神经网络控制

刘山
浙江大学控制科学与工程学院

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。

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。

神经网络在控制中的作用

- 在传统的控制系统中用以动态系统建模，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。

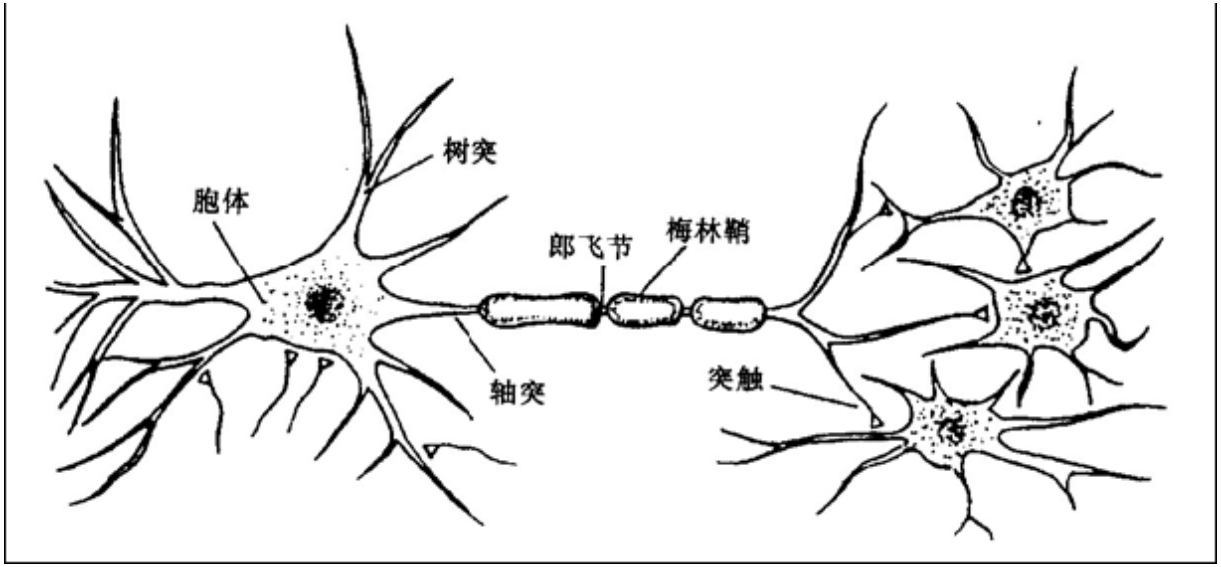
内容

- 1、人工神经网络概述
- 2、基于神经网络的系统建模与辨识
- 3、神经网络PID控制
- 4、神经网络控制结构方案
- 5、神经控制的应用

1、人工神经网络概述

生物神经元模型

- 输入与输出
- 兴奋与抑制



人工神经网络

- 人工神经网络是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型
- 神经网络是由许多神经元模型组成的信息处理网络，具有并行分布结构
- 每个构造起网络的神经元模型模拟一个生物神经元，表现出通用性
- 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
- 神经网络的结构是由基本处理单元及其互连方法决定的

神经网络要素

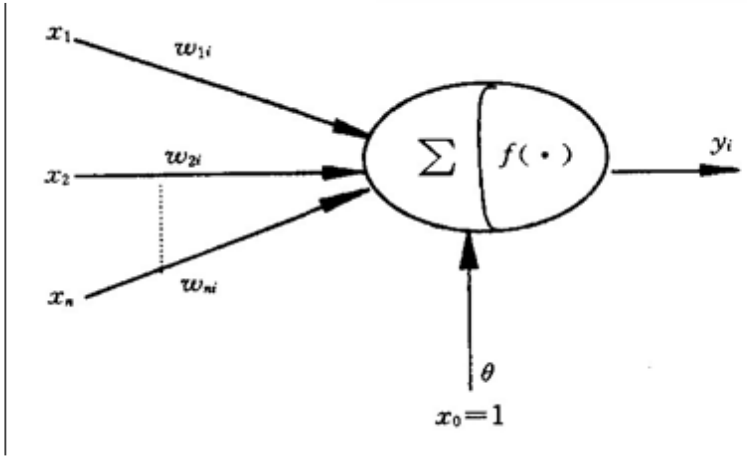
- 网络结构：一组单元和连接拓扑
  - 前向结构
  - 从输出到输入有反馈的前向网络
  - 层内互连前向网络
  - 互联网络
- 神经元模型：信息处理单元
  - 激发函数
- 学习方法（核心问题）：通过训练来调整权重，使神经网络学会解决问题的知识和经验
  - **Hebb**学习规则
  - δ学习规则
  - 相近学习规则

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

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

$$y_i = f(I_i)$$

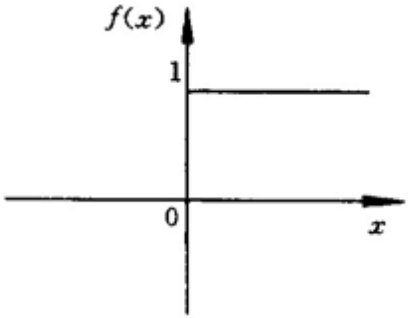


w_{ij}表示从神经元j到神经元i的连接权值；
θ_i为阈值；
$f(\bullet)$ 称为激发函数或作用函数。

常用激发函数

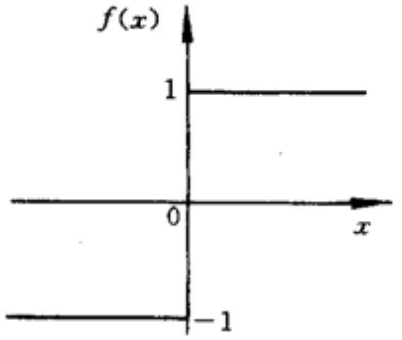
- 阈值型函数
  - 阶跃函数

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



  - 符号函数

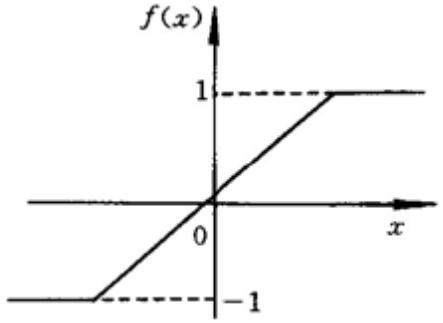
$$\mathrm{sgn}(x)=f(x)=\begin{cases}1, & x\geq 0\\ -1, & x<0\end{cases}$$



常用激发函数

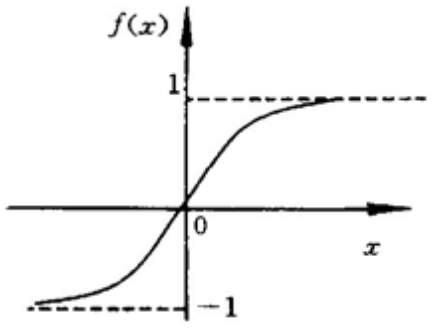
- 饱和型函数

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



- 双曲函数

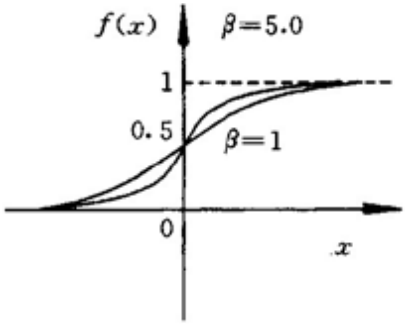
$$f(x)=\tanh(x)$$



常用激发函数

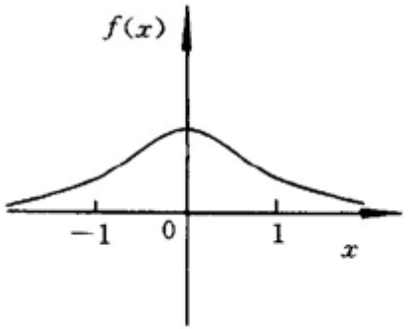
- **S 型函数（Sigmoid）**

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



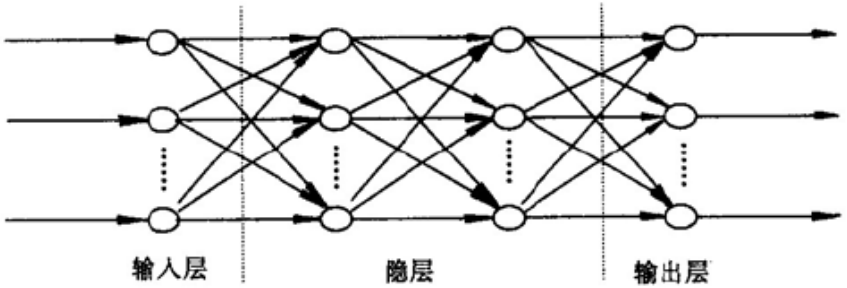
- **高斯函数（Radial Basis Function，RBF）**

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

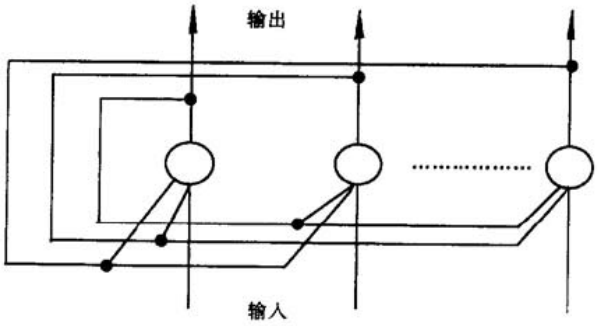


神经网络结构模型

- 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**）

神经网络的学习方法

- 有师学习
  - 网络的输出和期望的输出（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
  - 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。
- 强化学习算法
  - 强化学习是有师学习的特例。
  - 它不需要老师给出目标输出。
  - 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
  - 无教师学习
  - 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

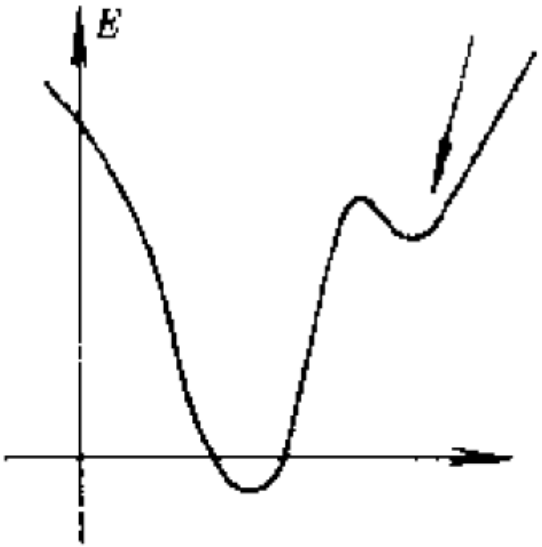
- **METROPOLIS准则**
  - 粒子在温度T时趋于平衡的概率为$\exp(-\Delta E/(kT))$，其中E为温度T时的内能，ΔE为其改变量，k为Boltzmann常数。
  - 用固体退火模拟优化问题，将内能E模拟为目标函数值f，温度T演化成控制参数t，得到解优化问题的模拟退火算法。
  - 由初始解i和控制参数初值t开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减t值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。
  - 退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的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

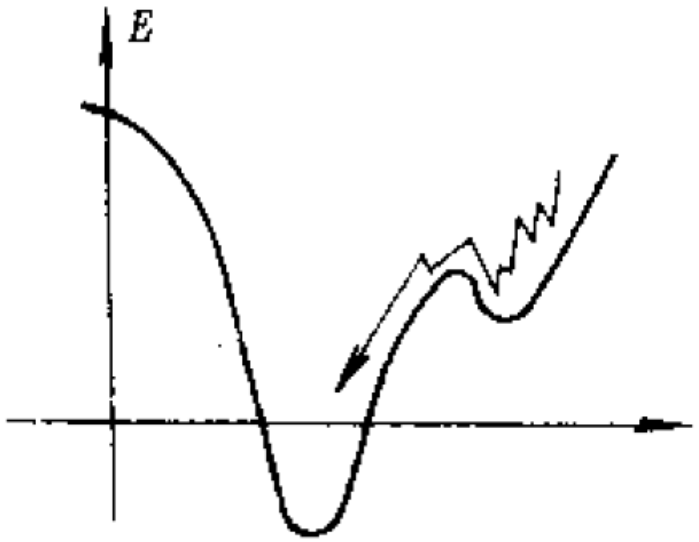
- 模拟退火算法的步骤：
  - 1、初始化。任选初始解S_0, 给定初始温度T_0, 终止温度T_f. 令迭代指标$k=0$, $S_k=S_0$, $T_k=T_0$.
  - 2、令$i=0$，$S_i=S_k$;（同一温度下，寻找平衡态）
  - 3、随机产生S_i的一个邻域解S_n, 计算目标值增量$\Delta f=f(S_n)-f(S_i)$
  - 4、若$\Delta f<0$, 令$S_{i+1}=S_n$, 转步骤5，（S_n比S_i好, 无条件接受）；否则按概率$\exp(-\Delta f/T_k)$接受$S_{i+1}=S_n$，（S_n比S_i差, 有条件接受)。
  - 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。

概率式学习

- 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

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：** $w_{ij}(k+1) = w_{ij}(k) + y_i y_j$

