第五章 人工神经网络
Artificial Neural Networks

邢文训
清华大学数学科学系
62787945
Email:wxing@tsinghua.edu.cn
答疑时间：周四下午4：00-5：00

发展历史

- 1890年——James的《心理学(Psychology(Briefer Course))》描述神经网络的基本原理：大脑皮层每一点的活力是由其他点势能释放的综合效能产生。这一势能同下面的因素有关：第一，相关其他点的兴奋次数；第二，兴奋的强度；第三，与其不相连的其他点所接受的能量。
- 1943年——McCulloch和Pitts建立了第一个人工神经网络模型，后被扩展为“认知(perceptron)”模型。
- 1969年——Minsky和Papert在《认知论(Perceptrons)》一书中指出：认知模型无法解决最为经典的“异或(XOR——exclusive-or)”问题。

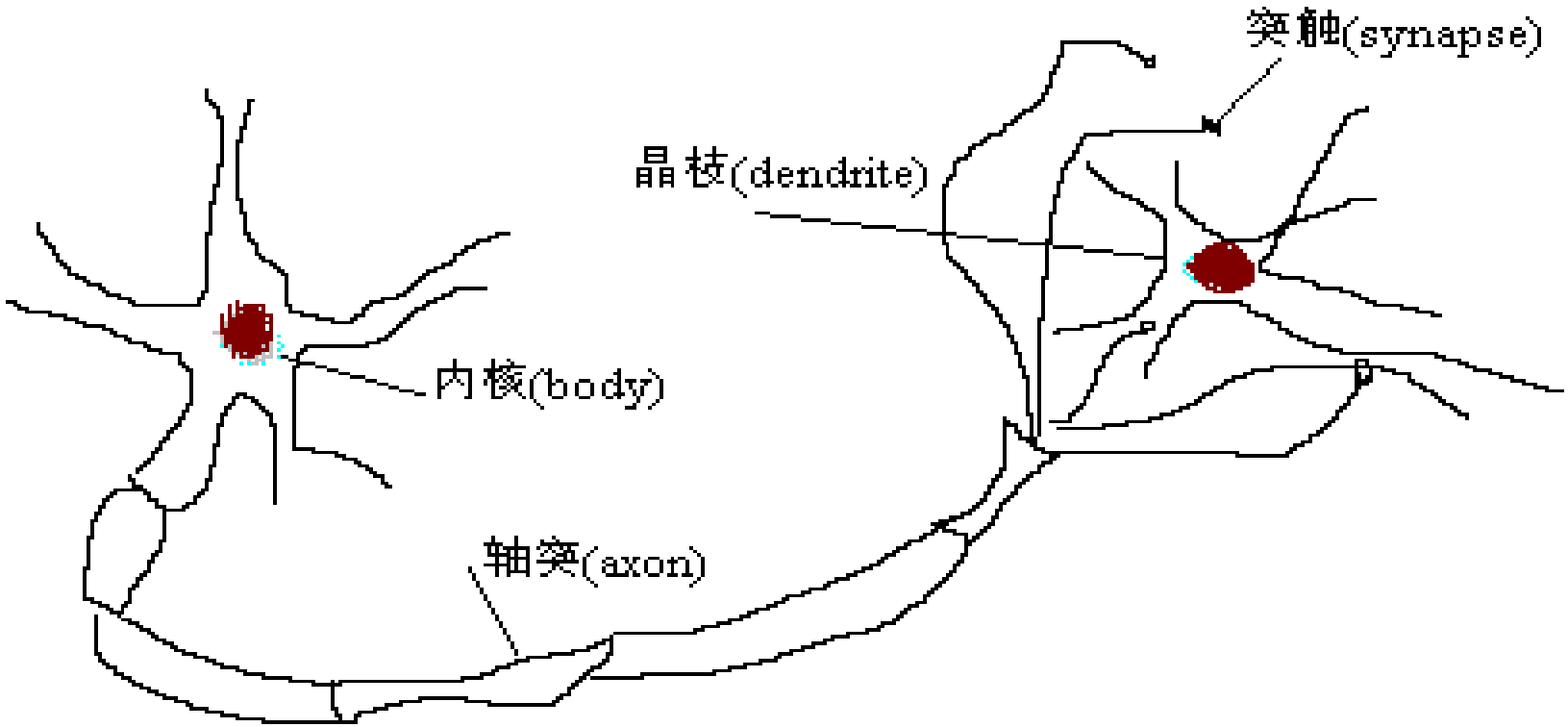
历史

- 二十世纪八十年代——Hopfield将人工神经网络成功地应用在组合优化问题
  - McClelland和Rumelhart构造的多层反馈学习算法成功地解决了单隐含层认知网络的“异或”问题及其他的识别问题。
- 2006年，深度学习再一次使ANN成为热点，2016年AlphaGo是一个成功应用标志。
- 2019 年3月27日，美国计算机协会(ACM)宣布了2018年的图灵奖获得者，人工智能科学家Yoshua Bengio、Geoffery Hinton 和 Yann LeCun共同获此荣誉。
  - Hinton, G. E., Osindero, S. and Teh, Y.,A fast learning algorithm for deep belief nets. Neural Computation 18:1527-1554, 2006
  - Yoshua Bengio, Pascal Lamblin, Dan Popovici and Hugo Larochelle, Greedy Layer Wise Training of Deep Networks, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 153-160, MIT Press, 2007
  - Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra and Yann LeCun Efficient Learning of Sparse Representations with an Energy-Based Model, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems (NIPS 2006), MIT Press, 2007

主要内容

- 人工神经网络的基本概念
- 单层前向神经网络
- 多层前向神经网络
- 竞争学习神经网络
- 反馈性Hopfield神经网络

5.1 人工神经网络的基本概念

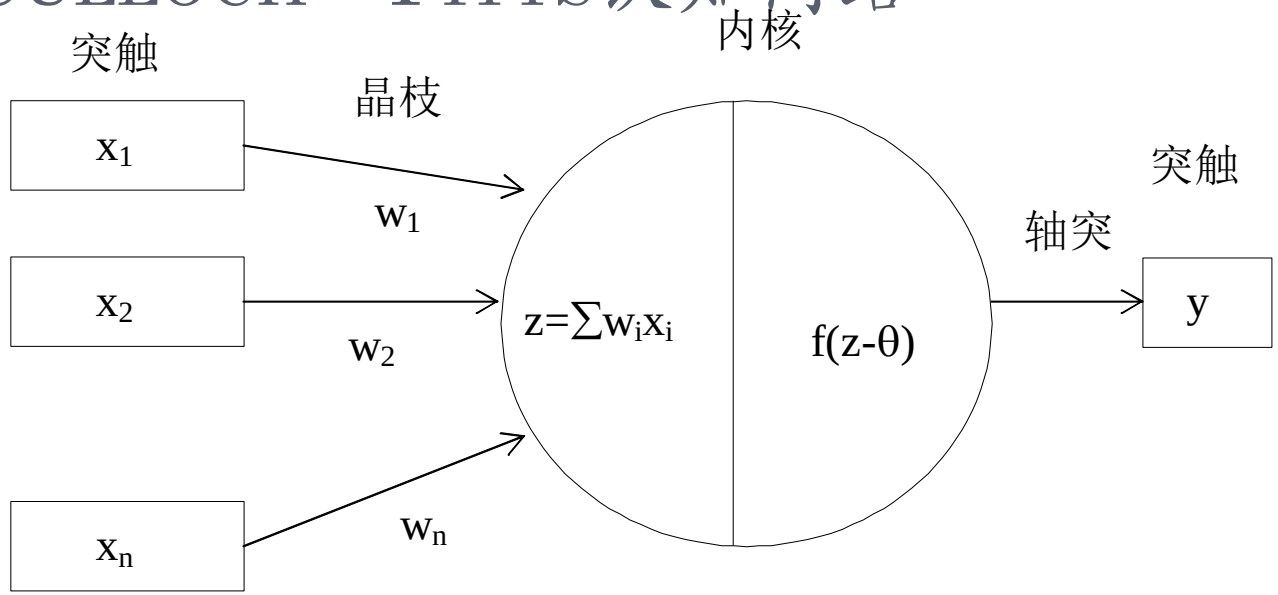


生物学中神经网络简图

工作原理

- 内核：接收到信号处理和发布信号。
- 轴突：传递信号渠道。
- 突触：发布信号终端。
- 晶枝：接收信号。

McCulloch—Pitts认知网络



w_i 为关联权，表示神经元对第 i 个晶枝接收到信息的感知能力。f 称为输出函数或激活函数(activation function)，$y = f(z-\theta)$ 为输出神经元的输出值。McClulloch—Pitts 输出函数定义为

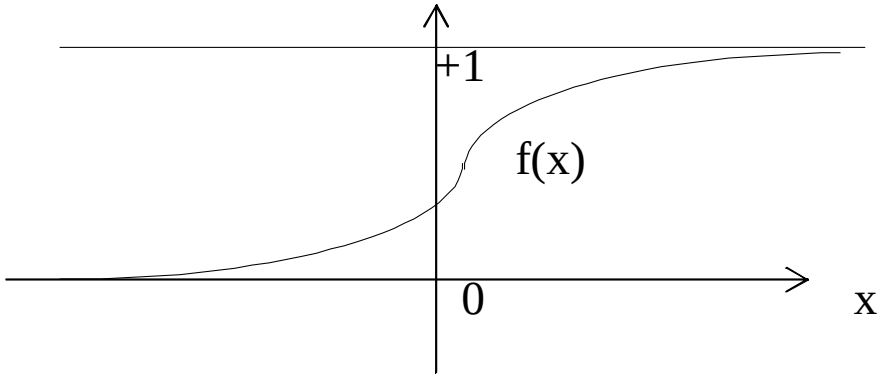
$$y = f(z-\theta) = \mathrm{sgn}\left(\sum_{i=1}^{n} w_i x_i - \theta\right), \quad \mathrm{sgn}(x) = \begin{cases} 1, x \geq 0, \\ 0, \text{其他}, \end{cases}$$

