系：当其输入的加权和大于或等于阈值时，输出为1，否则为－1（或为0）。

$$y = f(\sum_{i=0}^{n} w_i x_i)$$



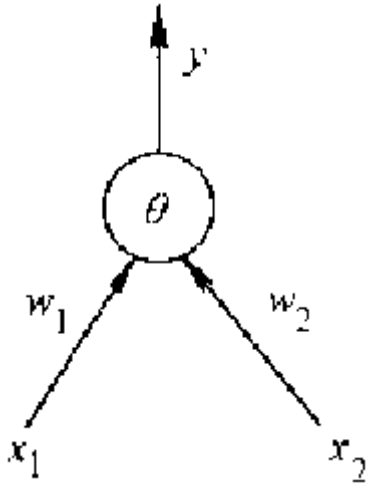


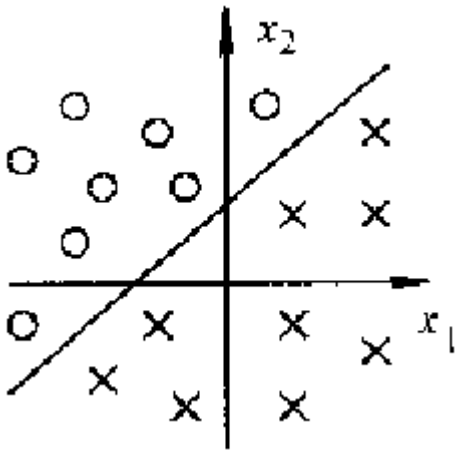
单层感知器为线性可分分类器

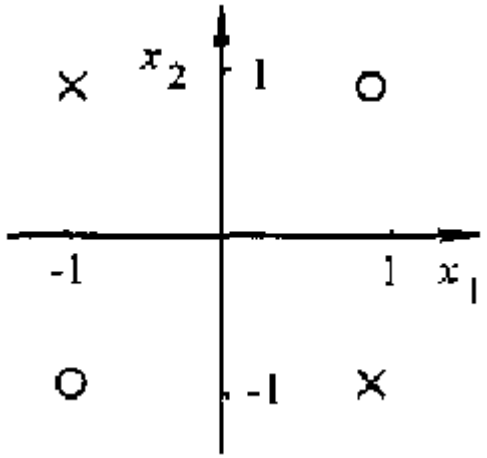
- 以二维空间为例，根据单层感知器的变换关系，分界线的方程为

$$w_1 x_1 + w_2 x_2 - \theta = 0$$

- 直线方程，只能区分线性可分模式类而无法区分如“异或”等非线性可分模式类。







单层感知器的一种学习算法

- 随机地给定一组连接权$w_i(0)$（较小的非零值）。
- 输入一组样本和期望的输出（亦称之为教师信号）。
- 计算感知器实际输出

$$y(k)=f\left(\sum_{i=0}^{n}w_i(k)x_i\right)=\begin{cases}1, & \sum_{i-0}^{n}w_i(k)x_i\geq 0\\ -1, & \sum_{i-0}^{n}w_i(k)x_i<0\end{cases}\qquad (x_0=1,\quad w_0(0)=-\theta)$$

- 修正权值

$$w_i(k+1)=w_i(k)+\eta[d(k)-y(k)]x_i\qquad i=0,1,2,\cdots,n$$

- 选取另外一组样本，重复上述的过程，直到权值对一切样本均稳定不变为止，学习过程结束。
- 算法收敛的充要条件为输入样本是线性可分的。

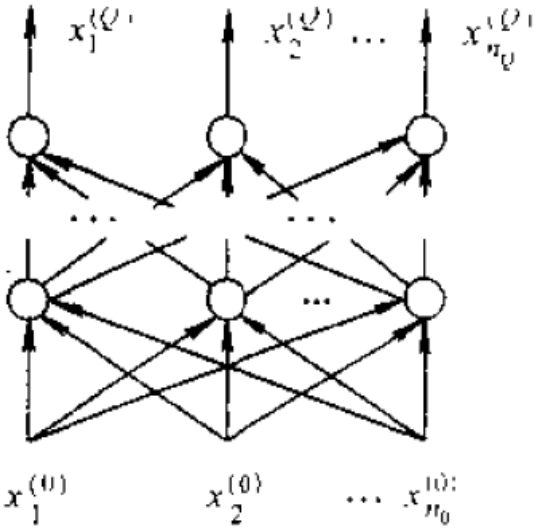
多层感知器

- 每一层为单层感知器，形成多层的组合。
- 多层感知器的输入输出变换关系

$$s_i^{(q)} = \sum_{j=0}^{n_{q-1}} w_{ij}^{(q)} x_j^{(q-1)} \qquad (x_0^{(q-1)} = \theta_i^{(q)}, w_{i0}^{(q)} = -1)$$

$$x_i^{(q)} = f(s_i^{(q)}) = \begin{cases} 1 & s_i^{(q)} \geqslant 0 \\ -1 & s_i^{(q)} < 0 \end{cases}$$

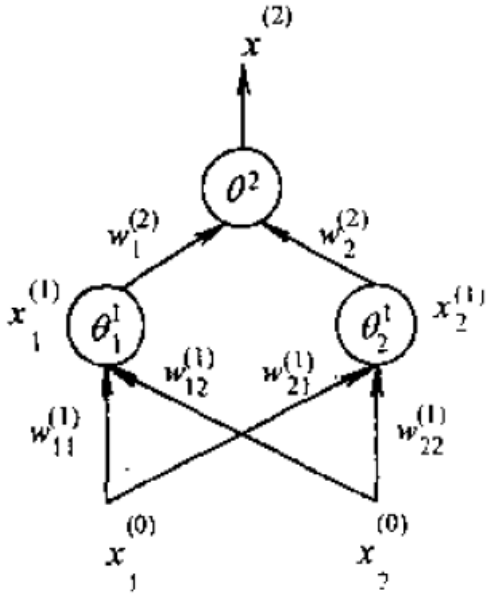
$$i = 1,2,\cdots,n_q \qquad j = 1,2,\cdots,n_{q-1} \qquad q = 1,2,\cdots,Q$$