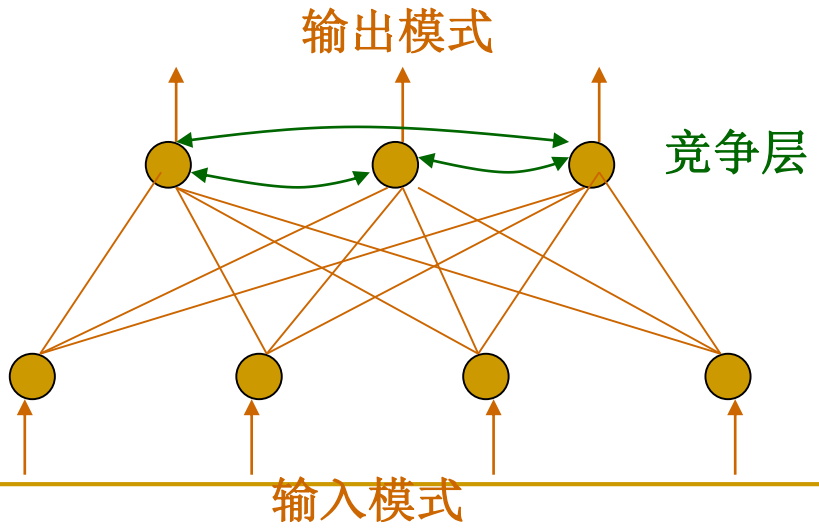
- 微分Hebb 学习方法：根据神经元状态变化来调整权值。

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

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。

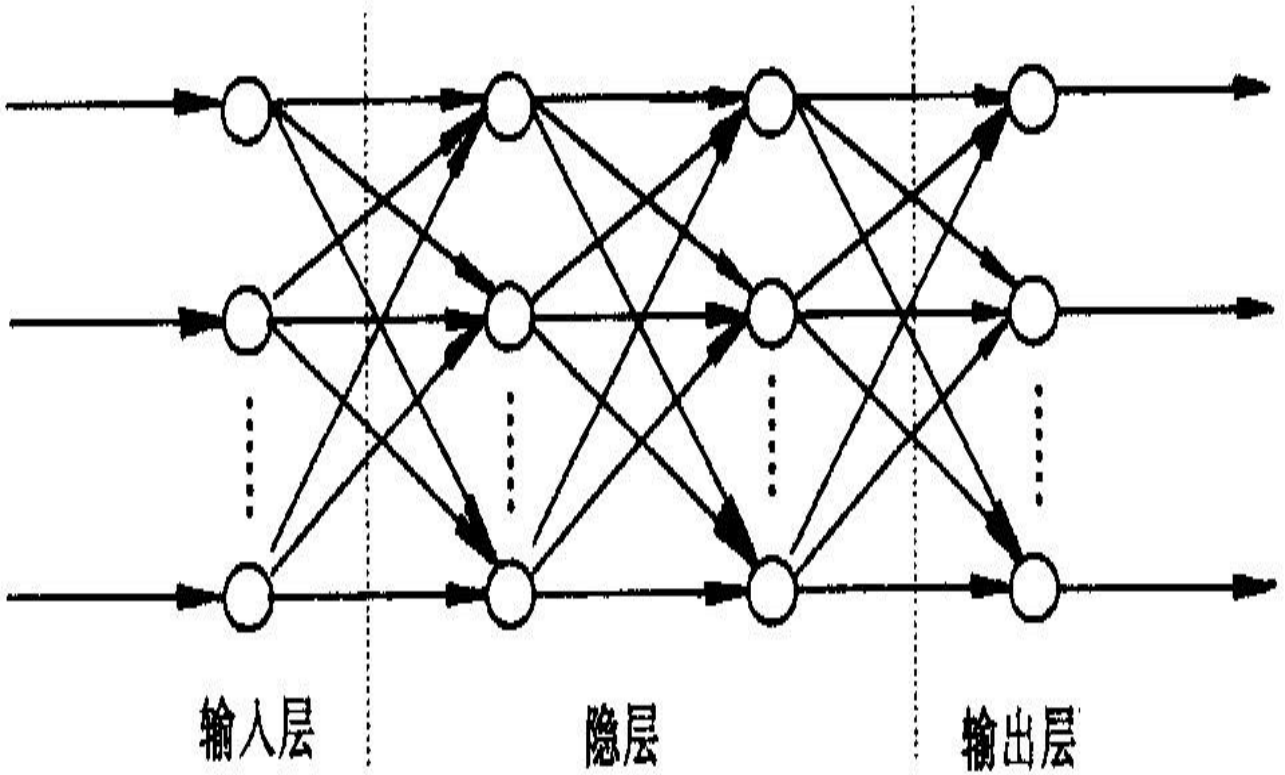
定义N_c，其半径逐渐减小至接近于零，权值的学习规则为

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

- 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。

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：

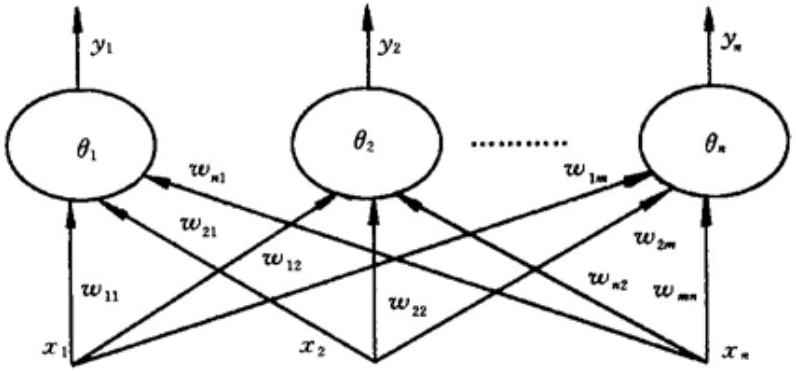
$$f(x)=\begin{cases}1 & x\geq 0\\ -1 & x<0\end{cases}$$

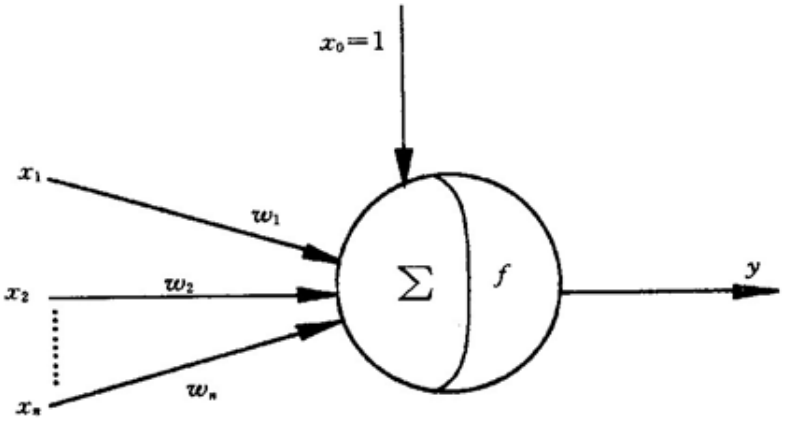
- 主要用于模式分类。

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）。

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



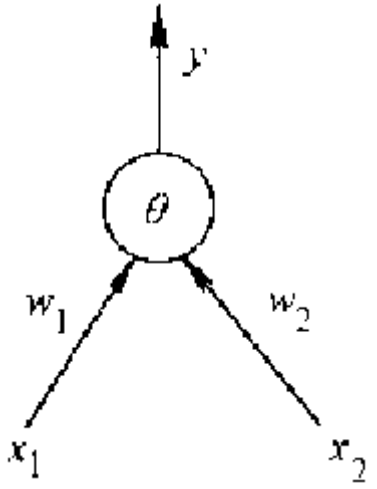


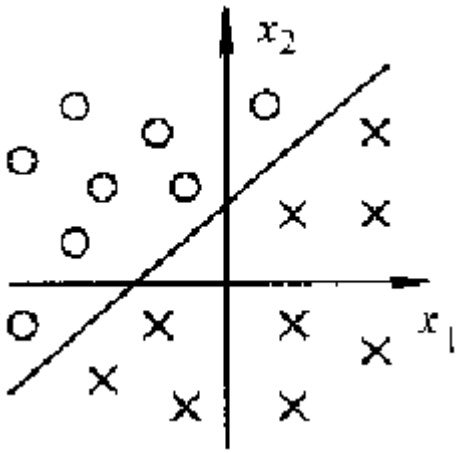
单层感知器为线性可分分类器

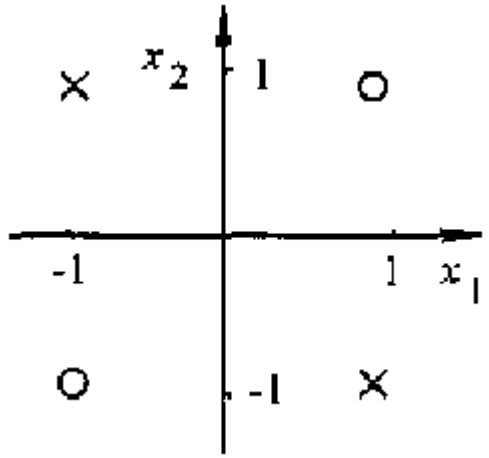
- 以二维空间为例，根据单层感知器的变换关系，分界线的方程为

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

- 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。

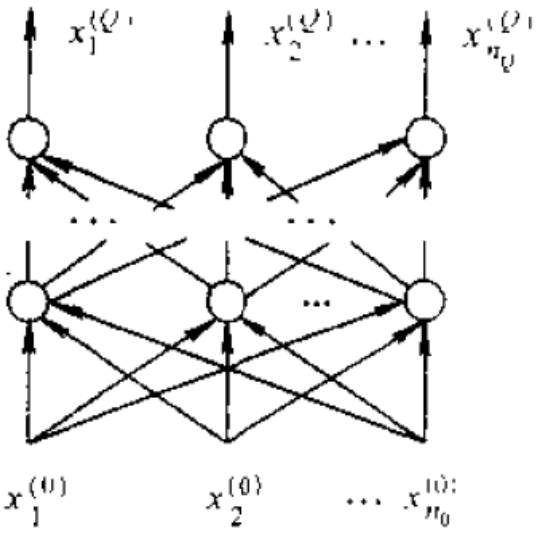
多层感知器

- 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

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

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

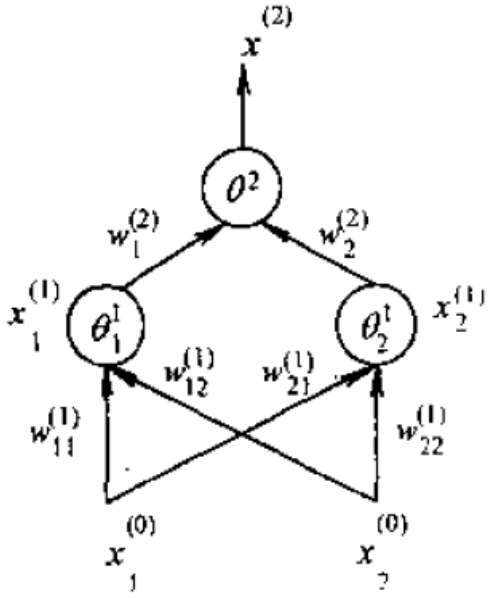
$$i = 1,2,\cdots,n_q \qquad j = 1,2,\cdots,n_{q-1} \qquad q = 1,2,\cdots,Q$$

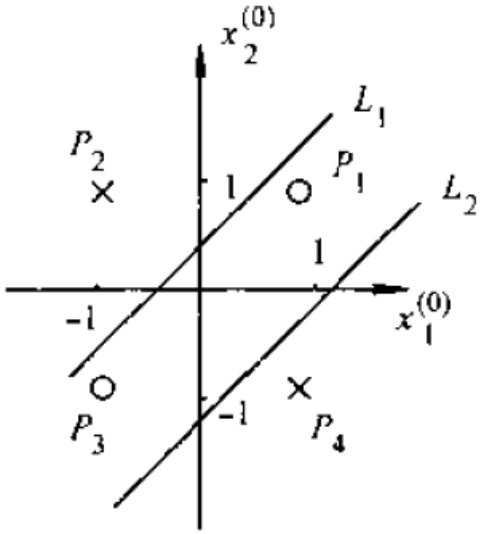


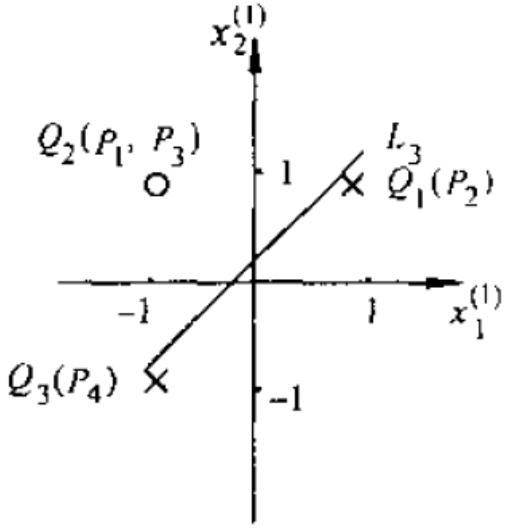
- 学习算法采用δ学习规则。

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。

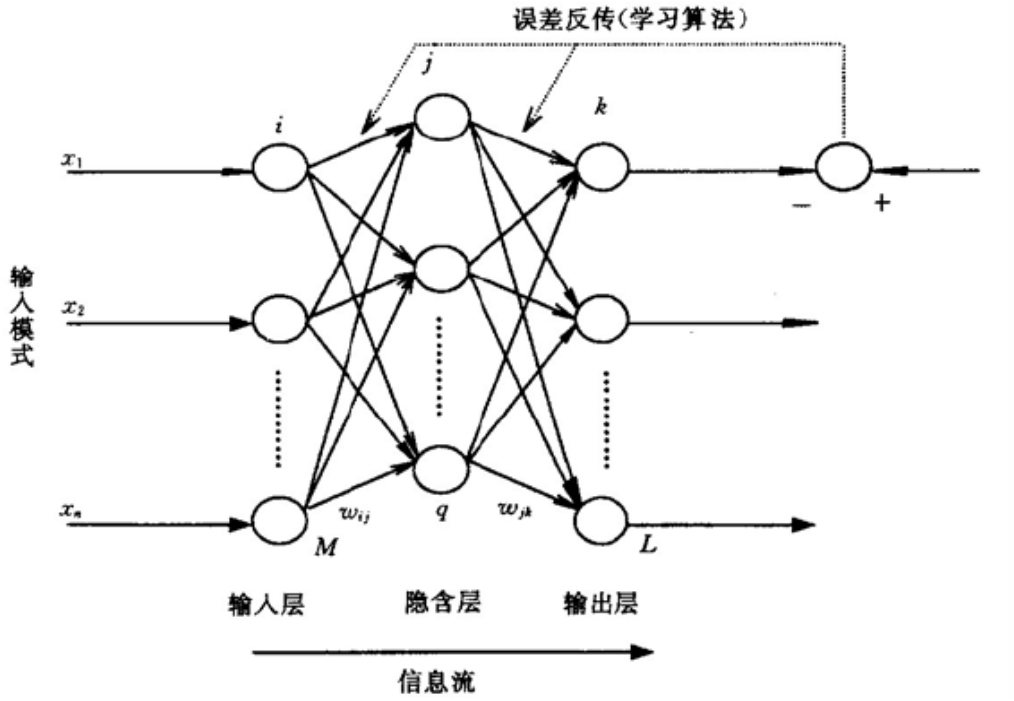






误差反向传播神经网络(BP 网络)

- 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 ($p = 1,2,\cdots, P$)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，$\dfrac{\partial E}{\partial w_{ij}}=\sum_{p=1}^{P}\dfrac{\partial E_p}{\partial w_{ij}}$
  - 关键是计算 $\dfrac{\partial E_p}{\partial w_{ij}}$

