

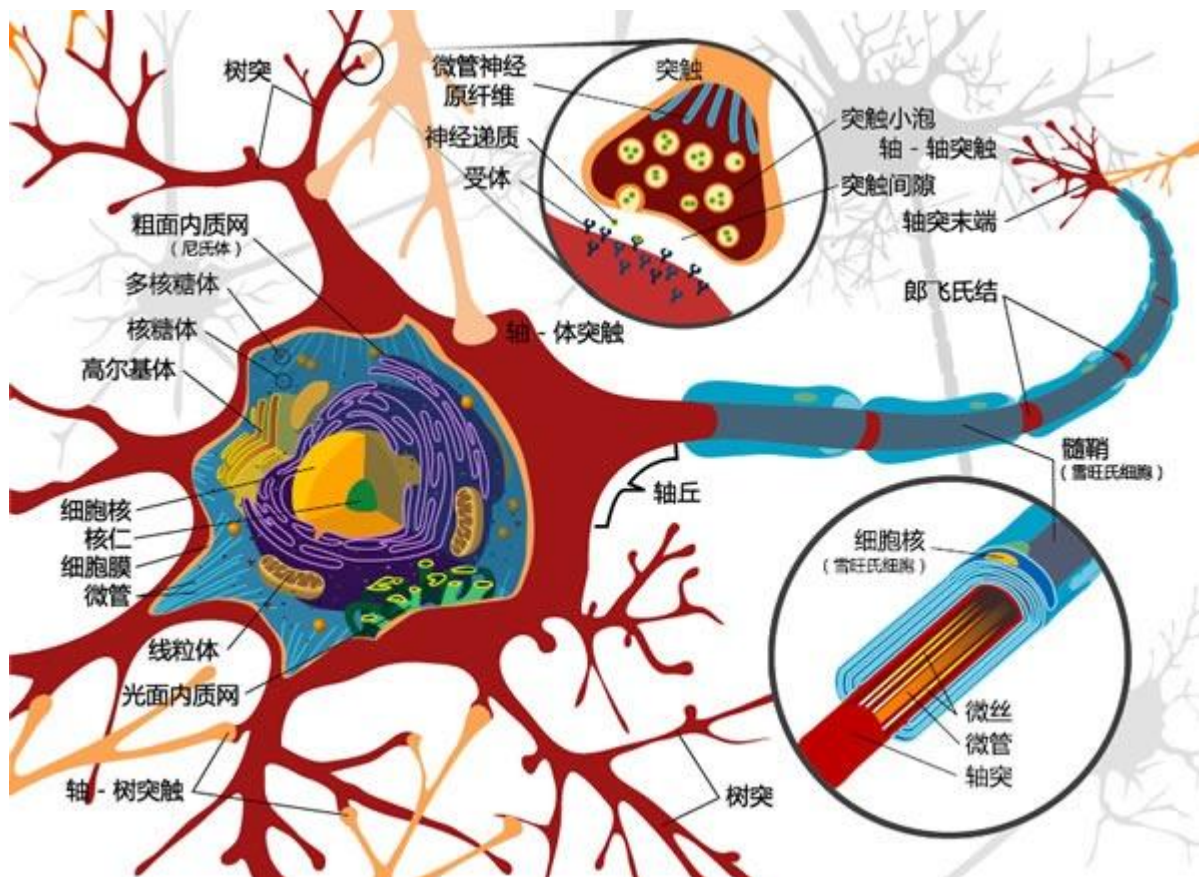
机器学习 第8周

【声明】 本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

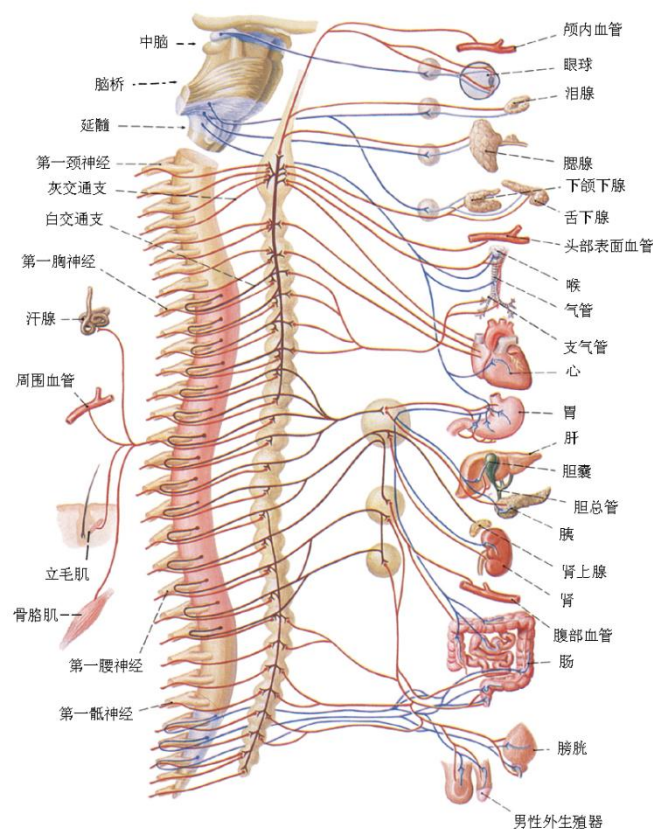
课程详情访问炼数成金培训网站

<http://edu.dataguru.cn>

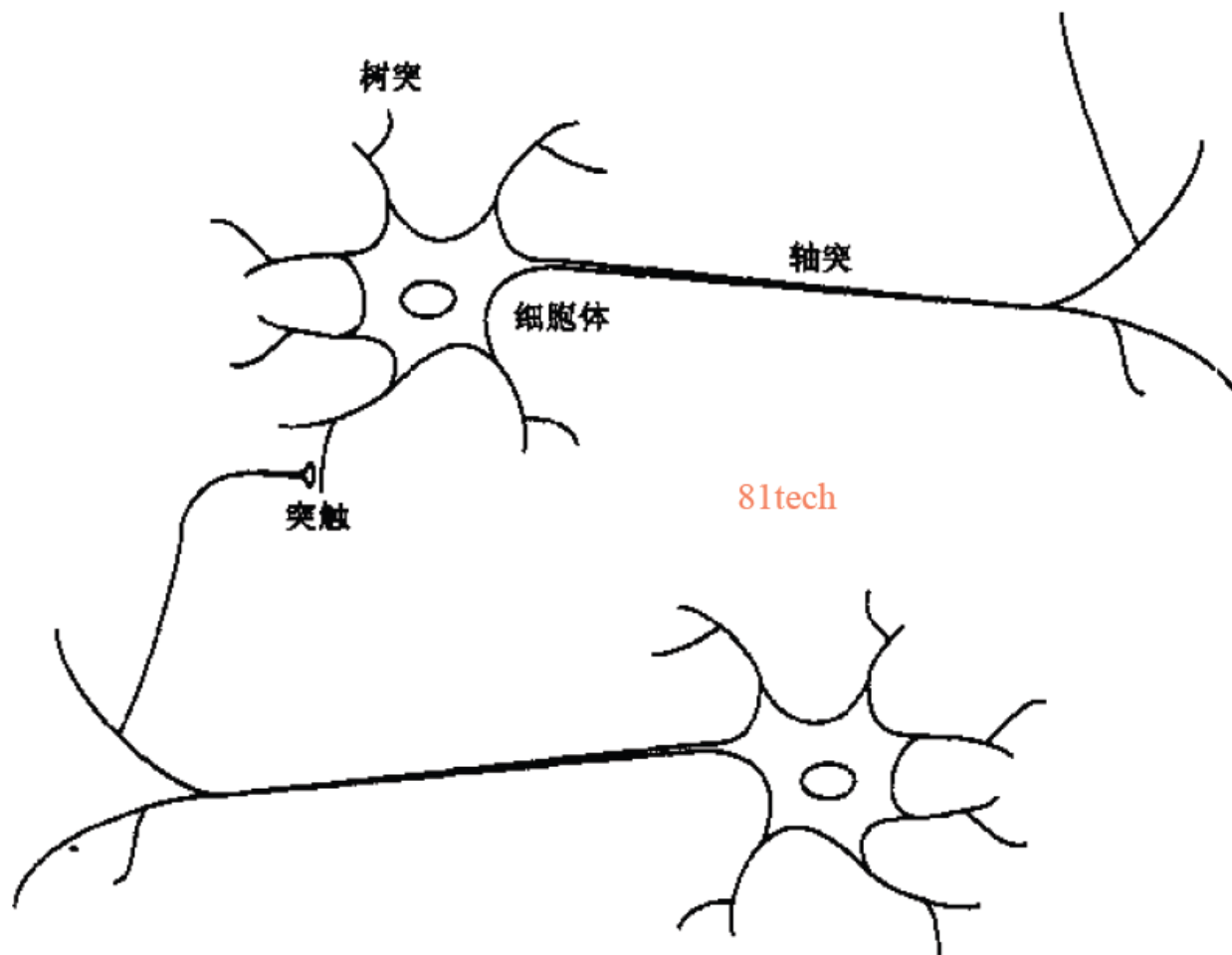
■ 人类神经系统原理



■ 人类神经系统



自主神经系概观



- 部分结构是与生俱来（非条件反射？）
- 大部分是经过后天学习建立连接（也可能消除连接）
- 记忆的本质是改变突触强度

- 1943年，心理学家W.S.McCulloch和数理逻辑学家W.Pitts建立了神经网络和数学模型，称为MP模型。他们通过MP模型提出了神经元的形式化数学描述和网络结构方法，证明了单个神经元能执行逻辑功能，从而开创了人工神经网络研究的时代。
