第6章 人工神经网络与连接学习

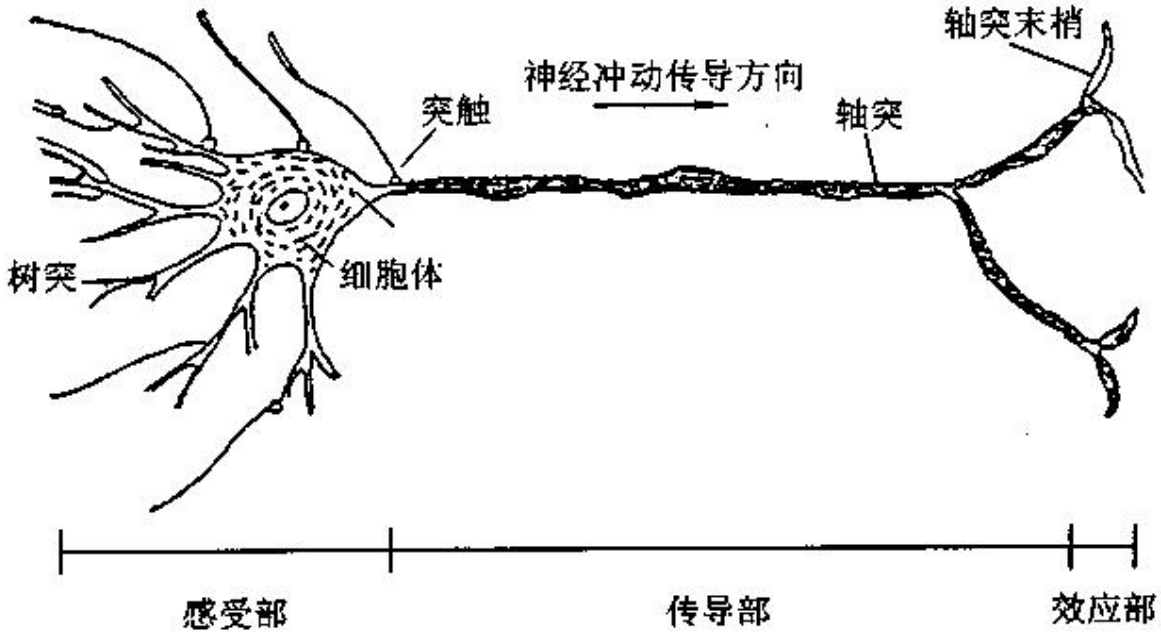
生物神经系统是人工神经网络的基础。人工神经网络是对人脑神经系统的简化、抽象和模拟，具有人脑功能的许多基本特征。

连接（联结）学习是一种基于人工神经网络的学习方式。

1. 生物神经系统简介[6.2]

生物神经元的结构

结构：细胞体、轴突、树突、突触



1. 生物神经系统简介

神经细胞及工作方式

细胞结构

细胞膜，细胞质，细胞核

基本状态：

抑制

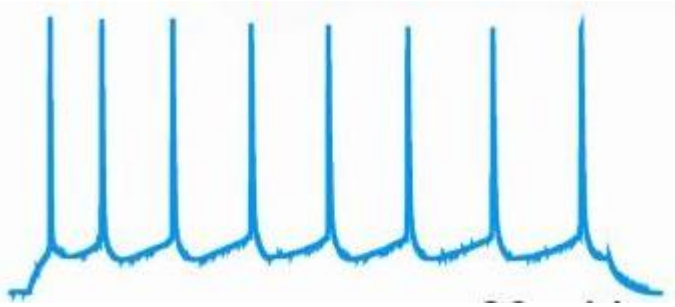
兴奋

工作方式：

刺激累积

瞬间冲动

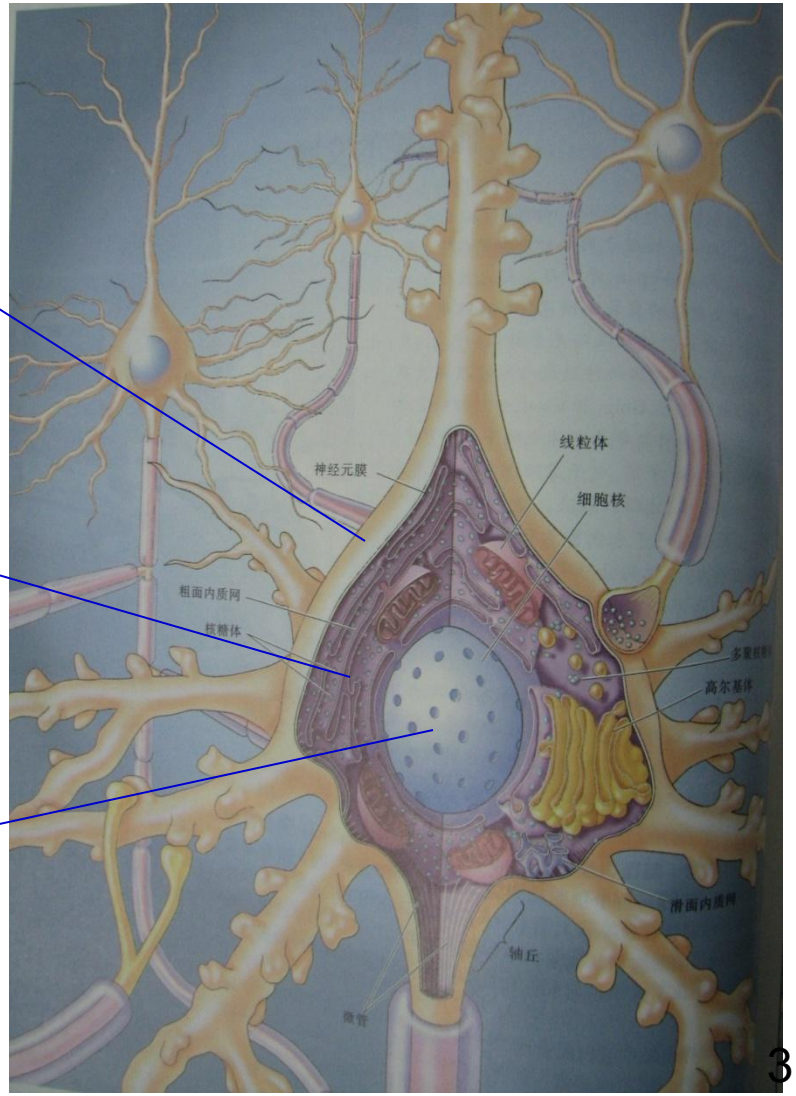
Integrate-and-Fire model



细胞膜

细胞质

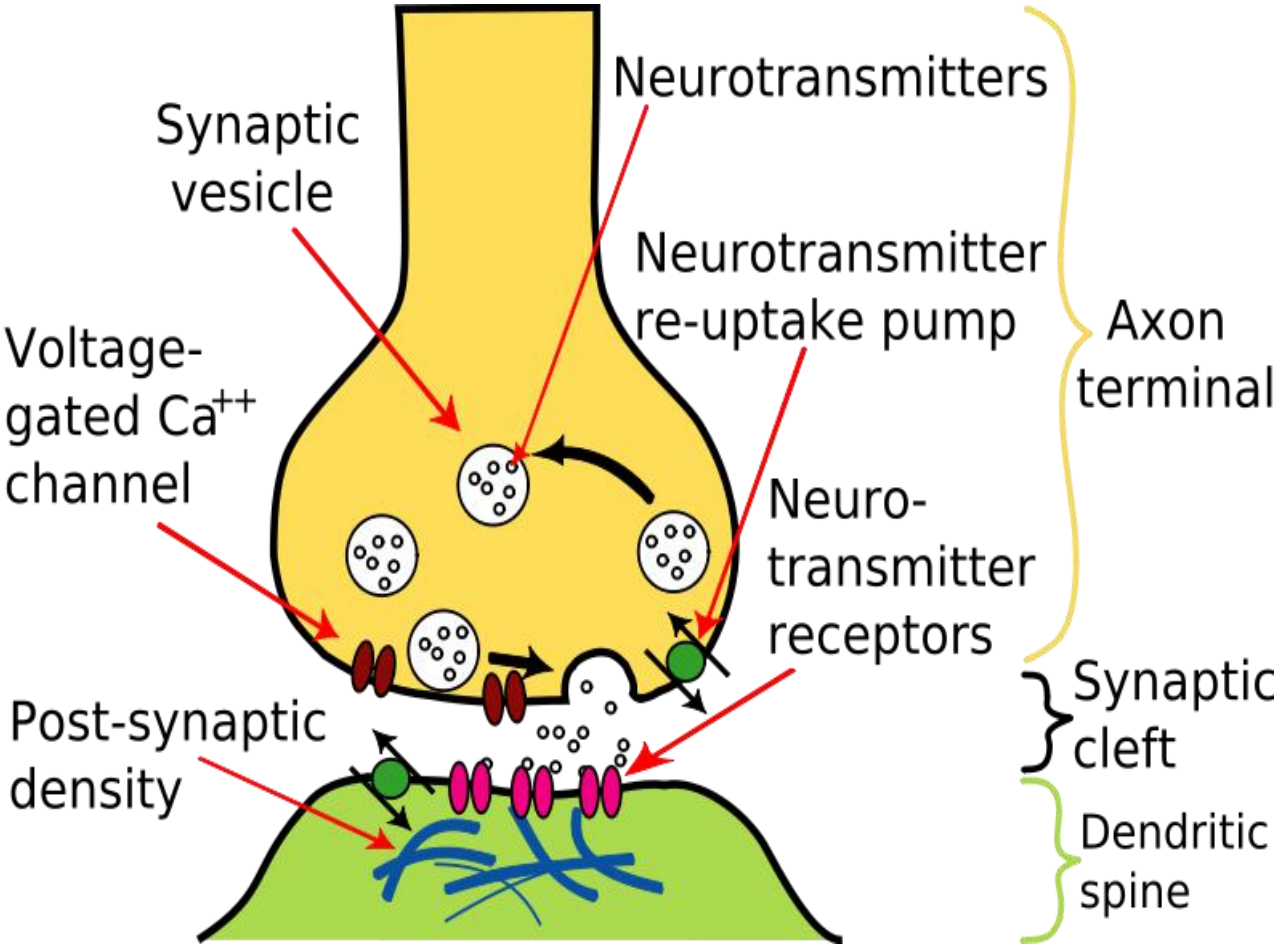
细胞核



1. 生物神经系统简介

突触传导

当一个神经冲动传到轴突末端时，促使小泡前移与突触前膜融合，并在融合处出现裂口，使其所含神经递质释放，释放出来的神经递质进入突触间隙。进入突触间隙的神经递质通过离子通道进入突触后膜，实现神经冲动的传递。



1. 生物神经系统简介

突触可塑性

人类大脑的神经元大约在10^{11}—10^{13}个左右。神经元之间的联系主要依赖其突触的联接作用。每个神经元大约有3×10^4个突触，小脑中每个神经元大约有10^5个突触。这种突触的联接是可塑的，也就是说突触特性的变化是受到外界信息的影响或自身生长过程的影响。生理学的研究归纳有以下几个方面的变化：

(1)突触传递效率的变化。首先是突触的膨胀以及由此产生的突触后膜表面积扩大，从而突触所释放出的传递物质增多，使得突触的传递效率提高。其次是突触传递物质质量的变化，包括比例成分的变化所引起传递效率的变化。

(2)突触接触间隙的变化。在突触表面有许多形状各异的小凸芽，调节其形状变化可以改变接触间隙，并影响传递效率。

(3)突触的发芽。当某些神经纤维被破坏后，可能又会长出新芽，并重新产生附着于神经元上的突触．形成新的回路。由于新的回路的形成，使得结合模式发生变化，也会引起传递效率的变化。

(4)突触数目的增减。由于种种复杂环境条件的刺激等原因，或者由于动物本身的生长或衰老，神经系统的突触数目会发生变化，并影响神经元之间的传递效率。

1. 生物神经系统简介

突触联结

神经元对信息的接受和传递都是通过突触来进行的。单个神经元可以从别的细胞接受多个输入。由于输入分布于不同的部位，对神经元影响的比例(权重)是不相同的。另外，各突触输入抵达神经元的先后时间也不一祥。因此，一个神经元接受的信息，在时间和空间上常呈现出一种复杂多变的形式，需要神经元对它们进行积累和整合加工，从而决定其输出的时机和强度。正是神经元这种整合作用，才使得亿万个神经元在神经系统中有条不紊、夜以继日地处理各种复杂的信息，执行着生物中枢神经系统的各种信息处理功能。

多个神经元以突触联接形成了一个神经网络。生物神经网络具有小世界特性。**[C. elegans 302个神经元]**

研究表明，生物神经网络的功能绝不是单个神经元生理和信息处理功能的简单叠加，而是一个有层次的、多单元的动态信息处理系统。

它们有其独特的运行方式和控制机制，以接受生物内外环境的输入信息，加以综合分折处理，然后调节控制机体对环境作出适当的反应。

Sebastian Seung: You are your connectome TED

1. 生物神经系统简介

神经信息处理

从信息系统研究的观点来看，人脑这个智能信息处理系统，有如下一些特征：

(1)并行分布处理的工作模式。

实际上大脑中单个神经元的信息处理速度是很慢的，毫秒(ms)级，比通常的电子门电路要慢几个数量级。每个神经元的处理功能也很有限，估计不会比计算机的一条指令更复杂。

但是人脑对某一复杂过程的处理和反应却很快，一般只需几百毫秒。例如要判定人眼看到的两个图形是否一样，实际上约需400 ms，而在这个处理过程中，与脑神经系统的一些主要功能，如视觉、记忆、推理等有关。按照上述神经元的处理速度，如果采用串行工作模式，就必须在百个串行步内完成，这实际上是不可能办到的。它是一个由众多神经元所组成的超高密度的并行处理系统。

1. 生物神经系统简介

神经信息处理

(2)神经系统的可塑性

神经系统的可塑性和自组织性与人脑的生长发育过程有关。从生理学的角度看，它体现在突触的可塑性和联接状态的变化，同时还表现在神经系统的自组织特性上。例如在某一外界信息反复刺激下．接受该信息的神经细胞之间的突触结合强度会增强。这种可塑性反映出大脑功能既有先天的制约因素，也有可能通过后天的训练和学习而得到加强。神经网络的学习机制就是基于这种可塑性现象，并通过修正突触的结合强度来实现的。

(3)信息处理与信息存贮合二为一。

大脑中的信息处理与信息存贮是有机结合在一起的，而不像现行计算机那样．存贮和处理是彼此分开的。由于大脑神经元兼有信息处理和存贮功能，所以在进行回忆时，不但不存在先找存贮地址而后再调出所存内容的问题，而且还可以由一部分内容恢复全部内容。

1. 生物神经系统简介

神经信息处理

(4) 信息处理的系统性

大脑是一个复杂的大规模信息处理系统，单个的元件“神经元”不能体现全体宏观系统的功能。实际上，可以将大脑的各个部位看成是一个大系统中的许多子系统。各个子系统之间具有很强的相互联系，一些子系统可以调节另一些子系统的行为。

(5) 能接受和处理模糊的、含混的、随机的信息。

(6) 求满意解而不是精确解。

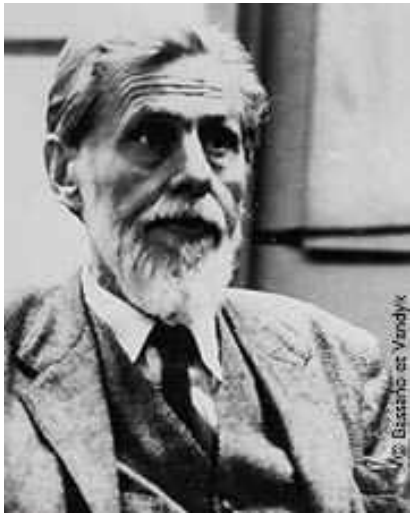
人类处理日常行为时，往往都不是一定要按最优或最精确的方式去求解，而是以能解决问题为原则，即求得满意解就行了。

(7) 系统的恰当退化和冗余备份(鲁棒性和容错性)。

2. 神经网络研究与发展[6.1]

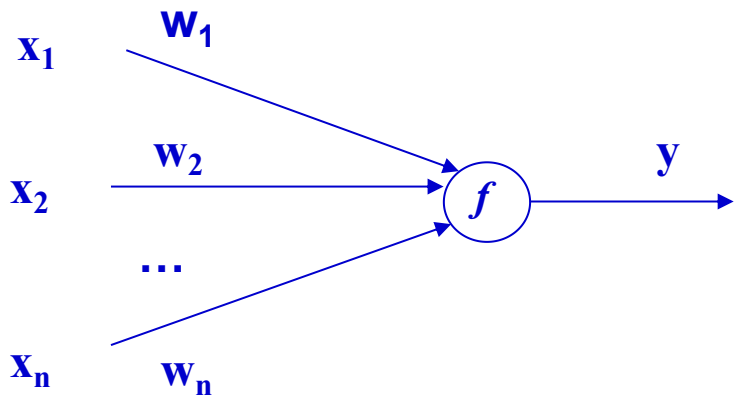
- 上世纪40年代初，美国McCulloch和Pitts从信息处理的角度，研究神经细胞行为的数学模型表达．提出了二值神经元模型。MP模型的提出开始了对神经网络的研究进程。

W. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity[J]. The Bulletin of Mathematical Biophysics, 1943, 5(4):115-133.



Warren McCulloch

Walter Pitts



逻辑与人生

MP模型

MP模型属于一种线性阈值元件模型，由McCulloch和Pitts提出的最早神经元模型之一。MP模型是大多数神经网络模型的基础。输入、输出、权值都是二值的。

w_{ij} —代表神经元i与神经元j之间的连接(模拟生物神经元之间突触连接)；+1兴奋，-1抑制；

u_i—代表神经元i的活跃值，即神经元状态；

v_j—代表神经元j的输出，即是神经元i的一个输入；

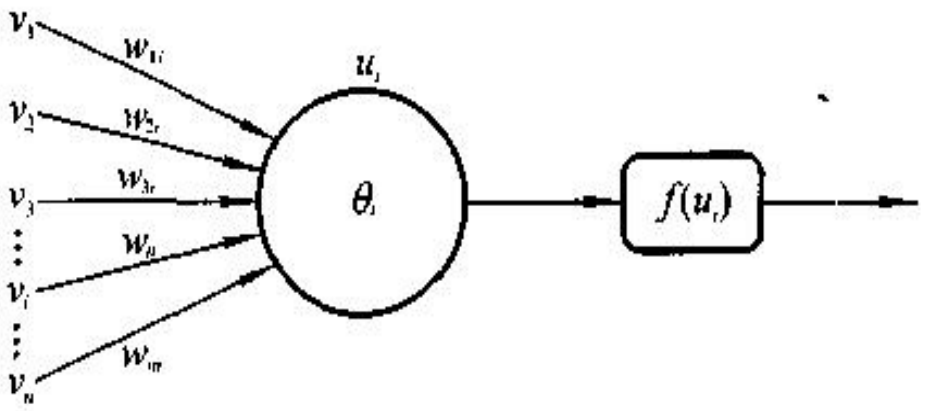
θ_i—代表神经元i的阈值。

函数f表达了神经元的输入输出特性。在MP模型中，f 定义为阶跃函数：

$$u_i = \sum_{j=1}^{n} w_{ji} v_j - \theta_i$$

$$v_i = f(u_i)$$

$$v_i = \begin{cases} 1, & u_i > 0 \\ 0, & u_i \le 0 \end{cases}$$



逻辑与人生：一颗数学巨星的陨落
启示

- 勇攀高峰、追求卓越；获得认同是以自身实力为基础的。具有硬实力，很难被埋没
- 树立健全的世界观，世界并非都是按纯粹的逻辑运行的
- 公正仁爱，“良言一句三冬暖，恶语伤人六月寒”
- 养成良好的生活习惯

2. 神经网络研究与发展

➢1949年心理学家Hebb提出著名的Hebb学习规则，即由神经元之间结合强度的改变来实现神经学习的方法。虽然Hebb学习规则在人们研究神经网络的初期就已提出，但是其基本思想至今在神经网络的研究中仍发挥着重要作用。

Cooper SJ, "Donald O. Hebb's synapse and learning rule: A history and commentary". *Neurosci & Biobehav Rev.* 2005 Jan; 28(8): 851-874.