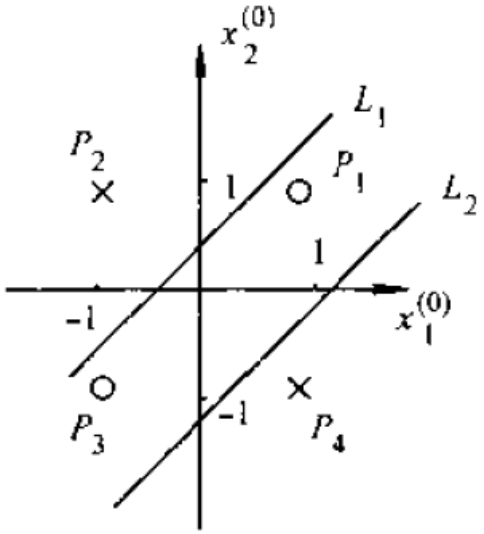


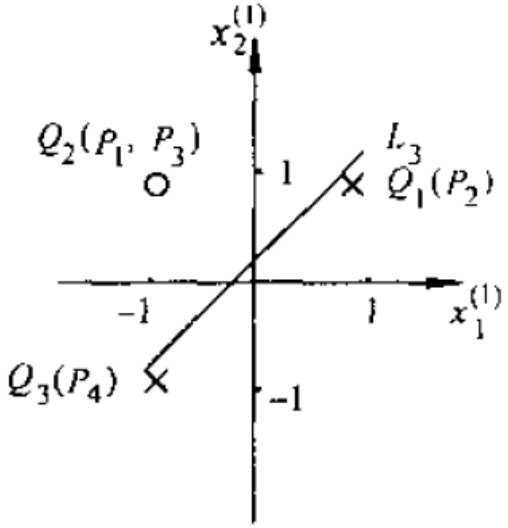
- 学习算法采用δ学习规则。

多层感知器

- 多层感知器可以处理“异或”。
- 定理：假定隐层的节点可以根据需要自由设置，那么用含有二层隐层的阈值网络可以实现任意的二值逻辑函数。

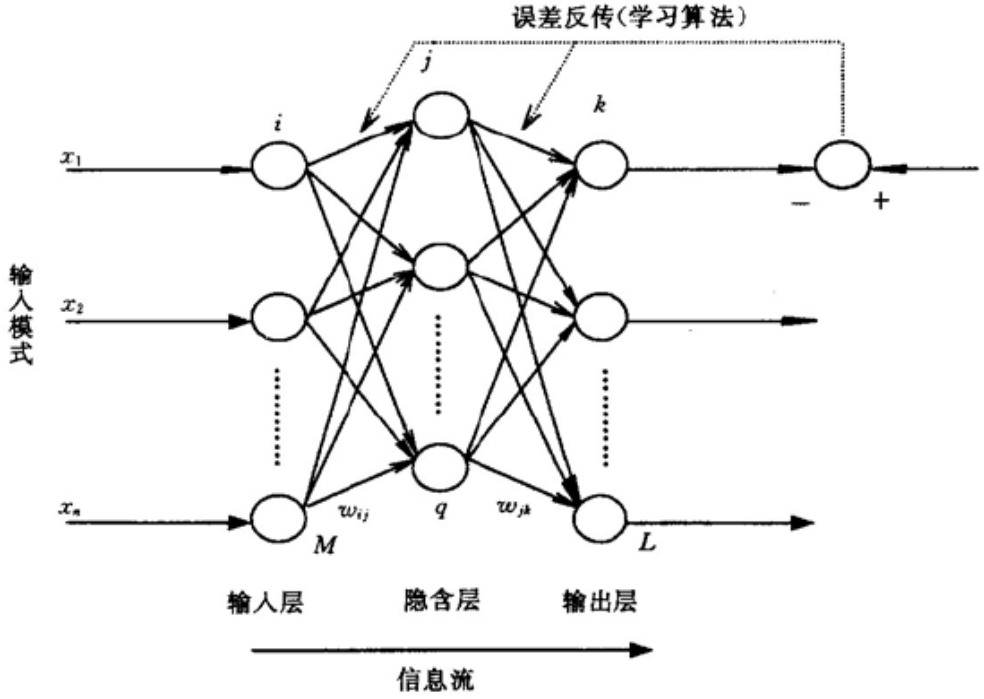






BP 网络

- 误差反向传播神经网络（**Back Propagation**，简称**BP**网络），是一种单向传播的多层前向网络。
- 在模式识别、图像处理、系统辨识、函数拟合、优化计算、最优预测和自适应控制等领域有着较为广泛的应用。



反向传播算法（BP 算法）

- 活化函数选择Sigmoid函数，连续可微。即

$$f(x)=\frac{1}{1+e^{-x}} \Rightarrow f'(x)=f(x)[1-f(x)]$$

- （1）BP 网络的前馈计算
  - 在训练网络的学习阶段，设有P个训练样本对，每个样本由输入输出模式X_p和$\{d_{pk}\}$组成，将样本p (p = 1, 2, · · ·, P)的输入通过网络前向计算，得到网络输出值，并与期望输出值比较，得到误差信号。
- （2）系统的误差代价函数为

$$E=\frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{L}(d_{pk}-O_{pk})^2=\sum_{p=1}^{P}E_p$$