- 60年代，人工神经网络得到了进一步发展，更完善的神经网络模型被提出，其中包括感知器和自适应线性元件等。M.Minsky等仔细分析了以感知器为代表的神经网络系统的功能及局限后，于1969年出版了《Perceptron》一书，指出感知器不能解决高阶谓词问题
- 1982年，美国加州工学院物理学家J.J.Hopfield提出了Hopfield神经网络模型
- 1986年，Rumelhart, Hinton, Williams发展了BP算法。Rumelhart和McClelland出版了《Parallel distribution processing: explorations in the microstructures of cognition》。迄今，BP算法已被用于解决大量实际问题。

- 1988年，Broomhead和Lowe用径向基函数(Radial basis function, RBF)提出分层网络的设计方法，从而将NN的设计与数值分析和线性适应滤波相挂钩。
- 90年代初，Vapnik等提出了支持向量机(Support vector machines, SVM)和VC(Vapnik-Chervonenkis)维数的概念。
- 这是迄今为止几乎最为成功的仿生学数学模型，是机器学习领域的热点，符合智能化机器的时代潮流
- 有统一的模型框架，很多算法问题可以划归为神经网络系统学习问题加以解决（例如SVM）
- 容易硬件化，元器件化，高集成化，并行化，性能优异
- 神经网络设计具有较高技巧。同时它也是一个灰箱系统，容易掩盖某些业务背景细节
- 容易产生过度拟合

■ 《神经网络设计》第4页

汽车

汽车自动导航系统，担保行为分析器

银行

支票和其他公文阅读器，信贷申请的评估器

国防

武器操纵，目标跟踪，目标辨识，面部识别、新型的传感器，声纳、雷达和图像信号处理(包括数据压缩、特征提取、噪声抑制、信号/图像的识别)