2. 神经网络研究与发展

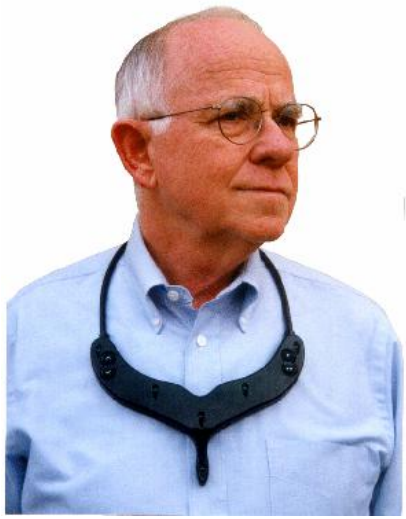
➢ 50年代末期，Rosenblatt提出感知机模型(Perceptron)，从工程角度出发，研究了用于信息处理的神经网络模型,它基本符合神经生理学的原理。虽然比较简单，却已具有神经网络的一些基本性质，如分布式存贮、并行处理、可学习性、连续计算等。这些神经网络的特性与当时流行串行的、离散的、符号处理的电子计算机及其相应的人工智能技术有本质上的不同，由此引起许多研究者的兴趣，在60代掀起了神经网络研究的第一次高潮。但是，当时人们对神经网络研究过于乐观，认为只要将这种神经元互连成一个网络，就可以解决人脑思维的模拟问题，然而，后来的研究结果却又使人们走到另一个极端上。

Frank Rosenblatt died in July 1971 on his 43rd birthday, in a boating accident in Chesapeake Bay

2. 神经网络研究与发展

- 1960年，Widrow和Hoff提出了自适应线性元件 Adaline是一个连续取值的线性阈值网络，在信号处理系统中应用十分广泛。

Least Mean Square (LMS)



2. 神经网络研究与发展

在60年代末，美国著名人工智能专家Minsky和Papert对Rosenblatt的工作进行了深人研究，出版了有较大影响的(Perceptron)一书，指出感知机的功能和处理能力的局限性，甚至连XOR(异或)这样的问题也不能解决，同时也指出如果在感知器中引入隐含神经元，增加神经网络的层次，可以提高神经网络的处理能力，但是却无法给出相应的网络学习算法。因此Minsky的结论是悲观的。

- In 1959, Minsky and McCarthy founded what is now known as the MIT Computer Science and Artificial Intelligence Laboratory.
- Minsky won the Turing Award in 1969, the Japan Prize in 1990, and the Benjamin Franklin Medal from the Franklin Institute in 2001.

←Marvin Minsky in 2008 (born in 1927)

2. 神经网络研究与发展

- 另一方面，由于60年代以来集成电路和微电子技术日新月异的发展，使得电子计算机的计算速度飞速提高，加上那时以功能模拟为目标、以知识信息处理为基础的知识工程等研究成果，给人工智能从实验室走向实用带来了希望，这些技术进步使人们认为它的潜力是无穷的，这就暂时掩盖了寻找新的智能途径的必要性和迫切性。另外，当时对大脑的计算原理、对神经网络计算的优点、缺点、可能性及其局限性等还很不清楚。

总之，认识上的局限性使对神经网络的研究进入了低潮。

2. 神经网络研究与发展

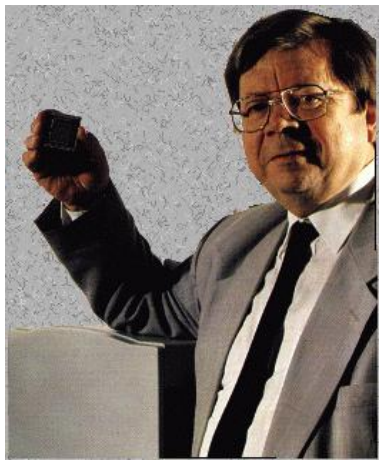
在这一低潮时期，仍有一些学者扎扎实实地继续着神经网络模型和学习算法的基础理论研究，提出了许多有意义的理论和方法。其中，主要有自适应共振理论，自组织映射，认知机网络模型理论，BSB模型等等，为神经网络的发展奠定了理论基础。

1. Boston大学的Grossberg和Carpenter提出了自适应共振理论ART网络。Cohen-Grossberg神经网络。

2. 芬兰的Helsinki大学的Kohonen提出了自组织映射网络。

3. 日本东京大学的Amari对神经网络进行了数学理论的研究，为神经网络的研究奠定了理论基础。



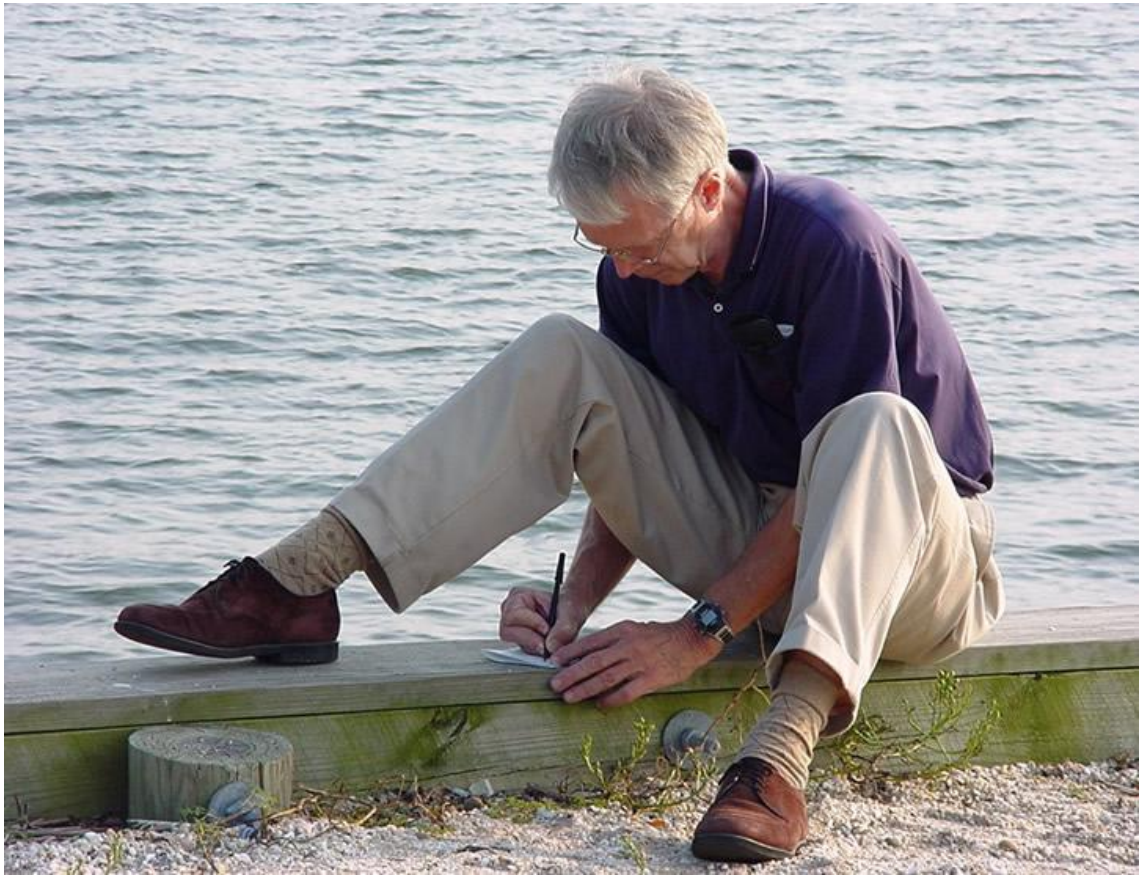



2. 神经网络研究与发展

- 进入80年代，首先是基于“知识库”的专家系统的研究和运用，在许多方面取得了较大成功。但在一段时间以后，实际情况表明专家系统并不像人们所希望的那样高明，特别是在处理视觉、听觉、形象思维、联想记忆以及运动控制等方面，传统的计算机和人工智能技术面临着重重困难。

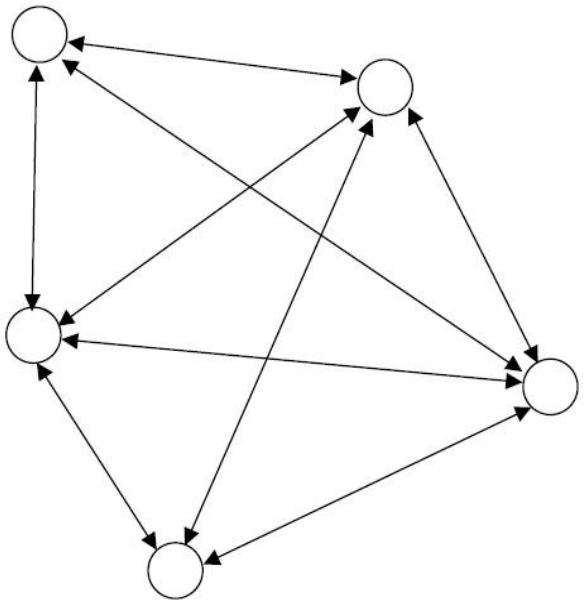
- 具有并行分布处理模式的神经网络理论又重新受到人们的重视。对神经网络的研究又开始复兴，掀起了第二次研究高潮。

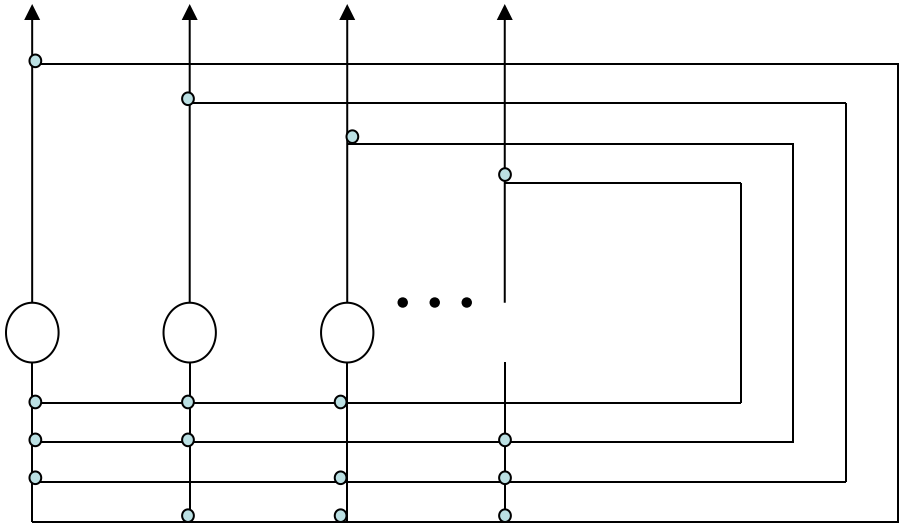
➢1982年，美国加州理工学院物理学家J．J．Hopfield提出了一种新的神经网络HNN。他引入了“能量函数”的概念，使得网络稳定性研究有了明确的判据。HNN的电子电路物理实现为神经计算机的研究奠定了基础，并将其应用于目前电子计算机尚难解决的计算复杂度为NP完全型的问题，例如著名的“旅行售货商问题”(TSP)，取得很好的效果。



Hopfield神经网络模型

Hopfield网络是由美国加州工学院物理学家霍普菲尔特1982年提出来的一种全互连的对称反馈网络模型。它可分为离散Hopfield网络和连续Hopfield网络。





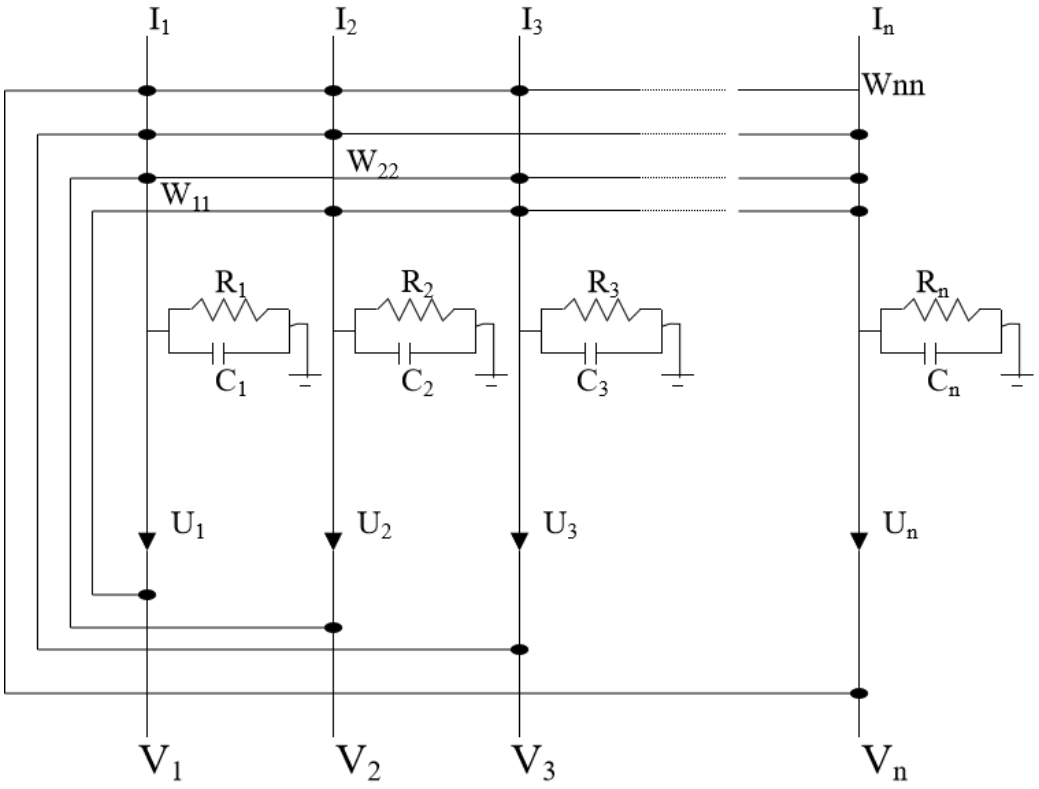
离散Hopfield神经网络模型

若用$y_j(t)$表示时刻t输出层神经元j的状态，则该神经元在时刻（t+1）的状态可由下式表示：

$$y_j(t+1)=sgn\left(\sum_{\substack{i=1\\ i\neq j}}^{n} w_{ij}y_i(t)-\theta_j\right)=\begin{cases}1 & 若\sum_{\substack{i=1\\ i\neq j}}^{n} w_{ij}y_i(t)-\theta_j\geq 0\\ 0(或-1) & 若\sum_{\substack{i=1\\ i\neq j}}^{n} w_{ij}y_i(t)-\theta_j<0\end{cases}$$

式中，sgn()为符号函数，θ_j为神经元j的阈值。输入神经元=输出神经元

连续Hopfield神经网络模型

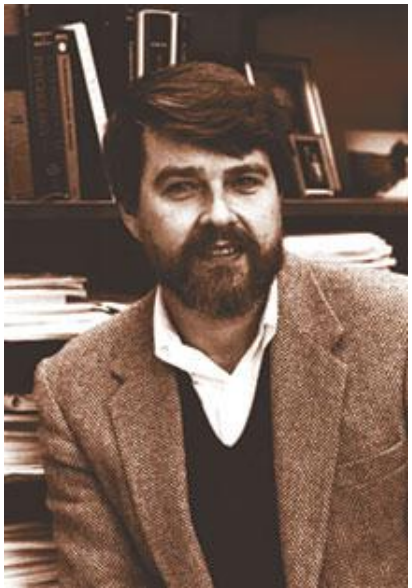


$$\begin{cases} C_i \dfrac{du_i}{dt} = -\dfrac{u_i}{R_i} + \sum_j w_{ij} V_j + I_i \\ V_i = f(u_i), \qquad i = 1,2,\cdots,n \end{cases}$$

➢1986年Rumelhart, Hinton等人（PDP) 在多层神经网络模型的基础上，提出了多层神经网络模型的反向传播学习算法(BP算法)，解决了多层前向神经网络的学习问题，证明了多层神经网络具有很强的学习能力，它可以完成许多学习任务，解决许多实际问题。

➢Hinton等人提出了Boltzman机模型。

➢1988年，蔡少堂(Leon O. Chua) 提出了细胞神经网络模型 。



2. 神经网络研究与发展

- 随后的二十来年，许多具备不同信息处理能力的神经网络已被提出来并应用于许多信息处理领域，如模式识别、自动控制、信号处理、决策辅助、人工智能等方面。神经计算机的研究也为神经网络的理论研究提供了许多有利条件，各种神经网络模拟软件包、神经网络芯片以及电子神经计算机的出现，体现了神经网络领域的各项研究均取得了长足进展。同时，相应的神经网络学术会议和神经网络学术刊物的大量出现，给神经网络的研究者们提供了许多讨论交流的机会。