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，经典的网络连接权值调整式为：

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

式中，$t+1$表示第$t+1$步。

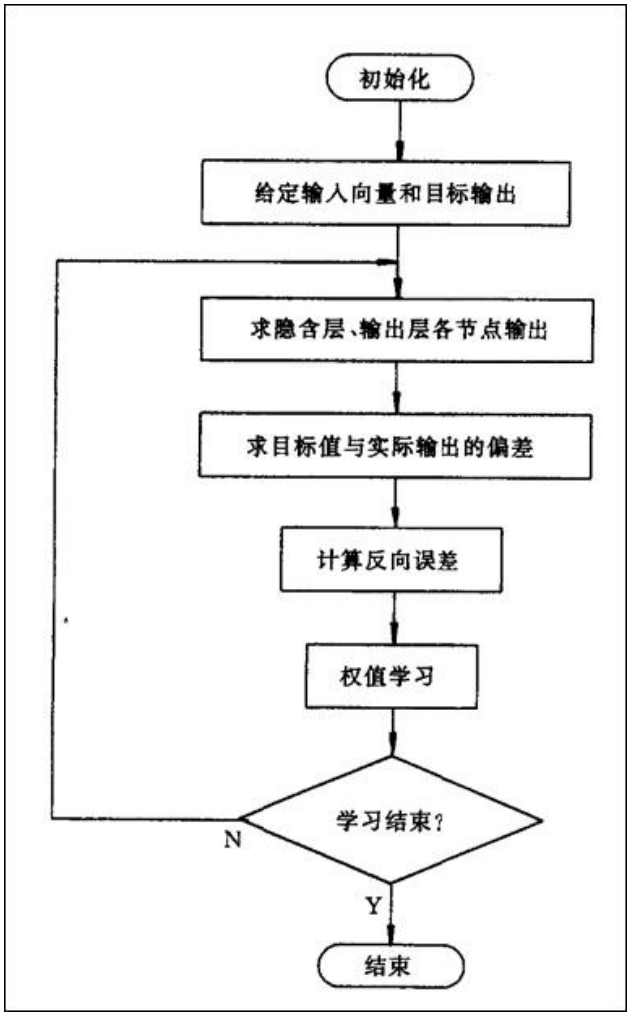
为加快算法收敛，引入松弛因子，得到改进的网络连接权值调整式

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

式中，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。

反向传播算法（BP 算法）

- 在使用BP算法时，要进行数据样本集的预处理和标准化。
  - 需要检查数据样本集中是否存在异常点（野点），若存在，这些点必须剔除。
  - 数据样本集的标准化过程是通过尺度变换将网络的输入输出样本数据映射到[0,1]区间（取S型激发函数时）或[-1,1]区间（取双曲型激发函数时）的归一化过程。
  - 输入数据归一化可以使不同量纲的输入分量具有同等重要的地位。
  - 输入数据归一化可避免神经元的输入量过大而处于激发函数的平坦区域，导致神经元输出饱和，造成梯度消失，从而权值学习失败。
  - 输出数据归一化才能使样本数据的输出满足激发函数的输出范围（即[0,1]或[-1,1]区间）。

反向传播算法（BP 算法）

- 数据样本集的标准化公式
  - 数据样本的归一化方法比较多，可以采用线性变换方法，也可以采用非线性变换方法。
  - 将训练样本输入输出数据归一化后进行训练，训练结束后使用网络时，需要将输入数据进行归一化，将网络的输出值进行反归一化。
  - 输入输出样本数据(*x*,*y*)的常用归一化线性变换式

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \qquad y_{\text{norm}} = \frac{y - y_{\min}}{y_{\max} - y_{\min}}$$

  - 网络应用时，网络输入的归一化与训练样本的归一化一致，对应的网络输出反归一化公式

$$y = \left(y_{\max} - y_{\min}\right) y_{\text{norm}} + y_{\min}$$

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；

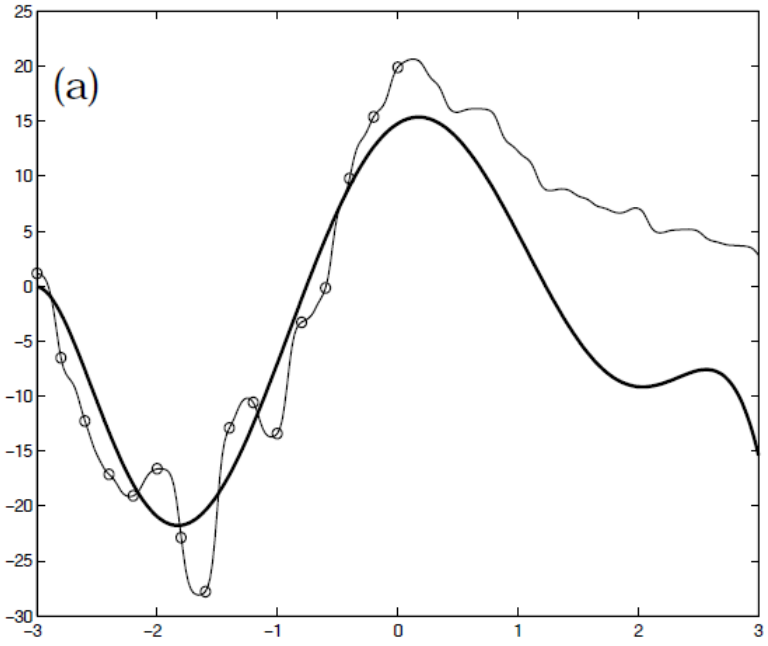
- 网络能通过学习带正确答案的实例集自动提取“合理的”求解规则，即具有自学习能力；

多层前向BP网络的缺点

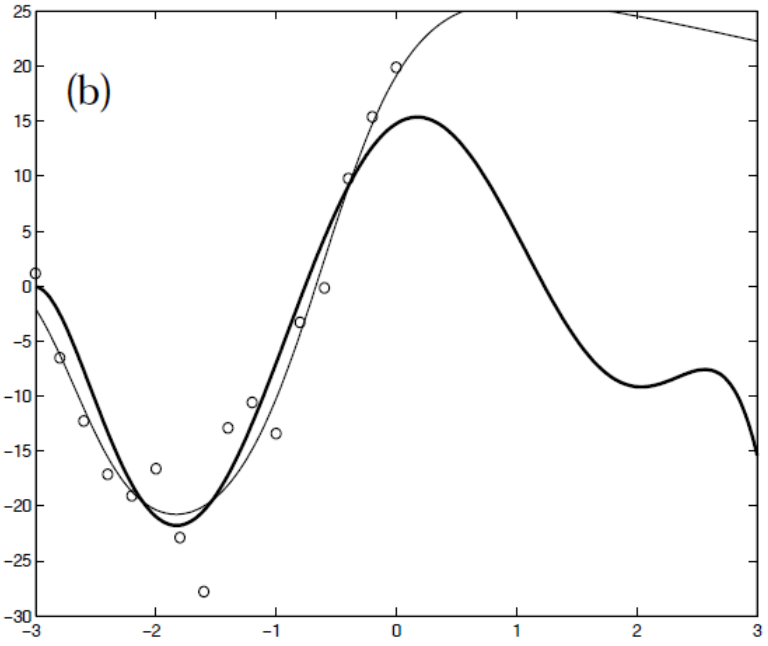
- 存在局部极值
- 学习速度很慢
- 网络训练失败的可能性较大
- 难以解决应用问题的实例规模和网络规模间的矛盾
- 网络结构的选择尚无一种统一而完整的理论指导，一般只能由经验选定
- 新加入的样本要影响已学习成功的网络，而且刻画每个输入样本的特征的数目也必须相同
- 网络的预测能力（也称泛化能力、推广能力）与训练能力（也称逼近能力、学习能力）的矛盾

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

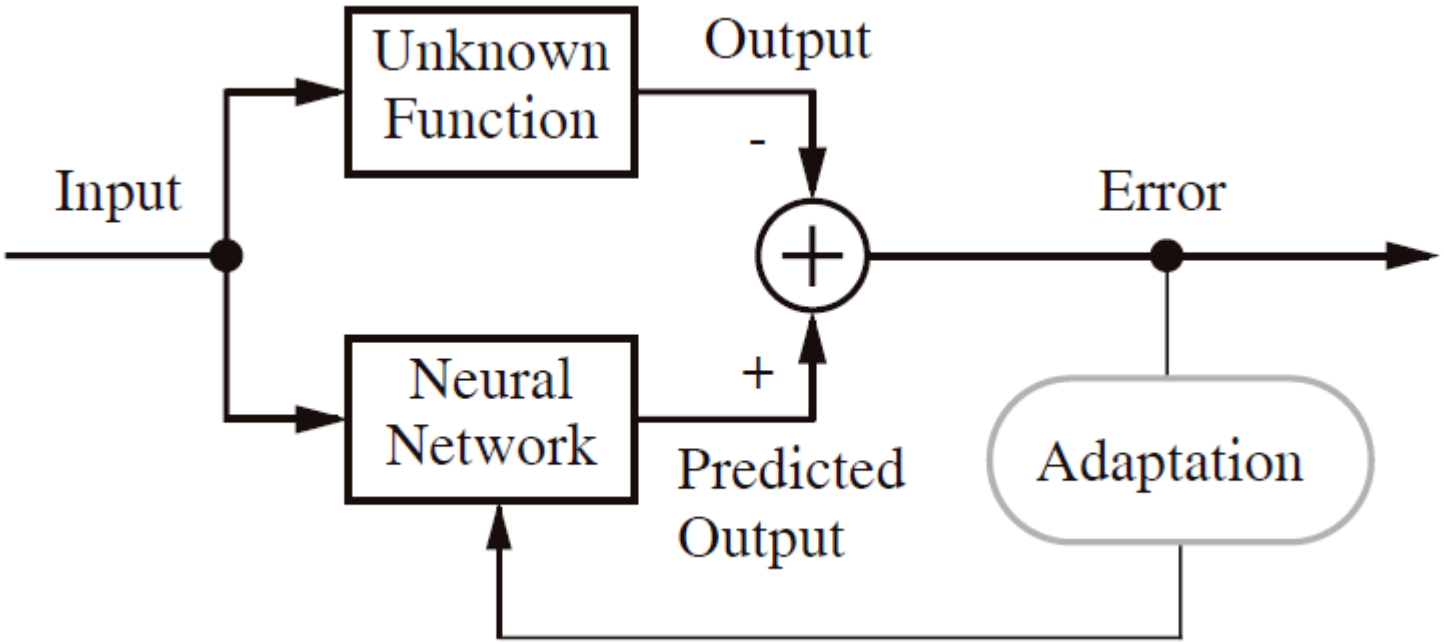
- 增加“惯性项”
  - 加快收敛速度。
- 采用动态步长
  - 加快收敛速度。
- 与其他全局搜索算法相结合
  - 为克服BP 算法全局搜索能力弱的缺点，将BP 算法与具有很强全局搜索能力的算法相结合，如与遗传算法相结合。
- 模拟退火算法
  - 为克服BP 算法易陷入局部极小的缺点。

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。

2、基于神经网络的系统建模与辨识

神经网络函数逼近

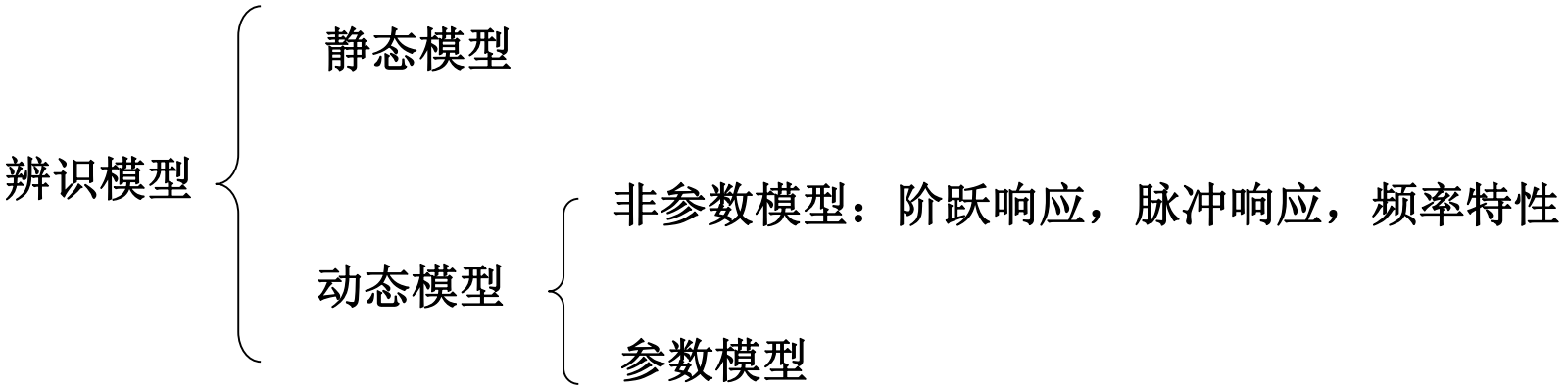

Unknown
Function
Output
-
Input
Error
+
Neural
Network
+
Predicted
Output
Adaptation

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。

辨识三要素

- 辨识是在输入和输出数据的基础上，从一组给定的模型中，确定一个与所测系统等价的模型。
- 辨识三要素
  - 输入/输出数据：能够量测到的系统输入与输出。
  - 模型类：所考虑系统的结构。
  - 等价原则：辨识的优化目标。

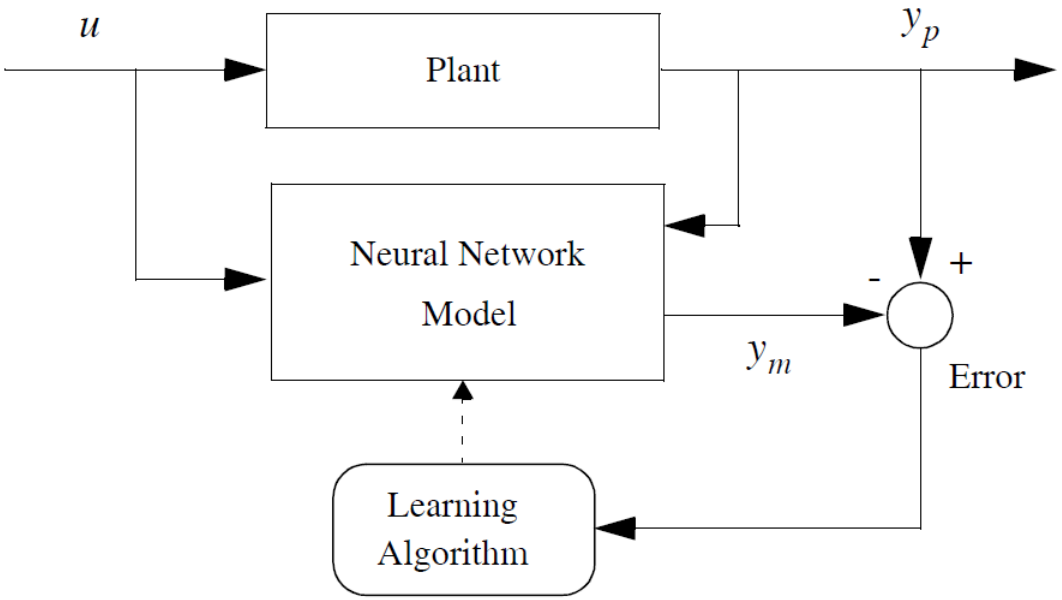


神经网络用于系统辨识

- 选择一个合适的神经网络模型来逼近实际的系统。
- 辨识准则为

$$\min_f \|e\| = \min_f \|y_p - y_m\|$$

- y_m，y_p分别是模型和系统对于系统输入的输出响应。



神经网络建模需要考虑的问题

（1）模型的选择

- 存在精确性和复杂性的矛盾。例，多层网络模型的节点数或隐层数的选择。

（2）输入信号的选择

- 输入信号必须满足一定的条件
  - 在辨识时间内，输入信号必须是持续激励的，即输入信号必须充分激励系统的所有模态。从频谱观点看，输入信号的频谱必须足以覆盖系统的频谱。
  - 所谓输入信号的最优设计问题，即设计输入信号使给定问题的辨识精度最高，常用的输入信号有白噪声或伪随机信号。