电子

代码序列预测，集成电路芯片布局，过程控制，芯片故障分析，机器视觉，语音综合，非线性建模

娱乐

动画，特技，市场预测

金融

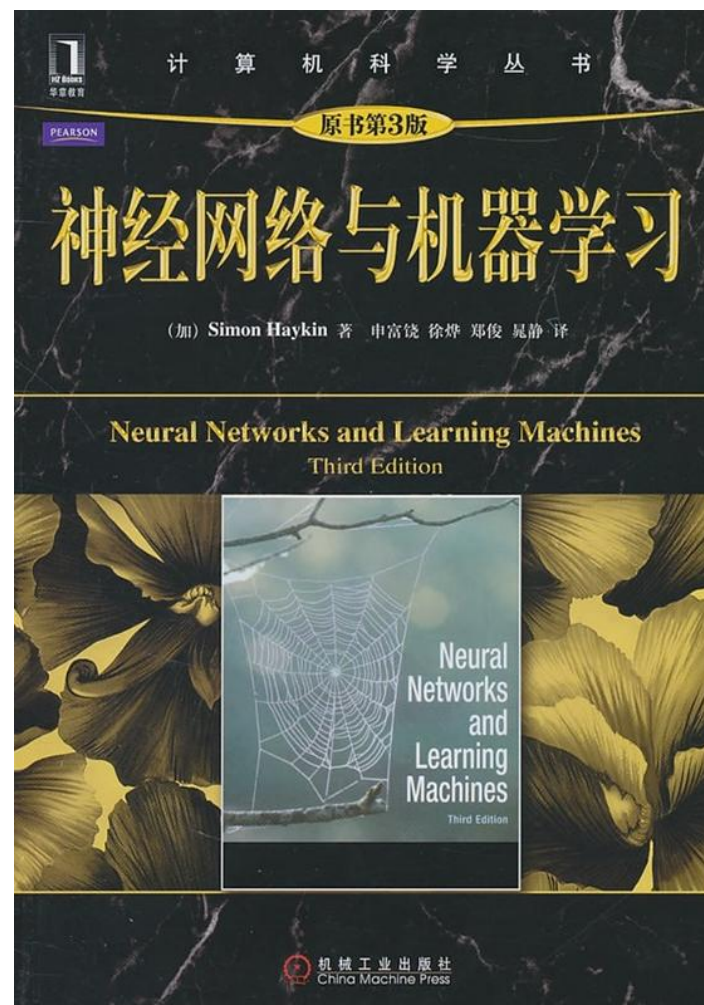
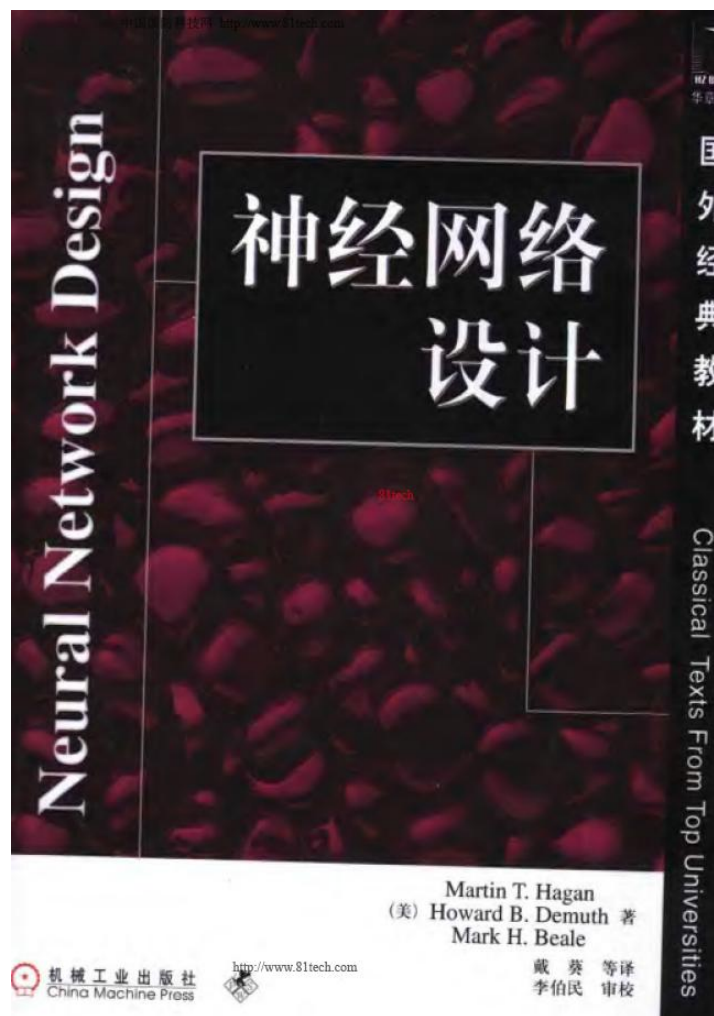
不动产评估，借贷咨询，抵押审查，公司证券分级，投资交易程序，公司财务分析，通货价格预测