$\theta \geq 0$ 时，θ 称为阈值；$\theta < 0$ 时，$-\theta$ 称为神经元的外部刺激值；一般称 θ 为神经元的激励值。

建立和应用的三个步骤

- 第一个步骤：网络结构的确定
  - 网络的拓扑结构
    - 前向型人工神经网络
      - 神经元分为层，每一层内的神经元之间没有信息交流，信息由后向前一层一层地传递。
    - 反馈型神经网络
      - 整个网络看成一个整体，神经元相互作用 。
  - 神经元激活函数的选取
    - 阶跃函数——sgn(x)
    - 线性函数 ——ax+b
    - S(Sigmiod)函数

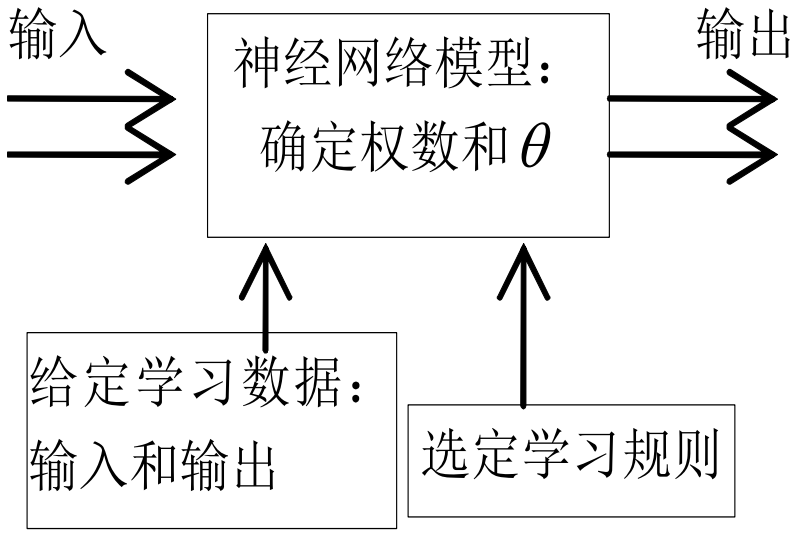
$$f(x)=\frac{1}{1+e^{-x}}$$



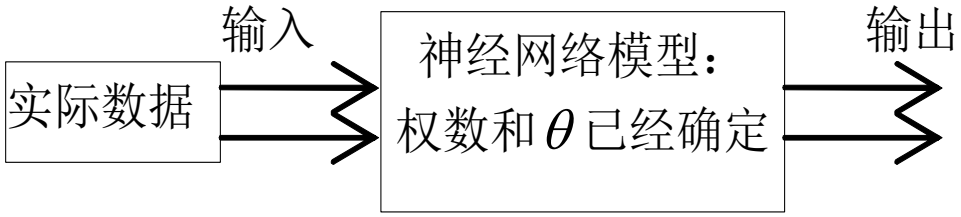
- 第二个步骤：关联权的确定
  - 权w_i和θ是通过学习（训练, train）得到
  - 分有指导学习和无指导学习两类
  - 梯度下降等方法(学习规则)
- 在已知一组正确的输入输出结果的条件下，人工神经网络依据这些数据，调整并确定权数w_i和θ，使得网络输出同理想输出偏差尽量小的方法称为有指导学习。
- 在只有输入数据而不知输出结果的前提下，确定权数的方法称无指导学习。
- 不同的目标函数得到不同的学习规则。

- 第三个步骤：工作阶段
  - 在权数和激活函数确定的基础上，用带有确定权数的神经网络去解决实际问题的过程称为工作。
- 人工神经网络计算过程——两个阶段

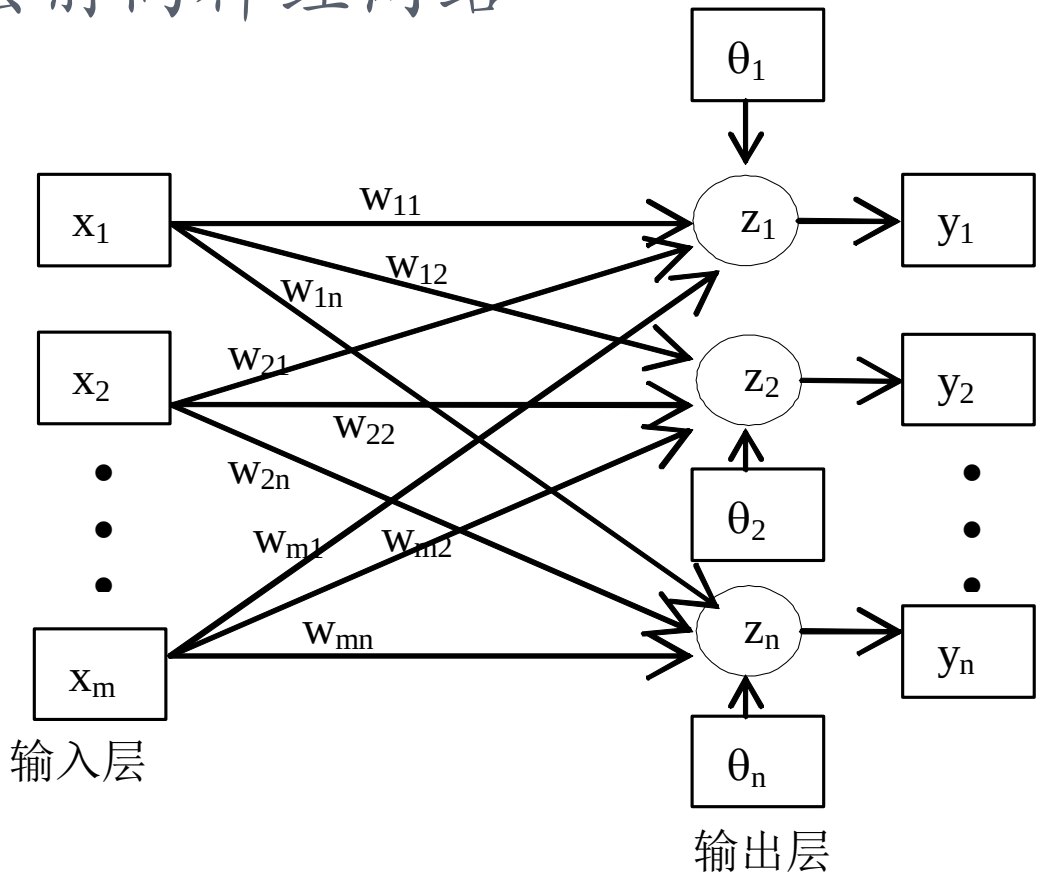
学习阶段



工作阶段



5.2 单层前向神经网络



线性网络 $Y = W^T X$

一致形式：虚拟一个对应阈值的神经元

$$y_j = z_j - \theta_j = \sum_{i=1}^{m} w_{ij} x_i - \theta_j$$

$$Y = \begin{pmatrix} w_{11} & \cdots & w_{m1} & \theta_1 \\ w_{12} & \cdots & w_{m2} & \theta_2 \\ & \cdots & \cdots & \\ w_{1n} & \cdots & w_{mn} & \theta_n \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_m \\ -1 \end{pmatrix}$$

$$Y = W^T X$$

XOR问题——单层线性网络无法正确识别

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

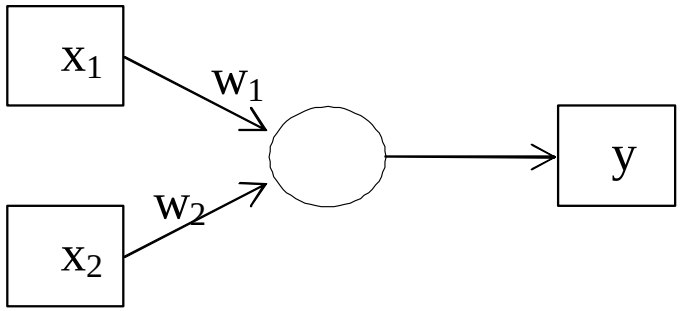


图 6.2.1 异或问题网络

矛盾方程

$$\begin{cases} w_2 = 1, \\ w_1 = 1, \\ w_1 + w_2 = 0. \end{cases}$$

学习规则——优化函数

整体学习

学习样本 $\{(X(1),D(1)),(X(2),D(2)),\cdots,(X(J),D(J))\}$

目标函数（最小二乘） $F(W)=\frac{1}{J}\sum_{t=1}^{J}(Y(t)-D(t))^T(Y(t)-D(t))$

求解过程 $$\frac{\partial F(W)}{\partial W}=2(\overline{XX}^T W-\overline{XD}^T)=0,$$