（3）误差准则的选择

- 误差准则是用来衡量模型接近实际系统的标准，它通常表示为一个误差的泛函。记作

$$J(\theta)=\sum_{k=1}^{L} f\left[e(k)\right]$$

用得最多的是平方函数，即 $f\left[e(k)\right]=e^2(k)$

系统辨识的常用方法

- 系统辨识是一个优化问题。
- 一般辨识算法的基本原理，就是通过建立系统依赖于参数θ的模型，把辨识问题转化成对模型参数的估计问题。
- 针对线性系统或可线性化的系统，主要有三种方法
  - 最小二乘法
  - 梯度校正法
  - 极大似然法

基于神经网络的辨识的特点

- 不要求建立实际系统的辨识格式。即可省去系统建模这一步，因为神经网络本质已作为一种辨识模型，其可调参数反映在网络内部的权值上。
- 可以对本质非线性系统进行辨识。辨识是非算法式的，由神经网络本身体现。辨识的结果为网络外部特性拟合系统的输入/输出特性，网络的内部特性归纳隐含在系统输入/输出数据中的系统特性。
- 辨识的收敛速度不依赖于待辨识系统的维数，只与神经网络本身及其所采用的学习算法有关，而传统的辨识算法随模型参数维数的增大变得很复杂。
- 神经网络具有大量连接，其连接权的权值在辨识中对应于模型参数，通过调节这些参数可使网络输出逼近系统输出。
- 神经网络作为实际系统的辨识模型，实际上也是系统的一个物理实现，可以用于在线控制。

神经网络用于系统辨识的一般结构

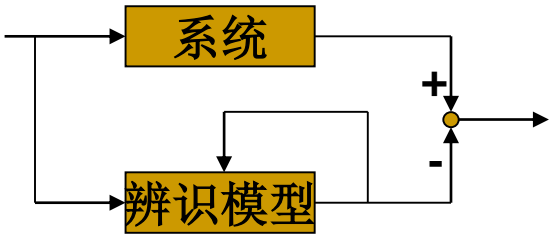
- 常利用多层静态网络进行系统辨识，要求预先给定系统的阶，即必须给出定阶的差分方程，典型的是NARMA模型(非线性自回归滑动平均模型 Nonlinear Auto Regressive Moving Average)。
- 一般可采用两种辨识模型
  - 并联模型：（动态反馈前向网络）

$$\hat{y}_p(k+1) = f\left[\hat{y}_p(k), \hat{y}_p(k-1), \cdots, \hat{y}_p(k-n+1); u(k), u(k-1), \cdots, u(k-m+1)\right]$$

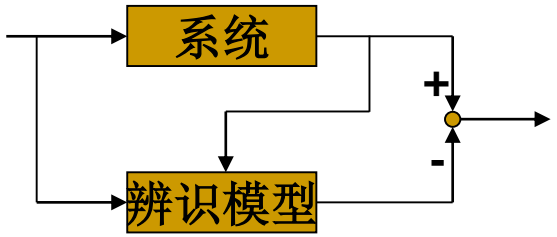
  - 串-并联模型：（静态前向网络）

$$\hat{y}_p(k+1) = f\left[y_p(k), y_p(k-1), \cdots, y_p(k-n+1); u(k), u(k-1), \cdots, u(k-m+1)\right]$$

- 一般，若并联模型可行，则优于串联模型，但学习算法需修改，比简单的基于误差的反向传播学习（BP）算法复杂。



并联辨识模式



串-并联辨识模式

例：基于BP网络的系统辨识

（1）结构设计

- 采用串-并联辨识模式，考虑以下形式的单输入单输出非线性动态系统

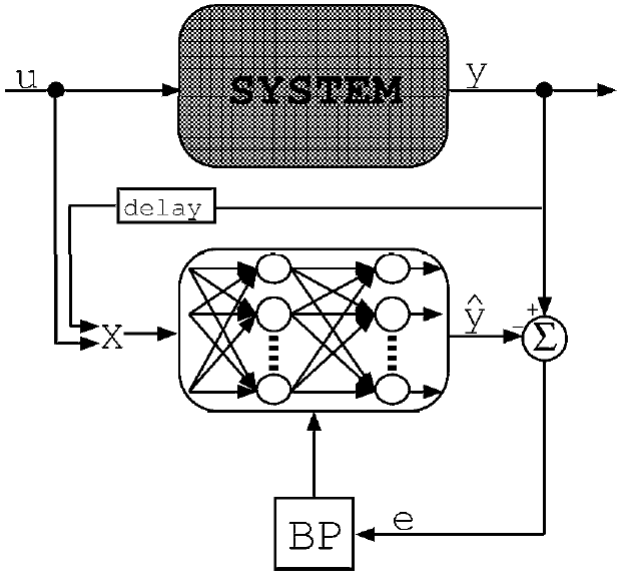
$$\hat{y}_p(k+1) = f\left[y_p(k), y_p(k-1), \cdots, y_p(k-n+1); u(k), u(k-1), \cdots, u(k-m+1)\right]$$

其中，m，n为输入、输出的阶次。

– 选择包含有一个隐含层的三层前向网络，活化函数取Sigmoid函数.

– 各层神经元节点数

- 输入层：$n_I = n+m$
- 输出层：$n_o = 1$
- 隐含层：一般要求$n_H \geq n_I$，可采用尝试训练后再比较性能指标的方法决定。



例：基于BP网络的系统辨识

（2）辨识算法

- a)取性能指标为 $$J=\frac{1}{2}\sum_{k}\left[y(k)-\hat{y}(k)\right]^2$$
- b)采用标准的BP算法（或改进的BP算法）修改网络权值。
- c)给定判定收敛的条件，如当所有网络连接权的变化充分小时，可判断辨识结束。

- 由于采用三层BP网络，结构比较简单，上述辨识算法的收敛性可由Lyapunov稳定性理论证明。

3、神经控制的结构方案

神经控制的结构方案

- 基于自适应方式的神经网络控制系统
- 由神经网络单独构成的系统
- 基于常规控制原理的神经网络控制
  - 与其它类型的控制原理相结合，采用常规的控制理论设计控制器结构，用神经网络取代其中部分内容或进行决策处理。
- 神经网络智能控制
  - 将神经网络与人工智能、模糊逻辑等相结合。
- 神经网络优化控制
  - 由于神经网络可以表达成任何非线性函数，因此，使用神经网络能完成各种复杂的优化运算。
  - 可将神经网络用于控制器的优化设计。
  - 例：利用**Hopfield**网络求解广义预测控制中矩阵求逆问题。
  - 例：利用神经网络在线辨识对象的数学模型。
  - 例：利用神经网络设计控制器。

基于神经网络的监督式控制

（1）选择网络的输入。

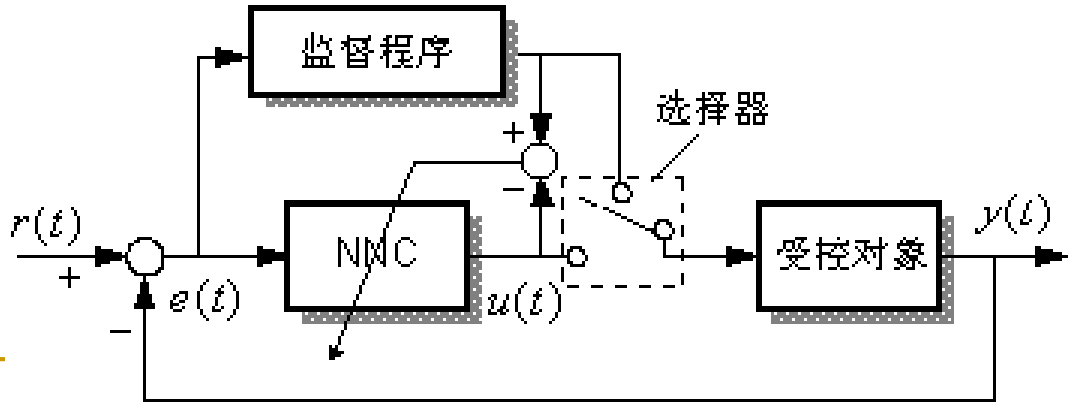
- 通过传感器和传感信息处理，调用必要的和有用的控制信息。

（2）选择和构造一个神经网络作为控制器。

- 构造神经网络，选择NN类型、结构参数和学习算法等。

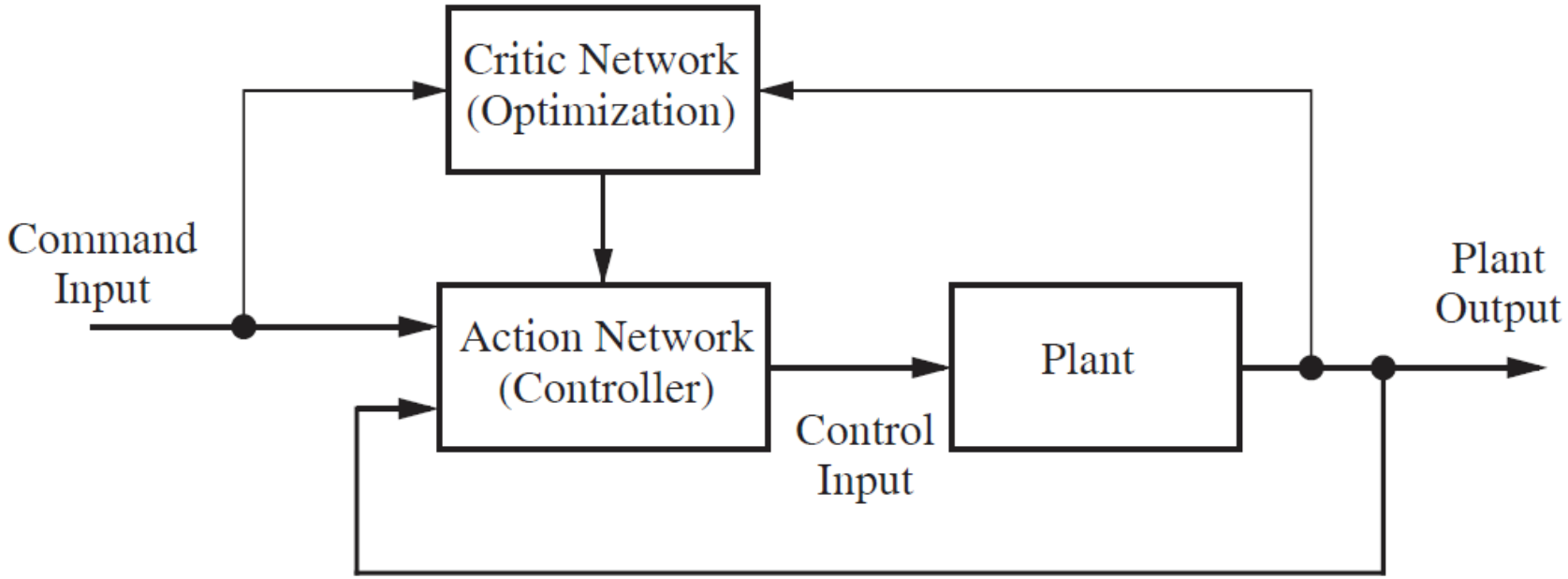
（3）采用参考控制器（监督程序）训练网络。

- 训练NN控制器，实现输入和输出间的映射，以便进行正确的控制。在训练过程中，可采用线性律、反馈线性化或解耦变换的非线性反馈作为导师（监督程序）来训练NN控制器。



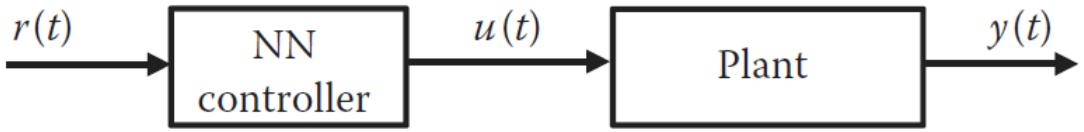
神经网络监督控制

- 一个神经网络作为逆模控制器，另一个神经网络进行控制性能评价，给出控制预测并调整逆模控制器参数。
- 评价网络经过训练以优化未来性能。这种训练是通过强化学习来进行的，强化学习是对动态规划的一种近似。



神经网络逆模型控制

- 采用受控系统的一个逆模型，神经网络与受控系统串接以便使系统在期望响应（网络输入）与受控系统输出间得到一个相同的映射。
- 网络(NN)直接作为前馈控制器，而且受控系统的输出等于期望输出。



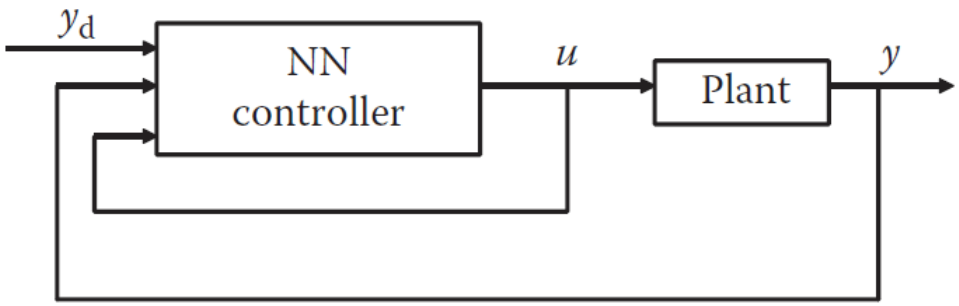
若系统为

$$y(k) = f(y(k-1), \ldots, y(k-n), u(k-1), u(k-2), \ldots, u(k-m))$$

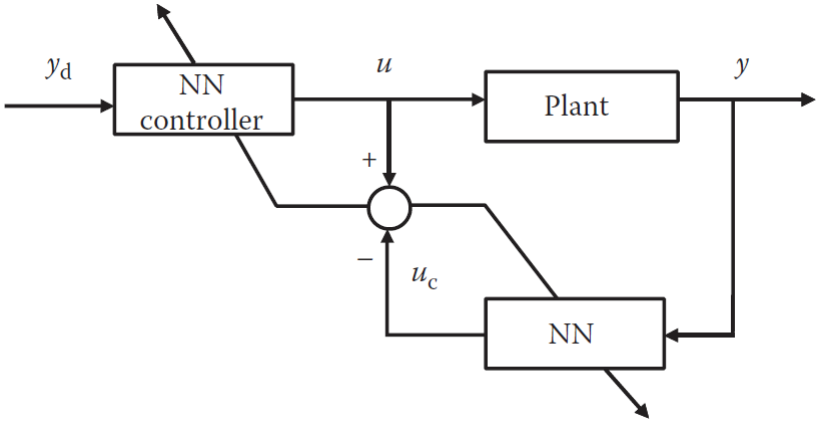
则欲训练的逆模型为

$$u(k) = \phi(y(k+1), y(k), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m+1))$$

