第八讲

模式识别

(Pattern Recognition)

武汉大学计算机学院

Email:18986211797@189.cn

第6章 神经网络

6.1 神经网络基本知识

6.2 感知器神经网络模型

6.3 BP神经网络模型

6.4 Hopfield神经网络模型

美国女教授课堂剃光头给学生讲解大脑结构

6.1 神经网络基本知识

神经网络(Neural Networks, NNs)，又称人工神经网络(Artificial Neural Networks, ANNs)，是模拟生物神经网络进行信息处理的一种数学模型。它以对大脑的生理研究成果为基础，其目的在于模拟大脑的某些机理与机制，实现一些特定的功能。

ANNs可以用硬件电路来实现，也可以用计算机程序来模拟，是人工智能研究的一种方法。

智能可以包含8个方面：

◆感知与认识客观事物、客观世界和自我的能力

------感知是智能的基础——最基本的能力

◆通过学习取得经验与积累知识的能力

------是人类在世界中能够不断发展的最基本能力

◆理解知识，运用知识和经验分析、解决问题的能力

这一能力可以算作是智能的高级形式。是人类对世界进行适当的改造，推动社会不断发展的基本能力。

◆联想、推理、判断、决策的能力

------是智能的高级形式的又一方面

------预测和认识

------“主动”和“被动”之分(联想、推理、判断、决策的能力是“主动”的基础)

◆运用语言进行抽象、概括的能力

上述这5种能力，被认为是人类智能最为基本的能力。

作为5种能力综合表现形式的3种能力：

- 发现、发明、创造、创新的能力
- 实时、迅速、合理地应付复杂环境的能力
- 预测、洞察事物发展、变化的能力

使用计算机模拟人类的这些能力是人工智能所研究的问题，神经网络是人工智能研究的一种方法。

一. 神经网络模型基本组成

1.生物神经元

科学研究发现：人脑大约有1000亿 (10^{11})个神经元，这些神经元通过1000万亿(10^{15})个连接构成一个大规模的神经网络系统。神经元是神经网络的基本信息处理单元。

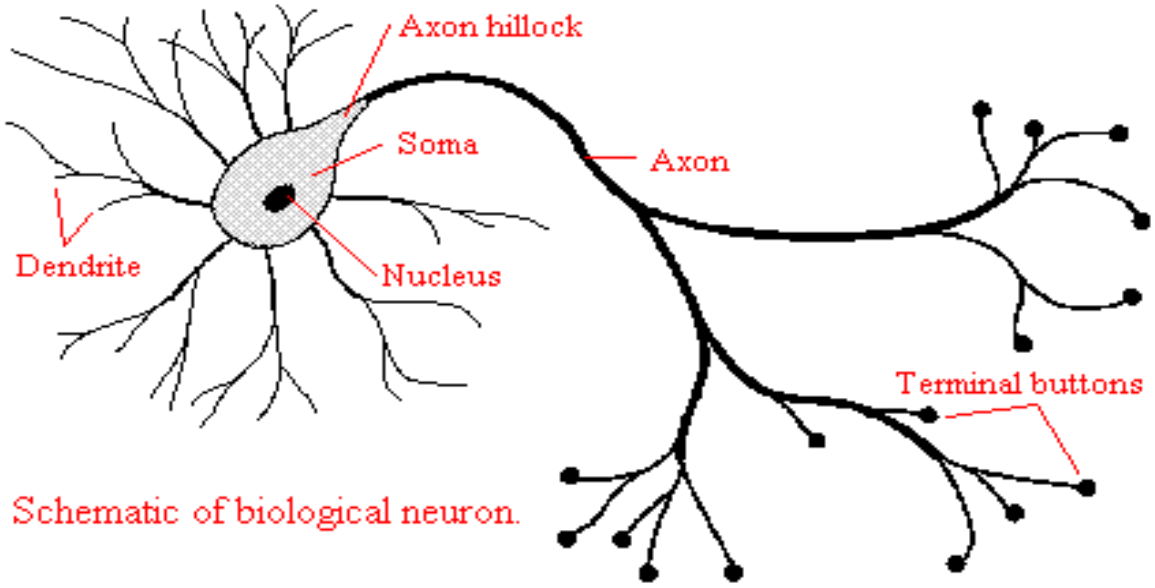
生物神经元的基本组成：

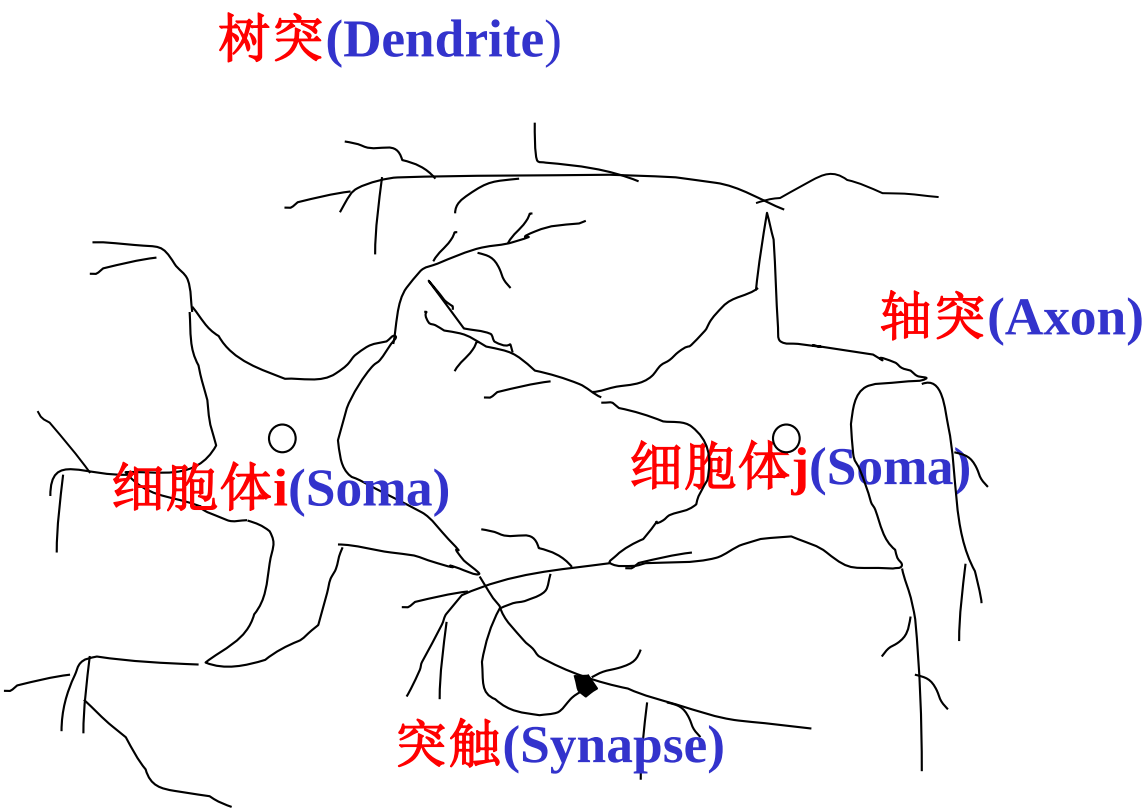
a.细胞体(soma)

b.树突(dendrite)

c.轴突(axon)

d.突触(synapse)




树突(Dendrite)
轴突(Axon)
细胞体i(Soma)
细胞体j(Soma)
突触(Synapse)

现代生理学研究表明：人类的大脑活动不是一个生物神经元所能完成的，也不是多个生物神经元功能的简单叠加，而是多个生物神经元构成的非线性动态处理系统。

在大脑神经系统中，每个神经元都通过突触与系统中的很多其他神经元相联系，突触的“连接强度”越大，接受的信号就越强，反之，突触的“连接强度”越小，接受的信号就越弱。突触的“连接强度”可以随着神经系统受到的训练而改变。例如，新记忆的形成就是通过改变突触的强度实现的，认识一位新朋友面孔的过程 包含了各种突触的改变过程。