结果 $$W=(\overline{XX}^T)^{-1}\overline{XD}^T.$$

数学结果完美，但不同于人类学习模式

学习规则——单样本学习

- 当学习样本一个个地到达，每到达一个样本，系统将学习并更新权数。

样本 $X=(x_1,x_2,\cdots,x_m)^T \quad D=(d_1,d_2,\cdots,d_n)^T$

极小化目标函数 $F(W)=\sum_{j=1}^{n}\left(\sum_{l=1}^{m}w_{lj}x_l-d_j\right)^2$

求解梯度 $\frac{\partial F(W)}{\partial w_{ij}}=2x_i\left(\sum_{l=1}^{m}w_{lj}x_l-d_j\right)=2(y_j-d_j)x_i$

权数修正量 $\delta w_{ij}=-\frac{\varepsilon}{2}\frac{\partial F(W)}{\partial w_{ij}}=\varepsilon(d_j-y_j)x_i, i=1,2,\cdots,m;\ j=1,2,\cdots,n,$

修正结果 $W(t+1)=W(t)+\delta W(t)=W(t)+\varepsilon_t\left((d_j(t)-y_j(t))x_i(t)\right)_{m\times n}$

收敛要求 $\sum_{t=1}^{\infty}\varepsilon_t=\infty, \qquad \sum_{t=1}^{\infty}\varepsilon_t^{\,2}<\infty$

以大数据的观点：整体学习与单样本学习

- 整体学习：最优解一次性获得，精准。但需要J个样本一次性输入。
- 单样本学习：每次自用一个样本，用到的计算信息有$(x_1, x_2, \ldots, x_m)$, $y=W^Tx$, $(d_1, d_2, \ldots, d_n)$。每一步的计算快，W收敛速度慢。
- 当样本数量J大的时候，计算机无法存储或计算，此时可利用单样本学习，W最优解的精度可能差！！！

与数学优化的关系

- 决策变量为W. 依据数据(*x*,*d*)到达方式，选取优化问题。
- 对应一个优化问题 $F(W)=\sum_{j=1}^{n}\left(\sum_{l=1}^{m}w_{lj}x_l-d_j\right)^2$
- 求解W的过程，对应学习规则。

$$\frac{\partial F(W)}{\partial w_{ij}}=2x_i\left(\sum_{l=1}^{m}w_{lj}x_l-d_j\right)=2(y_j-d_j)x_i$$

- 依据学习规则，研究最后得到解W*的理论性质。

$$W(t+1)=W(t)+\delta W(t)=W(t)+\varepsilon_t\left((d_j(t)-y_j(t))x_i(t)\right)_{m\times n}$$

$$\sum_{t=1}^{\infty}\varepsilon_t=\infty,\qquad \sum_{t=1}^{\infty}{\varepsilon_t}^2<\infty$$

LMS应用在自适应线性（Adaline，Adaptive Linear）网络得到的学习规则称为Widrow-Hoff学习规则，也称Hebb规则。

LMS、Hebb规则、Adaline规则有时混称。它们可以统称为误差修正规则。

定理 当 $X=(X(1),X(2),\cdots,X(m))$ 为标准正交向量时，若 W(0)=0，通过学习规则(Hebb 规则的变形)

$$\delta w_{ij}(t)=\varepsilon_t d_j(t)x_i(t), i=1,2,\cdots,m;\ j=1,2,\cdots,n,$$

其中，$D=(D(1),D(2),\cdots,D(m))$ 为理想输出，学习 m 次，则单层线性神经网络可以区分这 m 个向量，且对输入 $X(t)=(x_1(t),x_2(t),\cdots,x_m(t))^T(t=1,2,\cdots,m)$，认知为 $\varepsilon_t D(t)$。

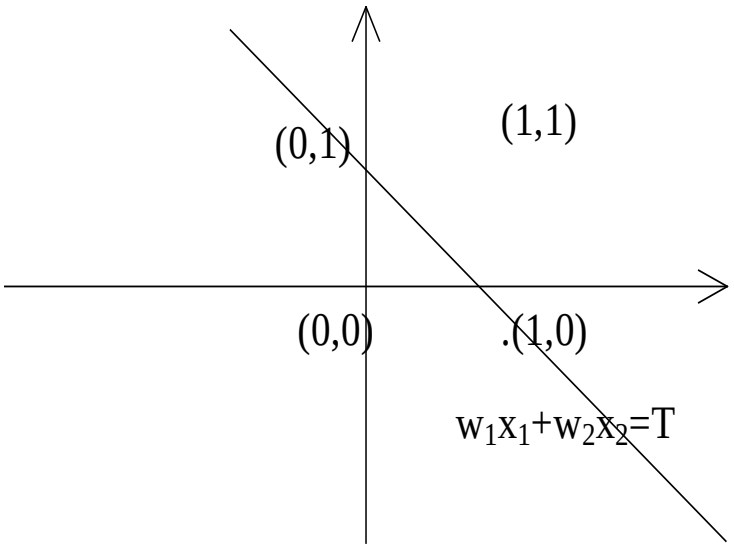
阶跃非线性单层网络

- 单层
- 激活函数：阶跃函数
  - f(z-T)=1, z-T>0; f(z-T)=0, z-T<=0.

非线性网络

- 采用符号函数的单层非线性神经网络同样无法实现XOR功能

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



$$\begin{cases} w_1 + w_2 \le T, \\ w_2 > T, \\ w_1 > T, \end{cases}$$

学习规则

给定输入X(t)，阈值θ(t)，理想输出D(t)，权数W(t)，

得到计算输出 $Y(t)=\mathrm{sgn}(W^T(t)X(t)-\theta(t))$

权数修正

$$W(t+1)=\varepsilon_t X(t)\{D(t)-Y(t)\}^T+W(t)$$

阈值修正

(1) 当 $d_j(t)=y_j(t)$ 时，表明第 j 个神经元对第 t 组输入 X(t)达到了理想输出，此时，$w_{ij}(t+1)=w_{ij}(t),i=1,2,\cdots,m$，即与神经元 j 相关的权数不修改。此时神经元 j 的激励值也不修改；

$$W(t+1)=\varepsilon_t X(t)\{D(t)-Y(t)\}^T+W(t)$$

(2) 当$d_j(t)=1, y_j(t)=0$时，表明第 j 个神经元对第 t 组输入 X(t)没有达到理想输出，此时有

$$z_j(t)=\sum_{i=1}^{m} w_{ij}(t)x_i(t)<\theta_j(t),$$

修改权数使得

$$z_j(t+1)=\sum_{i=1}^{m} w_{ij}(t+1)x_i(t)=\varepsilon_t\sum_{i=1}^{m} x_i^2(t)+\sum_{i=1}^{m} w_{ij}(t)x_i(t)\geq z_j(t),$$

激励值修改为

$$\theta_j(t+1)=\theta_j(t)-\varepsilon_t;$$

$$W(t+1)=\varepsilon_t X(t)\{D(t)-Y(t)\}^T+W(t)$$

（3）当$d_j(t)=0, y_j(t)=1$时，表明第 j 个神经元对第 t 组输入 X(t)没有达到理想输出，此时有

$$z_j(t)=\sum_{i=1}^{m} w_{ij}(t)x_i(t)\geq\theta_j(t),$$

修改权数使得

$$z_j(t+1)=\sum_{i=1}^{m} w_{ij}(t+1)x_i(t)=-\varepsilon_t\sum_{i=1}^{m} x_i^2(t)+\sum_{i=1}^{m} w_{ij}(t)x_i(t)\leq z_j(t),$$

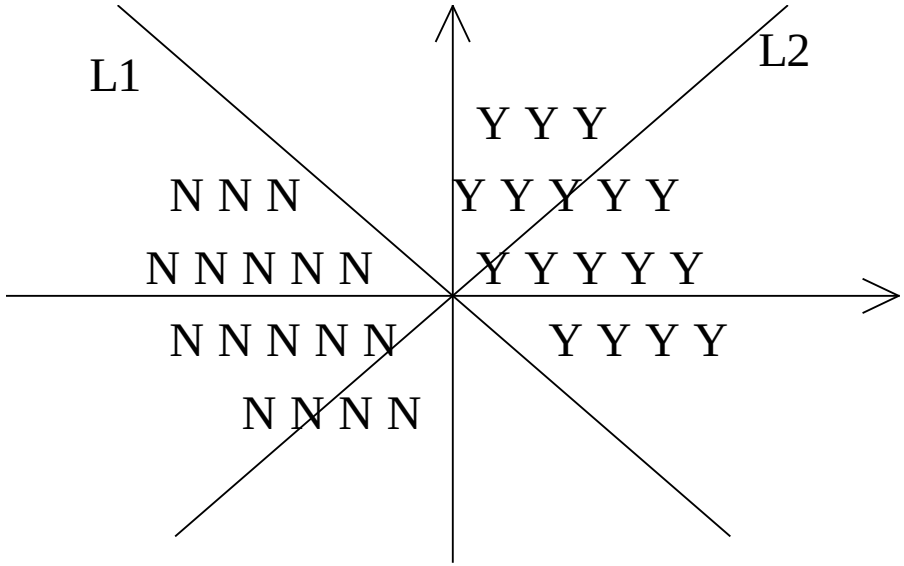
将激励值修改为

$$\theta_j(t+1)=\theta_j(t)+\varepsilon_t\text{。}$$

线性可分性

记 $\overline{X(t)} = (X(t), -1)^T$ $\overline{W(t)} = (W(t), \theta(t))^T$

$$u(t) = (u_1(t), u_2(t), \cdots, u_n(t))^T = (\overline{W(t)})^T \overline{X(t)}$$