离线训练结束，在上式中用期望***r*(*k*+1)**代替***y*(*k*+1)**即得到逆控制器。



间接型NN在线逆模型控制



- 策略：
  - 系统运行过程中，在线应用多层前向网络学习从系统输出到系统输入的逆模型，一般采用BP算法调整网络权值。
  - NN控制器为结构与逆模型相同的网络，权值在线取逆模型的相同的权值。
- 缺陷：
  - 由于逆模型为静态网络，与真实的系统动态逆模型不一致，导致控制方案的稳定性常出问题。

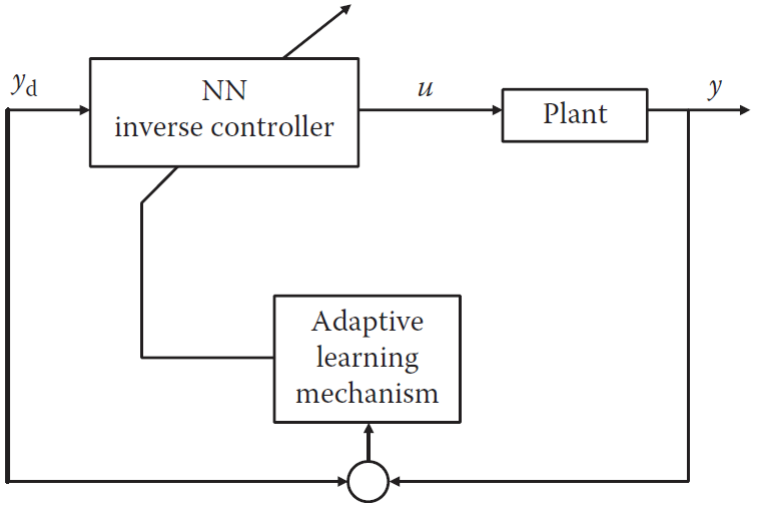
直接型NN在线逆模型控制

- 策略：
  - 在线通过期望输出与系统输出的误差直接调整逆模型网络的权值。
  - 若$e=y_d-y$，$P_i(\mathbf{u})$系统的第i个输出变量，通过反向传播方法，得到

$$\Delta w_{ji}^{(k)} = \mu \delta_j u_i^{(k)}$$

$$\delta_j^{(k-1)} = f'(\text{net}) \sum_i \delta_i^{\text{p}} \frac{\partial P_i}{\partial u_j}$$

$$\delta_i^{\text{p}} = y_{d_i} - y_i$$



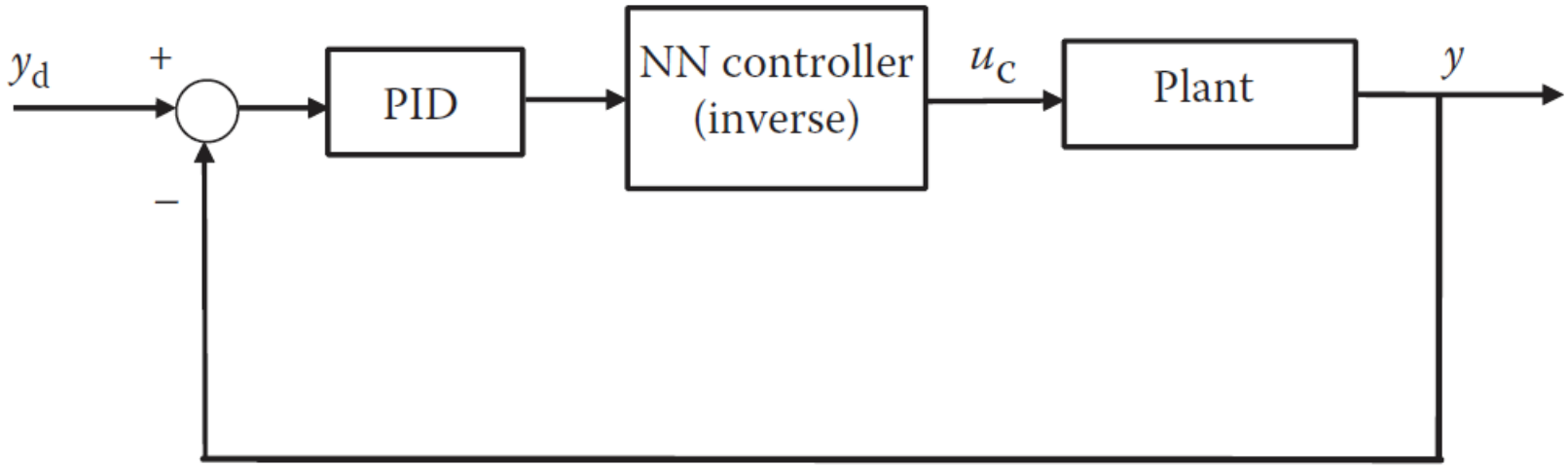
  - 由于一般系统的Jacobian未知，可采用结合学习步长的符号函数$\text{sgn}[\partial P_i/\partial u_j]$或如下的测试运行获得：

$$\frac{\partial P_i}{\partial u_j} \approx \frac{P_i(\mathbf{u}+\delta u_j) - P_i(\mathbf{u})}{\delta u_j}$$