生物神经系统六大特征：

(1)神经元及其连接；

(2)神经元之间的连接强度决定了信号的传递强弱；

(3)神经元之间的连接强度可以随训练而改变；

(4)信号可以是起刺激(excite)作用的，也可以是起抑制(inhibit)作用的；

(5)一个神经元接受信号的累积效果决定该神经元的状态；

(6)每个神元可以有一个阈值(threshold)。

2.人工神经元模型

人工神经元是人工神经网络操作的基本信息处理单位。

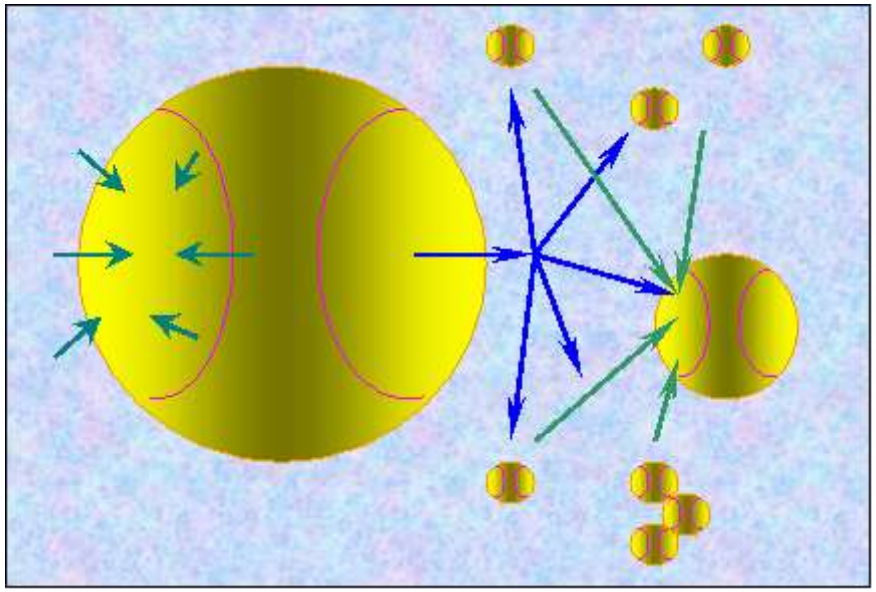
人工神经元的基本结构：

a.处理单元

b.连接

c.输入

d.输出



人工神经网络没有生物神经网络那么复杂，但它们之间有两个关键的相似之处。首先，两个网络的构成都是可计算单元(处理单元)的高度互连；其次，处理单元之间的连接决定了网络的功能。

(1)单输入神经元

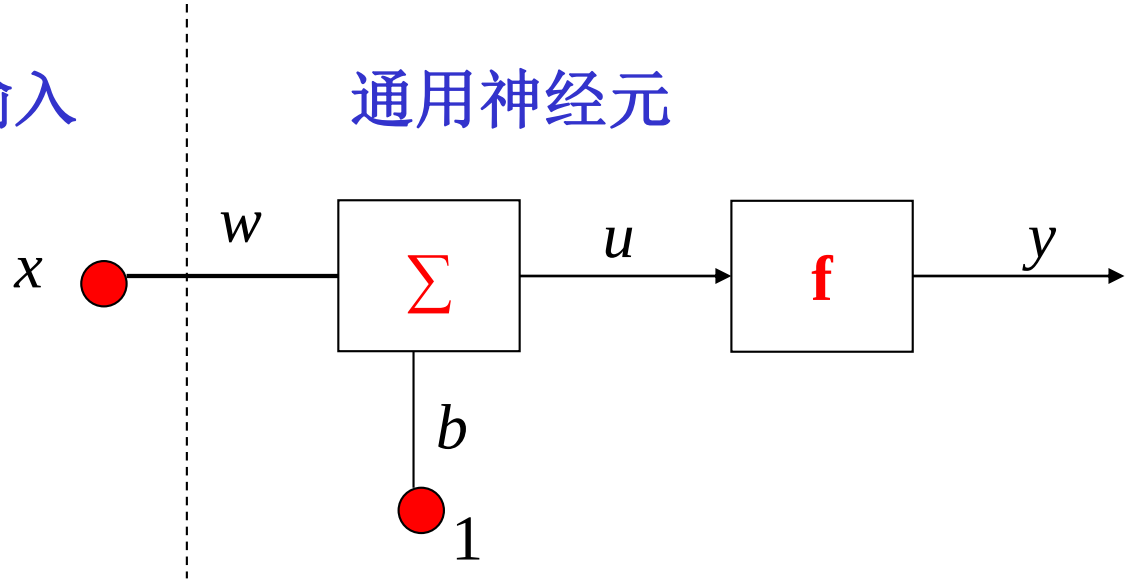


图　单输入神经元示意图

权值 偏置 净输入 传输函数

标量输入乘于标量权值w得到wx，再送到累加器，另一个输入1乘于偏置/阈值(bias/threshold)b，再将其送到加法器。加法器输出u通常被称为净输入(net input)，它被送到一个传输函数**f**(激活函数：用来限制神经元的输出幅度)，在**f**中产生神经元的标量输出。

与前面的生物神经元对照，权值w对应于突触的连接强度(w为正：激活，w为负：抑制)，细胞体对应于加法器和传输函数，神经元输出y代表轴突的信号。

神经元输出按下式计算：

$y=\mathbf{f}(wx+b)$

如，若$w=3$, $x=2$, $b=-1.5$, 则：

$y=f(3\times2-1.5)=f(4.5)$

偏置值除了有常数输入值1外，它很像一个权值。但是如果不想使用偏置值，也可忽略它。

w和b是神经元的可调整标量参数。设计者可以选择特定的传输函数，在一些学习规则中调整参数w和b，以满足特定的需要。

(2)传输函数

神经元中的传输函数可以是u的线性函数或非线性函数。这里讨论常用的几种激活函数(前三种)。

$y= \mathrm{f}(u)=\mathrm{f}(wx+b)$

a.硬极限传输函数(阶跃函数/阈值函数/阈值单元)

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

用该函数可以将输入分成两类。

MATLAB函数：y=hardlim(u)

b.线性传输函数(线性单元)

$$y = f(u) = u$$

MATLAB函数：y=purelin(u)

c.Sigmoid型传输函数(非线性单元)

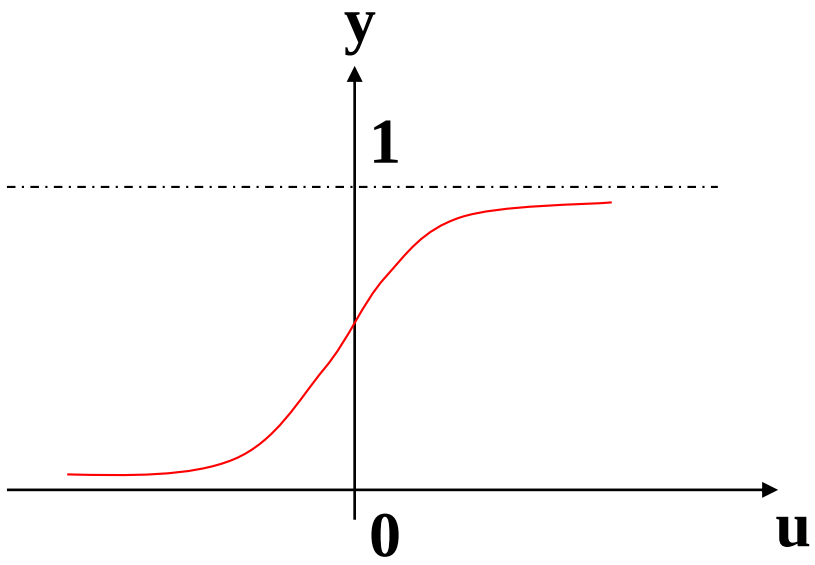
$$y = f(u) = \frac{1}{1+\exp(-u)}$$

$\{\exp(-u)$□ $e^{-u}\}$

该传输函数的输入在(-∞, ∞)区间取值，输出在0到1之间取值。

由于S型函数是可微的，能够体现数学计算上的优越性。在BP算法训练的多层网络中就采用了该传输函数。

S型函数的图形：



MATLAB函数： y=logsig(u)

d.其他传输函数

对称硬极限函数/sign函数:

$$y = f(u) = \begin{cases} -1, u < 0 \\ 1, u \geq 0 \end{cases}$$

{MATLAB函数：y=hardlims(u)}

饱和线性函数:

$$y = f(u) = \begin{cases} 0, & u < 0 \\ u, & 0 \leq u \leq 1 \\ 1, & u > 1 \end{cases}$$