- 实质上在2000年前后，又陷入了低潮（平台期）。

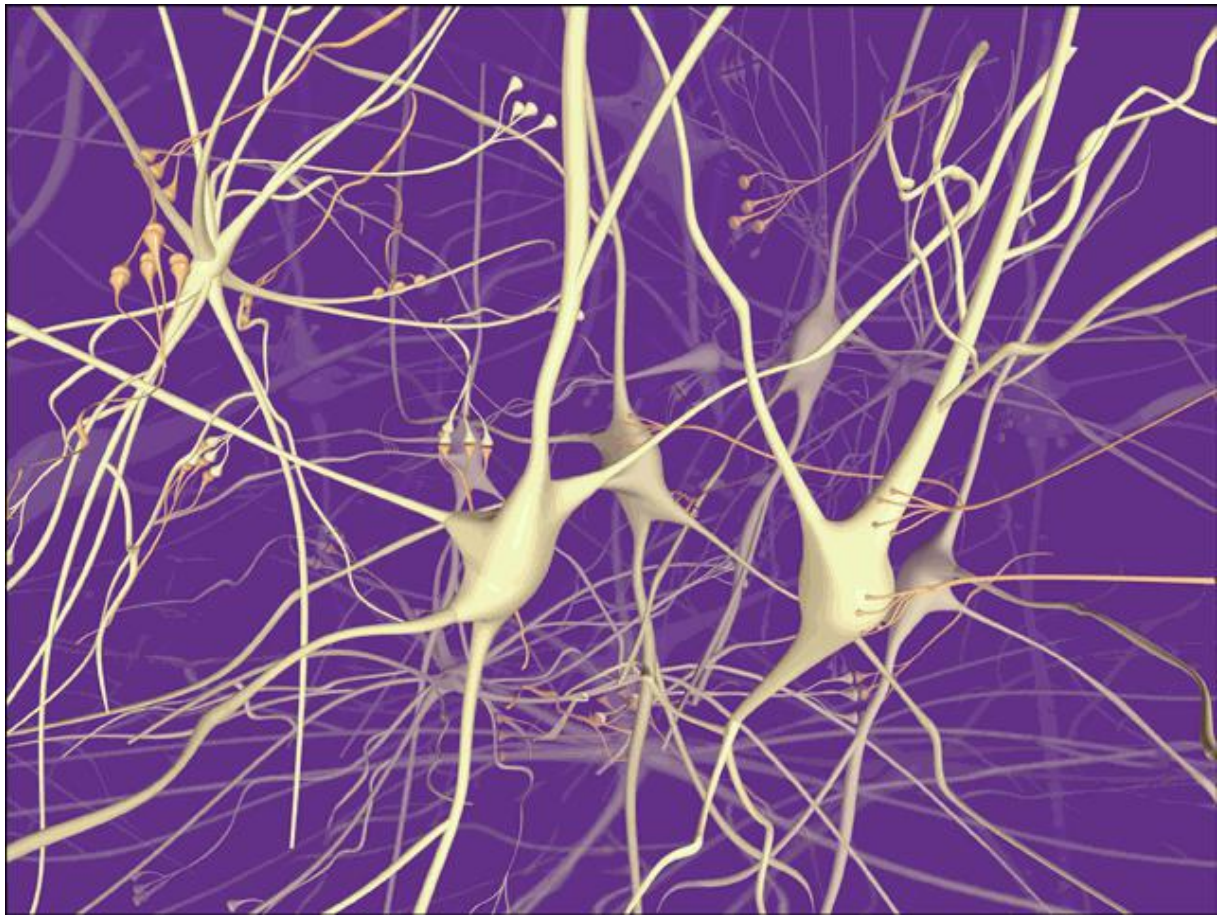
2. 神经网络研究与发展

- 2006年Hinton在Science发表论文以来，特别是2012年ImageNet竞赛以来，深度学习受到了非常大的关注。2016年AlphaGo推向高潮。
- 多层神经网络的训练
- 数据和计算资源的迅速提升
- 在图像与视觉、语音与自然语言处理等应用领域取得了很多好的结果。

3. 人工神经网络结构与类型[6.3]

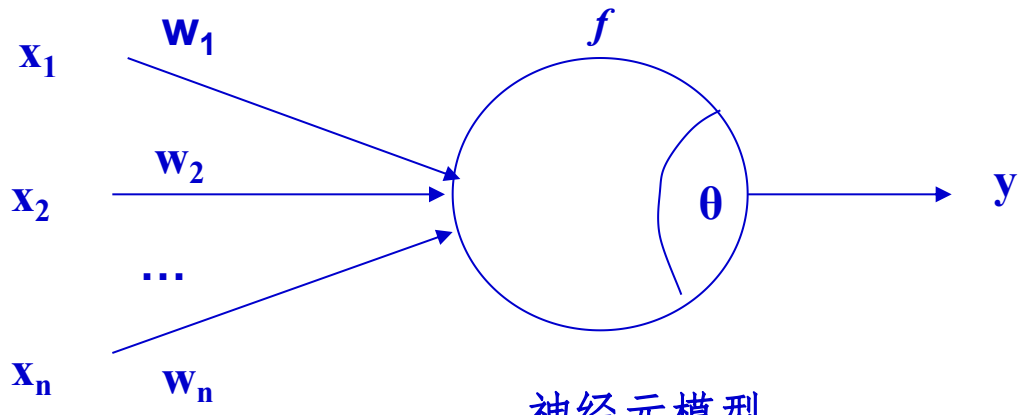
人工神经网络基本类型

- 神经元类型
- 网络结构类型
- 学习算法类型



3. 人工神经网络结构与类型

人工神经元的结构



神经元模型

图中的$x_1, x_2, \ldots, x_n$表示某一神经元的n个输入；w_i表示第i个输入的连接强度，称为连接权值；θ为神经元的阈值；y为神经元的输出。

人工神经元是一个具有多输入，单输出的非线性器件。其输入为：$\sum_{i=1}^{n} w_i x_i$

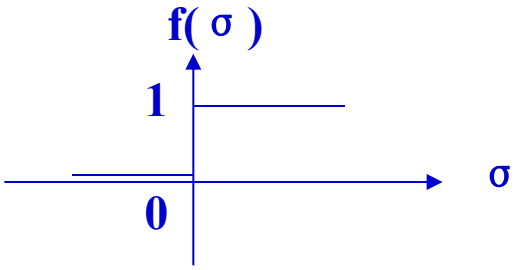
其输出为：

$$y = f(\sigma) = f(\sum_{i=1}^{n} w_i x_i - \theta)$$

3. 人工神经网络结构与类型

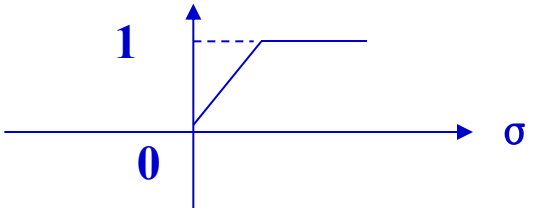
常用的人工神经元模型

激活函数(activation function)—执行对神经元所获得的输入的变换，也可以称为激励函数、活化函数, 根据激活函数的不同，可得不同的神经元模型。



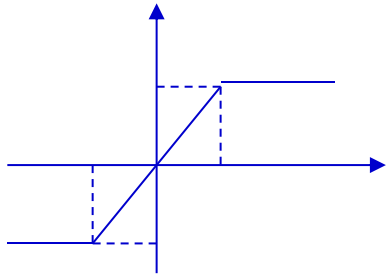
阈值型(Threshold)

这种模型的神经元没有内部状态，作用函数f是一个阶跃函数，他表示激活值σ和输出之间的关系。



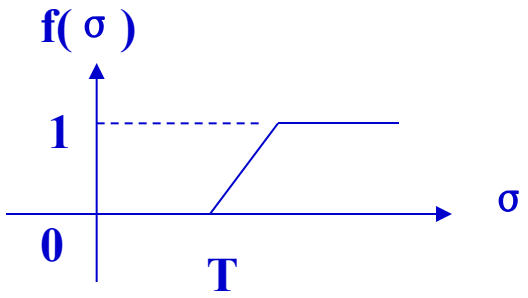
分段线性饱和型(Linear Saturation)

其输入/输出之间在一定范围内满足线性关系，一直延续到输出为最大值1为止。但当达到最大值后，输出就不再增大。



子阈累积型(Subthreshold Summation)

也是一个非线性函数，当产生的激活值超过T值时，该神经元被激活产生反响。在线性范围内，系统的反响是线性的。



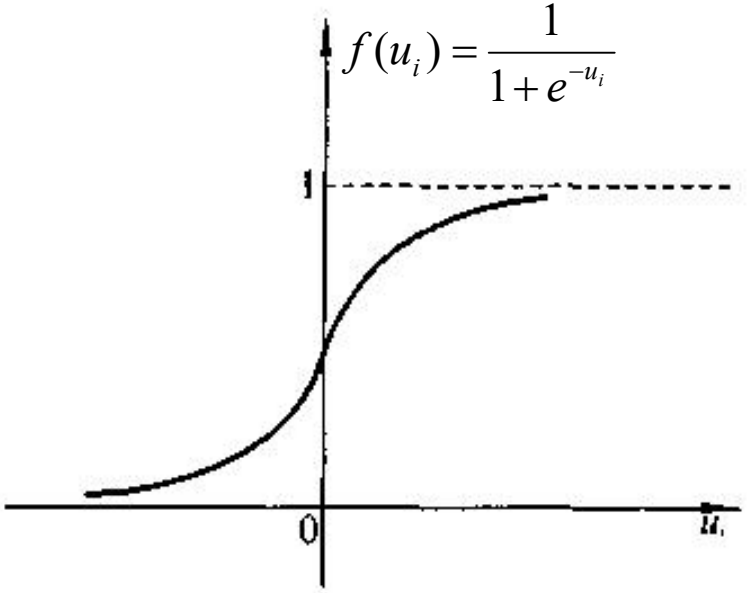
3. 人工神经网络结构与类型

常用的人工神经元模型

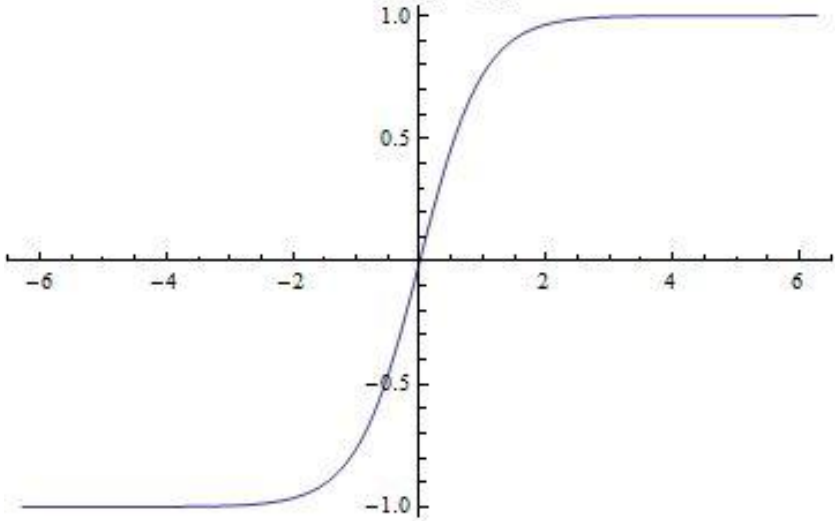
S型(Sigmoid)

这是一种连续的神经元模型，其输入输出特性反映的是神经元的饱和特性.

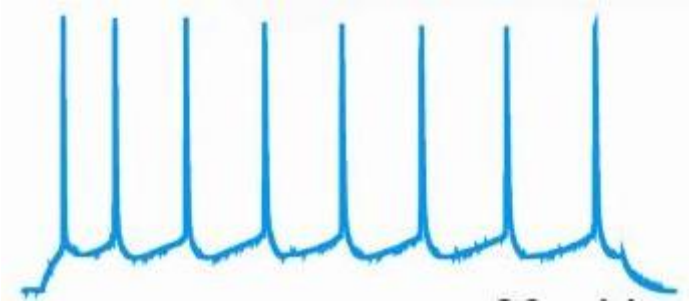
$$f(u_i)=\frac{1}{1+e^{-u_i}}$$

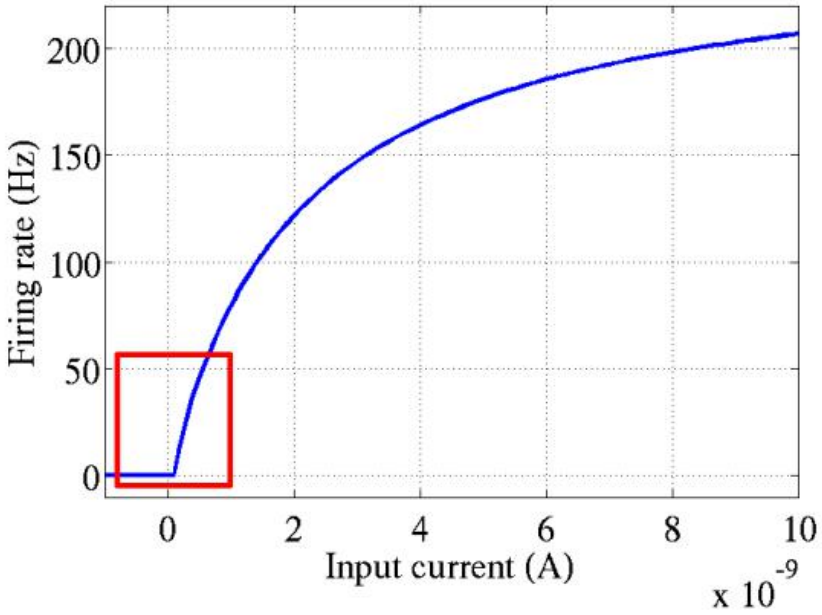


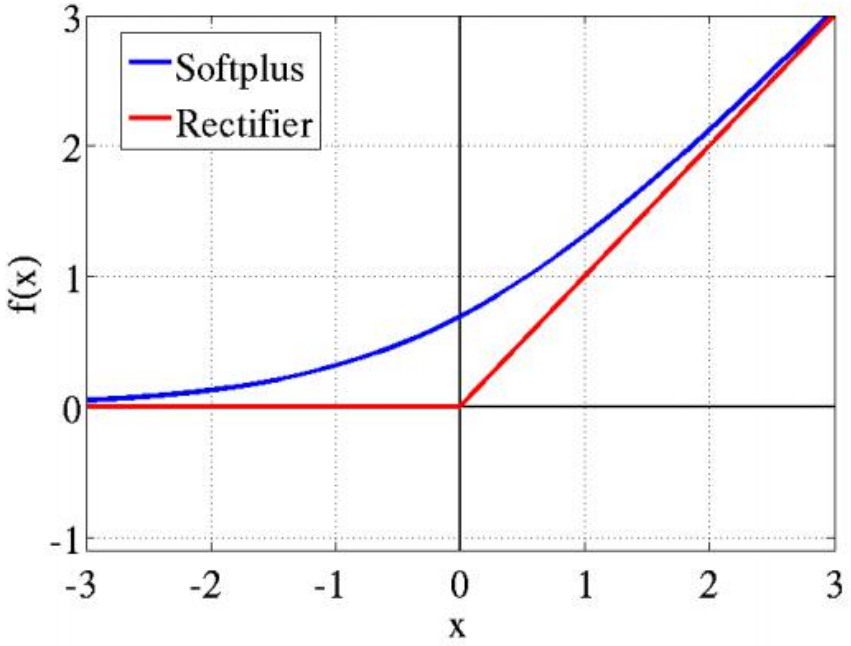
$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$