通过权数的修正，可以区别已经线性可分的数据

对数学优化的挑战

- 不同的数据信息考虑不同的数学模型
  - 如整体到达、个体到达用最小二乘目标函数。
- 数据随机到达：期望模型、不确定模型、鲁棒优化模型。
- 对W结构的特殊要求：如希望W稀疏（关联性少）等。
- 不同的输出结果评价标准：$l_p(0<p<1)$范数，1-范数，2范数等模型。

单阶段学习——线性单阶段模型：I

- 线性模型：y=Ax+b,其中A为n阶矩阵, x为n维向量。
- 一个阶段：学习样本$(x^1, d^1), (x^2, d^2),\ldots, (x^K, d^K)$。学习规则（最小二乘）：

$$\min_{A,b} \sum_{i=1}^{K} (y^i - d^i)^T (y^i - d^i)$$

- 对大脑权数（A，b）有特殊要求：系数有一定的稳定性，A的特征值越小越好，b的2范数越小越好。给定恒定误差δ，优化问题

$$\min \quad \| A \|_2^2 + \| b \|_2^2$$

$$s.t \quad \sum_{i=1}^{K} (y^i - d^i)^T (y^i - d^i) \le \delta$$

单阶段学习——线性单阶段模型：II

- 希望A的秩越小越好（表示输出与A的最少的列相关），||A||*表示A的核范数。

$$\min \quad \|A\|_* + \|b\|_2^2$$

$$s.t \quad \sum_{i=1}^{K}(y^i - d^i)^T(y^i - d^i) \le \delta$$

- 希望A，b中的非零数越少越好（稀疏问题）。

$$\min \quad \sum_{j=1}^{n}\|A_{\cdot j}\|_1 + \|b\|_1$$

$$s.t \quad \sum_{i=1}^{K}(y^i - d^i)^T(y^i - d^i) \le \delta$$

单阶段学习——非线性单阶段模型：III

- 非线性模型：y=f(Ax+b),其中A为n阶矩阵, x为n维向量。
- 一个阶段：学习样本$(x^1, d^1), (x^2, d^2),\ldots, (x^K, d^K)$。学习规则（最小二乘）：

$$\min_{A,b} \sum_{i=1}^{K} (y^i - d^i)^T (y^i - d^i)$$

  - 求解方法：梯度下降方法。参考多层神经网络学习的BP算法。
  - 对A和b有要求怎么办？

多阶段强化学习——I

- 线性模型：y=Ax+b,其中A为n阶矩阵, x为n维向量。
- T个阶段：样本D={$d^i \in R^n \mid i=1,2,\ldots,T$}, 系统输出 $y^1=d^1$, $y^i=Ay^{i-1}+b$, $i=2,3,\ldots,T$.
- 基本学习模型：

$$\min_{A,b} \sum_{i=2}^{T} (y^i - d^i)^T (y^i - d^i)$$

- 对大脑权数（A，b）有特殊要求：系数有一定的稳定性，A的特征值越小越好，b的2范数越小越好。给定恒定误差δ，优化问题

$$\min \quad \| A \|_2^2 + \| b \|_2^2$$

$$s.t \quad \sum_{i=2}^{T} (y^i - d^i)^T (y^i - d^i) \le \delta$$

多阶段强化学习——II

- 样本$D=\{d^i \in R^n \mid i=1,2,\ldots,T\}$,二次函数模型：$y^1=d^1$, $y^i=(y^{i-1})^T A y^{i-1}+B y^{i-1}+c$, $i=2,3,\ldots,T$, 其中$A \in R^{n\times n\times n}$, $B \in R^{n\times n}$，$c \in R^n$
- 学习规则：

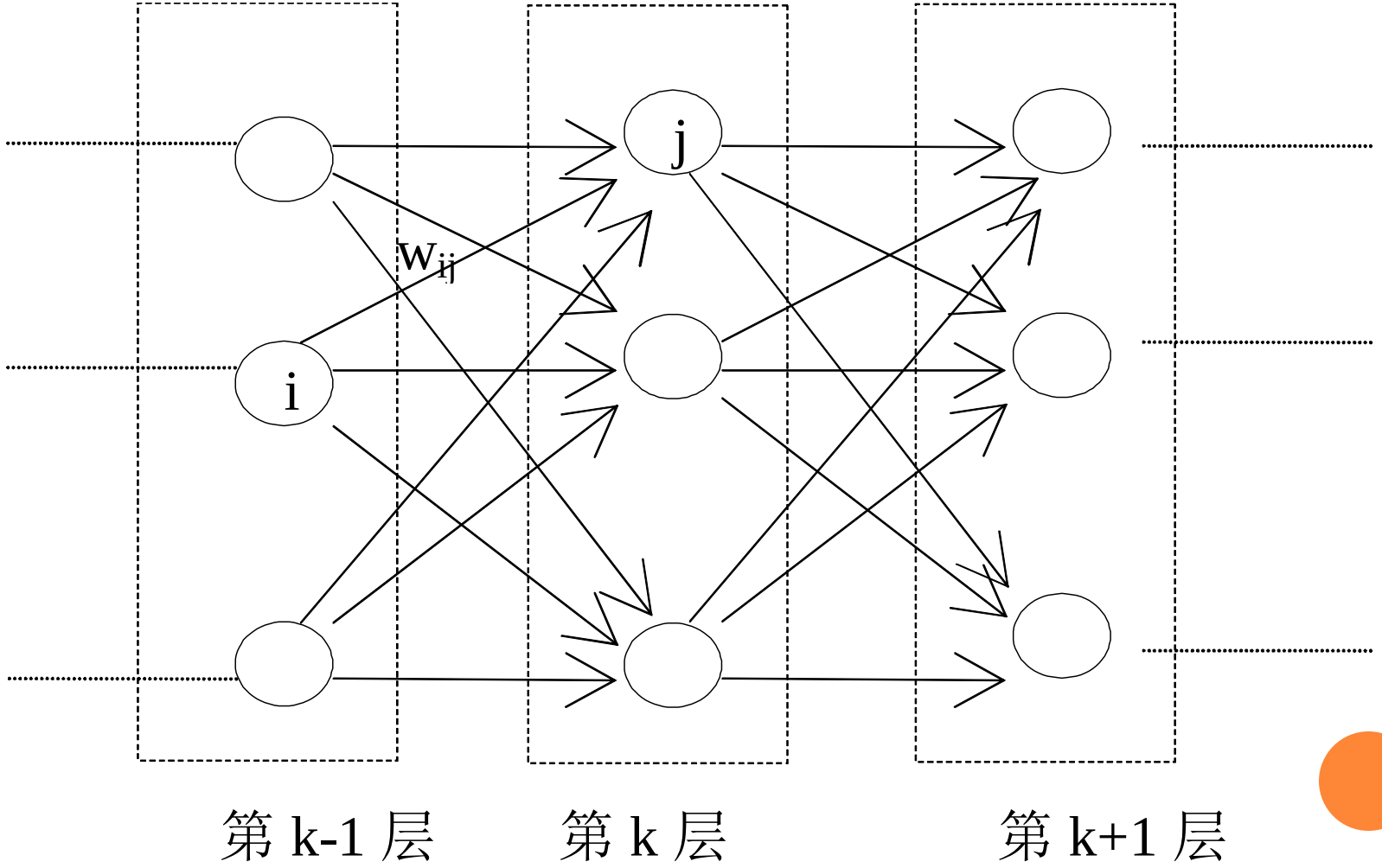
$$\min_{A,b} \sum_{i=2}^{T} (y^i - d^i)^T (y^i - d^i)$$

- 对A,B,c特殊要求如何建模和求解？

$$\min \quad \|A\|_2^2 + \|B\|_2^2 + \|c\|_2^2$$

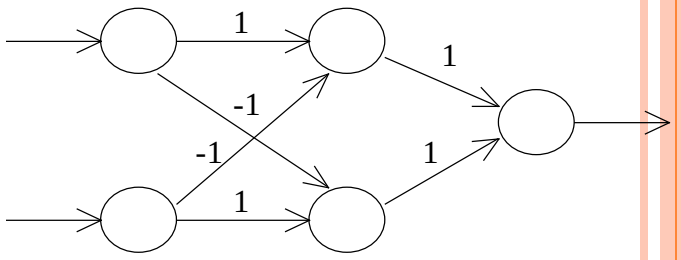
$$s.t \quad \sum_{i=2}^{T} (y^i - d^i)^T (y^i - d^i) \le \delta$$