- 缺陷：
  - 由于在线运行，系统缺乏持续激励，网络输入的适应范围受限。
  - 控制方案的抗干扰性弱。

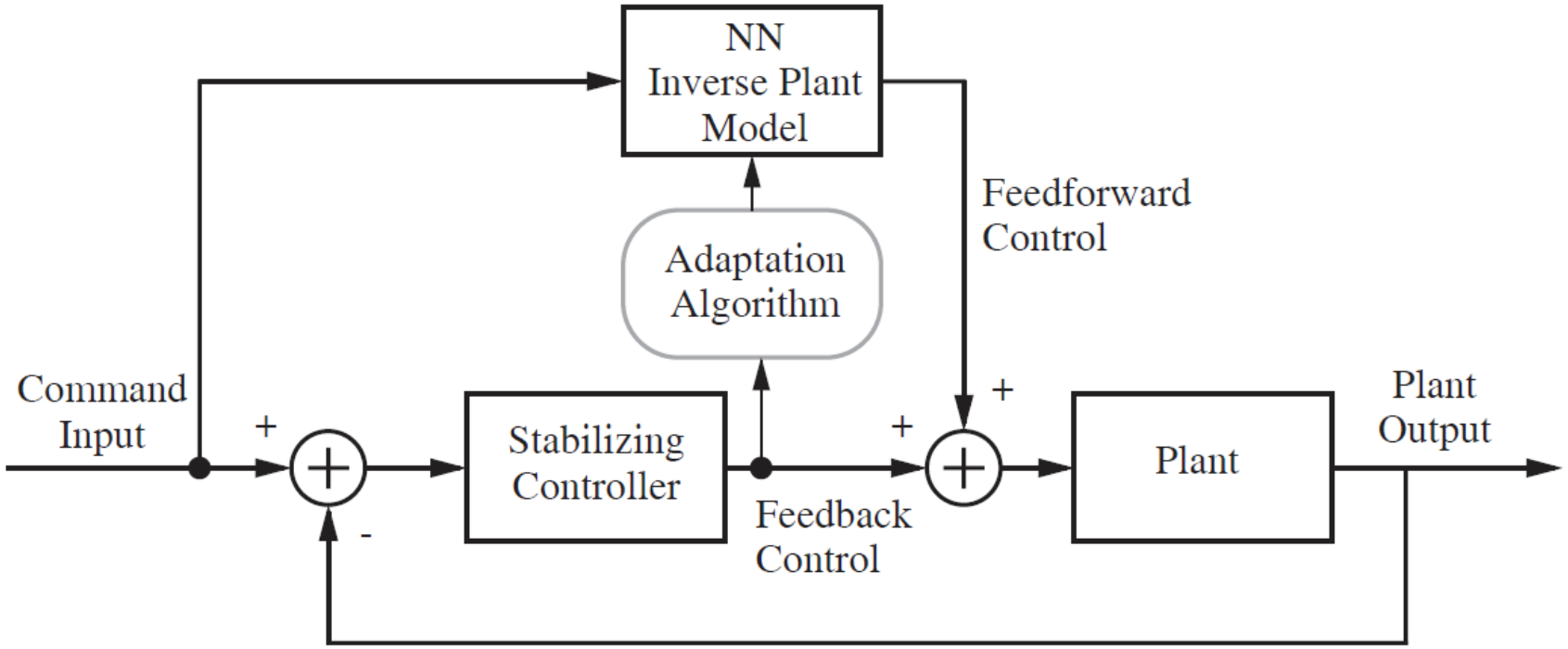
神经网络逆模型控制

- 结合**PID**控制，增加系统的鲁棒性和抗干扰能力



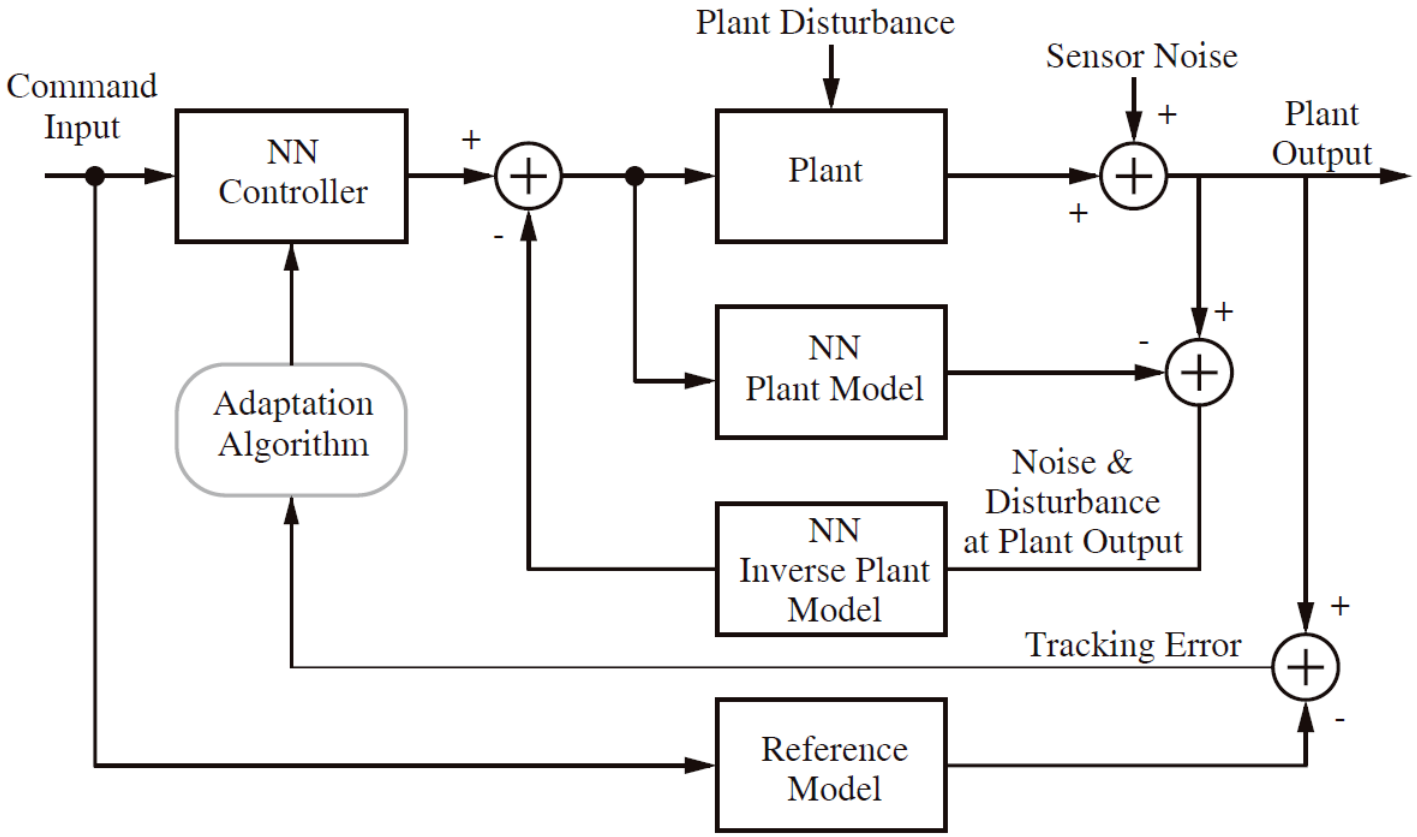
神经网络逆模型控制

- 在网络未充分学习的前提下，系统运行过程从一个稳定的控制开始。



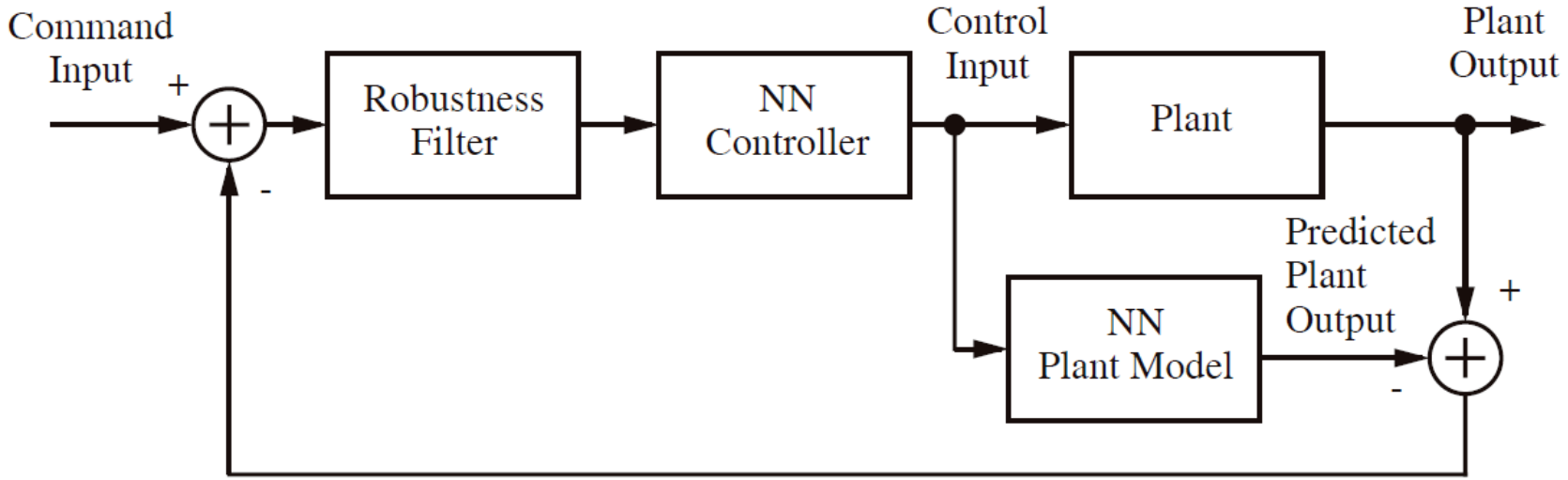
神经网络自适应逆模控制

- 增加一个逆系统模型，目的是消除系统干扰和测量噪声。



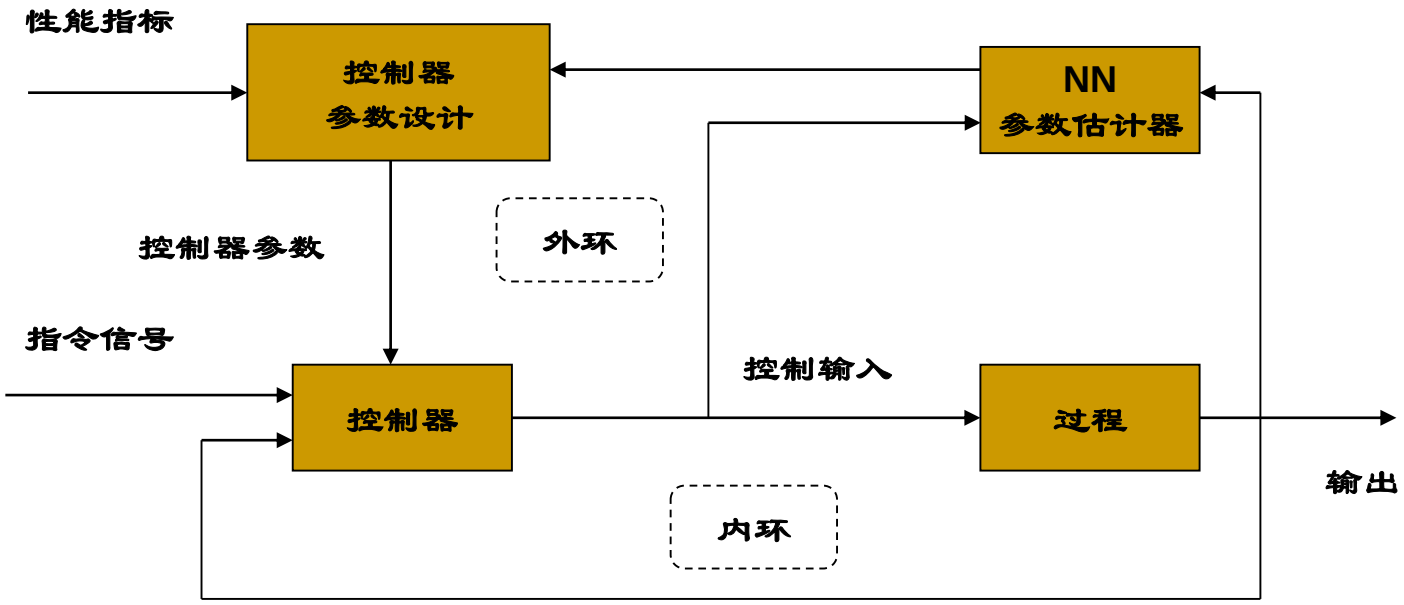
神经网络内模控制

- 系统模型(NN Plant Model)与实际系统并行设置。
- 反馈信号由系统输出与模型输出间的差得到，然后由NN Controller（在正向控制通道上一个具有逆模型的NN控制器）进行处理；
- NN控制器与系统的逆有关。



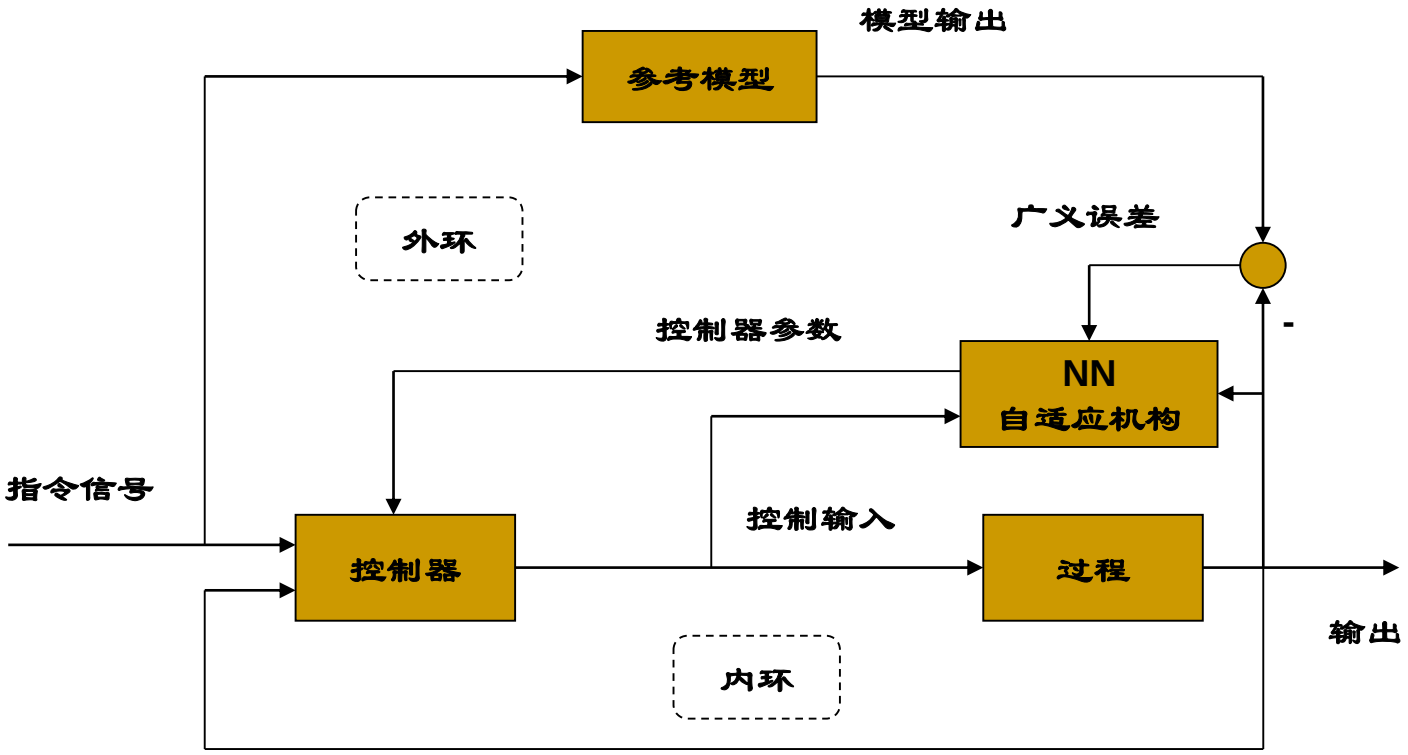
神经网络自适应控制

- NN自校正控制



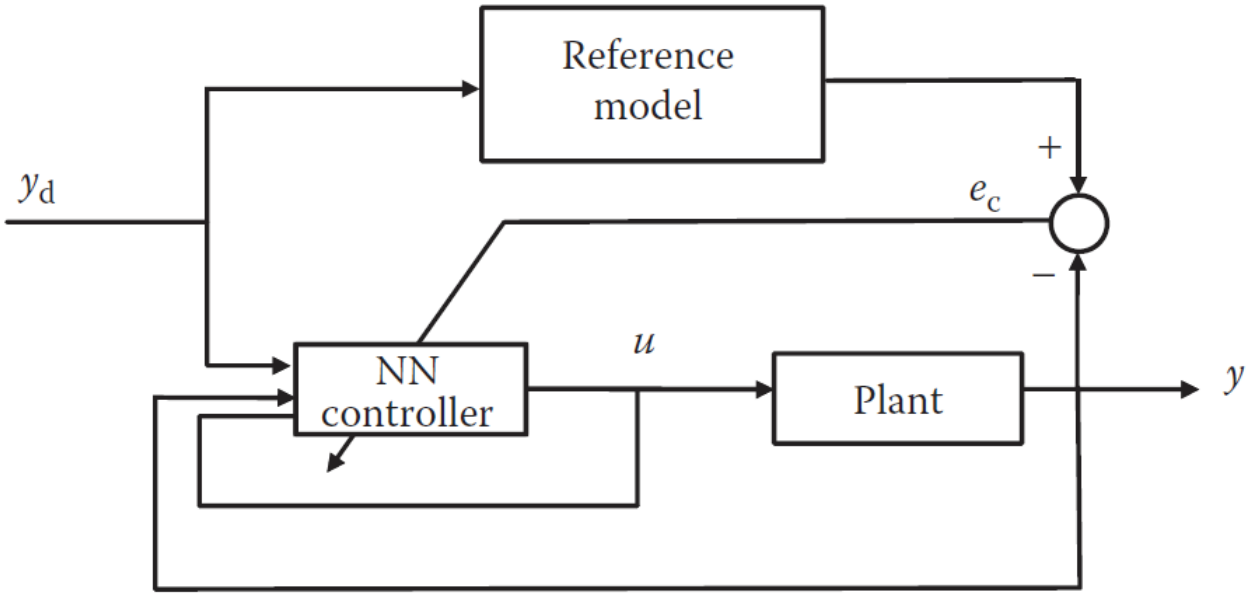
神经网络自适应控制

- NN模型参考自适应控制



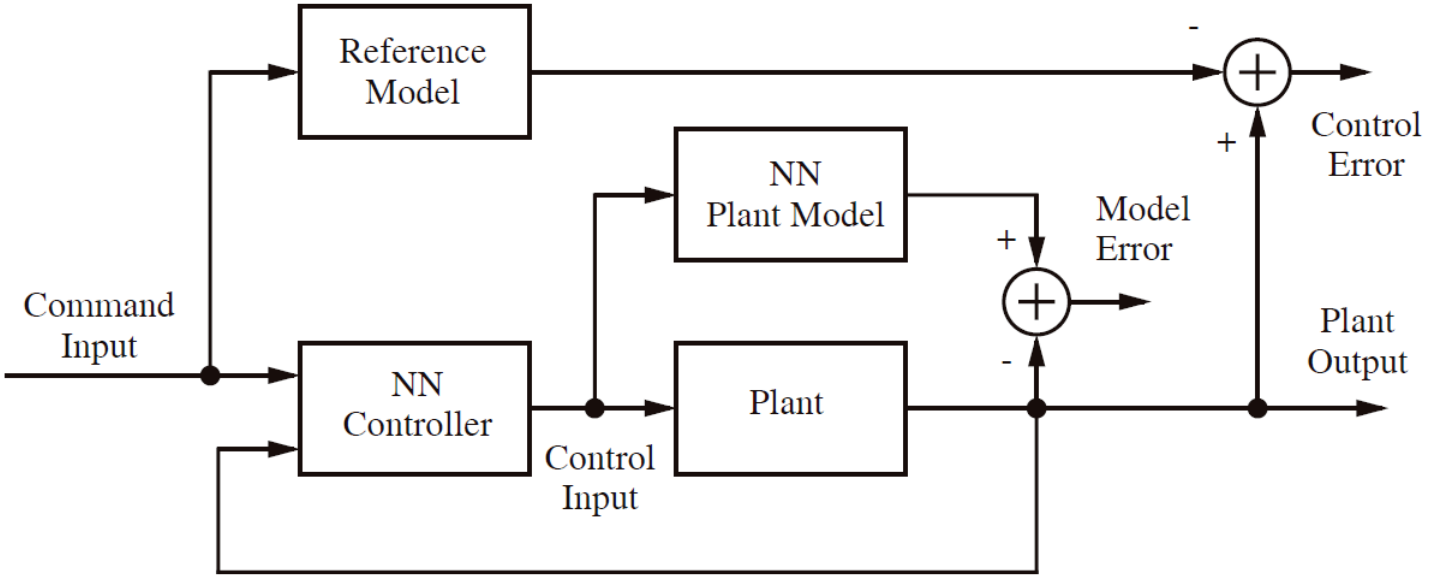
直接NN参考自适应控制

- NN在参考自适应控制中直接作为控制器



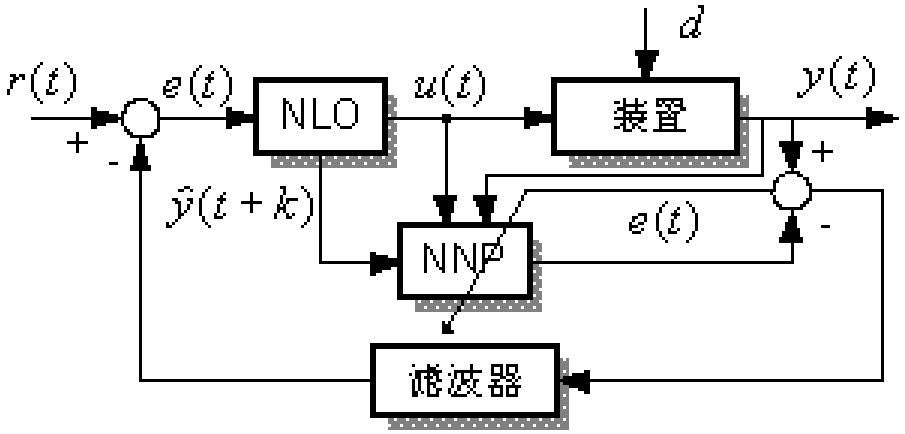
间接NN参考自适应控制

- NN系统模型先通过系统的历史数据离线训练好，并在在线运行过程中自适应调整。
- NN控制器先通过已训练好的NN系统模型进行仿真运行训练。
- 在线运行时，NN控制器通过参考模型输出与系统输出的误差在线调整。同时，自适应运行的NN系统模型针对控制器输入给出预测输出，并可根据预测误差调整NN控制器参数。