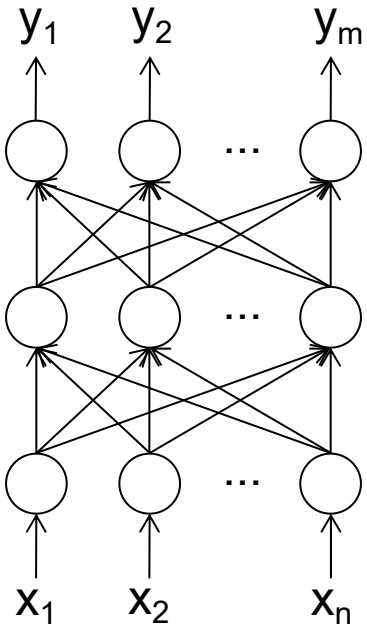
ReLU(Rectified Linear Unit) & Softplus



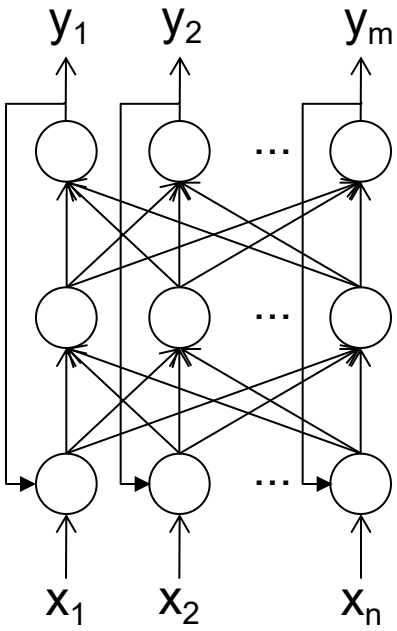




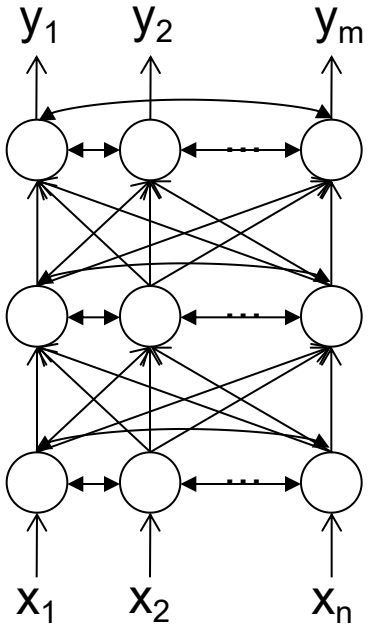
神经网络的结构类型



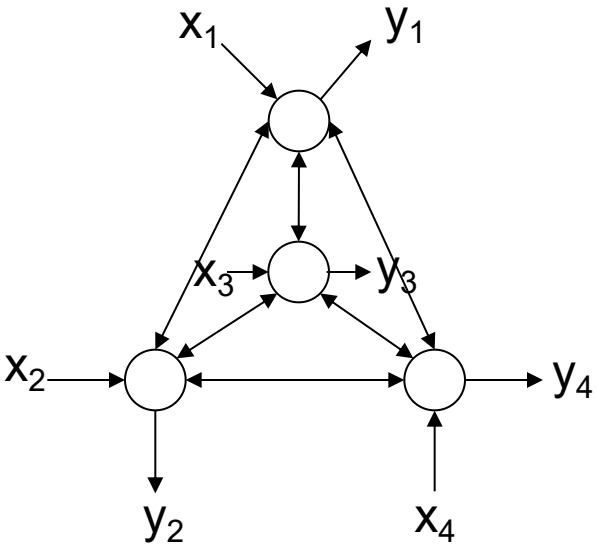
单纯前馈网络



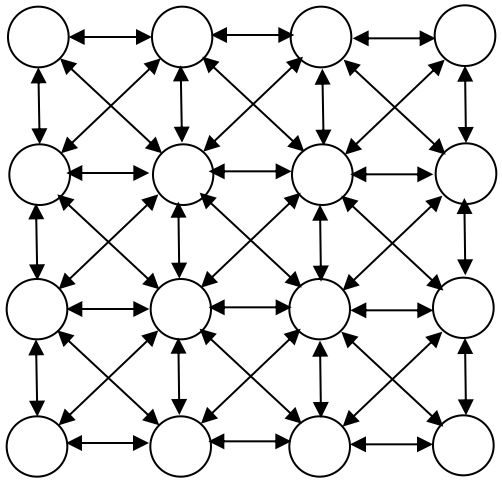
有反馈的前馈网络



内层互连前馈网络



全互连网络



局部互连网络

前馈网络

多层前馈网络是指那种除拥有输入、输出层外，还至少含有一个、或更多个隐含层（hidden layer)的前馈网络。

隐含层是指由那些既不属于输入层又不属于输出层的神经元所构成的处理层,也被称为中间层。

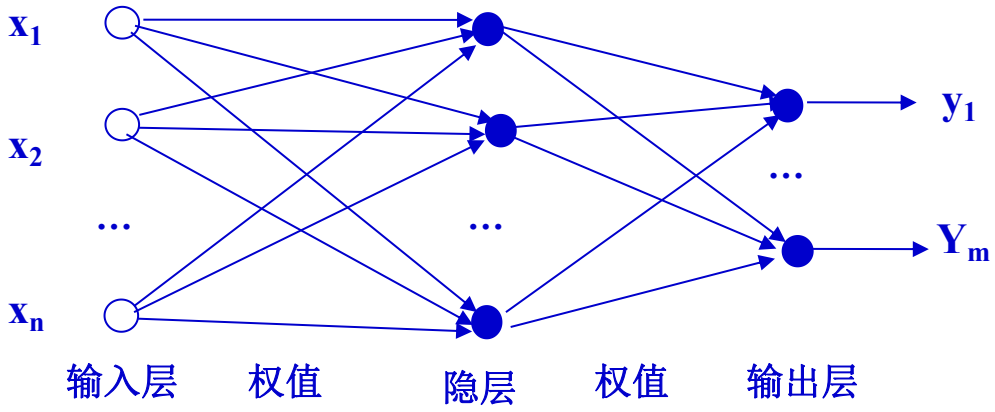


图4.9 多层前馈网络结构

多层前馈网络的输入层的输出向量是第一隐含层的输入信号，而第一隐含层的输出则是第二隐含层的输入信号，以此类推，直到输出层。

反馈神经网络

反馈网络是指允许采用反馈联结方式所形成的神经网络。所谓反馈联结方式是指一个神经元的输出可以被反馈至同层或前层的神经元。

反馈网络和前向网络不同：

前向网络属于非循环连接模式，它的每个神经元的输入都没有包含该神经元先前的输出。

反馈网络则不同，它的每个神经元的输入都有可能包含有该神经元先前输出的反馈信息，即一个神经元的输出是由该神经元当前的输入和先前的输出这两者来决定的。

反馈网络的一个典型例子是后面将要介绍的**Hopfield**网络

学习规则类型

- 监督学习 （δ学习律）
- 无监督学习
- 半监督学习
- 强化学习
- 竞争学习
- 。。。。。。

6.4 人工神经网络的典型模型

0. MP模型

MP模型属于一种线性阈值元件模型，由McCulloch和Pitts提出的最早神经元模型之一。MP模型是大多数神经网络模型的基础。输入、输出、权值都是二值的。

w_{ij} —代表神经元i与神经元j之间的连接(模拟生物神经元之间突触连接)；+1兴奋，-1抑制；

u_i—代表神经元i的活跃值，即神经元状态；

v_j—代表神经元j的输出，即是神经元i的一个输入；

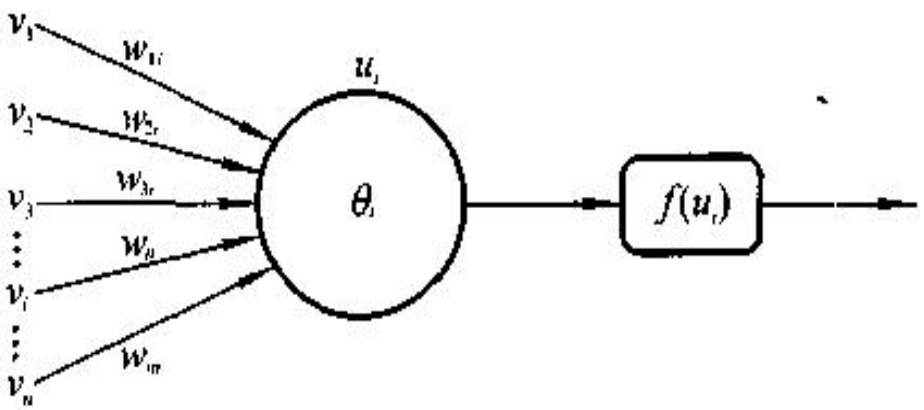
θ_i—代表神经元i的阈值。

函数f表达了神经元的输入输出特性。在MP模型中，f 定义为阶跃函数：

$$u_i = \sum_{j=1}^{n} w_{ji} v_j - \theta_i$$

$$v_i = f(u_i)$$

$$v_i = \begin{cases} 1, & u_i > 0 \\ 0, & u_i \le 0 \end{cases}$$



1. 感知器模型

单层感知器

感知器是美国学者罗森勃拉特（Rosenblatt）于1957年为研究大脑的存储、学习和认知过程而提出的一类具有自学习能力的神经网络模型，其拓扑结构是一种分层前向网络。它包括：单层感知器和多层感知器。

单层感知器是一种只具有单层可调节连接权值神经元的前向网络，这些神经元构成了单层感知器的输出层，是感知器的可计算节点。

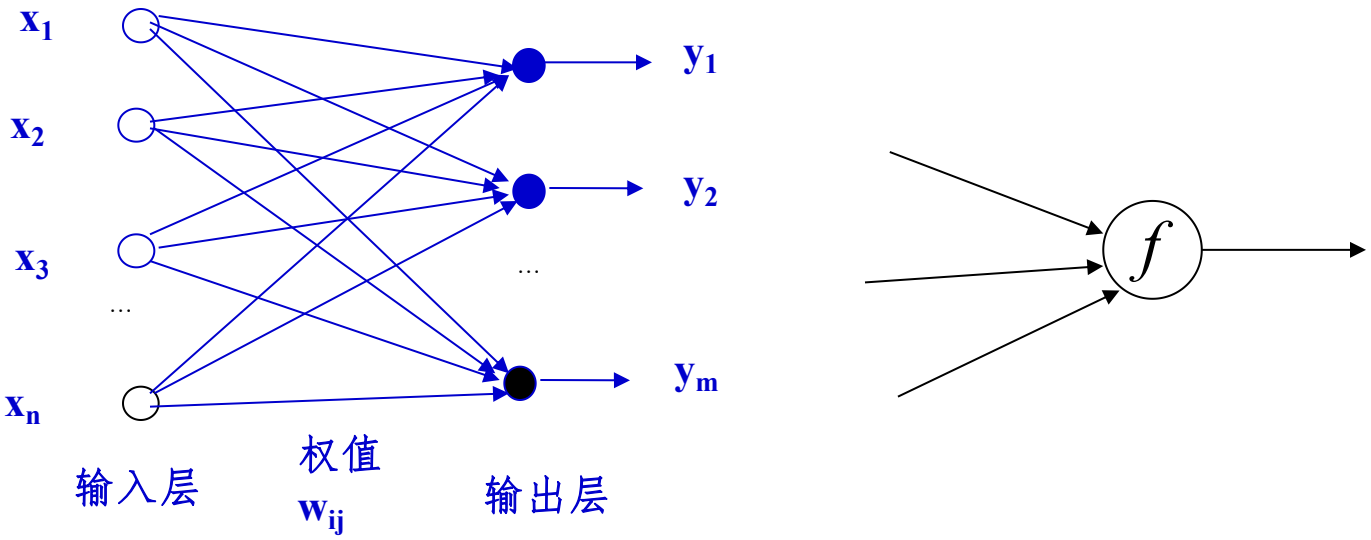
在单层感知器中，每个可计算节点都是一个线性阈值神经元。当输入信息的加权和大于或等于阈值时，输出为1，否则输出为0或-1。

单层感知器的输出层的每个神经元有一个输出，且该输出仅与本神经元的输入及联接权值有关，而与其他神经元无关。

1. 感知器模型

单层感知器

单层感知器的结构如下图



其中，输入向量为$X=(x_1,x_2,\ldots,x_n)$；输出向量为$Y=(y_1,y_2,\ldots,y_m)$；输入层各个输入到相应神经元的连接权值分别是w_{ij}，i=1,2,..,n，j=1,2,.., m。

1. 感知器模型

单层感知器

若假设各神经元的阈值分别是θ_j，j=1,2,…,m，则各神经元的输出y_j，j=1,2,..,m分别为：

$$y_j=f\left(\sum_{i=1}^{n} w_{ij}x_i-\theta_j\right) \qquad j=1,2,\ldots,m$$

其中，由所有连接权值w_{ij}构成的连接权值矩阵W为：

$$W=\begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix}$$

1.感知器模型

单层感知器

使用感知器的一个目的是为了对外部输入进行分类。

罗森勃拉特已经证明，如果外部输入是线性可分的（指存在一个超平面可以将它们分开），则单层感知器一定能够把它划分为两类。其判别超平面由如下判别式确定：

$$\sum_{i=1}^{n} w_{ij} x_i - \theta_j = 0 \qquad j = 1,2,\ldots,m$$

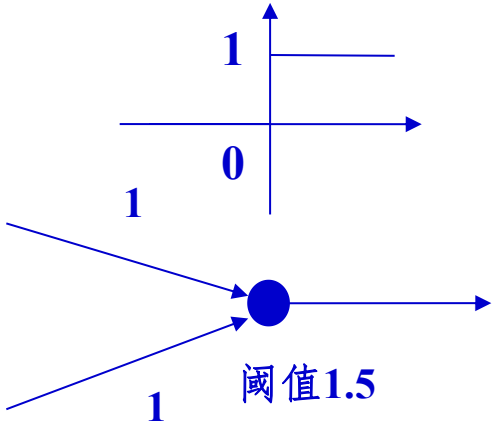
作为例子，下面讨论用单个感知器实现逻辑运算的问题。事实上，单层感知器可以很好地实现“与”、“或”、“非”运算，但却不能解决“异或”问题。

1. 感知器模型

单层感知器

例4.1“与”运算（$x_1 \wedge x_2$）

输入		输出	超平面	阈值条件
x_1	x_2	$x_1 \wedge x_2$	$w_1*x_1+ w_2* x_2-\theta=0$	
0	0	0	$w_1*0+ w_2*0 -\theta<0$	$\theta>0$
0	1	0	$w_1*0+ w_2*1 -\theta<0$	$\theta>w_2$
1	0	0	$w_1*1+ w_2*0 -\theta<0$	$\theta>w_1$
1	1	1	$w_1*1+ w_2*1-\theta\geq0$	$\theta\leq w_1+ w_2$



可以证明此表有解，例如取$w_1=1$，$w_2=1$，$\theta=1.5$，其分类结果如右图所示。

其中，输出为1的用实心圆，输出为0的用空心圆。后面约定相同。

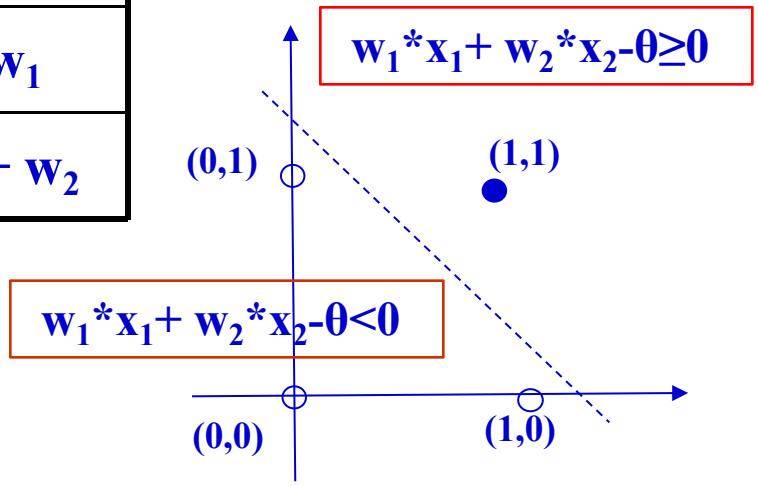


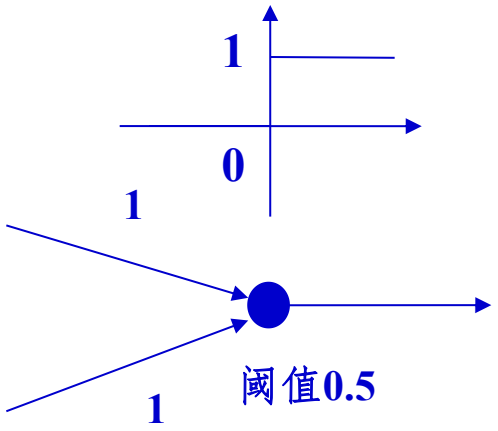
图4.10 与运算问题图示

1. 感知器模型

单层感知器

例4.2 “或”运算（$x_1 \vee x_2$）

输入		输出	超平面	阈值条件
x_1	x_2	$x_1 \vee x_2$	$w_1*x_1+ w_2* x_2-\theta=0$	
0	0	0	$w_1*0+ w_2*0 -\theta<0$	$\theta>0$
0	1	1	$w_1*0+ w_2*1 -\theta\geq0$	$\theta\leq w_2$
1	0	1	$w_1*1+ w_2*0 -\theta\geq0$	$\theta\leq w_1$
1	1	1	$w_1*1+ w_2*1-\theta\geq0$	$\theta\leq w_1+ w_2$



此表也有解，例如取$w_1=1$，$w_2=1$，$\theta=0.5$，其分类结果如右图所示。

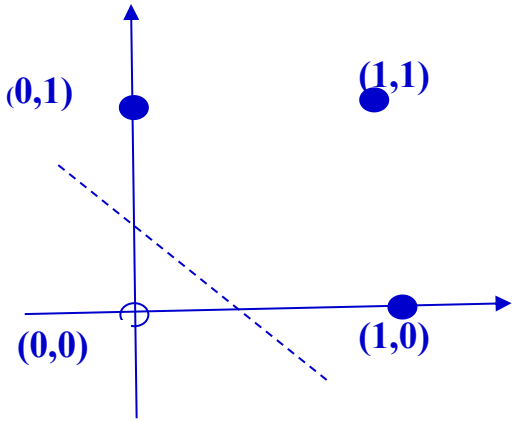


图4.11 或运算问题图示

1. 感知器模型

单层感知器

例4.3 “非”运算（$\neg x_1$）

输入	输出	超平面	阈值条件
x_1	$\neg x_1$	$w_1*x_1-\theta=0$	
0	1	$w_1*0-\theta \geq 0$	$\theta \leq 0$
1	0	$w_1*1-\theta<0$	$\theta>w_1$

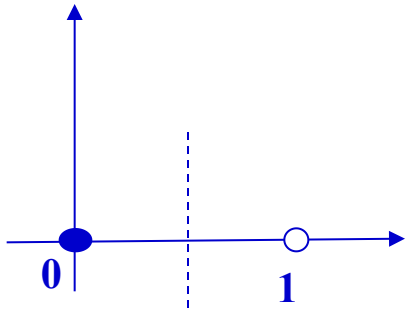


图4.12 非运算问题图示

此表也有解，例如取$w_1=-1$，$\theta=-0.5$，其分类结果如右图所示。

1. 感知器模型

单层感知器

例4.4 “异或”运算（x_1 XOR x_2）

输入		输出	超平面	阈值条件
x_1	x_2	x_1 XOR x_2	$w_1*x_1+ w_2*x_2-\theta=0$	
0	0	0	$w_1*0+ w_2*0 -\theta<0$	$\theta>0$
0	1	1	$w_1*0+ w_2*1 -\theta\geq0$	$\theta\leq w_2$
1	0	1	$w_1*1+ w_2*0 -\theta\geq0$	$\theta\leq w_1$
1	1	0	$w_1*1+ w_2*1-\theta<0$	$\theta>w_1+ w_2$