(saturating linear function)

{MATLAB函数：y=satlin(u)}

对称饱和线性函数(symmetrical saturating linear function)：

$$y = f(u) = \begin{cases} 0, & u < -1 \\ u, & -1 \le u \le 1 \\ 1, & u > 1 \end{cases}$$

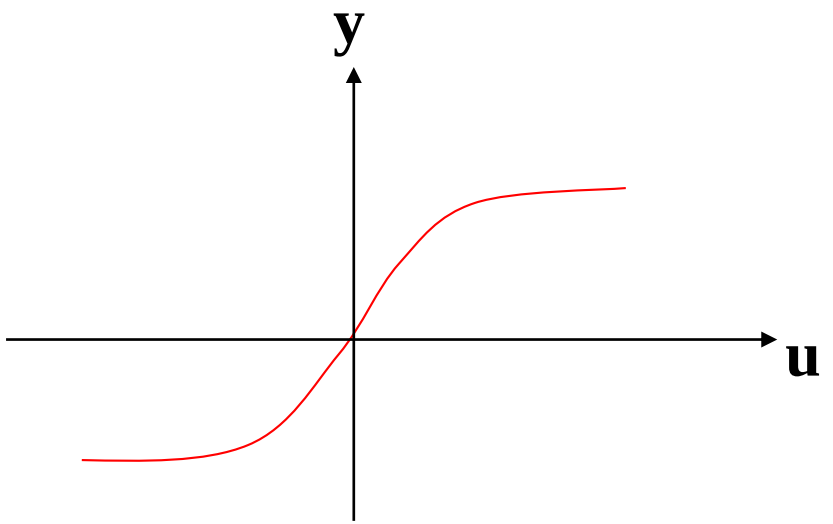
{MATLAB函数：y=satlins(u)}

对称饱和线性函数也称分段线性函数，该函数在[-1,1]线性区间内的放大系数是一致的，这种形式的传输函数可以看作是非线性放大器的近似。

双曲正切S型函数(hyperbolic tangent sigmoid function)：

$$y = f(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$$

{MATLAB函数：y=tansig (u)}



(3)多输入神经元

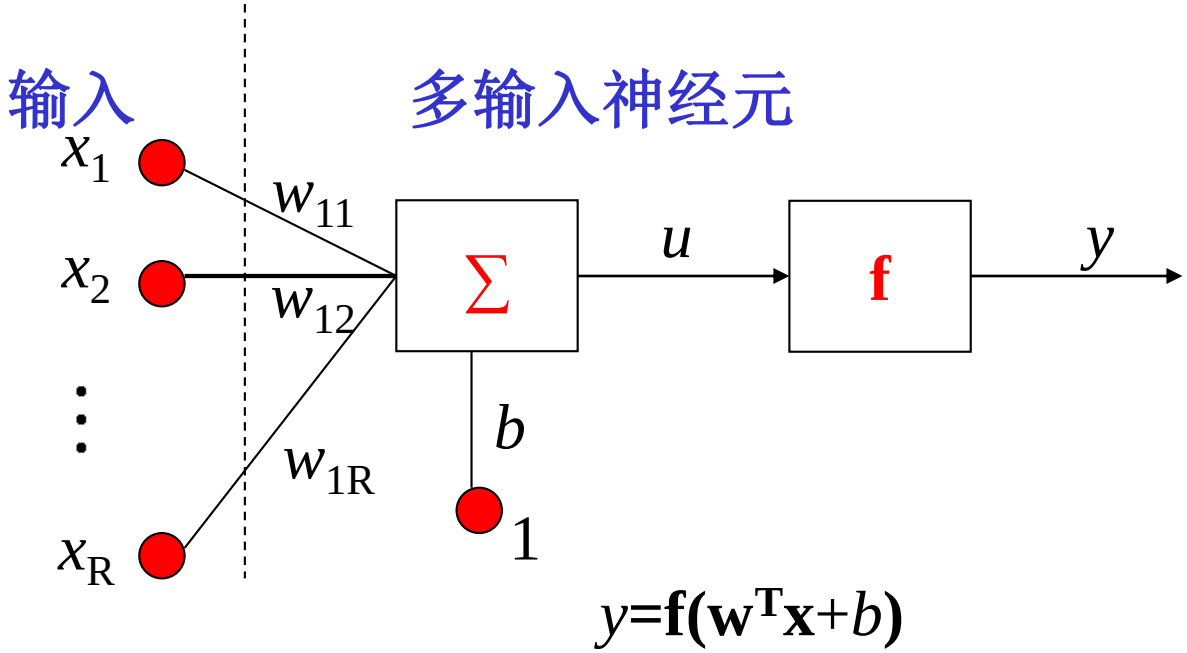


图　多输入神经元示意图

该神经元有一个偏置值b，它与所有输入的加权和累加，从而形成净输入u:

$$u = w_{11}x_1 + w_{12}x_2 + \ldots + w_{1R}x_R + b$$

该表达式可写成矩阵形式:

$$u = \mathbf{w}^T\mathbf{x} + b$$

其中,

$$\mathbf{w} = (w_{11}, w_{12}, \ldots, w_{1R})^T$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_R)^T$$