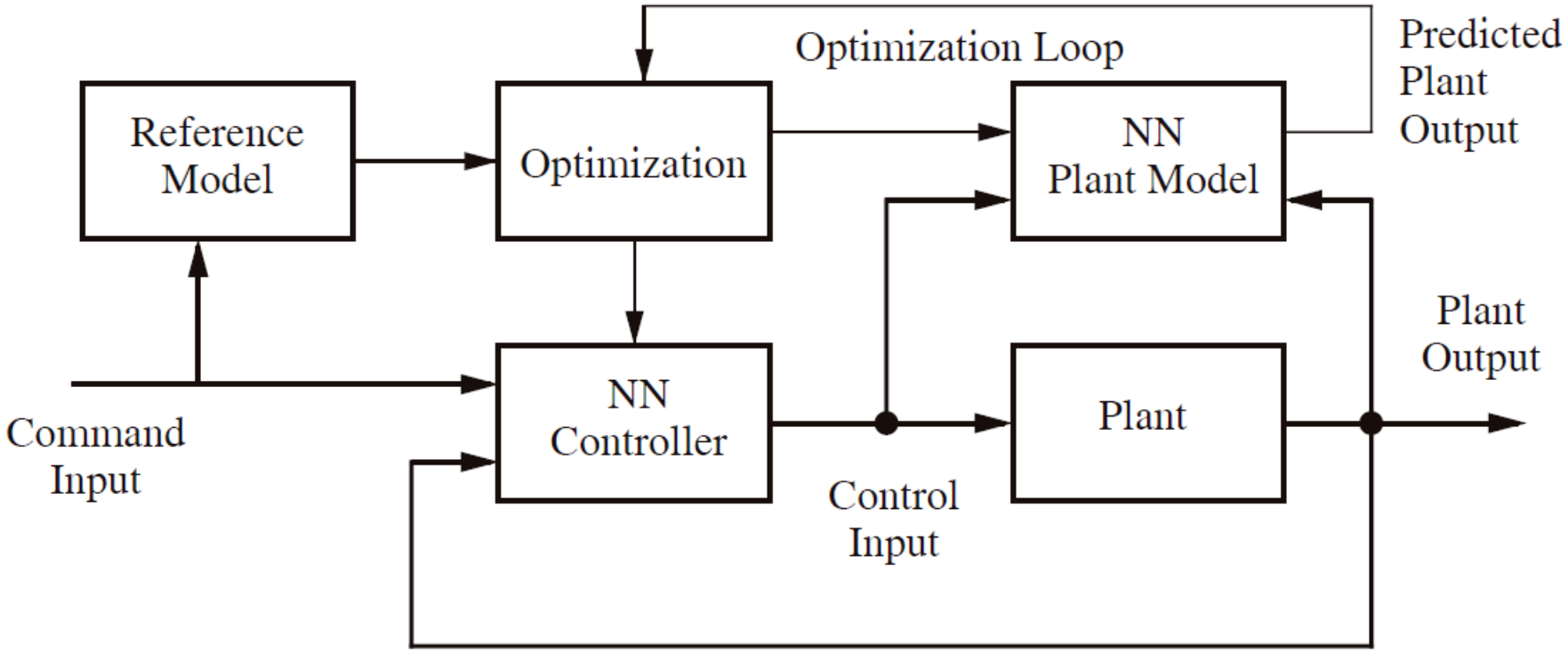
神经网络预测控制

- 预测控制具有预测模型、滚动优化和反馈校正等特点。
- 神经网络预测器NNP为一神经网络模型。
- 神经网络NLO为一非线性优化器。



神经网络预测控制

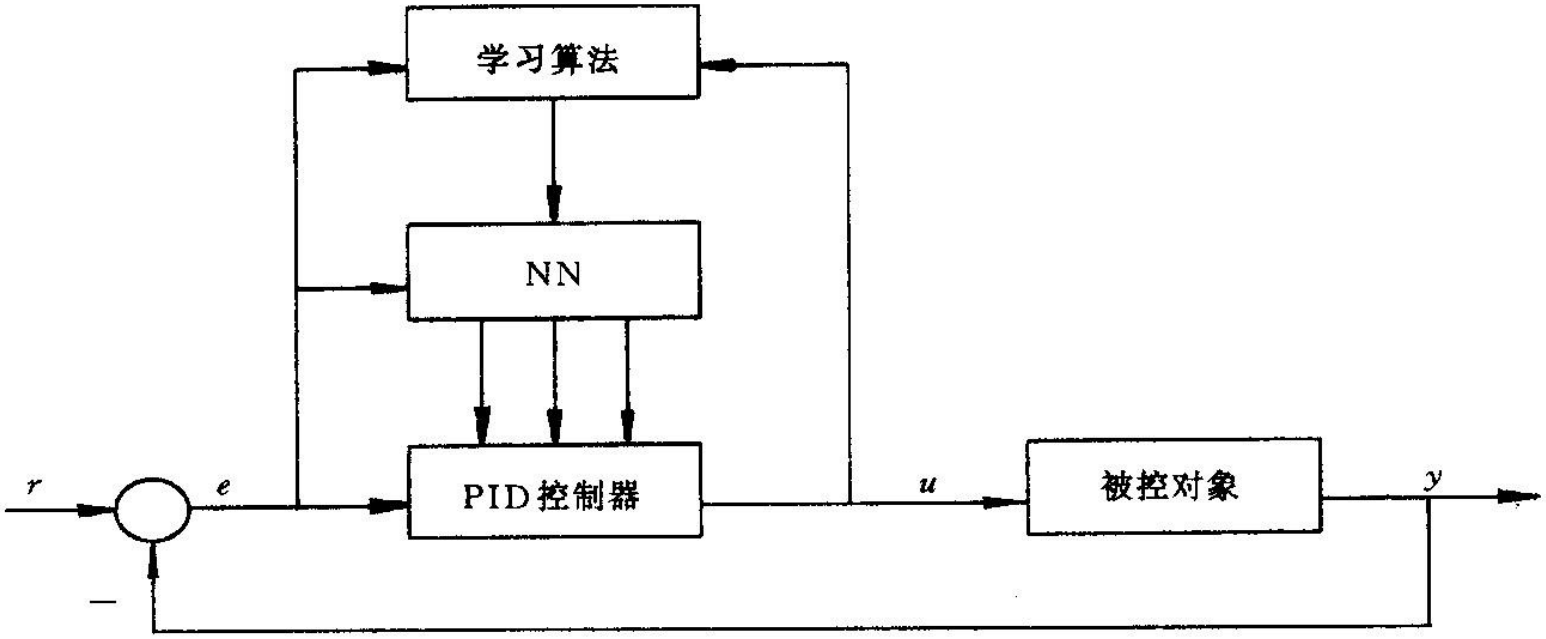
- 另一种策略是：两个神经网络分别作为预测控制中的系统预测模型和控制器。



4、神经网络PID控制

基于BP网络的PID 控制

- 神经网络可以直接作为控制器，也可以作为控制器参数的调节器。
- 通过BP神经网络自身的学习，找到某一最优控制律下的*P*，*I*，*D*参数。



控制器组成

- 经典的PID控制器
  - 直接对被控对象进行闭环控制，并且K_P, K_I, K_D三个参数为在线整定；
  - 经典增量式数字PID 的控制算式为

$$u(k)=u(k-1)+K_p[e(k)-e(k-1)]+K_I e(k)+K_D[e(k)-2e(k-1)+e(k-2)]$$

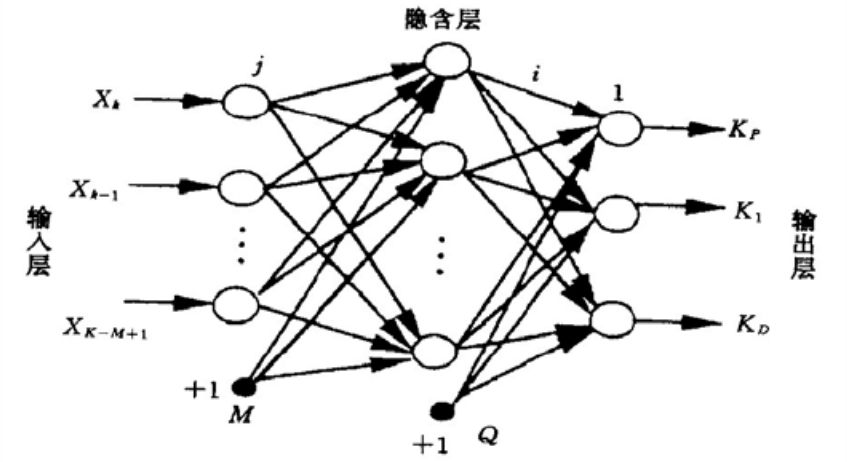
  - 因此，控制输出可表示为

$$u(k)=f[u(k-1),\ K_p,\ K_I,\ K_D,\ e(k),\ e(k-1),\ e(k-2)]$$

- 神经网络NN
  - 根据系统的运行状态，调节PID控制器的参数，以期达到某种性能指标的最优化。
  - 使输出层神经元的输出状态对应于PID控制器的三个可调参数K_P, K_I, K_D，通过神经网络的自学习、调整权系数，从而使其稳定状态对应于某种最优控制律下的PID控制器参数。

网络结构设计

- 用BP 神经网络通过训练和学习来找到一个最佳控制规律。
- 设BP 神经网络NN 是一个三层BP 网络，有M 输入节点、Q个隐层节点、3个输出节点。
- 输入节点对应所选的系统运行状态量，如系统不同时刻的输入量和输出量等，必要时要进行归一化处理。
- 输出节点分别对应PID 控制器的三个可调参数K_P, K_I, K_D
- 由于K_P, K_I, K_D不能为负值，所以输出层神经元的激发函数取非负的Sigmoid 函数，而隐含层神经元的激发函数可取正负对称的Sigmoid 函数。



网络输入输出

- **BP** 神经网络输入层节点的输出为

$$O_j^{(1)} = x_{k-j} = e(k-j), \quad j = 0,1,\cdots,M-1$$
$$O_M^{(1)} \equiv 1$$

- 网络隐含层输入、输出为

$$net_i^{(2)}(k) = \sum_{j=0}^{M} \omega_{ij}^{(2)} O_j^{(1)}(k)$$
$$O_i^{(2)} = f\left[net_i^{(2)}(k)\right], \qquad i = 0,1,\cdots,Q-1$$
$$O_Q^{(2)}(k) \equiv 1$$

- 网络输出层的输入、输出为

$$net_l^{(3)}(k) = \sum_{i=0}^{Q} \omega_{li}^{(3)} O_i^{(2)}(k)$$
$$O_l^{(3)} = g\left[net_i^{(3)}(k)\right], \qquad l = 0,1,2$$
$$K_p = O_0^{(3)}$$
$$K_I = O_1^{(3)}$$
$$K_D = O_2^{(3)}$$

权系数学习算法

- 取性能指标函数为 $$J=\frac{1}{2}[r(k+1)-y(k+1)]^2=\frac{1}{2}z^2(k+1)$$
- 依照最速下降法修正网络的权系数，按*J* 对权系数的负梯度方向搜索调整，并附加一个使搜索快速收敛全局极小的惯性项，有

$$\Delta\omega_{li}^{(3)}(k+1)=-\eta\frac{\partial J}{\partial\omega_{li}^{(3)}}+\alpha\Delta\omega_{li}^{(3)}(k)$$

$$\frac{\partial J}{\partial\omega_{li}^{(3)}}=\frac{\partial J}{\partial y(k+1)}\cdot\frac{\partial y(k+1)}{\partial u(k)}\cdot\frac{\partial u(k)}{\partial O_l^{(3)}(k)}\cdot\frac{\partial O_l^{(3)}(k)}{\partial net_l^{(3)}(k)}\cdot\frac{\partial net_l^{(3)}(k)}{\partial\omega_{li}^{(3)}}$$

- 由于$\partial y(k+1)/\partial u(k)$未知，所以近似用符号函数$\mathrm{sgn}[\partial y(k+1)/\partial u(k)]$替代，由此带来的计算不精确的影响可以通过调整学习速率η来补偿。

权系数学习算法

- 因此

$$\left.\begin{aligned}\frac{\partial u(k)}{\partial O_0^{(3)}(k)} &= e(k)-e(k-1)\\ \frac{\partial u(k)}{\partial O_1^{(3)}(k)} &= e(k)\\ \frac{\partial u(k)}{\partial O_2^{(3)}(k)} &= e(k)-2e(k-1)+e(k-2)\end{aligned}\right\}$$

- **BP 神经网络NN 输出层的权系数计算公式为**

$$\left.\begin{aligned}\Delta\omega_{li}^{(3)}(k+1) &= \eta\delta_l^{(3)}O_i^{(2)}(k)+\alpha\Delta\omega_{li}^{(3)}(k)\\ \delta_l^{(3)} &= e(k+1)\,\mathrm{sgn}\left(\frac{\partial y(k+1)}{\partial u(k)}\right)\times\frac{\partial u(k)}{\partial O_l^{(3)}(k)}g'[net_l^{(3)}(k)]\\ l &= 0,1,2\end{aligned}\right\}$$

权系数学习算法

- 隐含层权系数的计算公式为

$$\left.\begin{aligned}\Delta\omega_{ij}^{(2)}(k+1)&=\eta\delta_i^{(2)}O_j^{(1)}(k)+\alpha\Delta\omega_{ij}^{(2)}(k)\\ \delta_i^{(2)}&=f'[net_i^{(2)}(k)]\sum_{l=0}^{2}\delta_l^{(3)}\omega_{li}^{(3)}(k)\\ i&=0,1,\cdots,Q-1\end{aligned}\right\}$$

- 其中，由激发函数定义（输出层神经元的激发函数取非负的Sigmoid函数，而隐含层神经元的激发函数可取正负对称的Sigmoid 函数，即双曲正切函数。）

$$g(x)=\frac{1}{1-e^{-x}}$$

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

因此有：

$$g'[\bullet]=g(x)[1-g(x)]$$

$$f'[\bullet]=[1-f^2(x)]/2$$

基于BP神经网络的PID控制算法

1. 事先选定BP神经网络NN的结构，即选定输入层节点数M和隐含层节点数Q，并给出权系数的初值$w^{(2)}_{ij}(0)$, $w^{(3)}_{li}(0)$，选定学习速率η和平滑因子α，$k=1$；
2. 采样得到$r(k)$和$y(k)$，计算$e(k)=z(k)=r(k)-y(k)$；
3. 对$r(i)$, $y(i)$, $u(i-1)$, $e(i)$进行归一化处理，作为NN的输入；
4. 前向计算NN的各层神经元的输入和输出，NN输出层的输出即为PID控制器的三个可调参数$K_P(k)$, $K_I(k)$, $K_D(k)$；
5. 计算PID控制器的控制输出$u(k)$，参与控制和计算；
6. 计算修正输出层的权系数$w^{(3)}_{li}(k)$；
7. 计算修正隐含层的权系数$w^{(2)}_{ij}(k)$；
8. 置$k=k+1$，返回到“2”。

改进型BP网络PID控制

- 将神经网络用于控制器的设计或直接学习计算控制器的输出（控制量），一般都要用到系统的性质（如系统输出与系统输入的偏微分关系，即Jacobian）来计算权系数的修正量，然而，这些性质的定量值是不易获得的。