  - 如何调整连接权系数以使代价函数最小。采用梯度下降法，$\frac{\partial E}{\partial w_{ij}}=\sum_{p=1}^{P}\frac{\partial E_p}{\partial w_{ij}}$
  - 关键是计算 $\frac{\partial E_p}{\partial w_{ij}}$

反向传播算法（BP 算法）

- （3）反向传播计算
  - 为简便起见，略去下标**p**，有 $E = \frac{1}{2}\sum_{k=1}^{L}(d_k - O_k)^2$
  - （i）输出层权系数的调整

权系数的修正公式为 $\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$

其中，η为学习速率；

i,j,k表分别代表输入层，中间层和输出层。

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial w_{jk}}$$

net_k表示神经元节点的输入

定义反传误差信号为 $\delta_k = -\frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial O_k}\frac{\partial O_k}{\partial net_k}$

式中 $\frac{\partial E}{\partial O_k} = -(d_k - O_k)$ $\frac{\partial O_k}{\partial net_k} = \frac{\partial}{\partial net_k} f(net_k) = f'(net_k)$

$\delta_k = (d_k - O_k) f'(net_k) = O_k(1 - O_k)(d_k - O_k)$ $\frac{\partial net_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}(\sum_{j=1}^{q} w_{jk} O_j) = O_j$

反向传播算法（BP 算法）

– 由此得输出层的任意神经元权系数的修正公式为

$$\Delta w_{jk} = \eta(d_k - O_k) f'(net_k) O_j = \eta \delta_k O_j$$

– 或

$$\Delta w_{jk} = \eta O_k (1 - O_k)(d_k - O_k) O_j$$

– （ii）隐含层节点权系数的调整

计算权系数的变化量为

注意：i,j,k表分别代表输入层，中间层和输出层。

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial net_j} O_i$$

$$= \eta(-\frac{\partial E}{\partial O_j} \frac{\partial O_j}{\partial net_j}) O_i = \eta(-\frac{\partial E}{\partial O_j}) f'(net_j) O_i = \eta \delta_j O_i$$

式中，$\frac{\partial E}{\partial O_j}$不能直接计算，需通过其他间接量进行计算，即

$$-\frac{\partial E}{\partial O_j} = \sum_{k=1}^{L} (-\frac{\partial E}{\partial net_k}) \frac{\partial}{\partial O_j} (\sum_{j=1}^{q} w_{jk} O_j) = \sum_{k=1}^{L} (-\frac{\partial E}{\partial net_k}) w_{jk} = \sum_{k=1}^{L} \delta_k w_{jk}$$

反向传播算法（BP 算法）

- 因此有 $$\delta_j = f'(net_j)\sum_{k=1}^{L}\delta_k w_{jk}$$

注：此即为该算法名称的由来

- 此即为反向计算过程，即由δ_k计算δ_j，由δ_j计算δ_i
- （iii）将样本标记***p***记入公式后，有

对于输出节点***k*** ：

$$\Delta_p w_{jk} = \eta f'\left(net_{pk}\right)\left(d_{pk}-O_{pk}\right)O_{pj} = \eta O_{pk}\left(1-O_{pk}\right)\left(d_{pk}-O_{pk}\right)O_{pj}$$

对于隐含节点***j*** ：

$$\Delta_p w_{ij} = \eta f'(net_{pj})(\sum_{k=1}^{L}\delta_{pk}w_{jk})O_{pi} = \eta O_{pj}(1-O_{pj})(\sum_{k=1}^{L}\delta_{pk}w_{jk})O_{pi}$$

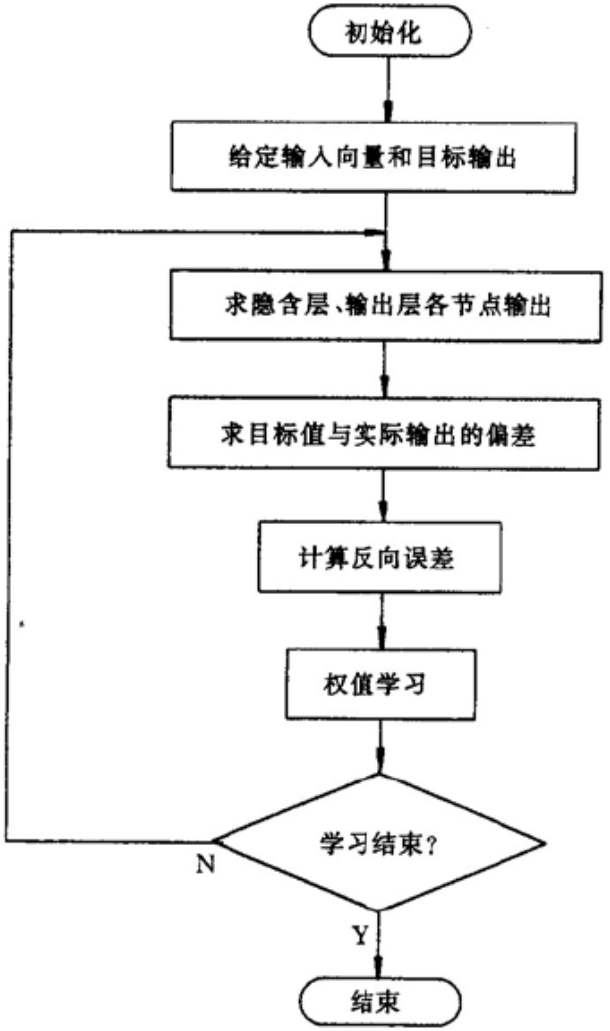
- （iv）最终网络连接权值调整式：

$$w_{ij}\left(t+1\right) = w_{ij}\left(t\right)+\eta\delta_j O_i+\alpha\left[w_{ij}\left(t\right)-w_{ij}\left(t-1\right)\right]$$