此表无解，即无法找到满足条件的w_1、w_2和θ，如右图所示。因为异或问题是一个非线性可分问题，需要用多层感知器来解决。

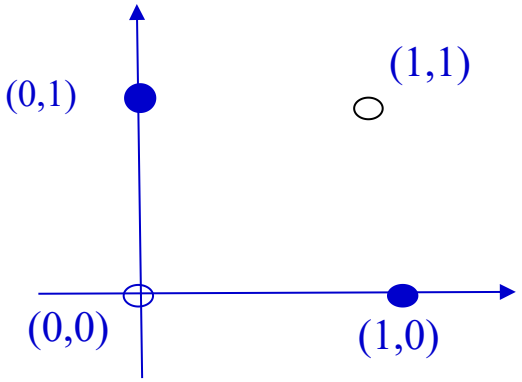


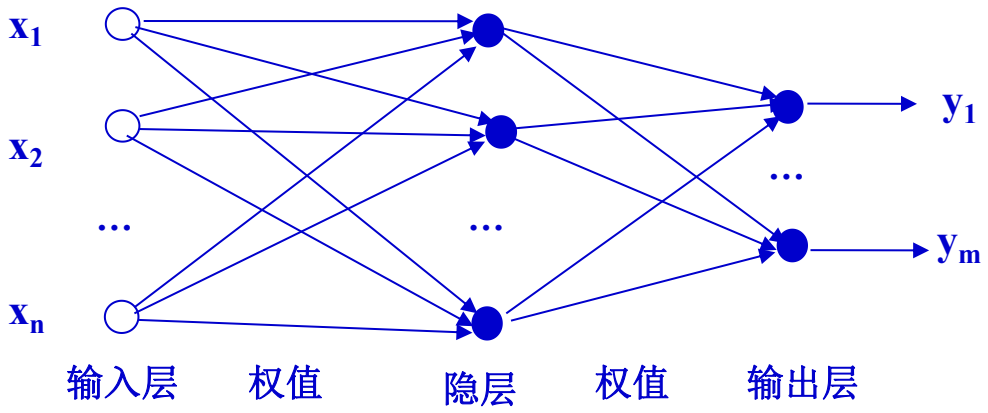
图4.13 异或运算问题图示

1. 感知器模型

多层 感知器

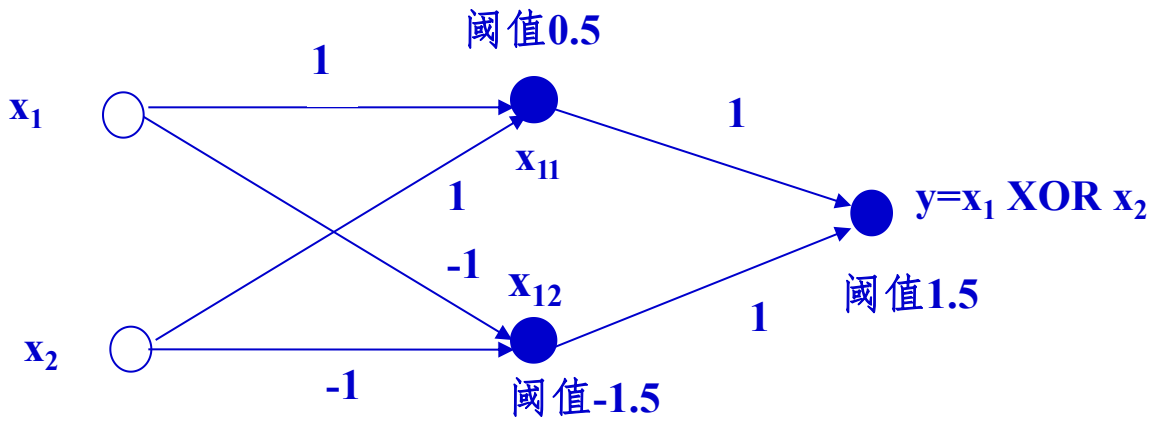
多层感知器

多层感知器是通过在单层感知器的输入、输出层之间加入一层或多层处理单元所构成的。



1. 感知器模型

多层 感知器



隐层神经元x_{11}所确定的直线方程为

$$1*x_1+1*x_2-0.5=0$$

可以识别一个半平面。

隐层神经元x_{12}所确定的直线方程为

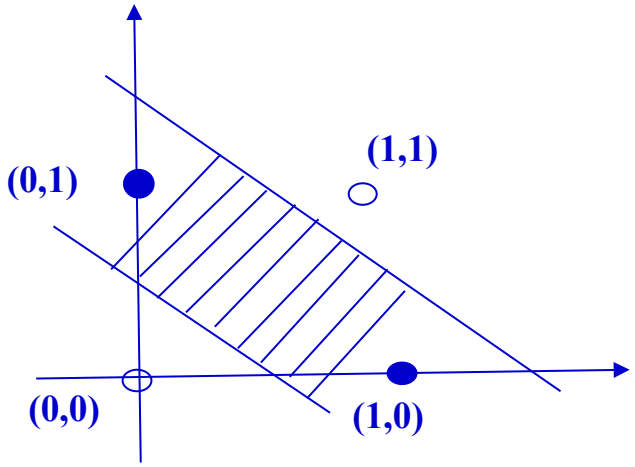
$$1*x_1+1*x_2-1.5=0$$

它也可以识别一个半平面。

输出层神经元所确定的直线方程为

$$1*x_{11}+1*x_{12}-1.5=0$$

它相当于对隐层神经元x_{11}和x_{12}的输出作“逻辑与”运算



连接学习

1. 监督学习
   1.1 ADALINE (LMS)
   1.2 BP算法
2.无监督学习
   2.1 Hebb 规则
   2.2 Oja 规则
3. 深度学习
   3.1 Autoencoder
   3.2 Convolutional Neural Network
   3.3 Generative Adversarial Network

6.6.1 连接学习规则

2. 纠错学习规则

纠错学习也叫误差修正学习，是一种有导师的学习过程。

基本思想：利用神经网络的期望输出与实际输出之间的偏差作为连接权值调整的参考，并最终减少这种偏差。

6.6.2 单层感知器学习算法

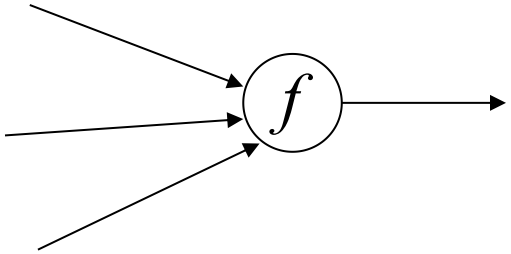
算法思想

单层感知器学习实际上是一种基于纠错学习规则，采用迭代的思想对连接权值和阈值进行不断调整，直到满足结束条件为止的学习算法。

假设X(k)和W(k)分别表示学习算法在第k次迭代时输入向量和权值向量，为方便，把阈值θ作为权值向量W(k)中的第一个分量，对应地把“-1”固定地作为输入向量X(k)中的第一个分量。即W(k)和X(k)可分别表示如下：

$X(k)=[-1, x_1(k), x_2(k), \ldots, x_n(k)]$

$W(k)=[\theta(k), w_1(k), w_2(k), \ldots, w_n(k)]$



即$x_0(k)=-1$，$w_0(k)=\theta(k)$。

单层感知器学习是一种有导师学习，它需要给出输入样本的期望输出。

假设一个样本空间可以被划分为A、B两类，定义：

期望输出：若输入样本属于A类，期望输出为+1，否则为-1。

6.6.2 单层感知器学习算法

算法描述

单层感知器学习算法可描述如下：

(1) 设t=0，初始化连接权和阈值。即给$w_i(0)(i=1, 2, \dots, n)$及$\theta(0)$分别赋予一个较小的非零随机数，作为初值。其中，$w_i(0)$是第0次迭代时输入向量中第i个输入的连接权值；$\theta(0)$是第0次迭代时输出节点的阈值；

(2) 提供新的样本输入$x_i(t)(i=1, 2, \dots, n)$和期望输出d(t)；

(3) 计算网络的实际输出：

$$y(t) = f(\sum_{i=1}^{n} w_i(t)x_i(t) - \theta(t)) = f(\sum_{i=0}^{n} w_i(t)x_i(t)) \qquad i = 1,2,\dots,n$$

(4) 若y(t)=d(t)，不需要调整连接权值，转(6)。否则，需要调整权值

(5) 调整连接权值

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

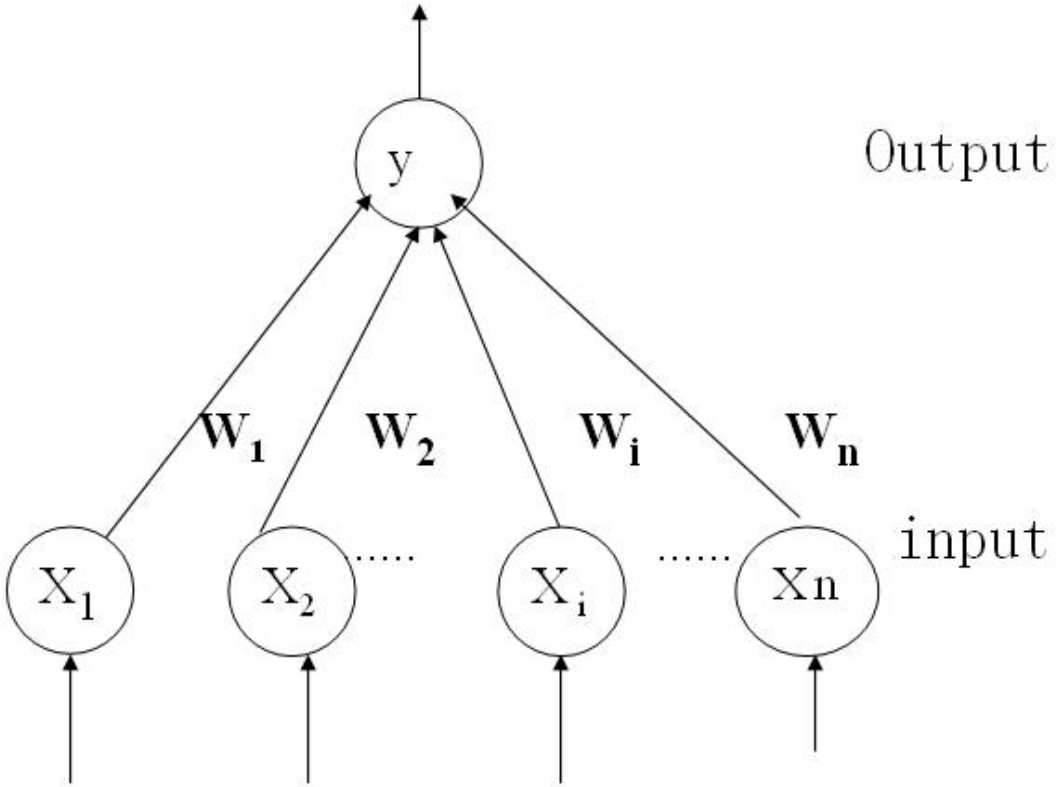
其中，η是一个增益因子，用于控制修改速度，其值如果太大，会影响$w_i(t)$的收敛性；如果太小，又会使$w_i(t)$的收敛速度太慢；

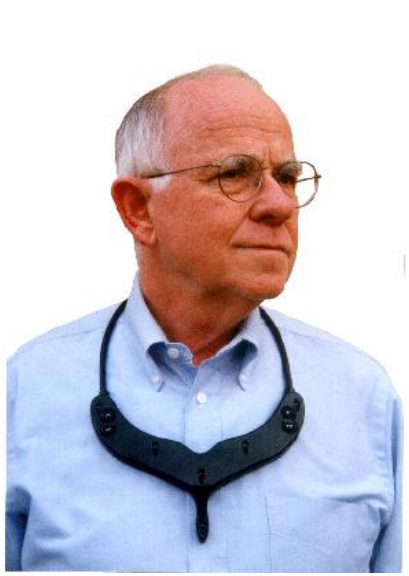
(6) 判断是否满足结束条件，若满足，算法结束；否则，将t值加1，转(2)重新执行。这里的结束条件一般是指$w_i(t)$对一切样本均稳定不变。

Adaline

- ADAptive LINear Element

  1962, Stanford University, Widrow





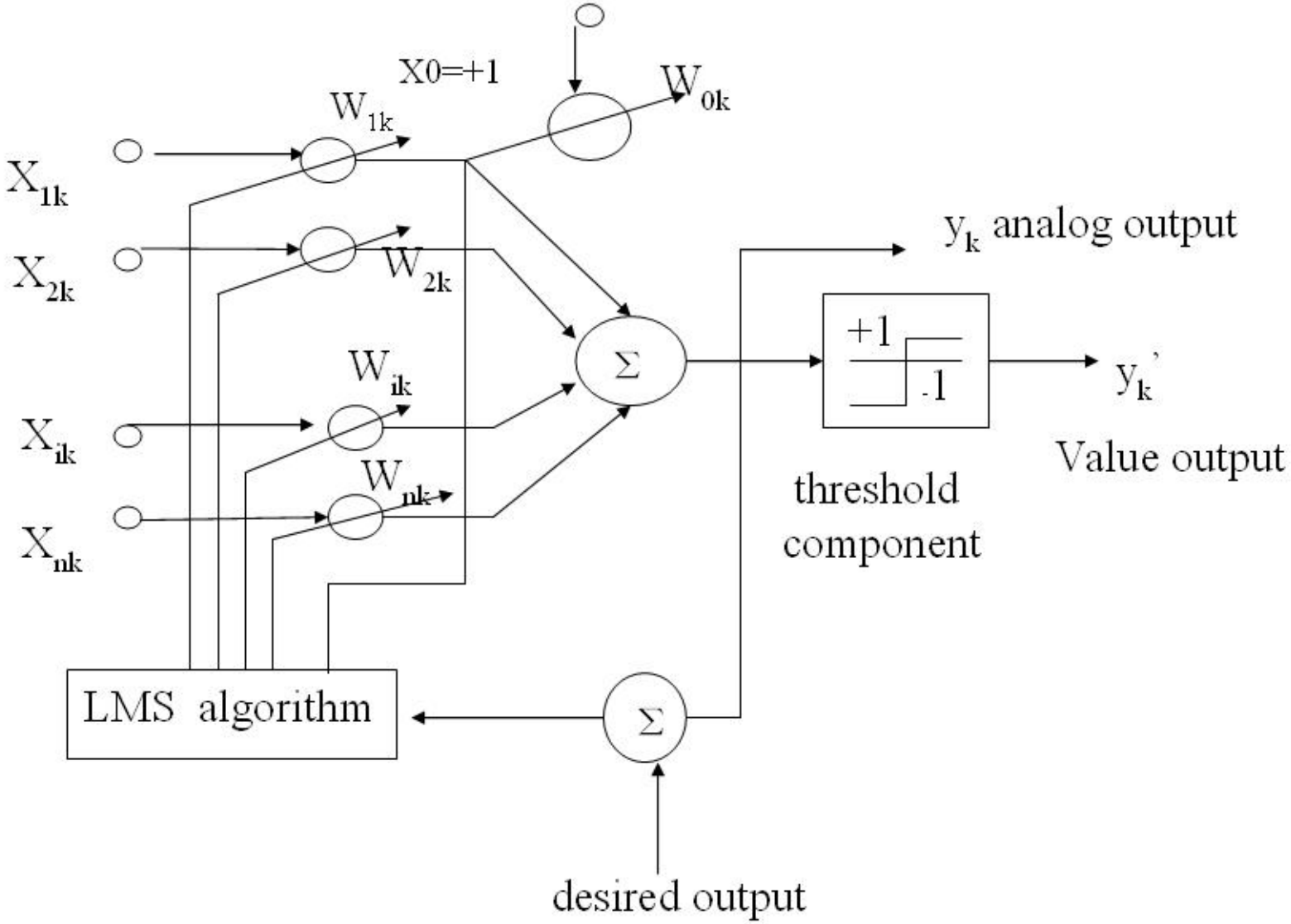


Fig.2 principle of Adaline

Bernard Widrow

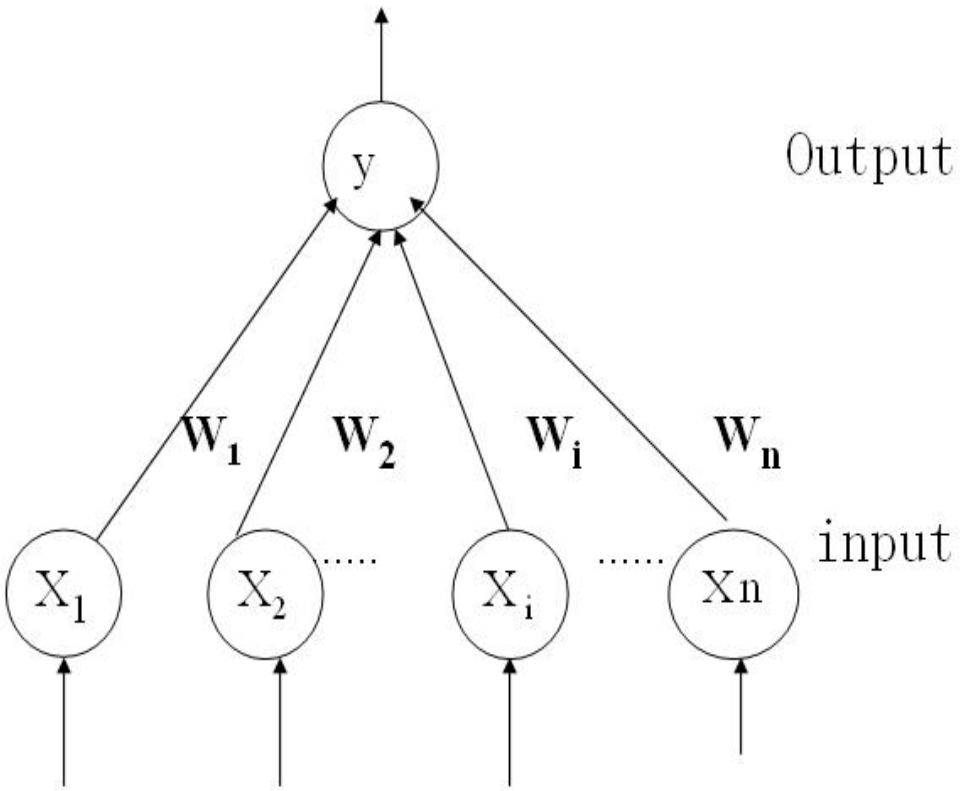
Thinking About Thinking: The Discovery of the LMS Algorithm

IEEE SIGNAL PROCESSING MAGAZINE [100] JANUARY 2005 1053-5888/05/$20.00©2005IEEE

Using this gradient estimate with steepest descent, we obtained a new adaptive algorithm as follows:

$$\begin{cases} W_{k+1} = W_k + 2\mu\varepsilon_k X_k \\ \varepsilon_k = d_k - X_k^T W_k \end{cases} \qquad (2)$$

where μ is known as the *learning rate*.