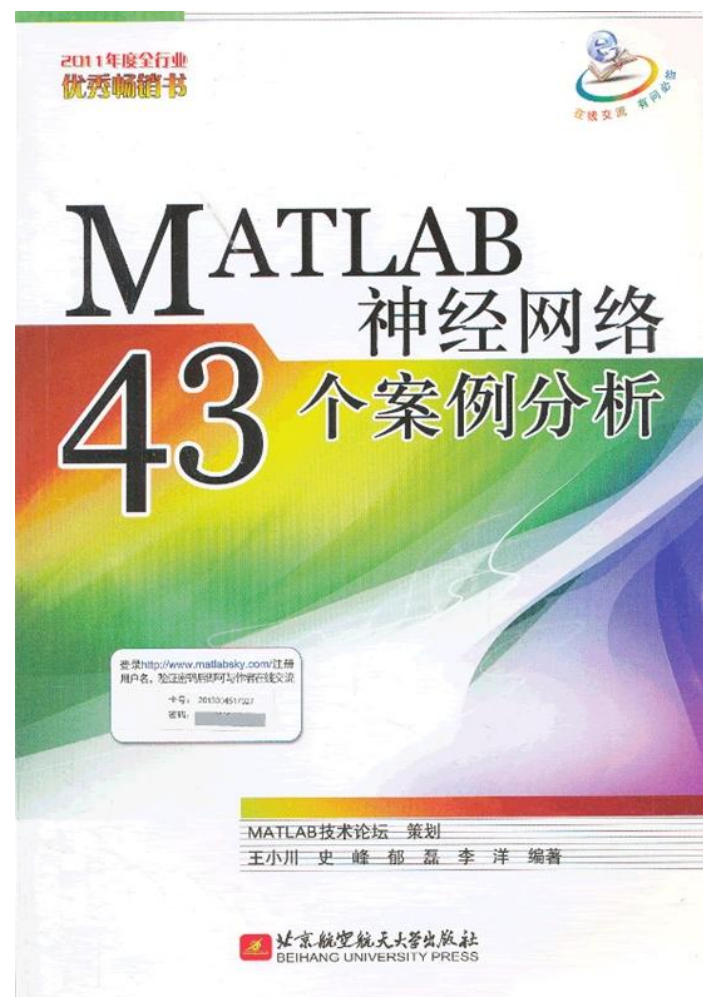
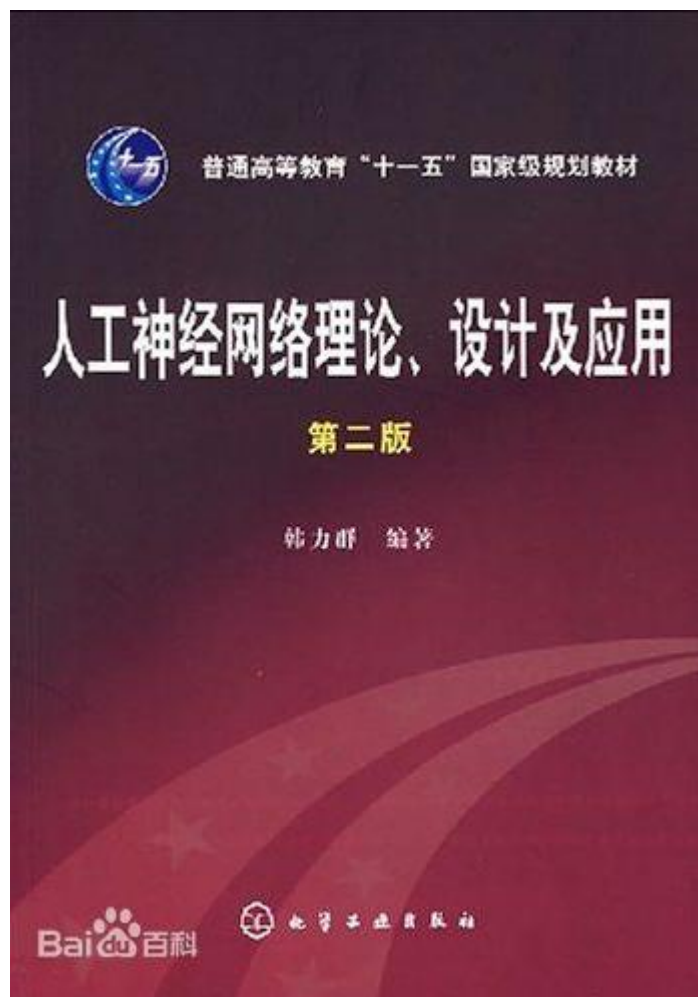
保险