5.3 多层前向神经网络



XOR的实现

- 线性网络等价单层网络
- 非线性神经网络
  - 激活函数采用符号函数，阈值为0



$$X=\begin{pmatrix}1\\0\end{pmatrix}$$

$$Z_1=W_1^T\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\-1\end{pmatrix},\ Y_1=f(Z_1)=\begin{pmatrix}1\\0\end{pmatrix},$$

$$Z_2=W_1^TY_1=1,\ Y_2=f(Z_2)=1.$$

$$X=\begin{pmatrix}0\\1\end{pmatrix}$$

$$Z_1=W_1^T\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}-1\\1\end{pmatrix},\ Y_1=f(Z_1)=\begin{pmatrix}0\\1\end{pmatrix},$$

$$Z_2=W_1^TY_1=1,\ Y_2=f(Z_2)=1.$$

$$X=\begin{pmatrix}1\\1\end{pmatrix}$$

$$Z_1=W_1^T\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix},\ Y_1=f(Z_1)=\begin{pmatrix}0\\0\end{pmatrix},$$

$$Z_2=W_1^TY_1=0,\ Y_2=f(Z_2)=0.$$

$$X=\begin{pmatrix}0\\0\end{pmatrix}$$

$$Z_1=W_1^T\begin{pmatrix}0\\0\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix},\ Y_1=f(Z_1)=\begin{pmatrix}0\\0\end{pmatrix},$$

$$Z_2=W_1^TY_1=0,\ Y_2=f(Z_2)=0.$$

BP算法

- 符号函数的多层前向神经网络有很强的分类功能，如对XOR问题的识别，但因没有很好的学习方法，故在实际问题中应用较少。
- 采用线性函数的多层前向神经网络，由于同单层线性神经网络的功效等价，因此，无法解决复杂问题的分类或识别问题。
- S形或线性函数有很好的函数特性，其效果又近似符号函数，因此现主要讨论采用S形和线性函数的前向多层神经网络的学习方法。

BP算法修正权数的简单思想

- 假设一个K层神经网络每一层的输入输出关系如下：从第0层到第一层的原始输入向量、权矩阵、接收值向量和输出向量及它们之间的关系分别为

$$X=\begin{pmatrix}x_1\\x_2\\\vdots\\x_{n_0}\end{pmatrix},\ W_1=\left(w_{ij}^1\right)_{n_0\times n_1},\ Z_1=\begin{pmatrix}z_1^1\\z_2^1\\\vdots\\z_{n_1}^1\end{pmatrix}=W_1^T X,\ Y_1=\begin{pmatrix}y_1^1\\y_2^1\\\vdots\\y_{n_1}^1\end{pmatrix},$$

- 第k-1层到第k层的权矩阵、接受值向量和输出向量及它们之间的关系分别为

$$W_k=\left(w_{ij}^k\right)_{n_{k-1}\times n_k},\ Z_k=\begin{pmatrix}z_1^k\\z_2^k\\\vdots\\z_{n_k}^k\end{pmatrix}=W_k^T Y_{k-1},\ Y_k=\begin{pmatrix}y_1^k\\y_2^k\\\vdots\\y_{n_1}^k\end{pmatrix},$$

$$y_i^k=f_i(z_i^k), i=1,2,\cdots,n_i, k=1,2,\cdots,K$$

- 单样本学习规则。学习的原则是：确定W，极小化

$$F(W)=(D-Y_K)^T(D-Y_K)$$

- 求解梯度

$$\frac{\partial F}{\partial w_{ij}^K}=-2(d_j-y_j^K)\frac{\partial y_j^K}{\partial w_{ij}^K},$$

$$\frac{\partial y_j^K}{\partial w_{ij}^K}=\frac{dy_j^K}{dz_j^K}y_i^{K-1},\qquad i=1,2,\cdots,n_{K-1},\ j=1,2,\cdots,n_K.$$

当 $k\le K-2$

$$\frac{\partial F}{\partial w_{ij}^k}=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\sum_{l_{K-1}=1}^{n_{K-1}}w_{l_{K-1}l_K}^K\frac{dy_{l_{K-1}}^{K-1}}{dz_{l_{K-1}}^{K-1}}\cdots$$

$$\sum_{l_{k+2}=1}^{n_{k+2}}w_{l_{k+2}l_{k+3}}^{k+3}\frac{dy_{l_{k+2}}^{k+2}}{dz_{l_{k+2}}^{k+2}}\sum_{l_{k+1}=1}^{n_{k+1}}w_{l_{k+1}l_{k+2}}^{k+2}\frac{dy_{l_{k+1}}^{k+1}}{dz_{l_{k+1}}^{k+1}}w_{jl_{k+1}}^{k+1}\frac{dy_j^k}{dz_j^k}y_i^{k-1},$$

$$i=1,2,\cdots,n_{k-1};\ j=1,2,\cdots,n_k,$$

$$\frac{\partial F}{\partial w_{ij}^{K-1}}=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{\partial y_{l_K}^K}{\partial w_{ij}^{K-1}}$$

$$=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\frac{\partial z_{l_K}^K}{\partial w_{ij}^{K-1}}$$

$$=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\sum_{l_{K-1}=1}^{n_{K-1}}w_{l_{K-1}l_K}^K\frac{\partial y_{l_{K-1}}^{K-1}}{\partial w_{ij}^{K-1}}$$

$$=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}w_{jl_K}^K\frac{dy_j^{K-1}}{dz_j^{K-1}}y_i^{K-2},$$

$$i=1,2,\cdots,n_{K-2};\ j=1,2,\cdots,n_{K-1}.$$

矩阵表示

$$B_{K+1}=\begin{pmatrix} d_1-y_1^K \\ d_2-y_2^K \\ \vdots \\ d_{n_K}-y_{n_K}^K \end{pmatrix},\quad W_{K+1}=I(\text{单位阵})$$

$$F(W)=(B_{K+1})^T B_{K+1}$$

$$B_K=diag(\frac{dy_1^K}{dz_1^K},\frac{dy_2^K}{dz_2^K},\cdots,\frac{dy_{n_K}^K}{dz_{n_K}^K})W_{K+1}B_{K+1}$$

$$\left(\frac{\partial F(W)}{\partial w_{ij}^K}\right)_{n_{K-1}\times n_K}=-2\begin{pmatrix} y_1^{K-1} \\ y_2^{K-1} \\ \vdots \\ y_{n_{K-1}}^{K-1} \end{pmatrix}(B_K)^T$$

$$B_{K-1}=diag(\frac{dy_1^{K-1}}{dz_1^{K-1}},\frac{dy_2^{K-1}}{dz_2^{K-1}},\cdots,\frac{dy_{n_{K-1}}^{K-1}}{dz_{n_{K-1}}^{K-1}})W_K B_K$$

$$\left(\frac{\partial F(W)}{\partial w_{ij}^{K-1}}\right)_{n_{K-2}\times n_{K-1}}=-2\begin{pmatrix} y_1^{K-2} \\ y_2^{K-2} \\ \vdots \\ y_{n_{K-1}}^{K-2} \end{pmatrix}(B_{K-1})^T$$

$$B_k=diag(\frac{dy_1^k}{dz_1^k},\frac{dy_2^k}{dz_2^k},\cdots,\frac{dy_{n_k}^k}{dz_{n_k}^k})W_{k+1}B_{k+1}$$

$$\left(\frac{\partial F(W)}{\partial w_{ij}^k}\right)_{n_{k-1}\times n_k}=-2\begin{pmatrix} y_1^{k-1} \\ y_2^{k-1} \\ \vdots \\ y_{n_{k-1}}^{k-1} \end{pmatrix}(B_k)^T$$

学习规则与BP算法

$$W_k(t+1) = W_k(t) + \delta W_k(t), \quad k = K, K-1, \cdots, 1,$$

$$\delta W_k(t) = -\frac{1}{2}\ \varepsilon_t \left(\frac{\partial F(W)}{\partial w_{ij}^k}(t) \right)_{n_{k-1} \times n_k} = \varepsilon_t \begin{pmatrix} y_1^{k-1}(t) \\ y_2^{k-1}(t) \\ \vdots \\ y_{n_{K-1}}^{k-1}(t) \end{pmatrix} (B_k(t))^T$$

反推学习算法

STEP1 选定学习的数组$\{X(t), D(t)\}, t = 1,2,\cdots, J$, 随机确定初始权矩阵 W(0);

STEP2 用学习数据 X(t)计算$Y_1(t), Y_2(t), \cdots, Y_K(t)$;

STEP3 用学习规则，反向修正 W(t)；直到学习完所有的数组。

BP算法实现中需注意的细节

- 第一，激活函数为S形函数或线性函数。由线性网络的讨论，如果网络中的全部神经元都取线性函数，那么，同线性网络没有任何区别。
- 第二，当激活函数为S形函数时，输出值只能趋近0或1。于是，在神经网络的工作期，判定一个数组的归类问题就需要给出一个分界值，如0.5，而不是绝对的1或0。

第三，W(0)的选取最好是随机的。当 W(0)的所有权数相等时，由 $Z_k = (W_k)^T Y_{k-1}, k = 1,2,\cdots,K$ 推出：无论什么样的输入 X，Z_1 中的全部分量相同；当第一层所有神经元的激活函数取同一个函数时，Y_1 的全部分量相同；当每一层的神经元选相同激活函数时，由于 Z_k 的各分量相同，所以 $\frac{dy_1^k}{dz_1^k} = \cdots = \frac{dy_{n_k}^k}{dz_{n_k}^k}$，再运算得 $\left(\frac{\partial F(W)}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k}$ 中的所有元素相等。由此得到的修正权对很多问题无法识别。