权值下标：采用人们习惯表示权值元素的下标。权值矩阵元素下标的第一个下标表示权值相应连接所指定的目标神经元编号，第二个下标表示权值相应连接的源神经元编号。

据此，w_{12}的含义是：该权值表示从第2个源神经元(此处即为第2个输入x_2)到第1个神经元的连接。

(4)神经元的层

一般来说，有多个输入的单个神经元并不能满足实际应用的要求。在实际应用中需要有多个并行的神经元。我们将这些可以并行操作的神经元组成的集合称为“层”。

下图是由S个神经元组成的单层网络。R个输入中的每一个均与每个神经元相连，权值矩阵有S行。

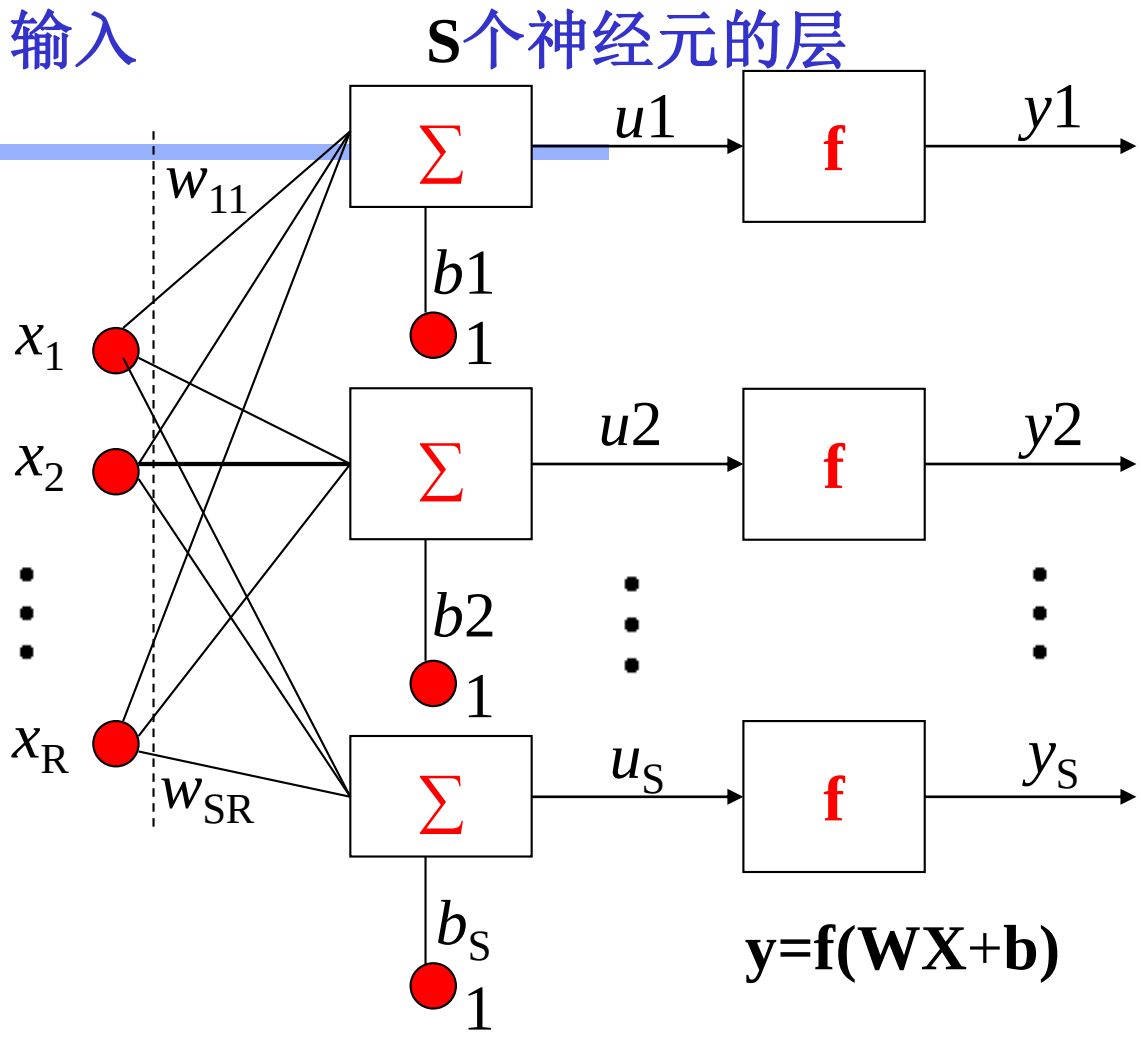

输入
S个神经元的层
w_{11}
x_1
x_2
x_R
w_{SR}
∑
∑
∑
$b1$
1
$b2$
1
b_S
1
$u1$
$u2$
u_S
f
f
f
$y1$
$y2$
y_S
y=f(WX+b)

图　S个神经元组成的层

该层包括权值矩阵**W**、加法器∑、偏置(阈值)向量**b**、传输函数框和输出向量**y**。

输入向量**x**的每个元素均通过权值矩阵**W**和每个神经元相连。每个神经元有一个偏置值b_i、一个加法器∑、一个传输函数**f**和一个输出y_i。将所有神经元的输出结合在一起，可以得到一个输出向量**y**。

通常，每层的输入个数并不等于该层中神经元的数目，即**S≠R**。

也许有人会问，同一层中所有神经元是否要有同样的传输函数？回答是否定的。可以把如上所述的两个并行操作网络组合在一起定义一种有不同传输函数的单个神经元(复合)层。两个网络都有同样的输入，而每个网络只产生一部分输出。

输入向量通过如下权值矩阵**W**进入网络：

$$\mathbf{W}=\begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1R} \\ w_{21} & w_{22} & \cdots & w_{2R} \\ \vdots & \vdots & & \vdots \\ w_{S1} & w_{S2} & \cdots & w_{SR} \end{pmatrix}$$

如前所述，矩阵**W**中元素的行下标代表该权值相应连接所指定的目的神经元，而列下标代表该权值相应连接的输入源神经元。那么，w_{32}的下标表示该元素是从第2个源神经元(此处即为第2个输入x_2)到第3个神经元的连接权值。

二. 神经网络结构

若将大量功能简单的神经元通过一定的拓扑结构组织起来，构成群体并行式处理的计算结构，这种结构就是人工神经网络。神经网络的特性及能力主要取决于网络拓扑结构及学习方法。

根据神经元的不同连接方式，可将神经网络分为两大类：分层网络和相互连接型网络。

1.分层网络

分层网络是将一个神经网络模型中的所有神经元按照功能分为若干层。一般有输入层、隐含层(中间层)和输出层，各层顺次连接。每一层的各神经元只能接受前一层神经元的输出,作为自身的输入信号。

输入层：接收外部输入模式，并由各输入单元传送至相连的隐含层各单元。

隐含层：神经网络的内部处理单元层，神经网络所具有的模式变换能力，如模式分类、特征提取等，主要体现在隐含层单元的处理，根据模式变换功能的不同，隐含层可以有多层，也可以一层都没有。

输出层：若某层的输出是网络的输出，那么称该层为输出层。输出层产生神经网络的输出模式。

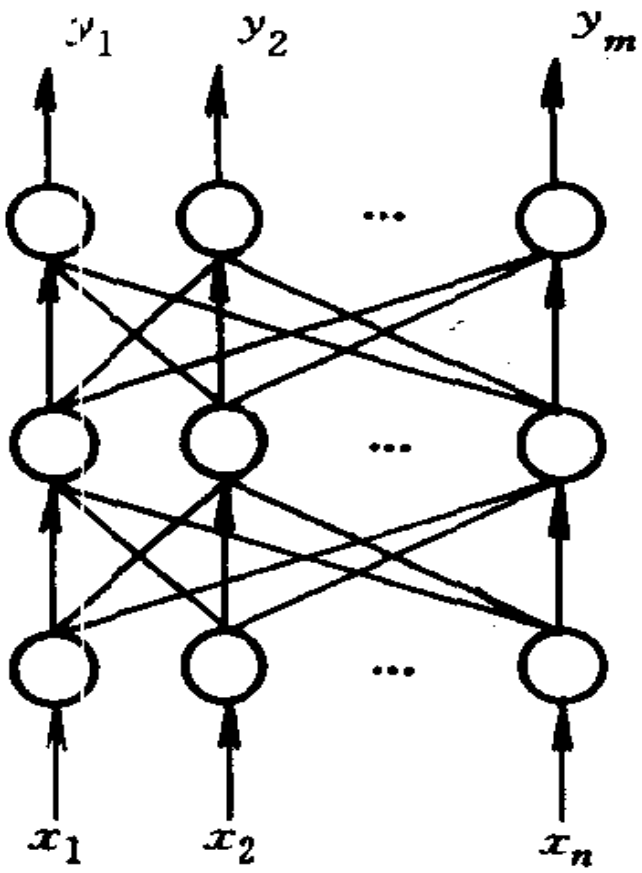
分层网络可细分为三种互连方式：

(1)前向网络(前馈网络)

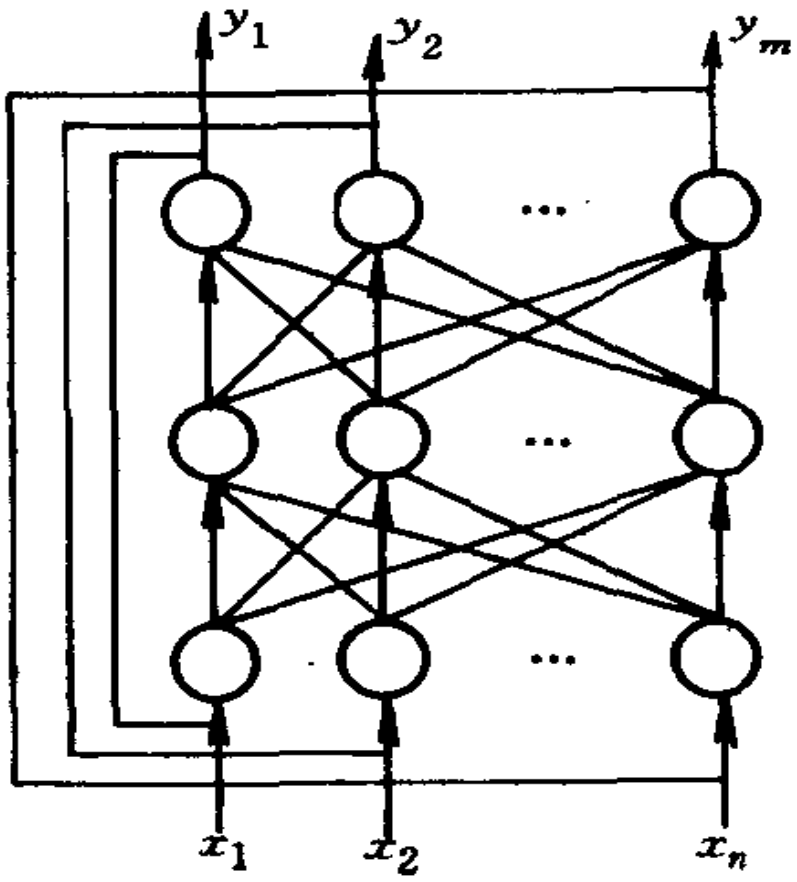
输入模式由输入层进入网络，经过中间各层的顺序模式变换，由输出层产生一个输出模式，完成一次网络状态的更新。见图a)所示。如感知器、BP(Back Propagation:误差反向传播，简称反向传播)神经网络、RBF(Radical Basis Function:径向基函数)网络采用此种连接方式。