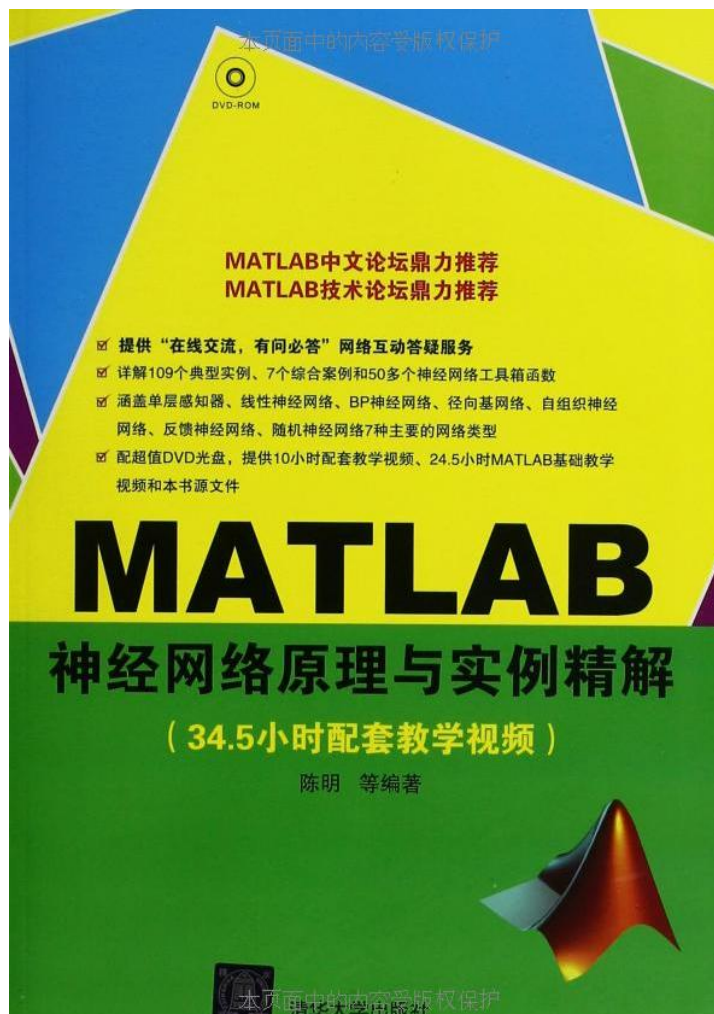
政策应用评估，产品优化

制造

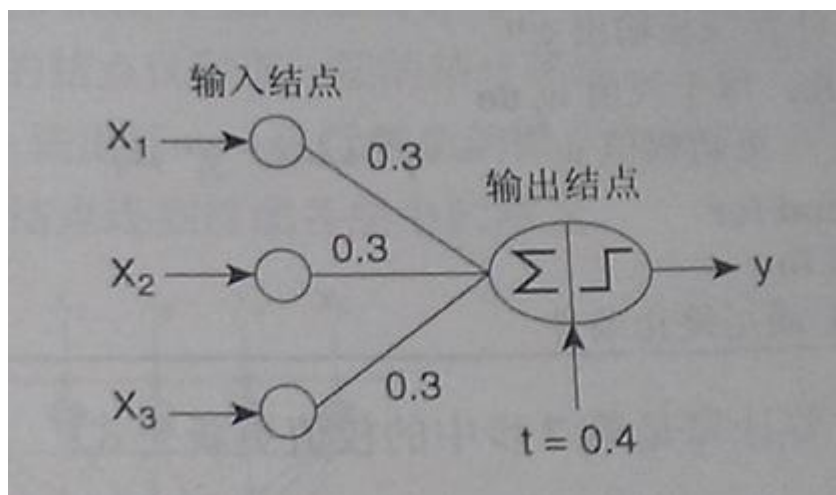
生产流程控制，产品设计和分析，过程和机器诊断，实时微粒识别，可视质量监督系统，啤酒检测，焊接质量分析，纸张质量预测，计算机芯片质量分析，磨床运转分析，化工产品设计分析，机器性能分析，项目投标，计划和管理，化工流程系统动态建模



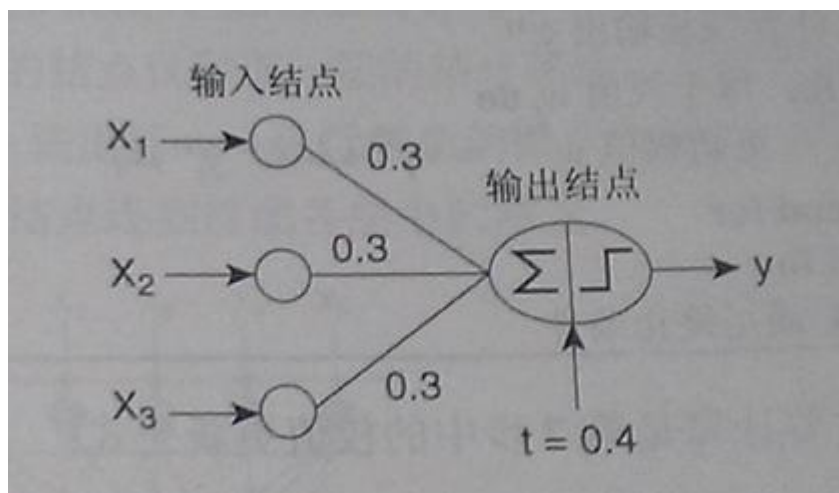




- ANN=Artificial Neural Networks , 人工神经网络
- 神经元 —— 单层感知器 , 最简单的人工神经网络



- 输入节点
- 输出节点
- 权向量
- 偏置因子
- 激活函数
- 学习率



建立数据

$x_1 = c(1, 1, 1, 1, 0, 0, 0, 0)$

$x_2 = c(0, 0, 1, 1, 0, 1, 1, 0)$

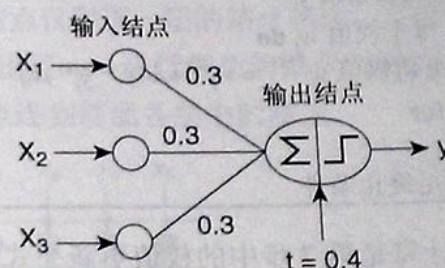
$x_3 = c(0, 1, 0, 1, 1, 0, 1, 0)$

$y = c(-1, 1, 1, 1, -1, -1, 1, -1)$

考虑图 5-14 中的图表。左边的表显示一个数据集，包含三个布尔变量(x_1, x_2, x_3)和一个输出变量 y ，当三个输入中至少有两个是 0 时， y 取 -1，而至少有两个大于 0 时， y 取 +1。