式中， ***t +1***表示第***t +1***步，α为平滑因子

反向传播算法（BP 算法）

- 误差反向传播的**BP** 算法基本思想是最小二乘算法。它采用梯度搜索技术，以期使网络的实际输出值与期望输出值的误差均方值为最小。
- **BP** 算法的学习过程由正向传播和反向传播组成。
- 在正向传播过程中，输入信息从输入层经隐含层逐层处理，并传向输出层，每层神经元（节点）的状态只影响下一层神经元的状态。
- 如果在输出层不能得到期望的输出，则转入反向传播，将误差信号沿原来的连接通路返回，通过修改各层神经元的权值，使误差信号最小。



反向传播算法（BP 算法）

- 在使用BP算法时，应注意的问题
  - 学习开始时，各隐含层连接权系数的初值应以设置较小的随机数较为适宜。
  - 采用S型激发函数时，由于输出层各神经元的输出只能趋于1或0，不能达到1或0. 在设置各训练样本时，期望的输出分量不能设置为1或0，以设置为接近于1或0 （如0.9或0.1）较为适宜。
  - 学习速率η的选择，在学习开始阶段， η 选较大的值可以加快学习速度。学习接近优化区时， η 值必须相当小，否则权系数将产生振荡而不收敛。

多层前向BP网络的优点

- 网络实现了一个从输入到输出的映射功能，已证明它具有实现任何复杂非线性映射的功能。这使得它特别适合于求解内部机制复杂的问题；

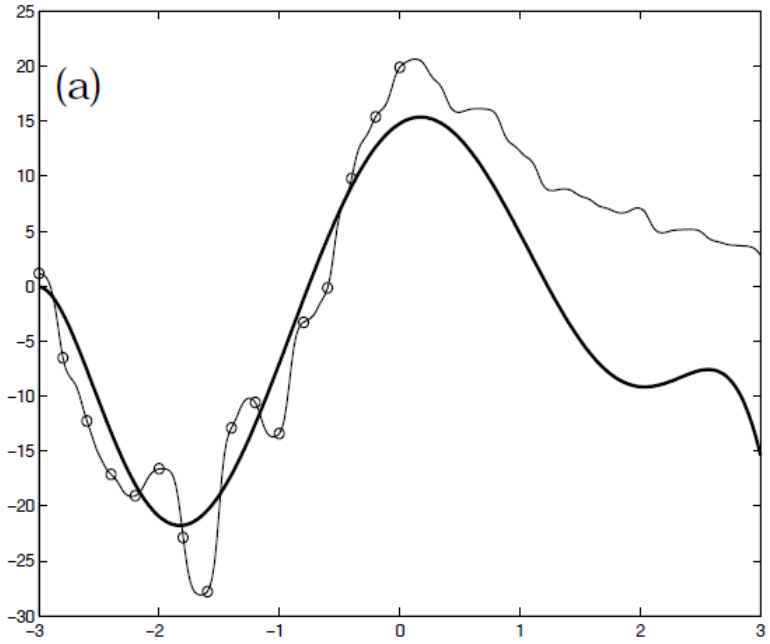
- 网络能通过学习带正确答案的实例集自动提取“合理的”求解规则，即具有自学习能力；

多层前向BP网络的缺点

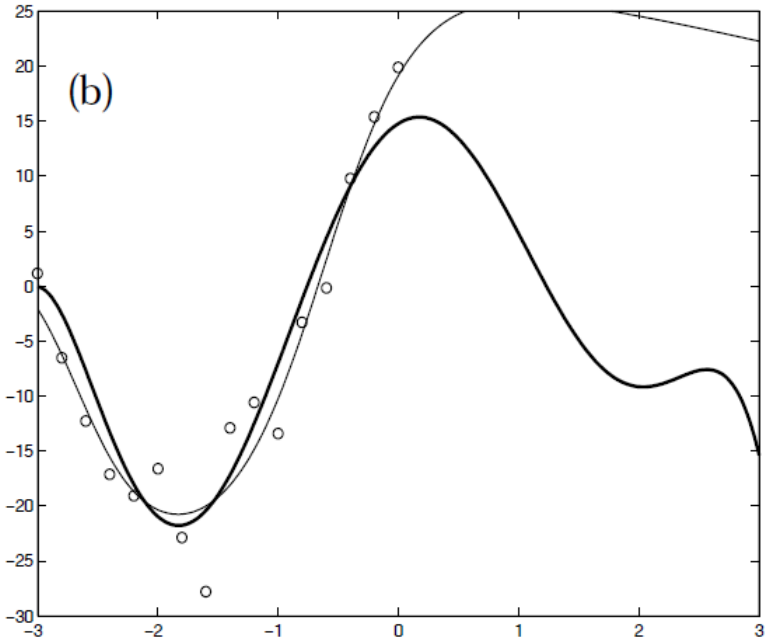
- 存在局部极值
- 学习速度很慢
- 网络训练失败的可能性较大
- 难以解决应用问题的实例规模和网络规模间的矛盾
- 网络结构的选择尚无一种统一而完整的理论指导，一般只能由经验选定
- 新加入的样本要影响已学习成功的网络，而且刻画每个输入样本的特征的数目也必须相同
- 网络的预测能力（也称泛化能力、推广能力）与训练能力（也称逼近能力、学习能力）的矛盾