第四，算法的全局最优性，即算法在什么样的条件下收敛？能否收敛到区局最优点？

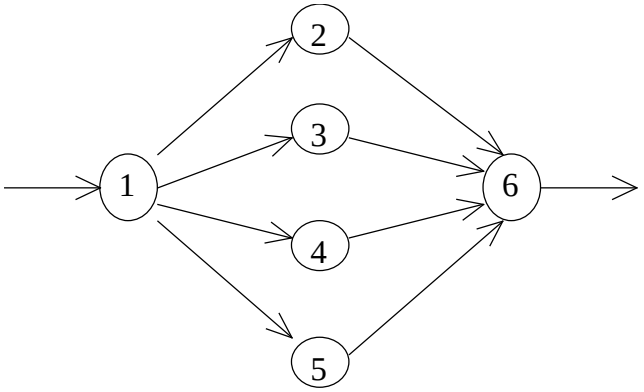
只要学习效率 ε_t 选取合适，BP 为目标下降算法，收敛到局部最优。为了避免局部最优解的一个改进方法是将学习规则中的 $\delta W_k(t)$ 修正为

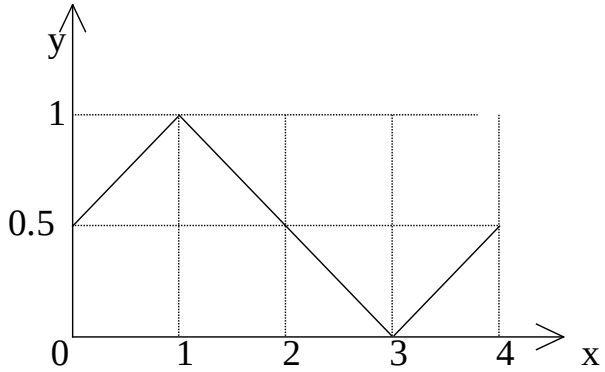
$$\delta W_k(t) = -\frac{1}{2}\varepsilon_t\left(\frac{\partial F(W)}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k} + \alpha_t \delta W_k(t-1),$$

右端的第二项相当于一个势能“惯性”，式中的 α_t 决定于权数两次变化的相互关系。

BP算法的一个示例

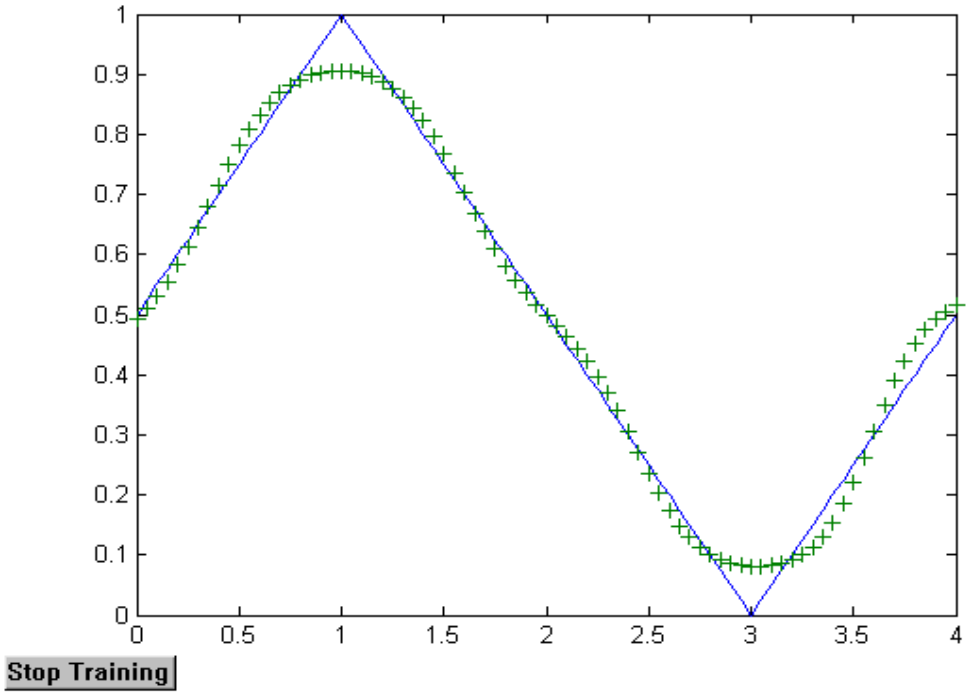
网络结构





示例

计算结果



\example12

一些技术问题

- 查手册获得不同的学习方法
- 比较不同学习方法的计算效率
- 一般能得到学习数据的较好的计算结果
- 应该将已有的数据分成学习数据和检验数据两部分
- 如果matlab命令中不用net=init(net); 上一次计算的权矩阵会被记录并重新使用，退出后或运行其他程序后才会被释放。

5.4 竞争学习神经网络

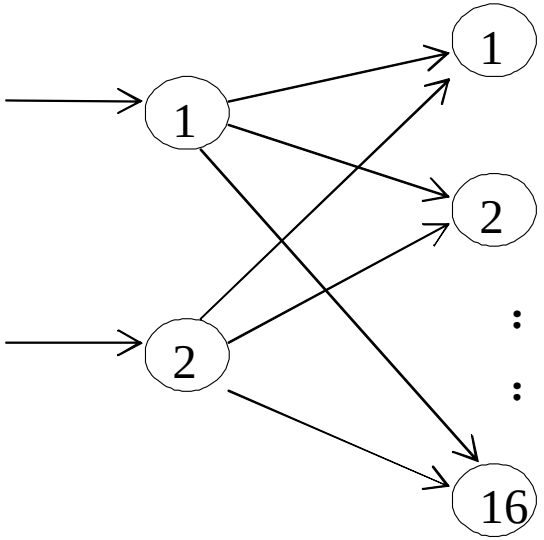
- 竞争学习神经网络是无监督神经网络中的一类。
- 给定输入数据，无监督学习不需要对应的理想输出。
- 它的输出结果表明输入数据中的相关或相似性。
- 一类简单的线性竞争学习模型

$$y_j = \begin{cases} 1, & \sum_{i=1}^{m} w_{ij} x_i = \max\limits_{1 \le k \le n} \left(\sum_{i=1}^{m} w_{ik} x_i \right), \\ 0, & \text{其他.} \end{cases}$$

学习规则是

$$w_{ij}(t+1) = w_{ij}(t) + \delta w_{ij}(t), \quad \delta w_{ij}(t) = y_j(t)(x_i(t) - w_{ij}(t))$$

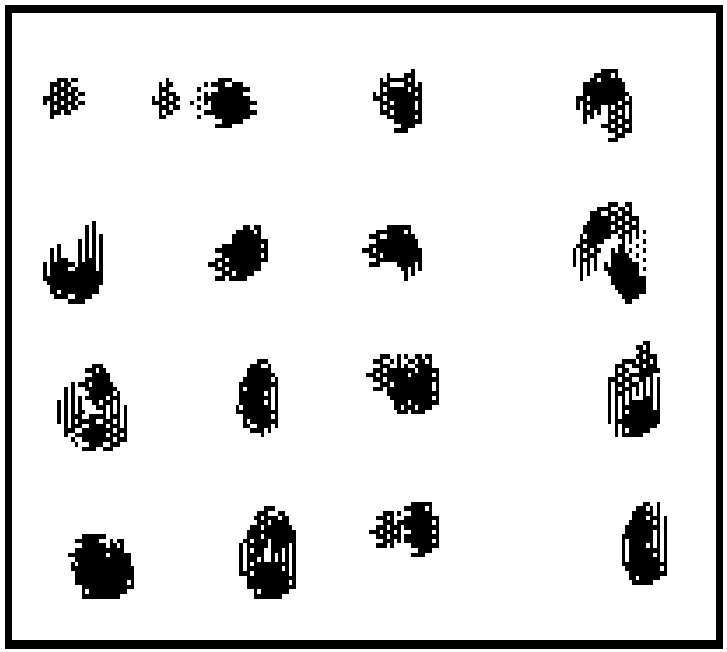
- 神经网络为



输入(x_1, x_2)服从[−1,1]的均匀分布，初始权数

$$W^{(j)} = \begin{pmatrix} w_{1j} \\ w_{2j} \end{pmatrix}$$ 在[−1,1]×[−1,1]随机选取。

通过24982次竞争学习后，竞争学习的权数模拟输出结果



深度学习(Deep learning)

- BP多层前向神经网络就是一种深度学习网络。
  - 只通过输入层和输出层的数据学习来确定隐层中的权数。
  - 计算效率可能不高。
- 改进
  - 对每一层有部分信息数据学习，类似强化学习。
  - 由于一般的多层中的权看成黑箱，待定的权数个数多，稀疏低秩的要求如何实现？
- 核心问题：优化模型的建立和求解（学习算法）。

- 2006后的深度学习概念。
  - 对隐层一层一层地进行训练，提取特征。
  - 有监督：如卷积神经网络
  - 无监督：如deep belief nets。
- 参考文献
  - Hinton, G. E., Osindero, S. and Teh, Y.,A fast learning algorithm for deep belief nets. Neural Computation 18:1527-1554, 2006
  - Yoshua Bengio, Pascal Lamblin, Dan Popovici and Hugo Larochelle, Greedy Layer Wise Training of Deep Networks, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 153-160, MIT Press, 2007
  - Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra and Yann LeCun Efficient Learning of Sparse Representations with an Energy-Based Model, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems (NIPS 2006), MIT Press, 2007

卷积神经网络简介

- 输入：灰度图像矩阵$X(l)=(x_{ij}(l))_{n\times n}$，$l=1,2,\ldots,L$, L样本量。 (n×n×k，彩色)。
- 给定池化样本：$Y(l)=(y_{ij}^{f}(l))_{p\times p,}$ f=1,2,…,q, q池化量
- 求解：滤子矩阵$K^{f}=(k_{ij}^{f})_{m\times m,}$ f=1,2,…,q.
- 满足关系：n>p, n>m.
- 卷积运算关系：图像矩阵与滤子矩阵
- 第一个算子：有效值卷积

$$z^{f}(u,v,l)=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}x_{i+u,j+v}(l)k_{i,j}{}^{f}\delta_{ij},\qquad \delta_{ij}=\begin{cases}1, & 1\le i,j\le m\\ 0, & others.\end{cases}$$

该矩阵将原有的图像规模X变成(n-m+1) × (n-m+1)的矩阵。当m规模较小时，存储没有实质性变化，通过加m-1行列边后，恢复n× n矩阵.