$$y(t) = w^T(t)x(t)$$

$$e(t) = d(t) - y(t)$$

$$E(t) = E[e^2(t)]$$

$$J(t) = e^2(t)$$

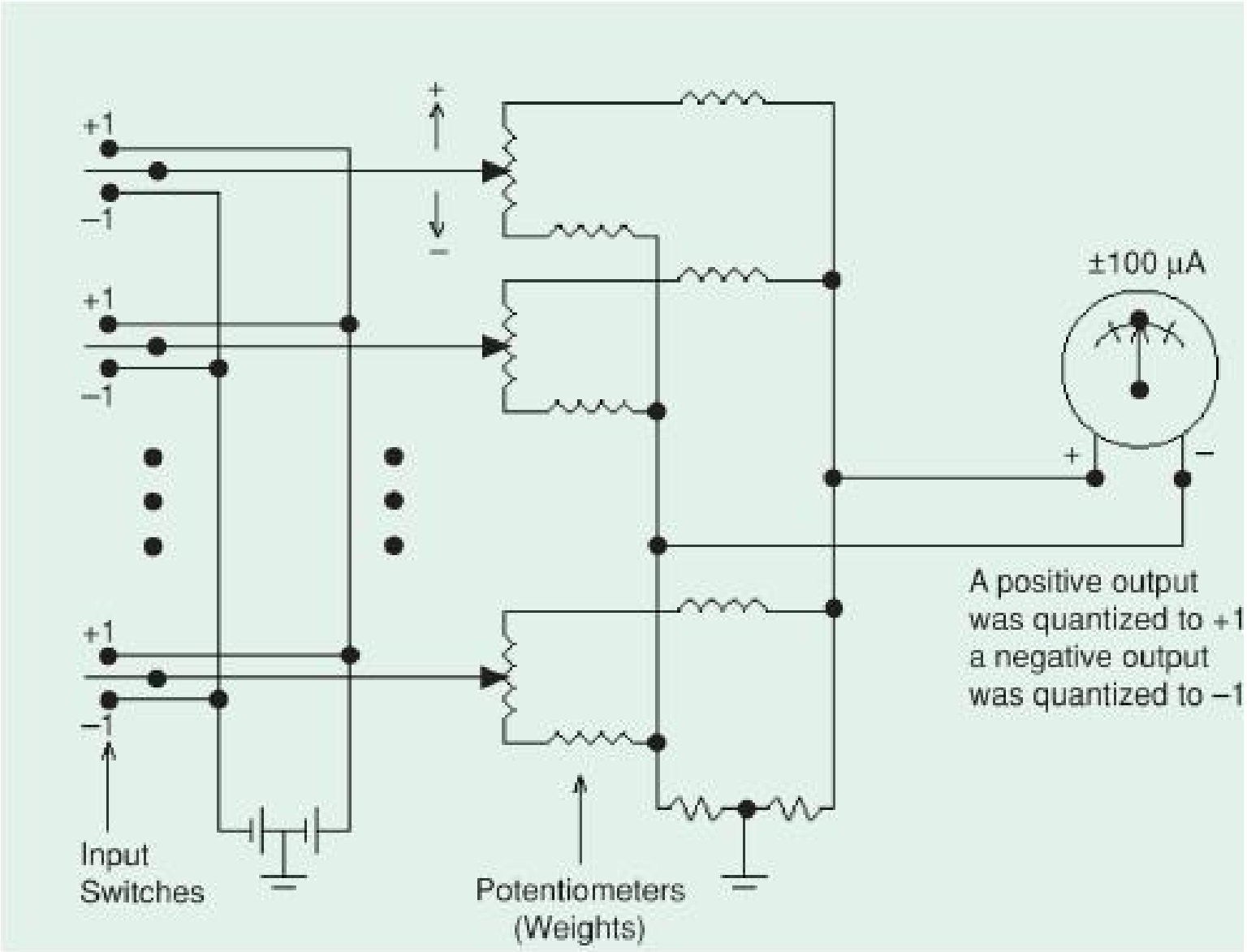
$$\frac{\partial J(t)}{\partial w(t)} = 2e(t)(-1)x(t)$$

$$w(t+1) = w(t) + \Delta w(t)$$

$$= w(t) + \mu(-\frac{\partial J(t)}{\partial w(t)})$$

$$= w(t) + 2\mu e(t)x(t)$$

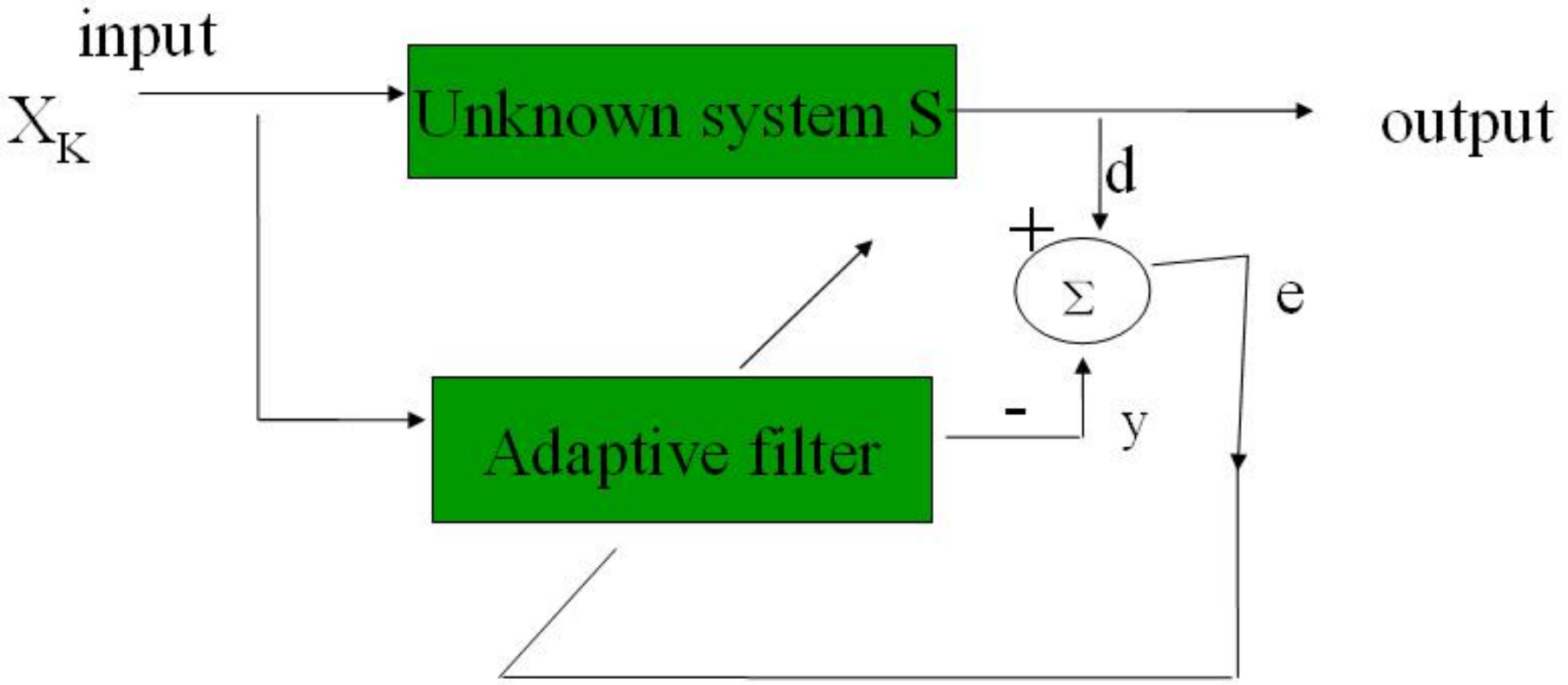
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



[FIG2] Circuit of the first adaptive neuron.

Adaline 应用举例

系统辨识



6.6.3 多层感知器学习问题

多层感知器可以解决非线性可分问题，但其隐层神经元的期望输出却不易给出。

而单层感知器学习是一种有导师指导的学习过程，因此其学习算法无法直接用于多层感知器。

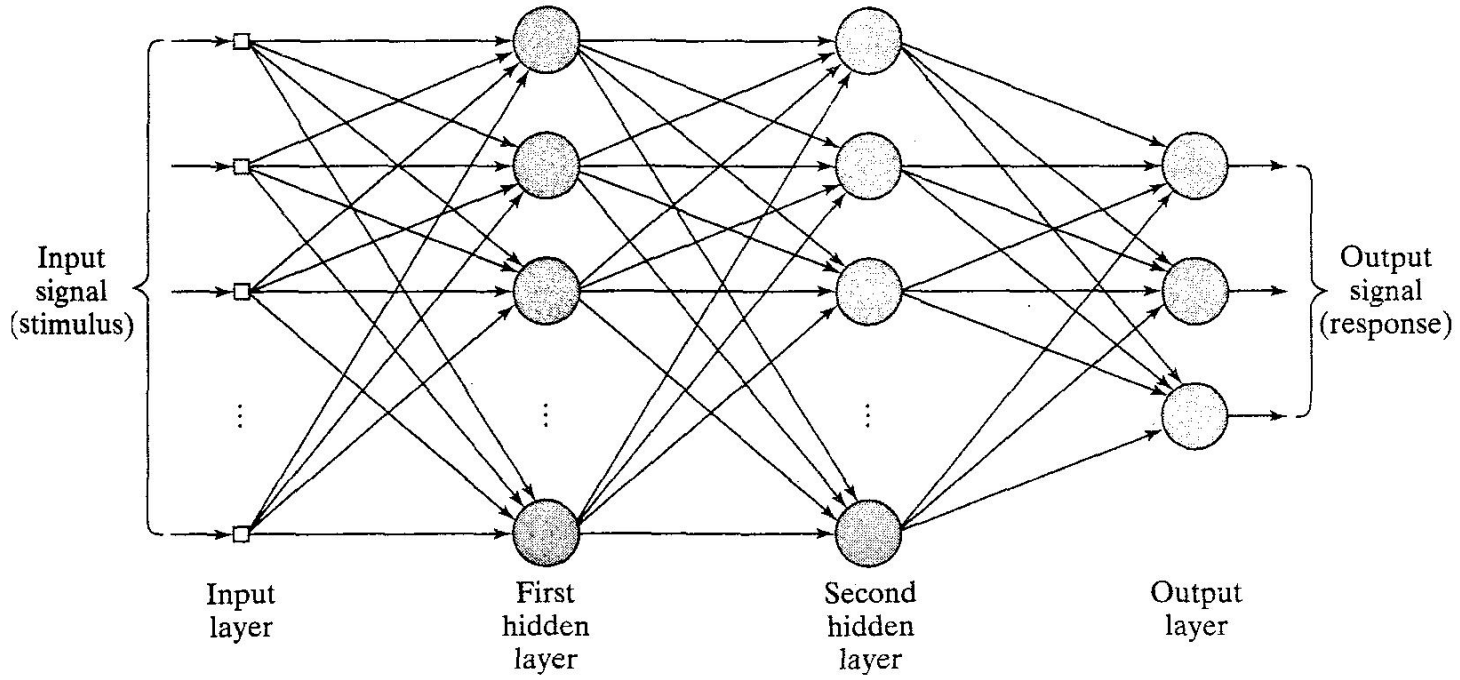


FIGURE 4.1 Architectural graph of a multilayer perceptron with two hidden layers.

BP算法

BackPropagation算法的提出：Paul Werbos于1974年首次给出了如何训练一般网络的学习算法，而人工神经网络只是其中的特例。不巧的，在当时整个人工神经网络社群中却无人知晓Paul Werbos所提出的学习算法。直到80年代中期，BP算法才重新被David Rumelhart、Geoffrey Hinton及Ronald Williams发现，并获得了广泛的关注。还有一些人声称更早发现BP算法，如Yann LeCun、Shun-ichi Amari。

BP算法的学习过程是由工作信号的正向传播和误差信号的反向传播组成的。所谓正向传播，是指输入模式经隐层到输出层，最后形成输出模式；所谓误差反向传播，是指从输出层开始逐层将误差传到输入层，并修改各层联接权值，使误差信号为最小的过程。

以下讲解没有采用书中的内容，原因：

（1）书中没有配图，而BP算法的推导不对应图很难理解

（2）书中是三层网络，我们希望给出多层感知机网络通用的BP算法公式，以便后边深度学习能够采用。

内容取自Simon Haykin, Neural Network and Learning Machine 一书。

Case 1 Neuron *j* Is an Output Node

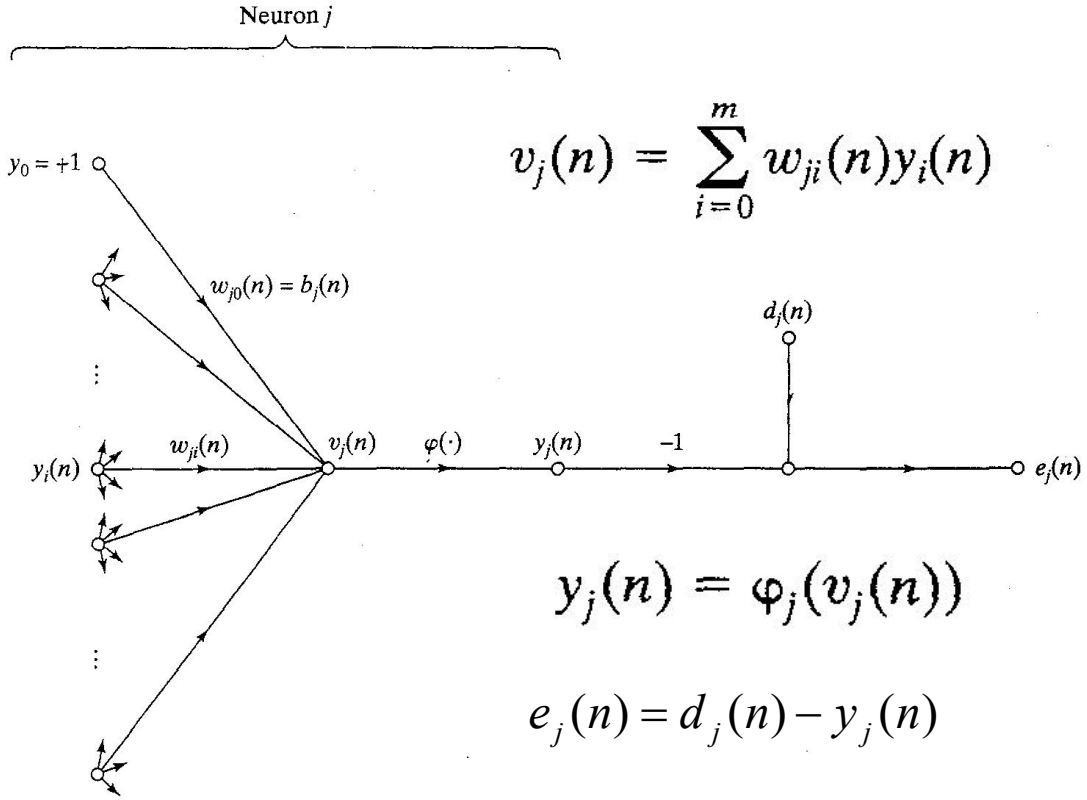


FIGURE 4.3 Signal-flow graph highlighting the details of output neuron *j*.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

$$e_j(n) = d_j(n) - y_j(n)$$

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial v_j(n)}$$

$$= -\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$= e_j(n) \varphi_j'(v_j(n))$$

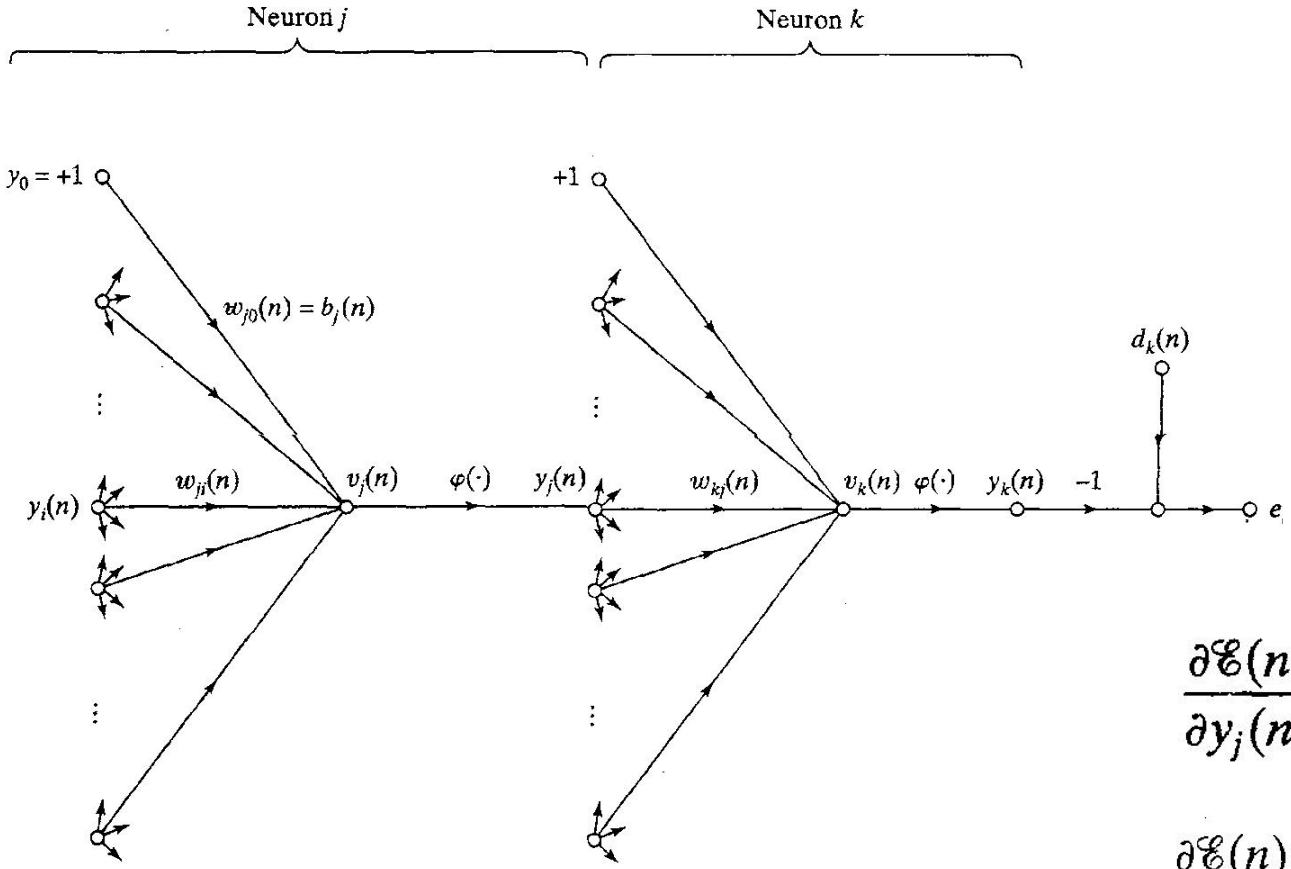
$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \varphi_j'(v_j(n)) y_i(n)$$

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n)$$

$$= w_{ji}(n) + \eta \delta_j(n) y_i(n)$$

Case 2 Neuron *j* Is a Hidden Node



$$\mathscr{E}(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n),$$

$$\Delta w_{ji}(n) = -\eta\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)}$$

$$\delta_j(n) = -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}$$

$$= -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)}\varphi_j'(v_j(n)),$$

$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = \sum_k e_k(n)\frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)}$$

$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi_k'(v_k(n))w_{kj}(n)$$

$$= -\sum_k \delta_k(n)w_{kj}(n)$$

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n)$$

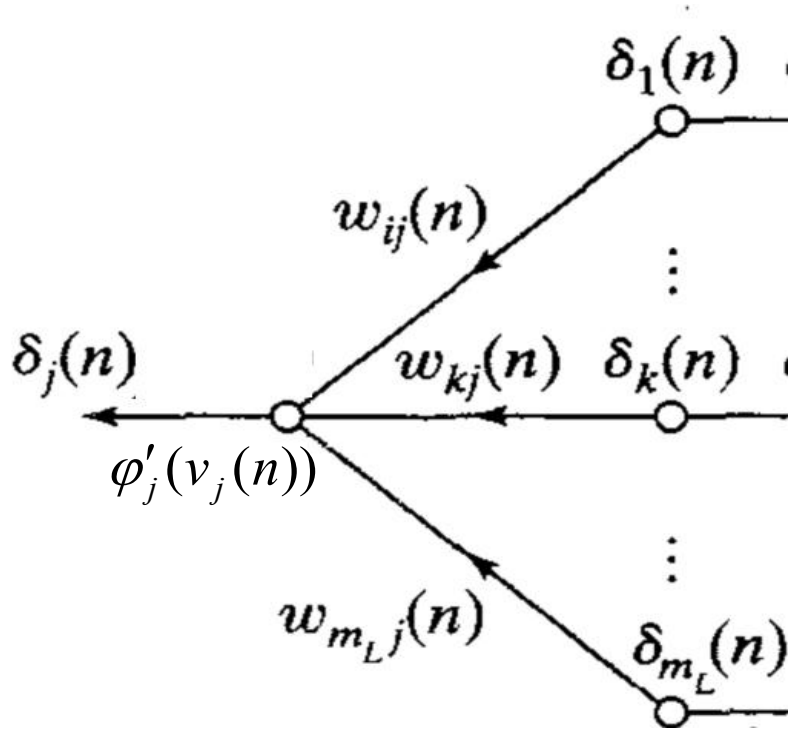
$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n)$$