过拟合和泛化能力

- 前向网络的泛化能力不足



过拟合



正常拟合

BP网络学习算法的改进

- 增加“惯性项”
  - 加快收敛速度。
- 采用动态步长
  - 加快收敛速度。
- 与其他全局搜索算法相结合
  - 为克服**BP** 算法全局搜索能力弱的缺点，将**BP** 算法与具有很强全局搜索能力的算法相结合，如与遗传算法相结合。
- 模拟退火算法
  - 为克服**BP** 算法易陷入局部极小的缺点。

反馈神经网络

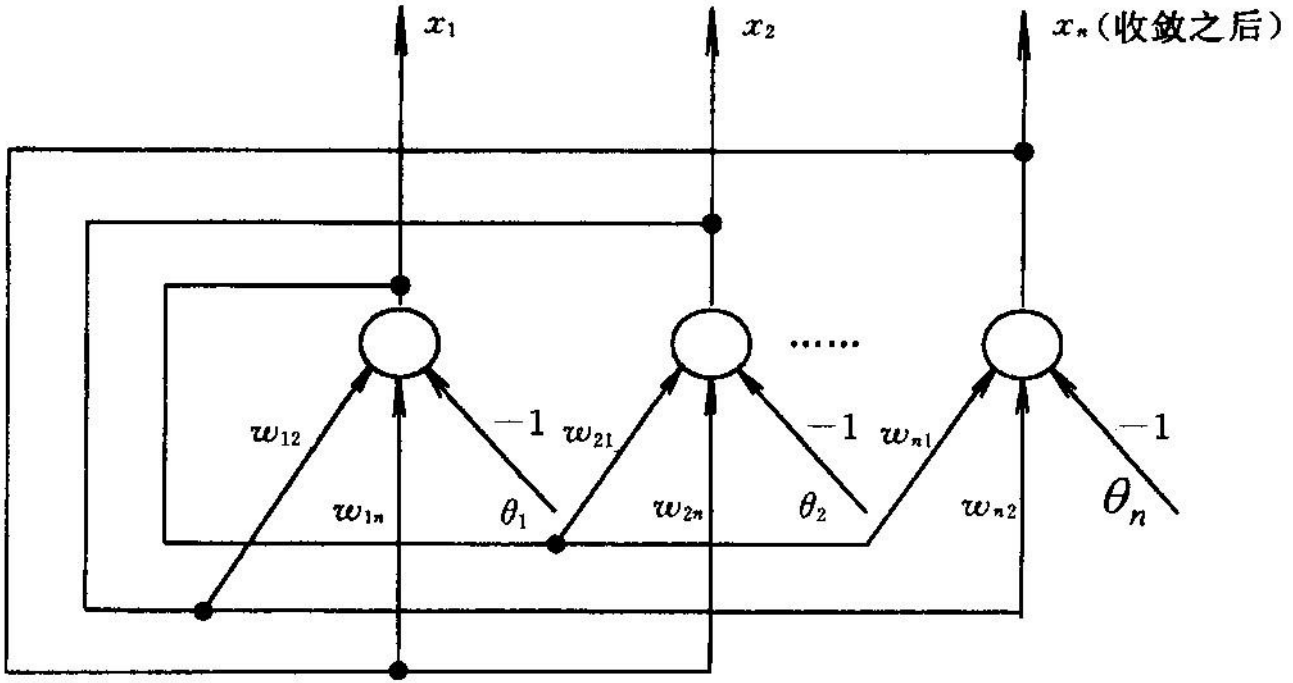
- 反馈网络（**Recurrent Network**），又称自联想记忆网络，其目的是为了设计一个网络，储存一组平衡点，使得当给网络一组初始值时，网络通过自行运行而最终收敛到这个设计的平衡点上。
- 网络的输出反馈到网络输入，输入信号决定反馈系统的初始状态，然后系统经过一系列状态转移后，逐渐收敛于平衡状态。
- 反馈网络能够表现出非线性动力学系统的动态特性。所具有的主要特性为：
  - 网络系统具有若干个稳定状态。当网络从某一初始状态开始运动，网络系统总可以收敛到某一个稳定的平衡状态；
  - 系统稳定的平衡状态可以通过设计网络的权值而被存储到网络中。

Hopfield网络

- **Hopfield网络是单层对称全反馈网络，根据其激活函数的选取不同，可分为**
  - **离散型的Hopfield网络(Discrete Hopfield Neural Network)**
    - **激活函数为二值型的，其输入、输出为{0，1}的反馈网络。**
    - **主要用于联想记忆。**
  - **连续型的Hopfield网络(Continuous Hopfield Neural Network)**
    - **激活函数的输入与输出之间的关系为连续可微的单调上升函数。**
    - **主要用于优化计算。**