卷积神经网络简介（续）

- 第二个算子：池化(pooling)。目的是将

$$Z=(z^f(u,v,l))_{(n-m+1)\times(n-m+1)}$$

降维到$T=(t^f(i,j,l))_{p\times p}$与样本$Y=(y_{ij}^f(l))_{p\times p}$可比较，达到学习的目的。p最好远小于n。常用方法：均值法或最大值法。

- 第三个算子：学习，求K^f, $f=1,2,\ldots,q$.

$$\min_{K^f,\,f=1,2,\ldots,q}\sum_{l=1}^{L}\sum_{f=1}^{q}\sum_{i,j=1}^{p}(y_{ij}^f(l)-f(t^f(i,j,l))^2$$

一个例子

- n=4, m=2, q=1, L=1. 给定：样本输入X和池化输出Y，单层求滤子K。

$$X=\begin{pmatrix}1&1&1&0\\0&1&1&1\\0&0&1&0\\0&1&1&0\end{pmatrix}\qquad Y=\begin{pmatrix}2&2\\2&2\end{pmatrix}$$

- 给定K，滤子卷积运算：$K=\begin{pmatrix}0&1\\1&0\end{pmatrix}$

$$\begin{pmatrix}1\times0&1\times1&1&0\\0\times1&1\times0&1&1\\0&0&1&0\\0&1&1&0\end{pmatrix}\rightarrow\begin{pmatrix}1&\\&\end{pmatrix}\rightarrow\begin{pmatrix}1&2\\&\end{pmatrix}\rightarrow\ldots\rightarrow\begin{pmatrix}1&2&1\\1&1&2\\0&2&1\end{pmatrix}=Z$$

- 池运算：最大值法

$$\begin{pmatrix}1&2&1\\1&1&2\\0&2&1\end{pmatrix}\rightarrow\begin{pmatrix}\begin{pmatrix}1&2\\1&1\end{pmatrix}&\begin{pmatrix}1\\2\end{pmatrix}\\\begin{pmatrix}0&2\end{pmatrix}&1\end{pmatrix}\rightarrow\begin{pmatrix}2&2\\2&1\end{pmatrix}=T$$

一个例子（续）

- 池运算：平均值法 $\begin{pmatrix}1&2&1\\1&1&2\\0&2&1\end{pmatrix}\rightarrow\left(\begin{pmatrix}1&2\\1&1\end{pmatrix}\ \begin{pmatrix}1\\2\end{pmatrix}\right)\rightarrow\begin{pmatrix}5/4&3/2\\1&1\end{pmatrix}=T$

- 有样本学习：给定池化学习样本Y和激活函数f(x)

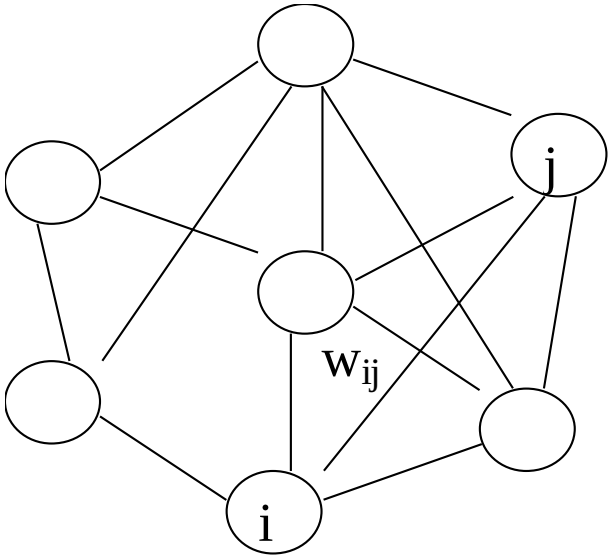
$$Y=\begin{pmatrix}2&2\\2&2\end{pmatrix}$$

$$\min_K \sum_{i,j=1}^{2}(2-f(t(i,j)))^2$$

- 最终得到K。神经网络学习过程结束，可以开始工作。

5.5 反馈型神经网络

- 反馈型神经网络(feed-back neural networks)的一般结构



模型

$$\frac{dz_i(t)}{dt} = -Az_i(t) + I_i(t) + \sum_{j=1}^{n} w_{ji} y_j(t),$$

$$y_i(t) = f_i(z_i(t)), \quad i = 1,2,\cdots,n,$$

一般性理论

动力系统微分方程组可以简记

$$\frac{dz(t)}{dt} = g(z,t) \qquad z(t) = (z_1(t), z_2(t), \cdots, z_n(t))^T$$

定义 如果 $g(z_e, t) = 0, \ \forall t,$

则称 z_e 是动力系统的平衡点，也称 z_e 为吸引子。

定义 满足下列条件的平衡点 z_e 称为稳定点：

任给 $\varepsilon > 0$ 和 $t_0 \geq 0$，存在 $\delta(\varepsilon, t_0) > 0$，当 $t \geq t_0$，$\|z_0 - z_e\| < \delta(\varepsilon, t_0)$ 时，有

$$\|z(t; z_0, t_0) - z_e\| < \varepsilon,$$

其中，$z(t; z_0, t_0)$ 表示以 z_0 为起始点，t_0 为起始时间满足微分方程的一条参数曲线。称动力系统在点 z_e 稳定。

定义 满足下列条件的稳定点 z_e 是渐近稳定点：

任给 $t_0 \geq 0$，存在 $\eta(t_0) > 0$，当 $\|z_0 - z_e\| < \eta(t_0)$ 时，

有 $\lim_{t\to\infty} z(t; z_0, t_0) = z_e$。称系统在点 z_e 渐近稳定。

定义 满足下列条件的点 z_0 组成的集合称为平衡点 z_e 的吸引域：

存在 $t_0 \geq 0$ 和 $\eta(t_0) > 0$，当 $\|z_0 - z_e\| < \eta(t_0)$ 时，有 $\lim_{t\to\infty} z(t; z_0, t_0) = z_e$。

反馈型神经网络的平衡点、平衡点的稳定性和吸引域等的性质的方法

- 研究动力系统和与它对应的Lyapunov能量函数的性质。
- 从物理学的角度来看有这样的现象：随着系统的运动，其储存的能量随时间的增长而衰减，直至趋于能量极小的平衡状态，这类动力系统称为耗散系统。
- 由此原理，反馈型神经网络的研究一般采用稳定性的Lyapunov第二方法，或称直接法，它首先给出微分方程和对应的能量函数或广义能量函数，或称Lyapunov函数，在研究函数的特性后，可以不用求解系统的运动方程，而给出系统平衡稳定性的信息。

HOPFIELD 的工作

- 利用模拟电路构造了反馈型人工神经网络的电路模型，建立的能量函数表达式为

$$E(y) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j + \sum_{i=1}^{n} \frac{1}{R'_i} \int_0^{y_i} f_i^{-1}(y)dy - \sum_{i=1}^{n} I_i y_i$$

$y_i = f_i(z_i)$, f_i 为 Sigmoid 或线性函数，w_{ij} 为电导参数，为 i 和 j 两个神经元的权数，具有对称性 $w_{ij} = w_{ji}$，R_i 是模拟电子线路电阻的大小，z_i 为第 i 个神经元的接受值，I_i 为外部偏置电流输入值，$\sum_{i=1}^{n} \frac{1}{R_i'} \int_0^{y_i} f_i^{-1}(y)dy$ 称为增益项。

$$\frac{1}{R_i'} = \frac{1}{R_i} + \sum_j w_{ij}$$

- 能量函数对应一个动力系统

$$\begin{cases} \dfrac{dz_i}{dt} = -A_i z_i + \sum_{j=1}^{n} w_{ij} y_j + I_i, \\ y_i = f_i(z_i), i = 1,2,\cdots,n, \end{cases}$$

$$A_i = \frac{1}{R_i'} \qquad \frac{1}{R_i'} = \frac{1}{R_i} + \sum_j w_{ij} \qquad w_{ij} = w_{ji}$$

且满足

$$-\frac{dz_i}{dt} = \frac{\partial E}{\partial y_i}$$

HOPFIELD神经网络的理论结果

定理 若能量函数正定（即对 $\forall z \neq z_e$，有 $E(z) > 0$），且满足 $\frac{dE(z)}{dt} \leq 0$，则系统的任何一个平衡点 z_e 是稳定的。

定理(Lasalle 不变原理) 若 S 是一个有界闭集，动力系统满足：$z(t_0; z_0, t_0) \in S$，都有 $z(t; z_0, t_0) \in S, t \geq t_0$，且 S 上存在能量函数 $E(z) \in C^1$，使得 $\frac{dE(z)}{dt} \leq 0, z \in S$，则 $z(t; z_0, t_0) \to Z_e = \{z \mid \frac{dE(z)}{dt} = 0, z \in S\}, t \to \infty$，当 Z_e 中只有一个点时，这一点是渐近稳定的。