(2)具有反馈的前向网络

反馈的结构形成封闭环路，从输出到输入具有反馈的单元也称为隐单元，其输出称为内部输出，而网络本身还是前向型的。见图b)所示。如Fukushima网络(福岛)采用此连接方式。



a)前向网络

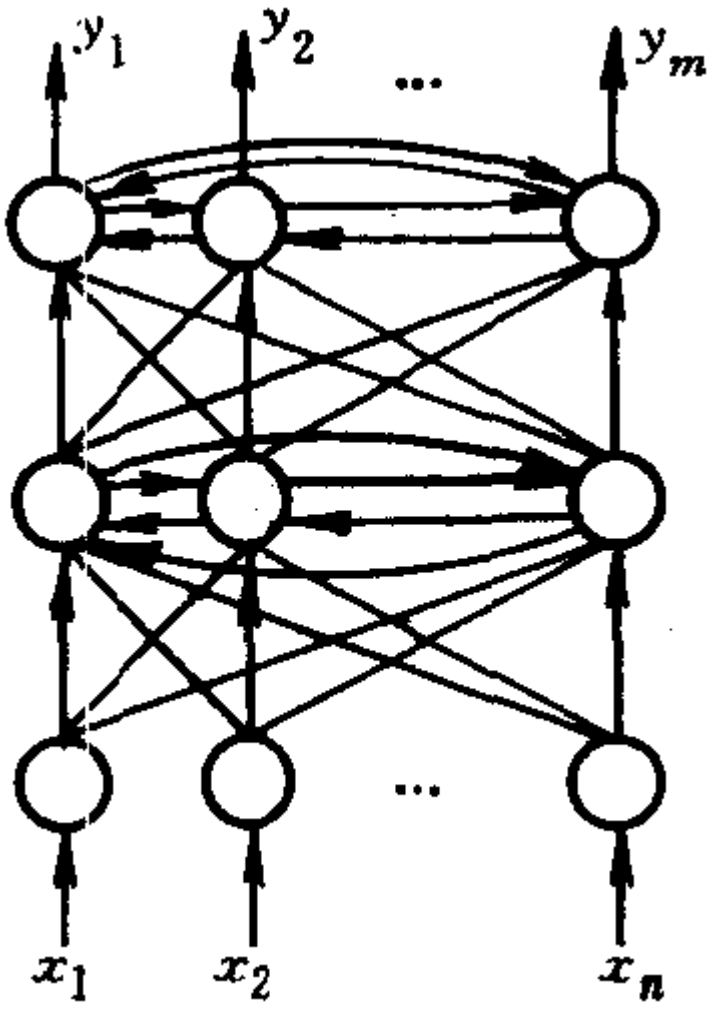


b)具有反馈的前向网络

图　神经网络的连接方式

(3)层内互连前向网络

同一层内单元的相互连接使它们彼此之间相互制约，限制同一层内能同时动作(激活)的神经元个数，而从外部看还是前向网络。见图c)所示。如很多自组织(竞争)神经网络采用此种连接。

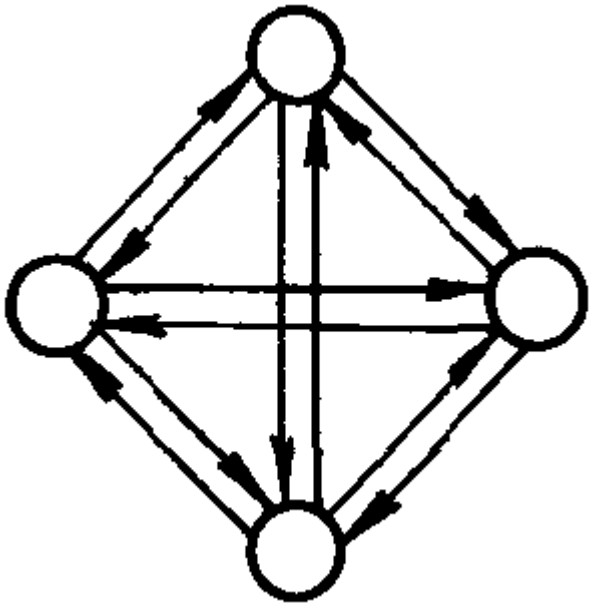


c)层内互连前向网络

图　神经网络的连接方式

2. 相互连接型网络

所谓相互连接是指网络中任意两个单元之间都是可达的，即存在连接路径。互连网络又细分为局部互连和全互连。全互连网络中每个神经元的输出都与其他神经元输入相连，而局部互连网络中，有些神经元之间没有连接关系。见图d)所示。如Hopfield网络和Boltzmann网络(又称Boltzmann machine)。



d)全互连网络

图　神经网络的连接方式

对于简单的前向网络，给定某一输入模式，网络能迅速产生一个相应的输出，并保持不变。但在相互连接的网络中，对于给定的某一输入模式，由某一网络参数出发，在一段时间内处于不断改变输出模式的动态变化中，网络最终可能产生某一稳定的输出模式，但也可能进入周期性振荡或混沌(chaos)状态。

三. 神经网络基本学习算法

学习方法是人工神经网络研究中的核心问题。

神经网络的学习也称为训练，指的是神经网络在外部环境刺激下调整神经网络的参数，使神经网络以一种新的方式对外部环境作出反应的过程。

能够从环境中学习和在学习中提高自身性能是神经网络最有意义的性质，神经网络经过反复学习达到对环境的了解。

1.学习方式

分为有导师学习、无导师学习和再励学习。

(1)有导师学习/有监督学习

(supervised learning)

需组织一批正确的输入输出数据对。对每一个输入训练样本，都有一个期望得到的输出值（也称导师信号），将它和实际输出值进行比较，根据两者之间的差值不断调整网络的连接权值，直到差值减小到允许范围之内。

(2)无导师学习/无监督学习/自组织学习

(unsupervised learning)

仅有一批输入数据。网络初始状态下，连接权值均设置为一小正数，网络按照预先设定的某种规则反复地自动调整网络连接权值，使网络最终具有模式分类等功能。 这一自组织方式，使网络具有某种“记忆”能力。

(3)再励学习/强化学习

(reinforcement learning)

介于有导师学习和无导师学习之间，外部环境对系统输出结果只给出评价(奖和罚)，而不给出正确答案，学习系统通过强化那些受奖励的动作来改善自身的性能。此学习方式更适合于控制系统应用领域。

(4)半监督学习* (Semi-supervised Learning, 了解)

是近几年来模式识别和机器学习领域的研究重点。它主要考虑如何利用少量的标注样本和大量的未标注样本进行训练和分类的问题。半监督学习对于减少标注代价，提高机器学习性能具有重大的实际意义。

基本假设:1)聚类假设：可直观地解释为如果一些模式紧凑地聚合在一起，它们就不可能属于两种类别；2)流形假设：认为高维数据总是落在低维流形上，据此假设，沿流形面上相近的点应该具有相似的类别标记，只要捕捉到数据所在的流形面就可以根据样本点之间的相似程度对未知样本的标记进行预测。

聚类假设反映的是模型的全局特征，而流形假设主要考虑是模型的局部平滑性，反映的是局部特征。

半监督学习的主要算法有五类：基于概率的算法；在现有的有监督算法基础上进行修改的方法，如半监督支持向量机方法等；直接依赖于聚类假设的方法；基于多试图的方法；基于图的方法。