- **Hopfield网络是利用稳定吸引子来对信息进行储存的，利用从初始状态到稳定吸引子的运行过程来实现对信息的联想存取的。**

Hopfield网络



离散Hopfield网络

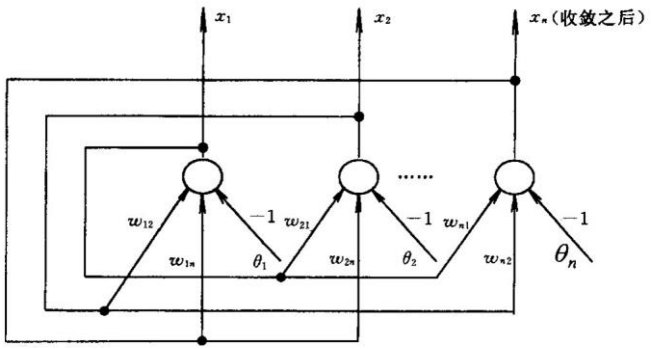
- 网络的结构
  - 单层网络。
  - 有***n***个神经元节点，每个神经元的输出均接到其它神经元的输入。
  - 各节点没有自反馈。即神经元的输出不作为自神经元的输入。
  - 活化函数

$$f(s)=\operatorname{sgn}(s)=\begin{cases}1 & s\geq 0\\ -1 & s<0\end{cases}$$

  - 系统方程

$$\begin{cases}s_i=\sum_{j=1}^{n}w_{ij}x_j-\theta_i\\ x_i=f(s_i)\end{cases}$$

其中$w_{ij}=w_{ji}$且$w_{ii}=0$



离散Hopfield网络

- 网络工作方式
  - 异步方式
    - 每次只有一个神经元节点进行状态的调整计算，其他节点的状态均保持不变。
    - 调整次序可以随机选定，也可按规定的次序进行。

$$x_i(k+1) = f\left(\sum_{j=1}^{n} w_{ij} x_j(k) - \theta_i\right)$$

$$x_j(k+1) = x_j(k), \qquad j \neq i$$

  - 同步方式
    - 所有神经元节点同时调整状态

$$x_i(k+1) = f\left(\sum_{j=1}^{n} w_{ij} x_j(k) - \theta_i\right) \qquad \forall i$$

离散Hopfield网络

- 稳定性和吸引子
  - 若网络的状态X 满足$X=f(WX-\theta)$，则称X为网络的稳定点或吸引子。

- 定理1：对于离散Hopfield 网络，若按异步方式调整状态，且连接权矩阵W为对称阵，则对于任意初态，网络都最终收敛到一个吸引子。

- 定理2：对于离散Hopfield 网络，若按同步方式调整状态，且连接权矩阵W为非负定对称阵，则对于任意初态，网络都最终收敛到一个吸引子。

离散Hopfield网络

- 吸引域
  - 为实现正确的联想记忆，对于每个吸引子应该有一定的吸引范围，这个吸引范围称为吸引域。
  - 对于异步方式，若吸引子与运行时的调整顺序有关，称为弱吸引；若对于任意调整次序都可以到达吸引子，称为强吸引。
  - 对所有$x \in R\left(x^{(a)}\right)$均有 x 弱（强）吸引到 $x^{(a)}$ ，则称$R\left(x^{(a)}\right)$为 $x^{(a)}$ 的弱（强）吸引域。

- 对于同步方式，吸引域无强弱之分。
- 对于异步方式，不同的调整顺序可能运行到不同的吸引子。

离散Hopfield网络

- 连接权设计（学习方法）
  - 为保证**Hopfield** 网络在异步方式工作时能稳定收敛，连接权矩阵***W***应是对称的。
  - 为保证同步方式收敛，要求***W***为非负定阵，这个要求比较高。
  - 一般设计***W***只保证异步方式收敛。
  - 对于给定的样本必须是网络的吸引子，而且要有一定的吸引域，这样才能正确实现联想记忆功能。
  - 为了实现上述功能，通常采用**Hebb** 规则来设计连接权，即当神经元输入与输出节点的状态相同（即同时兴奋或抑制）时，从第***j***个到第***i***个神经元之间的连接强度则增强，否则，则减弱。

离散Hopfield网络

- 联想记忆
  - 联想记忆功能是离散**Hopfield** 网络的一个重要应用范围。
  - 要想实现联想记忆，反馈网络必须具有两个基本条件：
    - 网络能收敛到稳定的平衡状态，并以其作为样本的记忆信息；
    - 具有回忆能力，能够从某一残缺的信息回忆起所属的完整的记忆信息。
  - 离散**Hopfield** 网络实现联想记忆的过程分为两个阶段：
    - 学习记忆阶段：设计者通过某一设计方法确定一组合适的权值，使网络记忆期望的稳定平衡点。
    - 联想回忆阶段：网络的工作过程。
  - 离散**Hopfield** 网络用于联想记忆有两个突出的特点
    - 记忆是分布式的
    - 联想是动态的

离散Hopfield网络

- 离散Hopfield 网络局限性
  - 记忆容量的有限性；
  - 伪稳定点的联想与记忆；
  - 当记忆样本较接近时，网络不能始终回忆出正确的记忆。
  - 网络的平衡稳定点并不可以任意设置的，也没有一个通用的方式来事先知道平衡稳定点。

连续Hopfield网络

- 单层的反馈网络。
- 实质上是一个连续的非线性动力学系统，可以用一组非线性微分方程来描述。

$$\begin{cases} s_i = \sum_{j=1}^{n} w_{ij} x_j - \theta_j \\ \dfrac{dy_i}{dt} = -\dfrac{1}{\tau} y_i + s_i \\ x_i = f(y_i) \end{cases}$$

引入一阶惯性环节

- 当给定初始状态，通过求解非线性微分方程组即可求得网络状态的运行轨迹。
- 若系统是稳定的，则它最终可收敛到一个稳定状态。

连续Hopfield网络

- 活化函数一般取Sigmoid函数。
- 网络可用电路实现。
- 网络的稳定性需要满足一定条件。
- 连续Hopfield网络主要用来进行优化计算。
- 网络的学习（权系数设计）
  - 网络的连接权系数需要根据具体问题加以设计。
  - 网络的目标函数通常根据优化的具体问题的要求写成系统的总的能量函数。
- 连续Hopfield网络成功地用于求解旅行商问题（TSP）。