HOPFIELD神经网络的计算步骤

STEP1 针对实际的组合优化问题构造能量函数，使得能量函数有好的稳定性，如满足上两个定理之一；

STEP2 由能量函数，根据Hopfield给出的关系求解出动力系统方程；

STEP3 用数值计算的方法求解动力系统方程的平衡点，具体的计算求解方法可通过自编程序或用现有的软件（如Matlab），由定理知平衡点是否为稳定点或渐近稳定，是否达到局部极小值。

HOPFIELD神经网络在TSP中的应用

- 解的表达形式
  - 用一个$n \times n$矩阵表示TSP的一个解，第i行表示商人到达该城市的顺序，第i行由0,1数字组成一个n维向量，其中只能有一个分量为1，1所在的序数表示商人到达的序数。

	1	**2**	**3**	**4**
城市1	0	1	0	0
城市2	0	0	1	0
城市3	1	0	0	0
城市4	0	0	0	1

表示商人行走的城市顺序为：3-1-2-4

神经网络的构造及求解

- 构造一个有$n \times n$个神经元的神经网络。对应(i,j)位置的神经元，接收到的值为在z_{ij}，其输出值为y_{ij}，激活函数采用Sigmoid函数。记两个城市x和y的距离是d_{ij}
- 期望每一行的和为1，希望下式为0。

$$E_1 = \sum_{u=1}^{n} \sum_{i=1}^{n} \sum_{j \neq i} y_{ui} y_{uj}$$

- 期望每一列的和为1，希望下式为0。

$$E_2 = \sum_{i=1}^{n} \sum_{u=1}^{n} \sum_{v \neq u} y_{ui} y_{vi}$$

- 保证每一行每一列正好有一个1，理想状态下式为0

$$E_3 = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} - n \right)^2$$

- 费用最小的最短路

$$E_4 = \sum_{u=1}^{n} \sum_{v \neq u} \sum_{i=1}^{n} d_{uv} y_{ui} (y_{vi-1} + y_{vi+1})$$

- 能量函数

$$E = \frac{A}{2} E_1 + \frac{B}{2} E_2 + \frac{C}{2} E_3 + \frac{D}{2} E_4 + \alpha E_5$$

$$E_5 = \sum_{i,j} \int_0^{y_{ij}} f_{ij}^{-1}(y) dy$$

$$\begin{cases} \dfrac{dz_{ui}}{dt} = -\dfrac{\partial E}{\partial y_{ui}} \\ = -\alpha z_{ui} - A\sum_{j \approx i} y_{uj} - B\sum_{v \neq u} y_{vi} - C(\sum_{v=1}^{n}\sum_{j=1}^{n} y_{vj} - n) - D\sum_{v \neq u} d_{uv}(y_{vi+1} + y_{vi-1}), \\ y_{ui} = f(z_{ui}), \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, i = j, \\ 0, i \neq j, \end{cases}$$

$$\begin{cases} w_{ui,vj} = -A\delta_{uv}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{uv}) - C - Dd_{uv}(\delta_{j,j+1} + \delta_{j,i-1}), \\ I_{ui} = nC, \end{cases}$$

HOPFIELD求解10个城市的TSP问题的参数设置

$$\alpha = 1, A = B = D = 500, C = 200$$

- Sigmoid函数为 $y_{ui} = f(z_{ui}) = \dfrac{1}{1 + e^{-\frac{2z_{ui}}{\mu_0}}}$ $\mu_0 = 0.02$

初始设置y_{ui}都相等,可以解出

$$\sum_{u=1}^{10}\sum_{i=1}^{10} y_{ui} = 10 \qquad z_{00} = -\frac{\mu_0}{2}\ln 9$$

z_{00}扰动 $z_{ui} = z_{00} + \delta z_{ui}, \quad -0.1\mu_0 \le \delta z_{ui} \le 0.1\mu_0$

其中，δz_{ui} 为$[-0.1\mu_0, 0.1\mu_0]$均匀分布。

离散系统神经网络

- 离散系统神经网络指激活函数采用阶跃函数的反馈型神经网络

$$\begin{cases} y_i(t+1) = f_i\left(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i\right), \\ i = 1,2,\cdots,n. \end{cases}$$

离散系统的平衡状态

$$Y_e = g(Y_e, t), \forall t \in I$$

- 异步算法：每次只调整一个神经元，表达式为

$$\begin{cases} y_i(t+1) = \mathrm{sgn}(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i), \\ y_j(t+1) = y_j(t), \quad j \neq i, \end{cases}$$

- 同步算法：同一时刻对所有神经元同时调整，表达式为

$$y_i(t+1) = \mathrm{sgn}(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i), i = 1,2,\cdots,n$$

例 三个神经元的反馈网络的权为

$$W=\begin{pmatrix}0 & 1 & 2\\ 1 & 0 & 3\\ 2 & 3 & 0\end{pmatrix},$$

在 t 时刻，一组输出 $Y(t)=(1,0,1)^T$，每个神经元的阈值都为 3。

第 t+1 时刻，同步算法使得每个神经元接受到的值为

$$Z(t+1)=WY(t)-\theta=(2,4,2)^T-(3,3,3)^T,$$

同步算法在 t+1 时刻的输出为

$$Y(t+1)=f(Z(t+1))=(0,1,0)^T。$$

异步算法的计算为：当 t+1 步迭代选择第一个神经元时，它的接收值为

$$z_1(t+1) = (0,1,2)Y(t) - \theta_1 = -1,$$

所以，t+1 时刻个神经元的输出为

$$y_1(t+1) = 0,\ y_2(t+1) = 0,\ y_3(t+1) = 1, \quad Y(t+1) = (0,0,1)^T 。$$

若 t+2 步迭代选择第二个神经元变化时，

$$z_2(t+2) = (1,0,3)Y(t+1) - \theta_2 = (1,0,3)\begin{pmatrix}0\\0\\1\end{pmatrix} - 3 = 0,$$

所以 $\quad y_1(t+2) = 0,\ y_2(t+2) = 0,\ y_3(t+2) = 1, \quad Y(t+2) = (0,0,1)^T 。$

若 t+3 步迭代选择第三个神经元变化时，

$$z_2(t+3) = (2,3,0)Y(t+2) - \theta_3 = (2,3,0)\begin{pmatrix}0\\0\\1\end{pmatrix} - 3 = -3,$$

所以

$$y_1(t+3) = 0,\ y_2(t+3) = 0,\ y_3(t+2) = 0, \quad Y(t+3) = (0,0,0)^T 。$$

离散动力系统的理论研究思路

- 沿用Hopfield求解连续神经网络的思想，利用离散动力系统来求解组合优化问题。
- 基本思路为：
  - 将组合优化问题对应一个能量函数
  - 建立一个离散动力系统
  - 找出离散动力系统的平衡点与能量函数极值点的关系。

- 假设组合优化问题对应的能量函数为

$$E(y) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j + \sum_{i=1}^{n} \theta_i y_i$$

- 对应的离散动力系统

$$\begin{cases} y_i(t+1) = f_i\left(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i\right), \\ i = 1,2,\cdots,n. \end{cases}$$

- 求离散动力系统的平衡点，期望达到优化问题的全局最优点。

定义 $Y \in \{0,1\}^n$ 的邻域为 $N(Y) = \{Y + \delta_i e_i | i = 1,2,\cdots,n\}$，

其中，e_i 表示第 i 个分量为 1 其他分量为 0 的 n 维向量，

$$\delta_i = \begin{cases} -1, & \text{当} y_i = 1, \\ 1, & \text{当} y_i = 0, \end{cases}$$

若能量函数在 Y*满足 $E(Y^*) \le E(Y),\ \forall Y \in N(Y^*)$，

则称 Y*为邻域 $N(Y^*)$ 的能量极小点，简称能量极小点。

- 异步算法：每次只调整一个神经元

STEP1 任选一个初始状态Y(0)，t=0；

STEP2 随机选一个神经元，更新状态；

$$\begin{cases} y_i(t+1) = \mathrm{sgn}(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i), \\ y_j(t+1) = y_j(t), \quad j \neq i, \end{cases}$$

STEP3 检验Y(t)是否为网络的平衡点，是转STEP4；否t:=t+1，转STEP2;

STEP4 输出Y(t).

人工神经网络的总体评价

- 第一，提供信息处理的一种新的手段。主要归功于人工神经网络的自适应性、学习能力；
- 第二，发展已较为成熟。成熟的三个方面是：（1）模型的建立和数学理论的支持，（2）计算工具的发展，（3）神经生物学的发展和对大脑的认识；
- 第三，人工神经网络发展仍然受到限制。尽管近三十年计算机发展迅速，同时有大量的数学理论支持和发展，但计算机设备在储存、速度和用户柔性方面的现有能力，限制了人工神经网络的发展；
- 第四，有成功应用的案例。人工神经网络在视觉、语言、信号处理和机器人等方面有成功应用的示例。虽说应用的范围有限，但其应用的问题及效果给人们的印象深刻。

- 人工神经网络的Hopfield模型直观可以发展神经网络型计算系统来替代传统的计算机。
- 产业化推广不成功！
- 在目前的条件下，还只能依赖传统的计算机来模拟人工神经网络的这些功能。
- 数值计算和分析是模拟中的一个重要部分。