2.基本学习算法

不同的学习算法对神经元的权值调整的表达式是不同的。没有一种独特的算法适用于设计所有的神经网络，选择或设计学习算法时还需考虑神经网络的结构及神经网络与外界环境相连接的形式。

权值的确定通常有两种方法：

①根据具体要求直接计算

如Hopfield网络作优化计算。

②通过学习得到

大多数人工神经网络都采用这种方法。

设v_j为神经元j的输出，v_i为神经元i对神经元j的输入，w_{ij}是神经元i与神经元j之间的连接权值，Δw_{ij}为连接权值w_{ij}的修正值，即$w_{ij}(n+1)=w_{ij}(n)+\Delta w_{ij}$。

(1)Hebb学习规则

它是由**Hebb**根据生物学中的条件反射机理，于**1949**年提出的神经元连接强度变化规则，属于无导师学习。其基本内容为：若两个神经元同时兴奋(即同时被激活)，则它们之间的突触连接加强，于是**Hebb**连接权值的学习规则可表示为：

$$\Delta w_{ij} = \eta v_i v_j$$

式中，η为学习速率参数。

Hebb学习规则是人工神经网络学习的基本规则，几乎所有神经网络的学习规则都可以看作**Hebb**学习规则的变形。

(2) δ学习规则/误差校正学习规则

误差校正学习规则是根据神经网络的输出误差对神经元的连接权值进行修正，属于有导师学习。

设d_i为神经元i的期望输出，y_i为神经元i的实际输出，$d_i - y_i$为误差信号或学习信号，神经元i到神经元j的连接权为w_{ij}。

在**Hebb**学习规则中引入教师信号，将**Hebb**学习规则公式中的y_i换成神经元i期望目标输出d_i与神经元i实际输出y_i之差，即为有监督**δ**学习规则：

$$\Delta w_{ij}(k) = \eta \cdot [d_i(k) - y_i(k)] \cdot y_j(k) = \eta \cdot \delta \cdot y_j(k)$$

$$\delta = d_i(k) - y_i(k)$$

上式表明，两个神经元之间的连接强度的变化量与教师信号$d_i(k)$和网络实际输出y_i之差成正比。

定义均方误差函数：

$$E=\frac{1}{2}\left(d_j-y_j\right)^2=\frac{1}{2}\left(d_j-f\left(\sum_k w_{kj}x_k\right)\right)^2$$

从而：
$$\frac{\partial E}{\partial w_{ij}} = -\left(d_j - y_j\right) f'\left(\sum_k w_{kj} x_k\right) x_i$$

要使期望误差达到最小，要求在负梯度方向上改变，因此有：

$$\begin{aligned}\Delta w_{ij} &= \eta\left(d_j - y_j\right) f'\left(\sum_k w_{kj} x_k\right) x_i \\ &= \eta\left(d_j - y_j\right) f'\left(neu_j\right) x_i \\ &= \eta \delta_j x_i\end{aligned}$$

其中，η为学习速率参数。 一般η选得很小；δ_j为误差函数对神经元j输入的偏导数。**BP(反向传播)**算法采用了δ_j规则。

(3) 竞争学习算法

有导师的学习算法不能充分反映人脑神经系统的高级智能学习过程，人脑神经系统在学习过程中各个细胞始终存在竞争。竞争学习网络由一组性能基本相同，只是参数有所不同的神经元构成。对于一个输入模式内各子模式的作用，每个神经元通过相互竞争来做出不同的反映，每个神经元的激活范围遵循某种特定的限制。

竞争学习的基本思想：对竞争获胜的神经元权值进行修正，获胜神经元的输入状态为**1**时，相应的权值增加，状态为**0**时权值减小；学习过程中，权值越来越接近相应的输入状态。竞争学习算法属于无导师学习。

四. 神经网络的特点及应用

1.神经网络的特点

(1)并行分布式处理

神经网络具有高度的并行结构和并行实现能力，具有高速寻找优化解的能力，能够发挥计算机的高速运算性能。

(2)非线性处理

人脑的思维是非线性的，故神经网络模拟人的思维也应是非线性的，这一特性有助于处理非线性问题。

(3)具有自学习功能

通过对过去的历史样本数据的学习，训练出一个具有归纳全部数据的特定神经网络，自学习功能对预测具有重要意义。

(4)神经网络的硬件实现

要使人工神经网络能更有效地解决大规模问题，可以采用超大规模集成电路(VLSI)实现，即把神经元和连接制作在一块芯片(多为CMOS)上构成ANNs。神经网络的VLSI设计方法近年来发展很快，硬件实现已成为ANNs的一个重要分支。

2.神经网络的应用领域

- 信号处理与分析
- 图像处理与分析
- 分类识别
- 自动控制中的智能控制
- 经济分析与决策
- 神经科学和生物学
- 信用分析
- 航空航天
- 医学诊断系统
- ……

6.2　感知器神经网络模型

一.单层感知器

感知器是由美国学者**Roseblatt**于1957年提出的有导师学习的神经网络模型。感知器是神经网络用来进行模式识别的一种最简单模型，属于前向神经网络模型，但是仅由一个神经元组成的单层感知器只能区分线性可分两类模式。

这里仅以只有一个神经元的单层感知器为例进行说明。

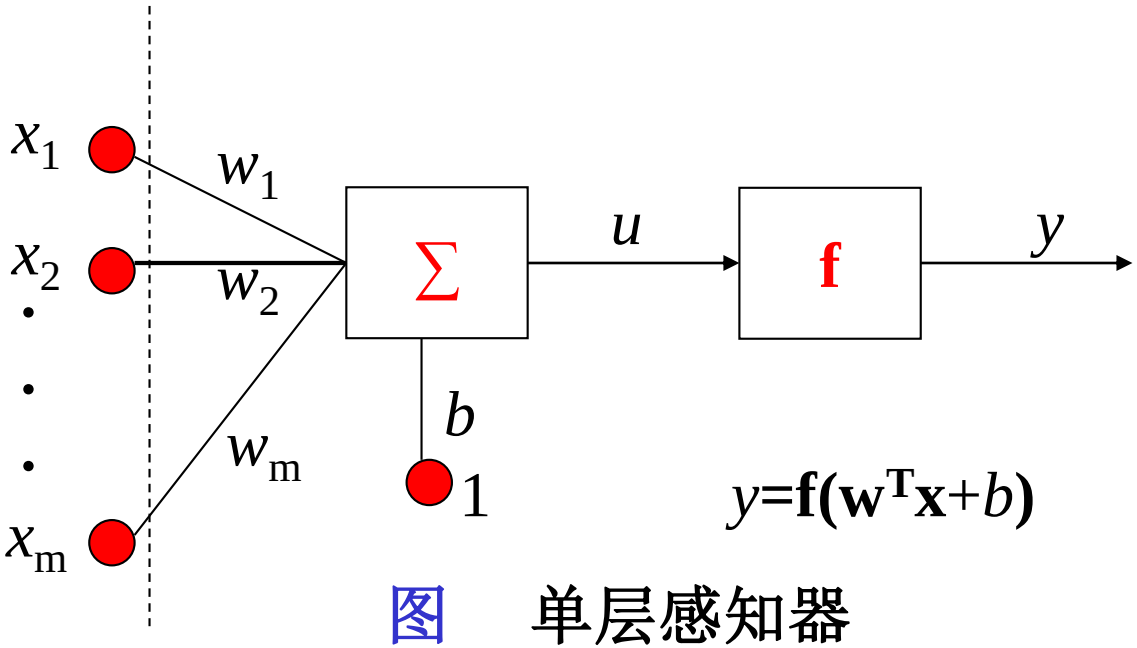


图　单层感知器

上图单层感知器中，通常**f**(u)使用符号函数或阶跃函数(二值阈值元件)。单层感知器将外部输入模式分成两类：当感知器的输出为+1时，模式**x**属于ω_1类，当感知器输出为-1时，模式**x**属于ω_2类。