Boltzmann 机

- 随机反馈型神经网络。
- 各神经元的状态可看作是一个随机变量，服从Boltzmann分布。
- 可用于模式分类、预测、组合优化及规划等方面。
- **Hopfield模型 + 隐层单元 + 模拟退火 = Boltzmann机**

Boltzmann 机

- 网络结构
  - **Boltzmann机网络是一个相互连接的单层反馈网络，形式上与离散Hopfield网络一样，具有对称的连接权系数。$w_{ij}=w_{ji}$且$w_{ii}=0$**
  - 网络构成
    - 可见单元（**Visible Unit**）：输入、输出
    - 隐单元（**Hidden Unit**）
  - 每个单元节点输出只取**-1**和**1**（或**0**和**1**）两种状态。

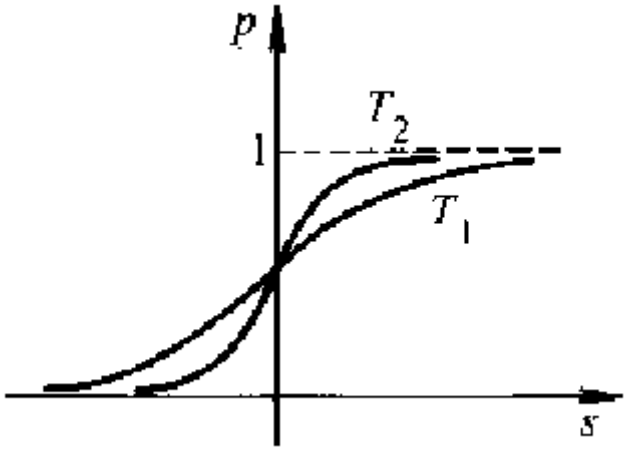
Boltzmann 机

- 实际运行
  - 与**Hopfield**网络相似，**Boltzmann**机的实际运行分为两个阶段：
    - 学习和训练阶段，即根据学习样本对网络进行训练，将知识分布地存储于网络的连接权中；
    - 工作阶段，即根据输入运行网络得到合适的输出，这一步实质上是按照某种机制将知识提取出来。

Boltzmann 机

- 神经元工作方式

$$\begin{cases} s_i = \sum_{i=1}^{n} w_{ij} x_j - \theta_i \\ p_i = \dfrac{1}{1+e^{-s_i/T}} \end{cases}$$



  - 其中，p_i是x_i=1的概率（即i神经元输出为1的概率）。
  - p_i是S形函数。T越大，曲线越平坦；T越小，曲线越陡峭。参数T通常称为“温度”。
  - 当$T\to 0$时，S形函数便趋于二值函数，随机神经网络便退化为确定性网络，与Hopfield网络相同。

Boltzmann 机

- 能量函数

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} x_i \theta_i$$

- 运行方式
  - 同步方式
  - 异步方式
- 分析可以得到， 在两种运行方式下，均有如下特点：随着系统状态的演变，从概率的意义上，系统的能量总是朝小的方向变化，所以系统最后总能稳定到能量的极小点附近。
- 由于这是随机网络，在能量极小点附近，系统也不会停止在某一个固定的状态。

Boltzmann 机

- 第*i*个神经元状态发生变化时，能量函数按下式变化

$$\Delta E_i = -\Delta x_i \left(\sum_{j-1}^{n} w_{ij} x_j - \theta_i \right)$$

- 分析可得到，对于网络中任意两个状态α和β出现的概率与它们的能量E_α和E_β之间也满足

$$\frac{p_\alpha}{p_\beta} = e^{-\left(E_\alpha - E_\beta\right)/T} = \frac{e^{-E_\alpha/T}}{e^{-E_\beta/T}}$$

- 上式为**Boltzmann**分布，故该网络称为**Boltzmann**机。
- 特点：
  - （在同一 *T* 下，）**Boltzmann**机处于某一状态的概率主要取决于在此状态下的能量，能量越低，概率越大。
  - 此概率还取决于温度参数*T*，*T*越大，不同状态出现概率的差异便越小，比较容易跳出能量的局部极小点而到达全局的极小点。

Boltzmann 机

- 工作阶段的算法是指在网络连接权已确定的情况下，网络运行的算法。为有效地演化到网络能量函数的全局极小点，通常采用模拟退火的方法。
  - 即开始采用较高的温度T，此时各状态出现概率的差异不大，比较容易跳出局部极小点进入到全局极小点附近，
  - 然后再逐渐减小温度T，各状态出现概率的差别逐渐拉大，从而一方面可较准确地运动到能量的极小点，同时阻止它跳出该最小点。
- 工作阶段算法：采用模拟退火方法寻找全局最优。
  - 初始化，令$T=T_0$，$\{w_{ij}\}$，$\{\theta\}$根据设计计算给定，$w_{ii}=0$，（$i,j=1,2,\ldots,n$）
  - （1）从n个神经元中随机选择一个神经元。（异步方式）
  - （2）计算神经元的输入，即该神经元的$s_i(k+1)$。

Boltzmann 机

- （3）按p_i将神经元i的状态更新为1，更新概率为

$$p_i\left[x_i\left(k+1\right)=1\right]=\frac{1}{1+e^{-s_i\left(k+1\right)/T}}$$

- （4）i以外神经元的输出状态保持不变，即

$$x_j\left(k+1\right)=x_j\left(k\right),\quad \left(j=1,2,\cdots,n;\quad j\neq i\right)$$

- （5）从n个神经元中另选一个神经元，重复上述步骤（1）-（4），直到在该温度下网络达到“热平衡”状态。
- （6）以某种方式取$T_1<T_0$，令$T=T_1$
- （7）运行过程是否结束，否则返回第（1）步。