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n)$$
$$= w_{ji}(n) + \eta\delta_j(n)y_i(n)$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

$$\begin{pmatrix} \text{Weight} \\ \text{correction} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{learning-} \\ \text{rate parameter} \\ \eta \end{pmatrix} \cdot \begin{pmatrix} \text{local} \\ \text{gradient} \\ \delta_j(n) \end{pmatrix} \cdot \begin{pmatrix} \text{input signal} \\ \text{of neuron } j \\ y_i(n) \end{pmatrix}$$

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$



Activation function

1. *Logistic Function*

$$\varphi_j(v_j(n)) = \frac{1}{1 + \exp(-av_j(\mathrm{n}))} \qquad a > 0$$

$$\varphi_j'(v_j(n)) = \frac{a\exp(-av_j(n))}{[1 + \exp(-av_j(n))]^2}$$

$$\varphi_j'(v_j(n)) = ay_j(n)[1 - y_j(n)]$$

Activation function

2. *Hyperbolic tangent function*

$$\varphi_j(v_j(n)) = a \tanh(bv_j(n)), \qquad (a,b) > 0$$

$$\begin{aligned}\varphi_j'(v_j(n)) &= ab\,\mathrm{sech}^2(bv_j(n)) \\ &= ab\left(1 - \tanh^2(bv_j(n))\right) \\ &= \frac{b}{a}[a - y_j(n)][a + y_j(n)]\end{aligned}$$

Summary of BP *Algorithm*

1. Initialization
2. Presentation of a training example
3. Forward computation

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \qquad (4.44)$$

where $y_i^{(l-1)}(n)$ is the output (function) signal of neuron i in the previous layer $l-1$ at iteration n and $w_{ji}^{(l)}(n)$ is the synaptic weight of neuron j in layer l that is fed from neuron i in layer $l-1$. For $i = 0$, we have $y_0^{(l-1)}(n) = +1$ and $w_{j0}^{(l)}(n) = b_j^{(l)}(n)$ is the bias

Summary of BP Algorithm

$$y_j^{(l)} = \varphi_j(v_j(n))$$

If neuron j is in the first hidden layer (i.e., $l = 1$), set

$$y_j^{(0)}(n) = x_j(n)$$

where $x_j(n)$ is the jth element of the input vector $\mathbf{x}(n)$. If neuron j is in the output layer (i.e., $l = L$, where L is referred to as the *depth* of the network), set

$$y_j^{(L)} = o_j(n)$$

Compute the error signal

$$e_j(n) = d_j(n) - o_j(n) \tag{4.45}$$

where $d_j(n)$ is the jth element of the desired response vector $\mathbf{d}(n)$.

Summary of BP Algorithm

4. *Backward Computation.* Compute the δs (i.e., local gradients) of the network, defined by

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\,\varphi_j'(v_j^{(L)}(n)) & \text{for neuron } j \text{ in output layer } L \\ \varphi_j'(v_j^{(l)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) & \text{for neuron } j \text{ in hidden layer } l \end{cases} \tag{4.46}$$

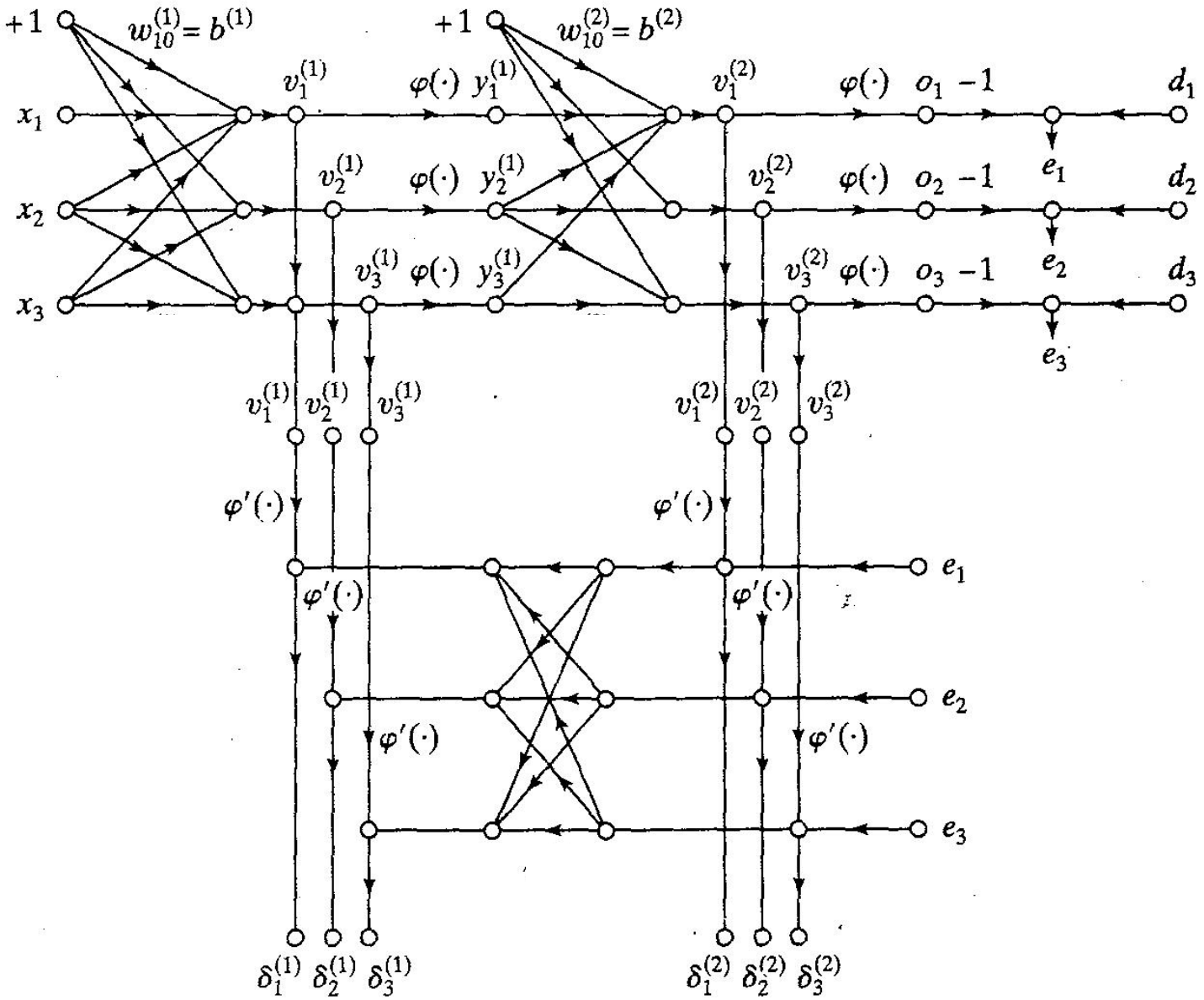
where the prime in $\varphi_j'(\cdot)$ denotes differentiation with respect to the argument. Adjust the synaptic weights of the network in layer l according to the generalized delta rule:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta\delta_j^{(l)}(n) y_i^{(l-1)}(n) \tag{4.47}$$

where η is the learning-rate parameter and α is the momentum constant.

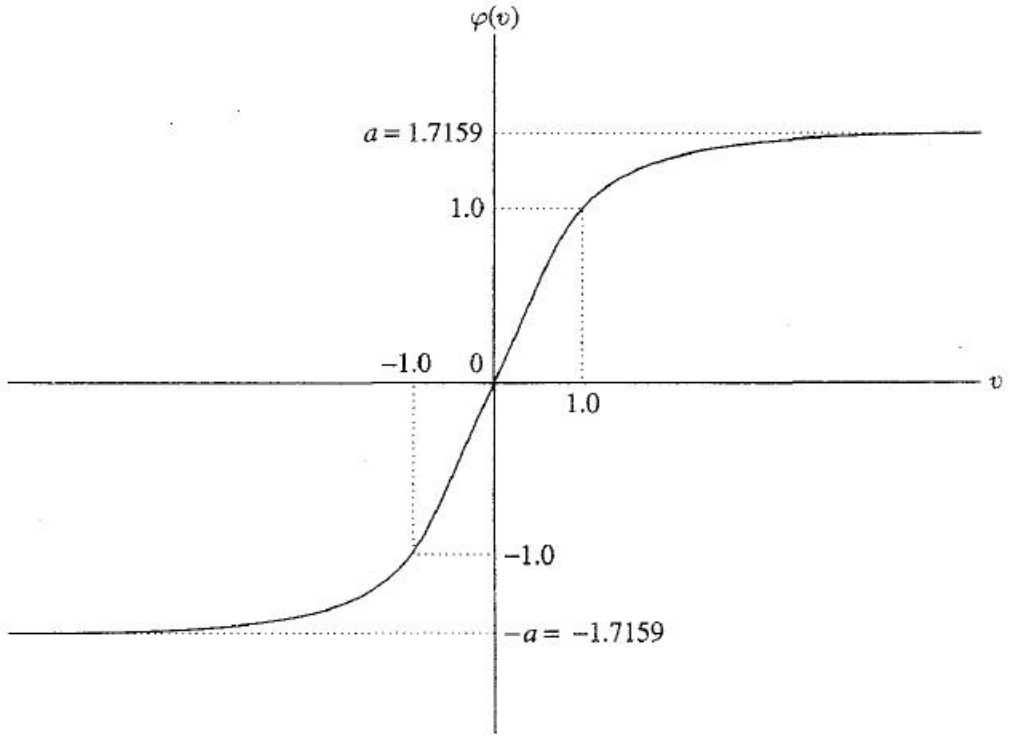
5. *Iteration.* Iterate the forward and backward computations under points 3 and 4 by presenting new epochs of training examples to the network until the stopping criterion is met.

Summary of BP Algorithm



Heuristics for performing better

- Activation function



odd

φ′(0)≈1

φ"(0)最大

a=1.7159

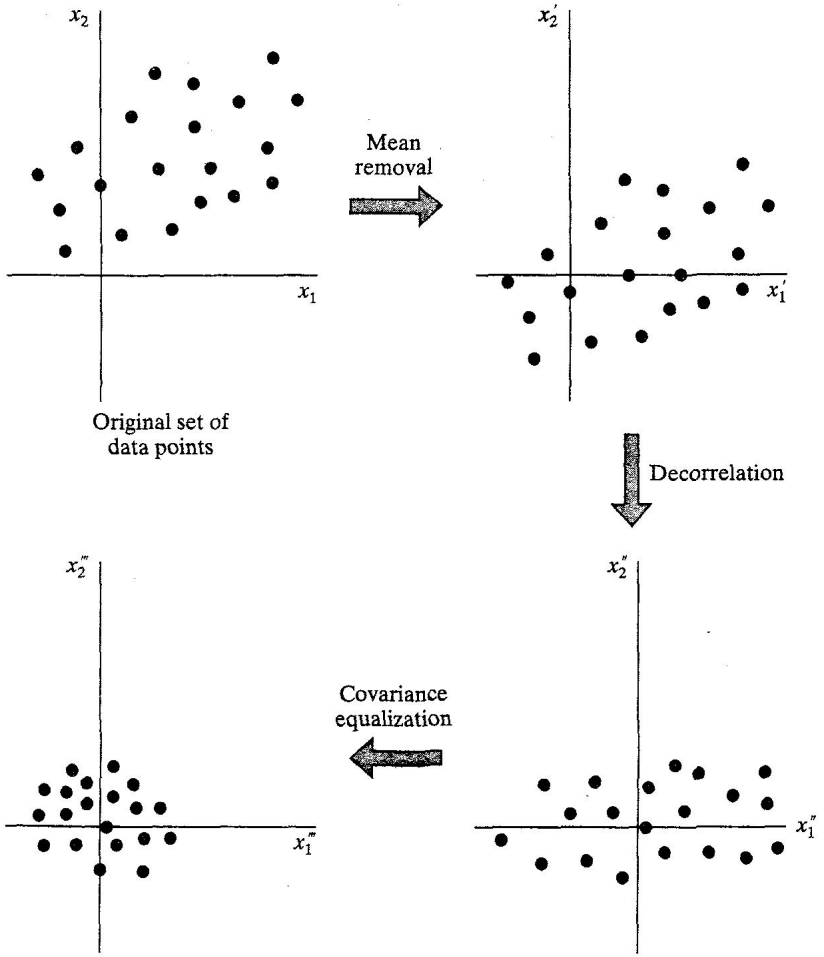
b=2/3

$$\varphi(-v) = -\varphi(v) \quad \varphi(v) = a\tanh(bv)$$

by Yann LeCun

Heuristics for performing better

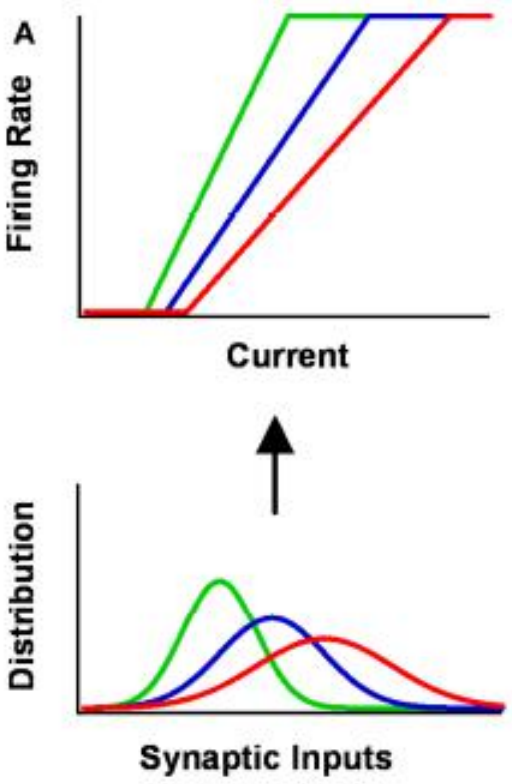
- Normalizing the inputs



Heuristics for performing better

➢ Intrinsic Plasticity

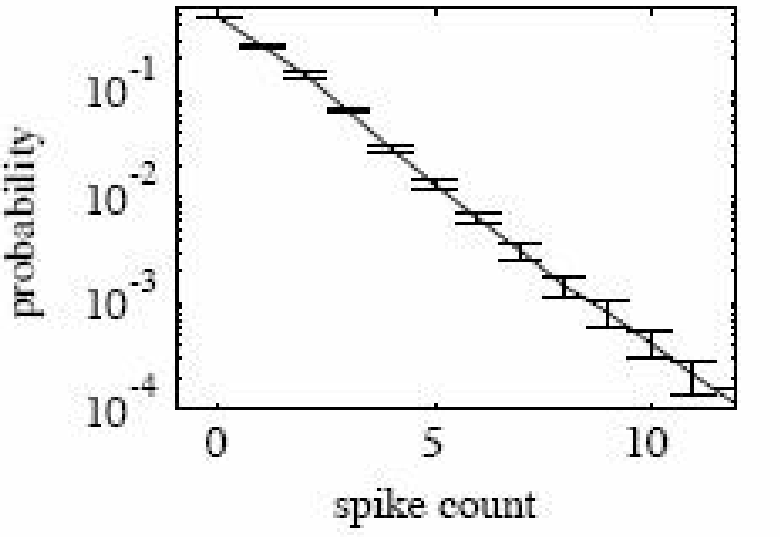
神经元自身也具有可塑性，被称为内在可塑性(Intrinsic Plasticity)。神经元可以根据过去接收到的历史输入动态地调整其响应特性，以使得在外界环境和输入变化时，神经元脉冲发放率(firing rate)的平均值仍然能大致保持在一个相对恒定的数值（homeostasis，动态平衡、自稳）。



神经元的响应曲线可随着输入分布的变化动态地调整，也就是说神经元是为了“迎合”输入的变化而做出非常聪明的改变，这样神经元就可以总是用输入-输出响应曲线的最灵敏区域去和输入的分布相对应。

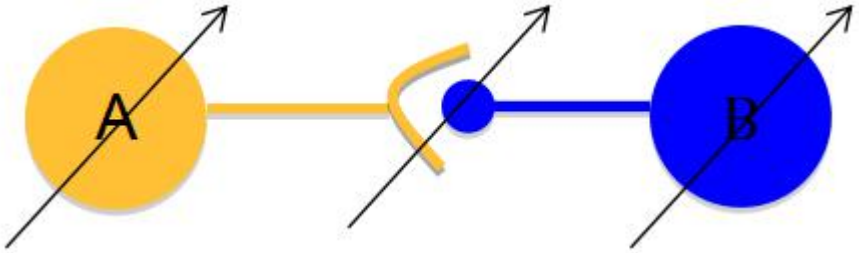
Heuristics for performing better

- Intrinsic Plasticity



- 如果在神经元发放的脉冲序列中取小的时间窗，并数其中的脉冲数目(spike count)，那么各个时间窗内的spike count 大致服从指数分布。
- 指数分布可以说明，一个时间窗内脉冲数目较小的时间窗的数量多，而脉冲多的时间窗少，换句话说，神经元趋向于保持一个较低的发放率，这样对节省信息传递的能量消耗有好处。
- 另外，由信息论知识我们知道，在固定均值的情况下的最大熵分布具有指数形式，也就是说服从指数分布则神经元可以用最小的平均放电率实现最大限度地表示信息。

Heuristics for performing better



突触可塑性和神经元内在可塑性是同时存在并互相影响的，突触的改变会影响突触后神经元的输入并进而影响其响应；突触后神经元的响应的变化又会影响其和突触前神经元之间的突触的变化，所以二者是相互影响的。

内在可塑性 + 突触可塑性　协同

李雨珂，神经网络协同学习理论及应用研究，浙大博士论文

Heuristics for performing better

- Initialization
- Learning rate
- Momentum term
- Order of training data
- …….