单层感知器进行模式识别的决策面为：

$$\sum_{i=1}^{m} w_i x_i + b = 0$$

当它用于两类问题的模式分类时，相当于在高维样本空间中，用一个超平面将两类样本分开。

二.单层感知器的学习算法

单层感知器的学习算法是基于迭代的思想，通常采用误差校正学习规则的学习算法。考虑偏置值b及输入常量1，可将输入向量和权值向量写成增广矩阵的形式：

$$\mathbf{X}(k)=\left(x_1(k),\quad x_2(k),\quad \cdots\quad x_m(k),\quad 1\right)^T$$

$$\mathbf{W}(k)=\left(w_1(k),\quad w_2(k),\quad \cdots\quad w_m(k),\quad b(k)\right)^T$$

式中k为迭代次数,若$b(k)$用$w_0(k)$表示, 则二值阈值元件的输入可写为:

$$u=\sum_{i=0}^{m} w_i(k)x_i(k)=\mathbf{W}(k)^T\mathbf{X}(n)$$

令上式等于0,即$\mathbf{W}(k)^T\mathbf{X}(n)=0$,可得到在$m$维空间的单层感知器判决超平面。

感知器学习算法如下:

Step 1:设置变量和参数

$\mathbf{X}(k)=\left(x_1(k),\quad x_2(k),\quad \cdots\quad x_m(k),\quad 1\right)^T$为输入向量,也可看成是训练样本

$\mathbf{W}(k)=\left(w_1(k),\quad w_2(k),\quad \cdots\quad w_m(k),\quad b(n)\right)^T$为权向量

$b(k)$为偏置值, $f(.)$为传输函数, $y(k)$为网络实际输出, $d(k)$为期望输出, η为迭代次数, e为实际输出与期望输出的误差.

Step 2: 初始化

置迭代次数k=0, 给权值向量$\mathbf{W}(0)$的各个分量赋一个较小的随机非零值.

Step 3:输入样本模式$\mathbf{X}(k)$和它的期望输出$d(k)$

Step 4:计算实际输出

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

Step 5:求出期望输出和实际输出的误差e

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

根据误差e判断当前输出是否满足条件. 若满足条件则算法结束, 否则$w(k+1) = w(k)+\eta[d(k)-y(k)]x(k), k \leftarrow k+1$, 转到Step 3, 进入下一次计算过程.

在以上的学习算法中，**Step 5**需要判断是否满足算法结束条件，算法结束条件可以是误差小于设定的值 ε 或者是权值的变化已经很小。另外，在实现过程中还应设定最大的迭代次数，以防止算法不收敛时，学习算法进入死循环。

在单层感知器学习算法中，最关键的因素是引入了一个量化的期望输出，这样可以利用误差校正学习规则对权值向量逐步进行修正，最终达到问题所需要的精度。

对于线性可分的两类模式，可以证明单层感知器的学习算法是收敛的，即通过调整神经网络各连接权值可以得到合适的决策边界，从而正确区分两类模式；而对于线性不可分的两类模式，无法用一条直线区分两类模式，此时，单层感知器的学习算法不收敛，即单层感知器无法正确区分线性不可分的两类模式。

三.单层感知器的MATLAB实现

MATLAB神经网络工具箱为单层感知器的设计、训练和学习等提供了丰富的工具函数。下面介绍常用函数并在此基础上给出一个完整的例子。

1. newp-创建感知器

语法：**net=newp(PR,S,TF,LF)**

功能：创建感知器神经网络。

参数： 输入

PR-R×2矩阵，由**R**个输入的边界范围界定；

S-神经元数目；

TF-传输函数，缺省为’**harmlim**’

LF-网络学习函数，缺省为’**learnp**’

输出 **net**-所创建的感知器网络(**net**以结构体形式存储)

net常用调整参数：

net.trainParam.epochs-最大训练次数(缺省为0)

net.trainParam.goal-要求训练的目标误差(缺省为0)

……..

net权值和偏置值：

net.iw{1}:输入权值矩阵

net.b{1}:偏置值向量

2. train-神经网络训练

语法：[net,tr,Y]=train(NET,P,T,Pi,Ai,VV,TV)

功能：神经网络训练函数。

参数：

输入

NET-训练前的网络

P-网络输入

T-网络目标，缺省值为0

Pi-初始的输入延时，缺省值为0

Ai-初始的层延时，缺省值为0

VV-验证向量

TV-测试向量

输出

net-训练后的网络

tr-训练结果

Y-网络输出向量

3. sim-神经网络仿真

语法：[net,PF,AF]=sim(NET,P, Pi,Ai)

功能：神经网络仿真函数。

参数：

输入

NET-要测试的网络

P-网络输入

Pi-初始的输入延时

Ai-初始的层延时

输出

net-训练后的网络

PF-最终的输入延时

AF-最终的层延时

4. mae(mean absolute error)

语法：**perf=mae(E,w, pp)**

功能：平均绝对误差性能函数。

参数：

输入	输出
E-误差向量或矩阵 (即目标向量和输出向量之差)	**perf**-平均绝对误差
w-所有权值及偏置值向量	
pp-性能参数	

5. plotpv-在坐标图上绘制样本点

语法：**plotpv(P,T)/plotpv(P,T,V)**

功能：根据目标向量绘制感知器的样本点。

参数：

输入

P-R×Q样本矩阵

(R=2/R=3，Q为样本个数)

T-样本点的类别，为Q维向量

V-设置绘图坐标轴的显示范围

V=[x_min x_max;
y_min y_max]

若T含一维向量，则目标为0的输入向量在在坐标图中用符号’。’表示，目标为1的输入向量在在坐标图中用符号’+’表示。

若T含二维向量，则输入向量在坐标图中所采用的符号表示：[0 0]-’。’，[0 1]-’+’，[1 0]-’*’，[1 1]-’×’。

6. plotpc-在已绘制的图上加分类线

语法：plotpc(W,B,H)

功能：在存在的感知器图中绘制分类线函数。

参数：

输入

W-weight matrix

B-bias vector

H-handle to last plotted line

(H句柄用于在绘制新分类线前删除旧线)

7. learnp-感知器权值和阈值学习函数

此略。

利用MATLAB实现神经网络的步骤：

Step 1：根据应用创建一个神经网络；

Step 2：设定神经网络的训练参数，利用给定的样本对创建的神经网络进行训练；

Step 3：输入待测试模式，测试训练好的神经网络性能。

感知器MATLAB程序举例：

从待分类的模式样本中取出一部分样本及其对应的类别作为训练样本数据，使用测试样本对训练好的神经网络进行测试，并给出测试结果。

MATLAB程序清单：

```
%Filename:ex8_1.m
%给定3个训练样本
P= [-0.4 -0.5 0.6; 0.9 0 0.1];
%给定训练样本所对应的类别，用1和0来表示两种类别
T= [1 1 0];
%创建一个2个(特征)输入、样本数据各特征的取值范围都在[-1,1]之间，并且网络只有一个神经元的感知器神经网络
net=newp([-1 1;-1 1],1);
%设置网络的最大训练次数为20次，即训练20次后结束训练
net.trainParam.epochs = 20;
```

```
%使用训练函数对所创建的网络进行训练
net=train(net,P,T);
%对训练后的网络进行仿真，即根据训练后的网络和样本数据给出输出
Y=sim(net,P)
%计算网络的平均绝对误差，表示网络错误分类
E1=mae(Y-T)
%给定3个待测试的样本，检测训练好的神经网络的性能
Q=[0.6 0.9 -0.1; -0.1 -0.5 0.5];
%使用测试样本对网络进行仿真，仿真输出即为分类的结果
Y1=sim(net,Q)
```

```
%创建一个新的绘图窗口
figure;
%在坐标图中绘制测试数据点，并根据数据所对应的类别用约定的符号画出
plotpv(Q,Y1);
%在坐标图中绘制分类线
plotpc(net.iw{1},net.b{1})
```

本程序运行后命令行窗口显示结果：

```
TRAINC, Epoch 0/20
TRAINC, Epoch 3/20
TRAINC, Performance goal met.

Y =

     1     1     0

E1 =

     0

Y1 =

     0     0     1
```