- 降温策略

$$T\left(k\right)=\frac{T_0}{\log\left(k+1\right)}$$

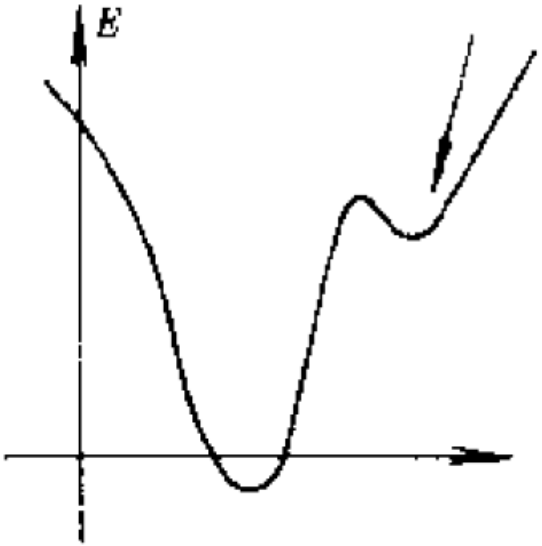
保证全局极小，但收敛慢。

$$T\left(k\right)=\frac{T_0}{1+k}$$

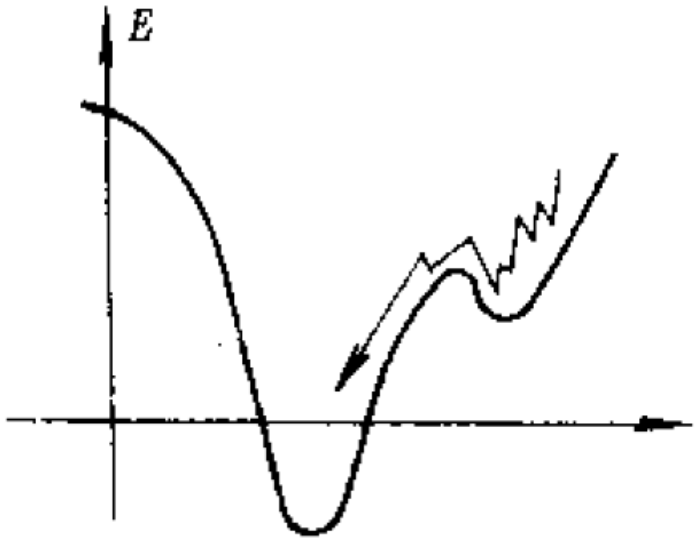
快速降温策略。

Boltzmann 机

- 工作阶段，采用模拟退火算法，可以避免局部极小，而达到全局能量最小的状态，通用性强。



快速下降法



模拟退火法

Boltzmann 机

- 网络的学习和训练
  - 网络学习的目的是通过给出一组学习样本（即已知输入输出的例子），经学习后得到**Boltzmann**机各种神经元之间的连接权w_{ij}，使网络在自由状态下的输入输出与样本的输入输出一致。
  - 由于**Boltzmann**机是概率型网络，与确定性网络不同，其学习样本通常是一组在输入模式下得到的期望输出状态的概率分布。
  - **Boltzmann**机不但能通过工作运行实现优化，而且通过学习，可以模拟外界给定的概率分布，实现联想记忆。

- **Boltzmann**机网络学习的策略：
  - 将网络节点分为输入、输出和隐含部分。
  - 第一阶段首先限制网络的输入和输出部分与样本一致，隐含部分自由运动，系统平衡后，得到各状态出现的概率；
  - 第二阶段，仅限制网络的输入与样本一致，隐含和输出部分自由运动，系统平衡后，得到各状态出现的概率；
  - 通过学习调整网络权值，使上述两组概率一致。

Boltzmann 机

- **Boltzmann机网络学习的步骤如下：**
  - **(1)随机设定网络的连接权$w_{ij}(0)$及初始高温。**
  - **(2)按照已知的概率$p(x_\alpha)$，依次给定学习样本。**
  - **(3)在学习样本（即输入节点约束和输出节点约束）的约束下，按照模拟退火程序运行网络，直至达到平衡状态，统计出各p_{ij}（i和j神经元输出值均为1出现的频率）；**
  - **(4)在无约束条件（即仅有输入节点约束，输出节点自由）下，按模拟退火程序同样的步骤并同样的次数运行网络，统计出各p'_{ij}（i和j神经元输出值均为1出现的频率）；**
  - **(5)按下述公式修改权值**

$$w_{ij}(k+1)=w_{ij}(k)+\eta(p_{ij}-p'_{ij}),\ \eta>0$$

  - **(6)重复上述步骤(3)-(5)，直到$p_{ij}-p'_{ij}$小于一定的容限。**

学习结束，网络自由运行，能够与学习样本一致

Boltzmann 机

- 调整权值的原则是：当 $p_{ij}>p'_{ij}$时，则权值增加，否则减少权值。这种权值调整公式称为**Boltzmann** 机学习规则。

- 由于模拟退火过程要求高温使系统达到平衡状态，而冷却（即退火）过程又必须缓慢地进行，否则容易造成局部最小，所以这种学习规则的学习收敛速度较慢。

神经网络的训练

- 获取训练样本集
  - 获取训练样本集合是训练神经网络的第一步，也是十分重要和关键的一步。它包括训练数据的收集、分析、选择和预处理等。
- 选择网络类型与结构
  - 神经网络的类型很多，需要根据任务的性质和要求来选择合适的网络类型。
- 训练与测试
  - 最后一步是利用获取的训练样本对网络进行反复训练，直至得到合适的映射结果。