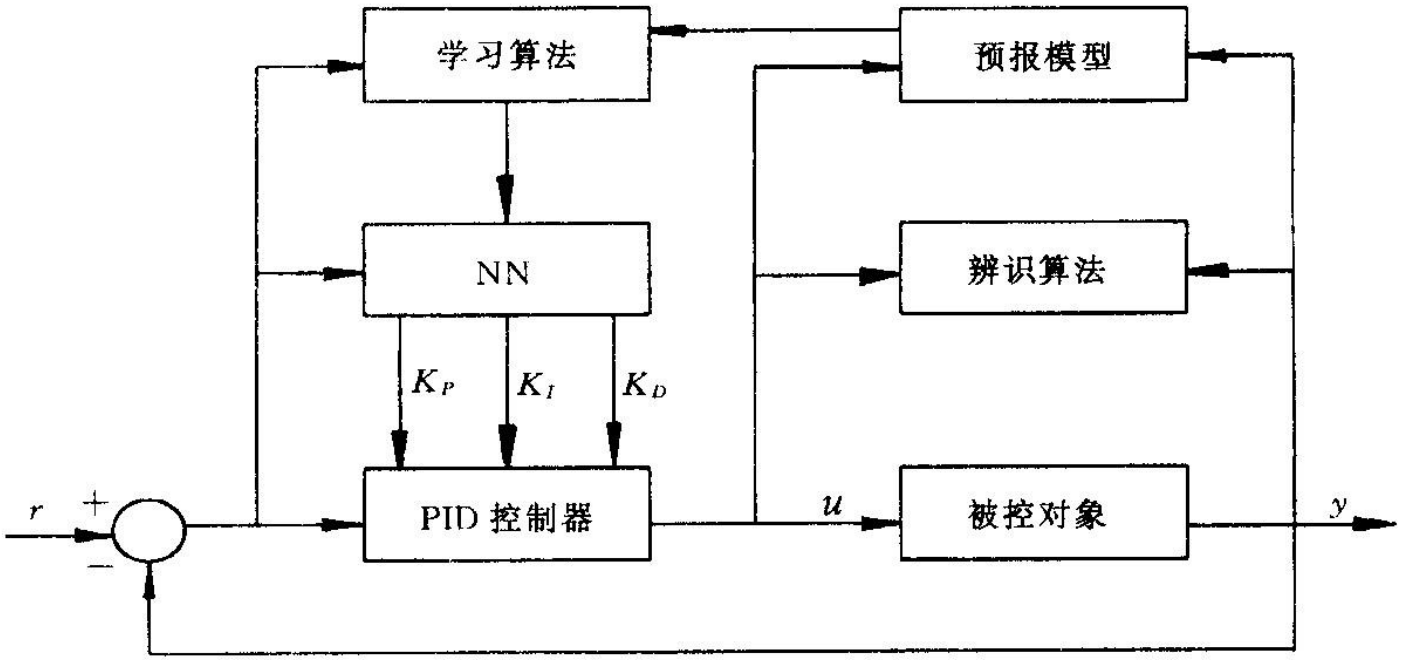
- 通常的做法是建立被控对象的预测数学模型，用该模型所计算的预测输出来取代预测处的实测值，以提高控制效果。

- 如BP网络PID控制中的$\partial y(k+1)/\partial u(k)$未知，也可由预测模型的估计值替代。

采用线性预测模型

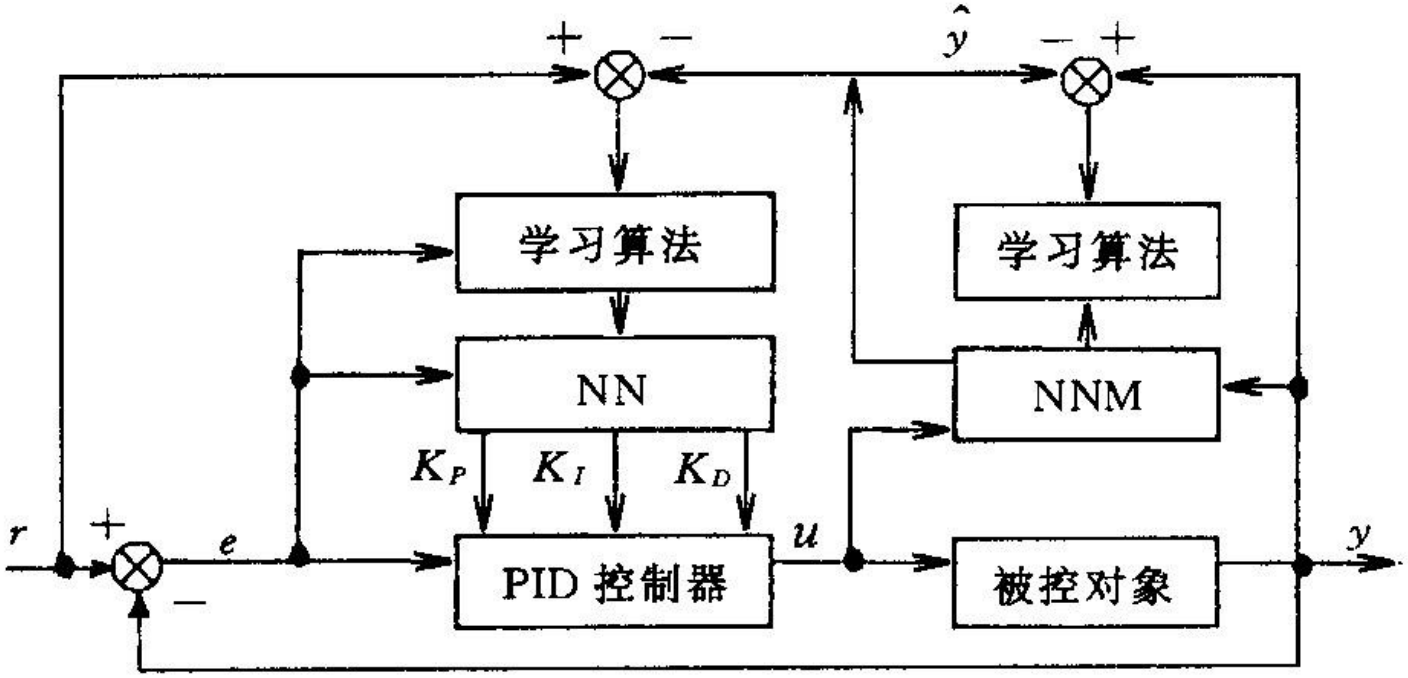
- 预报模型可用线性模型描述

$$A(z^{-1})y(k) = z^{-d}B(z^{-1})u(k) + v(k)$$



采用非线性预测模型

- 预报模型的非线性模型可以选择为三层BP 神经网络模型（NNM）



5、神经网络控制的应用

基于神经网络的板带材轧制过程自适应厚度控制

- 板带材的轧制过程是一个复杂的非线性过程。
- 厚度自动控制(**Automatic Gauge Control, AGC**)是轧制过程自动化的重要组成部分。
- 传统的**AGC**技术是从轧制过程所发生的基本现象出发，把轧机的弹性变形与轧件的塑性变形均近似为线性变形，并设计出线性控制器，但对于一些无法忽略的非线性和不确定性因素，不得不采取各种改进或补偿措施。
- 由于神经网络具有极强的处理非线性及不确定性问题的能力，神经网络用于轧制过程，能改善控制性能。

轧制过程的数学模型

- 轧制过程的基本数学模型
  - 轧机弹跳方程 $h = S_0 + \frac{F}{M} + \Sigma$
  - 轧制力方程 $F = \varphi(h, H, T, t, f, B, R)$

h —— 出口厚度；
S_0 —— 空载辊缝；
F —— 轧制力；
M —— 轧机模数；
Σ —— 补偿量(包括油膜补偿、热膨胀补偿等)；
H —— 入口厚度；
T、t —— 机架前、后张力；
f —— 轧辊与轧件之间的摩擦因数；
B —— 带材宽度；
R —— 轧辊半径；
φ —— 非线性函数，因建模方法不同而呈现不同形式

- 由于系统的非线性及不确定性，表示成一般的非线性模型

$$h(k) = \psi(h(k\text{-}1), H(k), H(k\text{-}1), S_0(k\text{-}1))$$

k —— 当前采样时刻

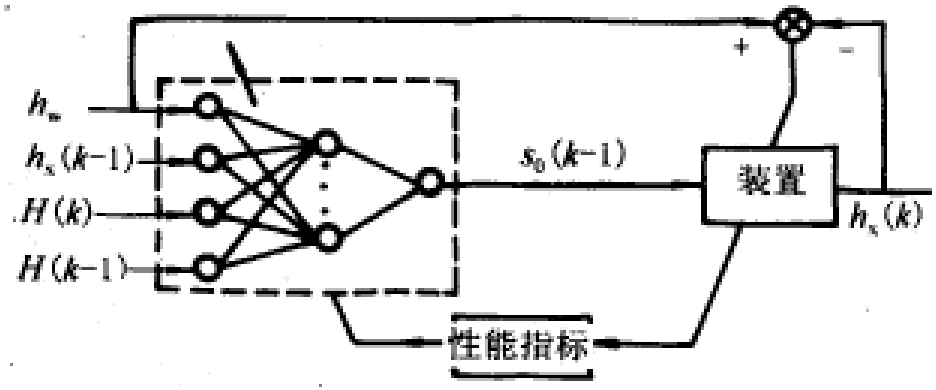
基于神经网络的自适应厚度控制(NNA-AGC)方案

- AGC的控制手段是调节空载辊缝S_0，采用直接神经网络控制，由轧制模型和轧制过程的时滞性，可知前馈神经网络控制器可表示为

$$S_0(k\text{-}1)=N(h(k\text{-}1),\ H(k),\ H(k\text{-}1),\ h_{ref}(k))$$

h_{ref} —— 期望的出口厚度

- NNA-AGC(neural networks based adaptive AGC)控制方案



网络设计及训练算法

- 适于过程控制的输出层网络激励函数

$$g(x)=k_v\left(\frac{1}{1+e^{-x}}-0.5\right)$$

k_v ——由所需要的控制信号幅度所决定

- 隐层的网络激励函数为 $f(x)=\frac{1}{1+e^{-x}}$

- **NNA-AGC系统中的网络学习训练用的偏差函数可定义为**

$$E_k=\frac{1}{2}\left(h_{ref}(k)-h(k)\right)^2$$

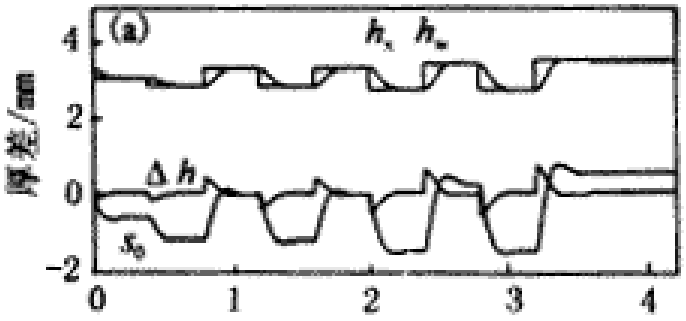
- 为加快学习速度，采用惯性**BP**学习算法

$$w_{ij}(k+1)=w_{ij}(k)-\eta\frac{\partial E_k}{\partial w_{ij}}+\beta\cdot\Delta w_{ij}(k)$$

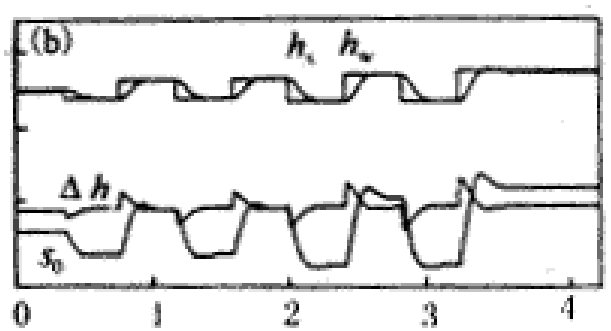
计算时要需要利用$\partial h(k)/\partial S_0(k\text{-}1)$，由于该值无法直接计算，因此，在实际控制中，根据轧制过程模型的性质，用符号$\mathrm{sgn}(\partial h(k)/\partial S_0(k\text{-}1))$代替$\partial h(k)/\partial S_0(k\text{-}1)$

NNA-AGC系统跟随性能

- 入口厚度基准值H_{REF} = 5.0 mm
- 学习率η = 0.1
- 惯性因数β = 0.05
- 期望的出口厚度为方波（仿真结果）

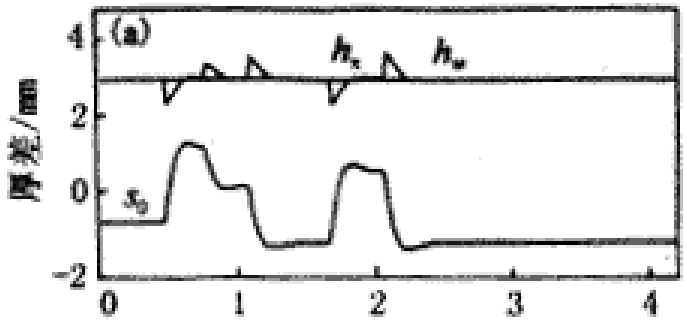


(a) 一般BP算法

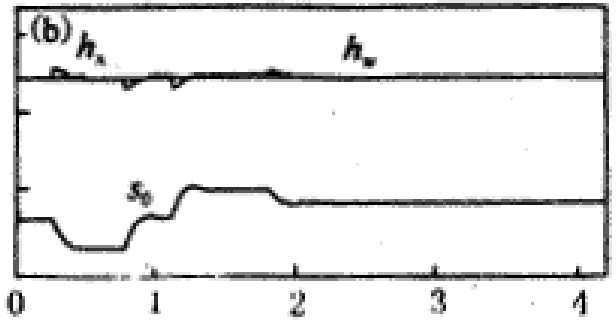


(b) 改进算法

ANN-AGC系统抗扰性能



(a)入口厚度波动时的抗扰调节过程



(b)轧件塑性系数波动时的抗扰调节过程

神经网络控制特点

- 新的神经网络控制结构不断出现；
- 每种神经网络控制结构各有优缺点；
  - 例，模型参考自适应不能保证稳定性；
  - 例，逆模控制技术需要系统的逆存在且稳定；
- 神经网络本身的泛化能力（外推的能力）有限，特定的神经网络控制结构一般均只能应用于具有特殊属性的控制对象类型；
- 神经网络控制需要训练权值，因此，需要大量样本数据，控制对象需要充分激励。

神经网络控制学习方法

- 经典神经网络控制多采用BP算法
- 理论性强的方法多采用Lyapunov技术
- 结合模糊逻辑的方法
- 采用计算智能的学习方法