使用TRAINC作为神经网络训练函数，第0次训练，最大训练次数为20次；

第3次训练；

达到目标误差要求，训练结束；

对训练样本进行仿真并给出输出；

网络平均绝对误差为0；

对测试样本进行仿真并给出输出(分类情况见后面的“训练后的分类线”图)；

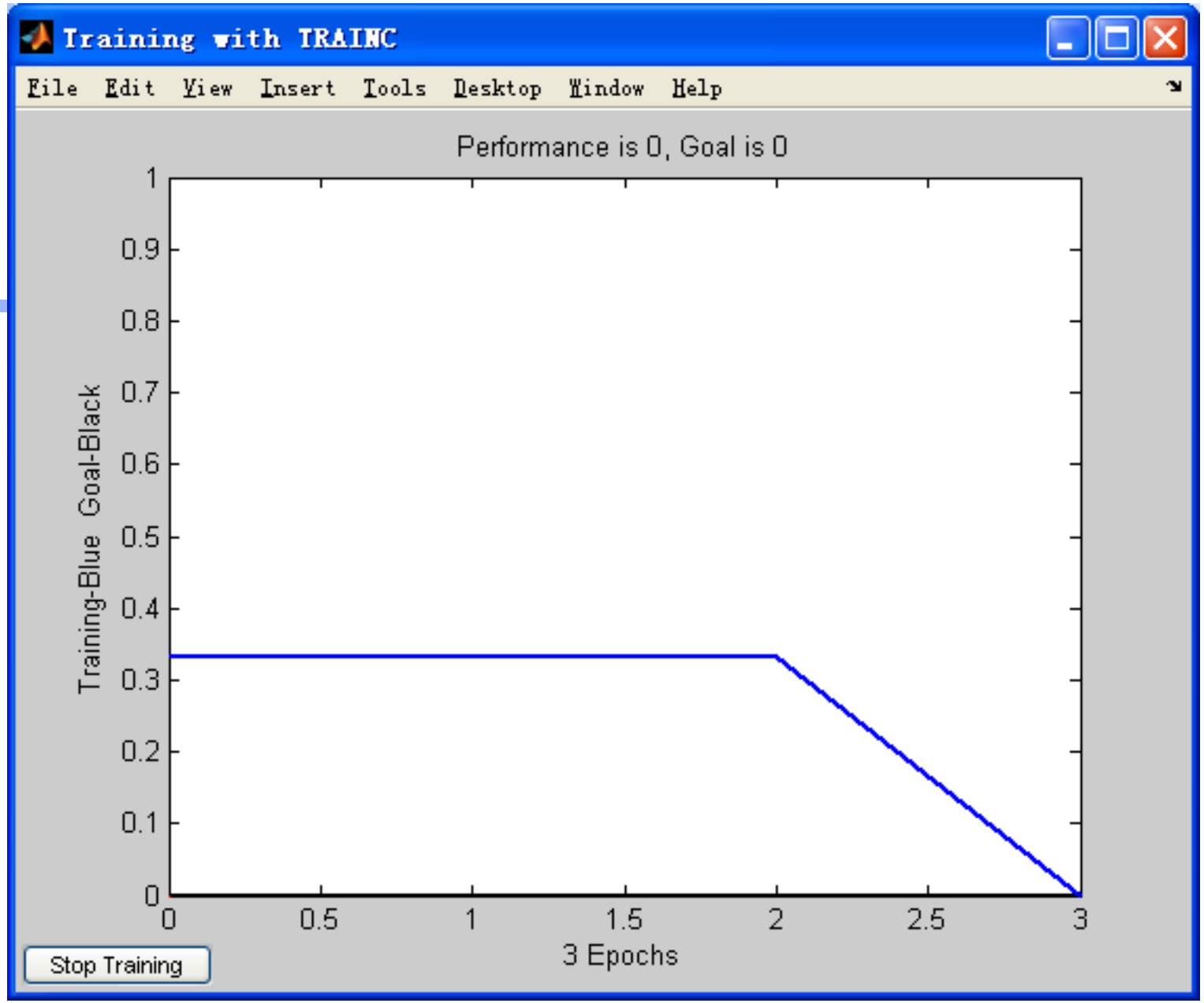

Training with TRAINC
File Edit View Insert Tools Desktop Window Help
Performance is 0, Goal is 0
Training-Blue Goal-Black
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 0.5 1 1.5 2 2.5 3
3 Epochs
Stop Training

图 训练误差曲线

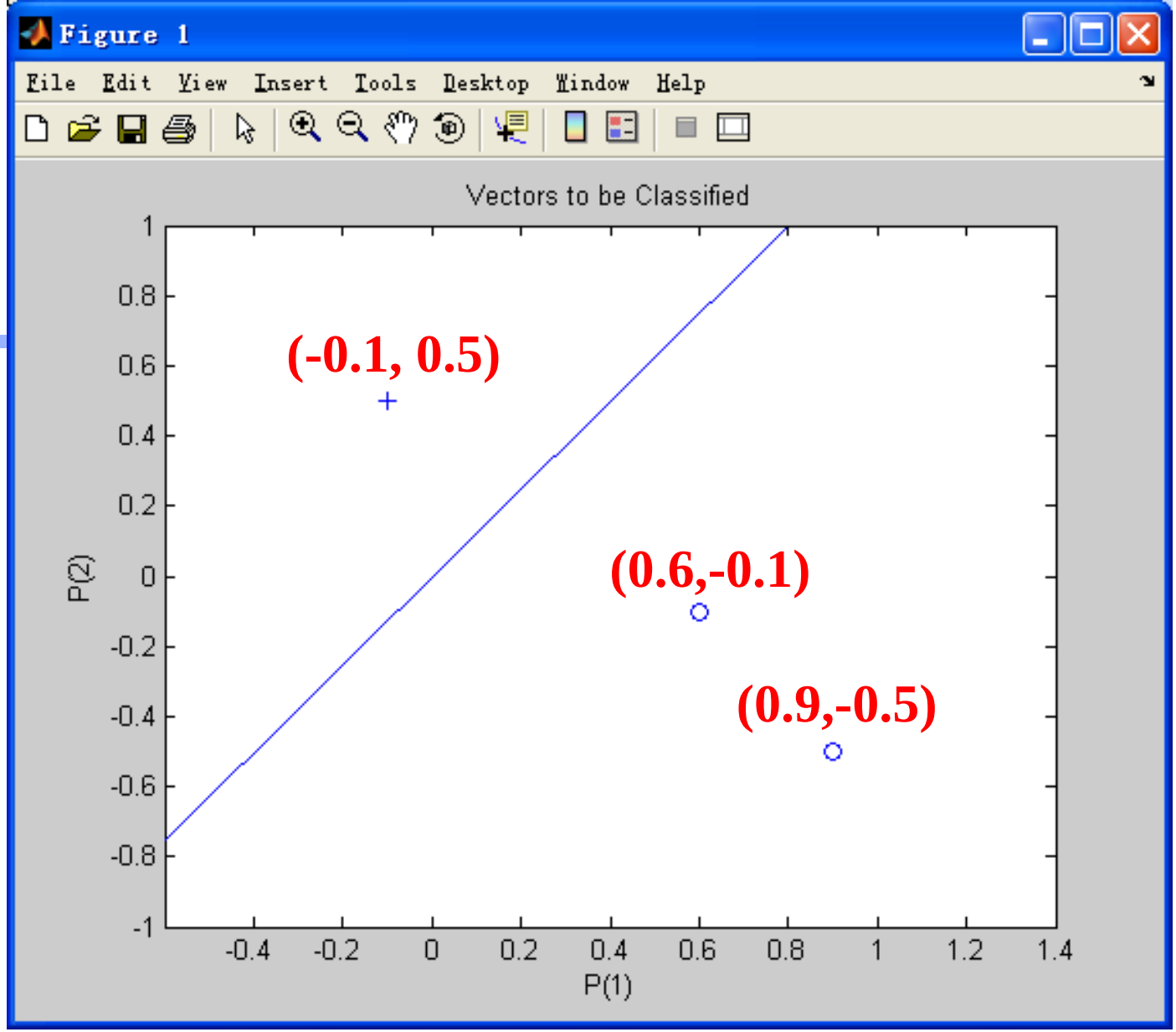


Figure 1
File Edit View Insert Tools Desktop Window Help
Vectors to be Classified
(-0.1, 0.5)
(0.6,-0.1)
(0.9,-0.5)
P(2)
P(1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4

图　训练后的分类线