4. BP算法的讨论

BP算法是目前使用较多的一种神经网络学习算法，其主要优、缺点如下：

优点

（1）算法的优点是算法推导清楚，学习精度较高；

（2）从理论上说，多层前馈网络可学会任何可学习的东西；

（3）经过训练后的网络，运行速度快，可用于实时处理。

缺点

（1）由于其数学基础是非线性优化问题，因此可能陷入局部最小区域；

（2）算法收敛速度慢，通常需要数千步或更长，甚至还可能不收敛；

（3）网络中隐层结点数的设置无理论指导。

上述缺点的解决办法

对于局部最小区域问题，通常需要采用模拟退火算法或遗传算法。

对于算法收敛慢的问题，其主要原因在于误差是时间的复杂非线性函数。为提高算法收敛速度，可采用逐次自动调整学习因子η，或修改激活函数$f(x)$的方法来解决。

4. BP算法的讨论

- **Cybenko's Theorem**

  G. Cybenko (1989) Approximation by superpositions of a sigmoidal function

- **Extreme Learning Machine**

  random kitchen sinks, fastfood, random approximation

6.6.1 连接学习规则

1. Hebb学习规则

Hebb学习的基本思想

如果神经网络中某一神经元同另一直接与它连接的神经元同时处于兴奋状态，那么这两个神经元之间的连接强度将得到加强，反之应该减弱。其对连接权值的调整可表示为：

$$w_{ij}(t+1) = w_{ij}(t) + \eta[x_i(t)x_j(t)]$$

其中，$w_{ij}(t+1)$表示对时刻t的权值修正一次后所得到的新的权值；

η是一正常量，也称为学习因子，它取决于每次权值的修正量；

$x_i(t)$、$x_j(t)$分别表示t时刻第i个和第j个神经元的状态。

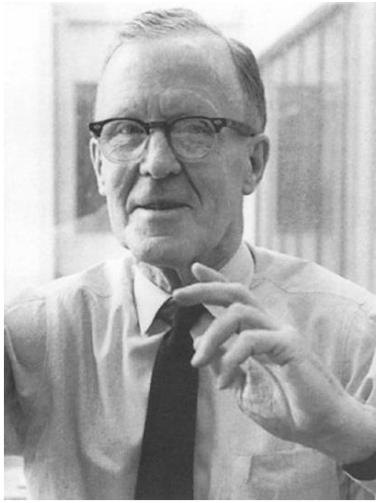
Hebb学习分析

Hebb学习是联结学习中影响较大的一种学习方法，认为对神经元重复同一刺激就可以产生性质相同、程度增强的反应。

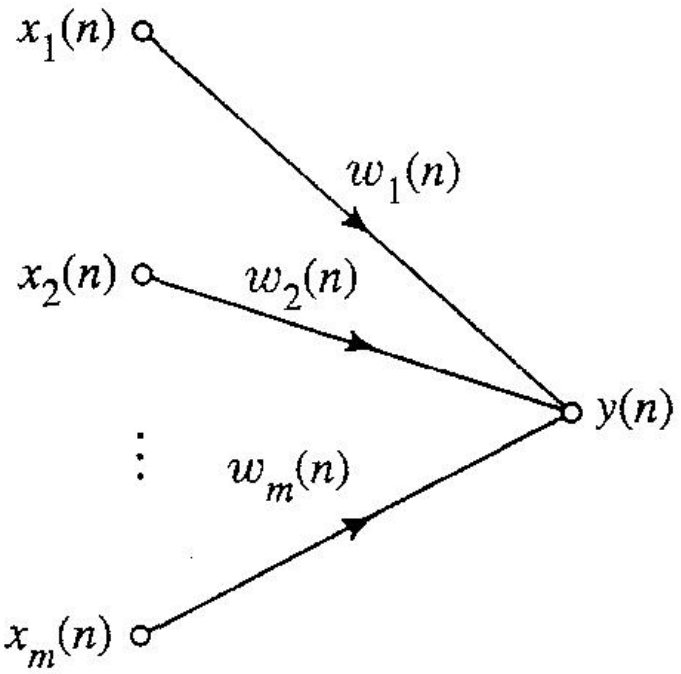
但神经生理学并没有得到直接证据，它不符合生物机理。[X]

Hebbian Rule

- When an axon of cell A is near enough to excite a cell B and repeatedly of persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B is increased .
- Expand and rephrase it as a two-part rule:
  - If two neurons on either side of a synapse (connection) are activated simultaneously (i.e. synchronously), then the strength of that synapse is selectively increased.
  - If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.



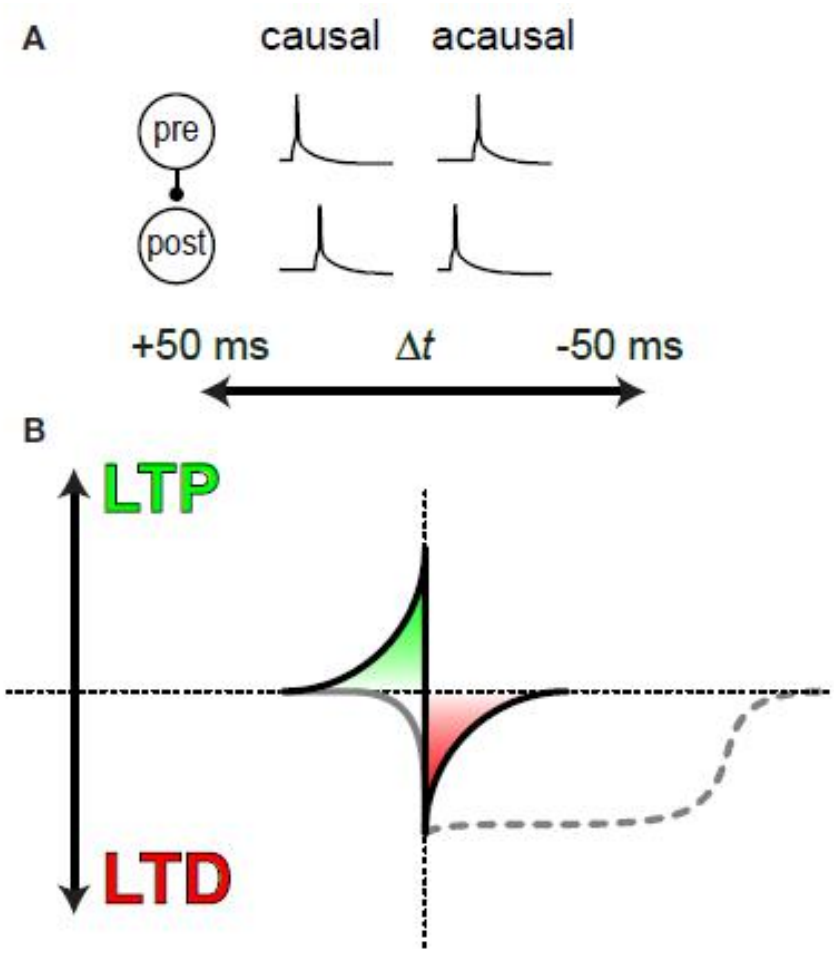
$$y = \sum_{i=1}^{m} w_i x_i$$



$$w_i(n + 1) = w_i(n) + \eta y(n) x_i(n), \qquad i = 1, 2, \ldots, m$$

where n denotes discrete time and η is the *learning-rate parameter*. However, this learning rule in its basic form leads to unlimited growth of the synaptic weight w_i,

Spike-Timing Dependent Plasticity



Gerstner, W., Kempter, R., van Hemmen, J.L., and Wagner, H.(1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature* 383, 76–81.

Henry Markram
Mu-ming Poo, Yang Dan

Henry Markram, Wulfram Gerstner and Per Jesper Sjöström, A history of spike-timing-dependent plasticity, *Frontiers in Synaptic Neuroscience*, 2011.

Oja's rule

$$w_i(n+1) = \frac{w_i(n) + \eta y(n) x_i(n)}{\left(\sum_{i=1}^{m} [w_i(n) + \eta y(n) x_i(n)]^2\right)^{1/2}} \tag{8.38}$$

where the summation in the denominator extends over the complete set of synapses associated with the neuron. Assuming that the learning-rate parameter η is small, we may expand Eq. (8.38) as a power series in η, and so write

$$w_i(n+1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)] + O(\eta^2) \tag{8.39}$$

where the term $O(\eta^2)$ represents second- and higher-order effects in η. For small η, we may justifiably ignore this term, and therefore approximate Eq. (8.38) to first order in η as follows:

$$w_i(n+1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)] \tag{8.40}$$

$$x_i'(n) = x_i(n) - y(n)w_i(n) \tag{8.41}$$

which may be viewed as the *effective input* of the ith synapse. We may now use the definition given in Eq. (8.41) to rewrite the learning rule of Eq. (8.40) as follows:

$$w_i(n+1) = w_i(n) + \eta y(n) x_i'(n) \tag{8.42}$$

E. Oja, A Simplified Neuron Model as a Principal Component Analyzer, *J. Math. Biol.* (1982) 15:267-273.

Unsupervised Learning

"Many of the unsupervised learning algorithms that have been suggested for neural networks can be seen as variations on two basic methods: Principal Components Analysis (PCA) and Vector Quantization (VQ) which is also called clustering or competitive learning".

--- Geoffrey E. Hinton

7.1.2 联结学习的学习规则

3. 竞争学习规则

基本思想：竞争中获胜神经元的连接权会向着对其刺激相应模式更为有利的方向发展；而竞争失败神经元的刺激响应模式受到抑制。

竞争型学习是一种典型的无导师学习，学习时只需要给定一个输入模式集作为训练集，网络自行组织训练模式，并将其分成不同类型。

学习过程：竞争型学习的简单形式是任一时刻只允许一个神经元被激活

① 将一个输入模式送给输入层LA；

② 将LA层神经元的激活值送到下一层LB；

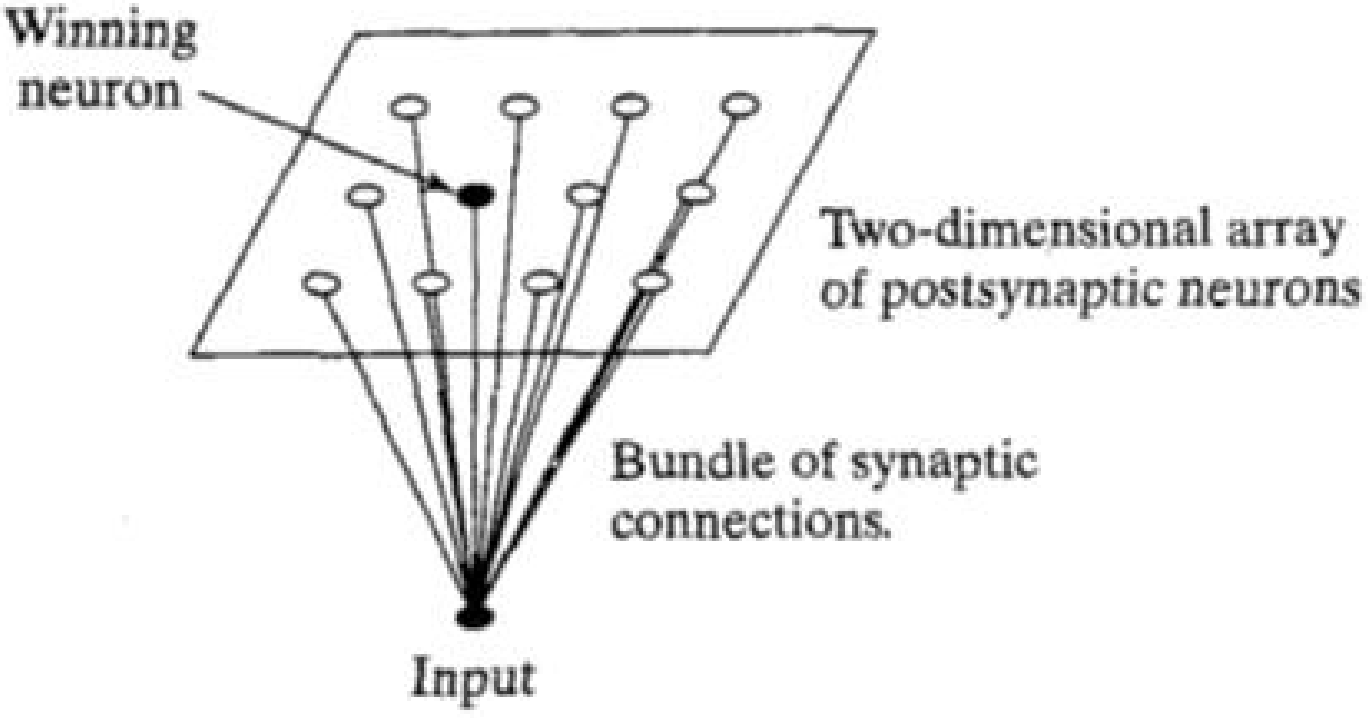
③ LB层神经元对LA层送来的刺激模式进行竞争，每个神经元将一个正信号送给自己（自兴奋反馈），同时将一个负信号送给该层其它神经元（横向邻域抑制）；

④ LB层中输出值最大的神经元为竞争获胜神经元，该神经元被激活，连接权值增强；其它神经元为竞争失败神经元，受到抑制，连接权则不变。

缺点：当有一个神经元失效时，网络中的所有信息都将丢失；并且不能表示层次结构知识等。

解决方法：放松仅有一个获胜神经元的限制，允许多个获胜者出现，学习发生在胜者集合中各神经元的连接权上。

Self-Organizing Map Model



Kohonen model

三种学习的对比

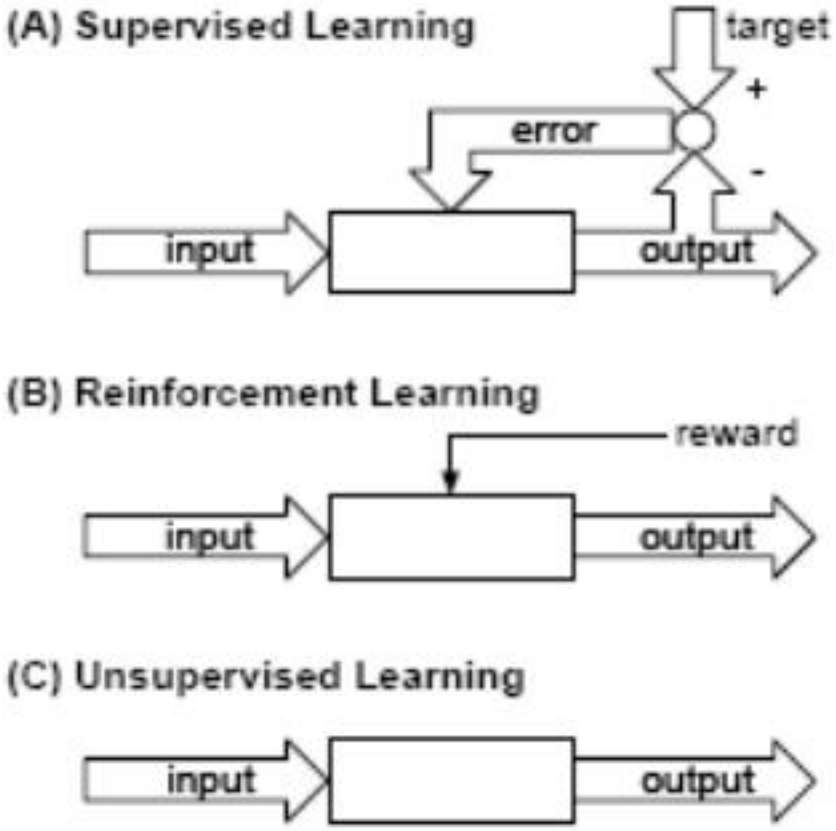
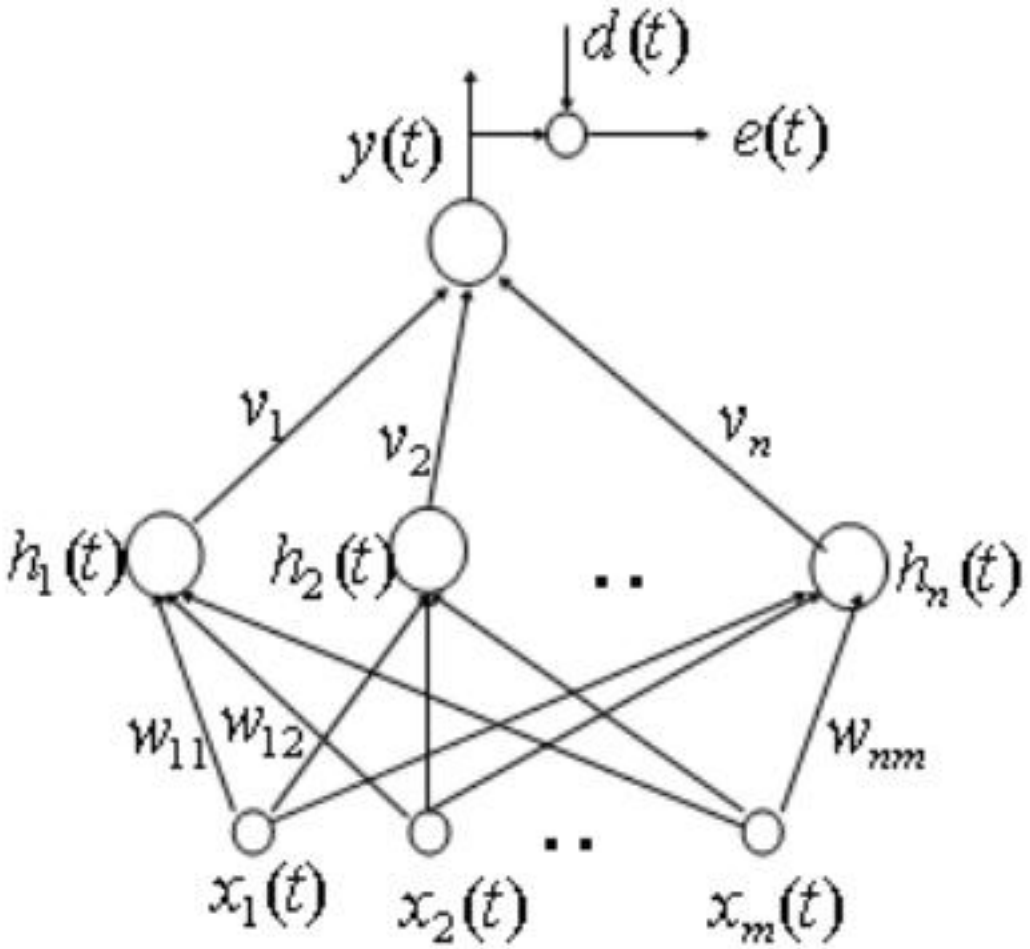


Fig. 2. Three basic learning paradigms. (a) Supervised learning by error vectors; (b) reinforcement learning by scalar reward signal; (c) unsupervised learning by statistics of input signal itself.

作 业 题

6.4 简述BP算法的思想

完成三层前馈神经网络的BP算法程序[从基础上自己写,尽量用向量表示]



$$y(t) = v^Y h(t)$$

$$h(t) = \varphi(s(t))$$

作业题

6.12　设有如下输入特征图和卷集核，请求出卷积操作后的输出特征图。

2	1	2	3
3	2	1	3
2	2	3	1
2	3	1	2

（输入特征图）

0	1
−1	0

（卷集核）

6.13　设有如下输入特征图，给定池化窗口为 2×2，请分别用最大池化法和平均池化法求出池化后的输出特征图。

3	5	2	4
2	4	5	3
6	3	5	7
5	8	6	4

（输入特征图）