x_1	x_2	x_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

(a) 数据集

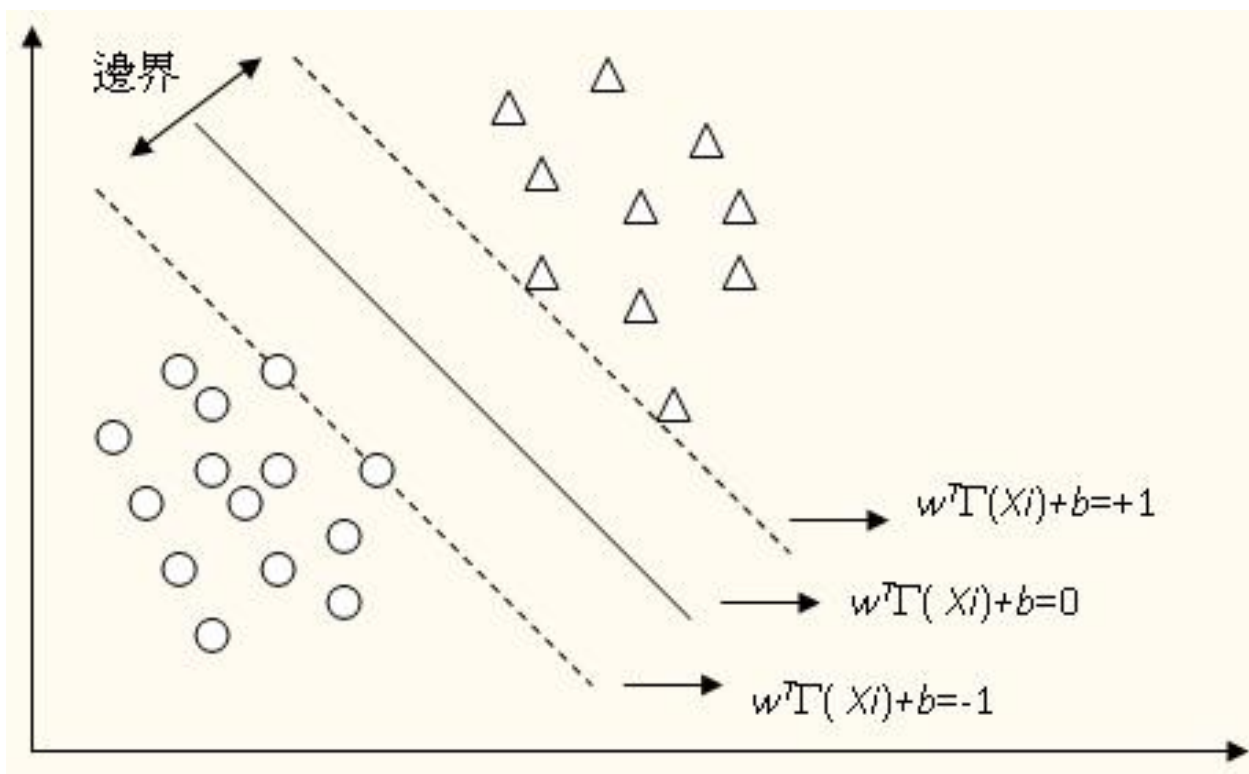


(b) 感知器

图 5-14 使用感知器模拟一个布尔函数

$$\hat{y} = \begin{cases} 1 & \text{如果 } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0 \\ -1 & \text{如果 } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 < 0 \end{cases}$$

■ 分离超平面



算法 5.4 感知器学习算法

- 1: 令 $D = \{(\mathbf{x}_i, y_i) | i=1, 2, \dots, N\}$ 是训练样例集
- 2: 用随机值初始化权值向量 $\mathbf{w}^{(0)}$
- 3: **repeat**
- 4: **for** 每个训练样例 $(\mathbf{x}_i, y_i) \in D$ **do**
- 5: 计算预测输出 $\hat{y}_i^{(k)}$
- 6: **for** 每个权值 w_j **do**
- 7: 更新权值 $w_j^{(k+1)} = w_j^{(k)} + \lambda (y_i - \hat{y}_i^{(k)}) x_{ij}$
- 8: **end for**
- 9: **end for**
- 10: **until** 满足终止条件

- 误差（实际输出与期望输出的差的绝对值）小于某个预先设定的较小的值
- 两次迭代之间的权值变化已经很小
- 设定最大迭代次数
- 单层感知器只对线性可分问题收敛

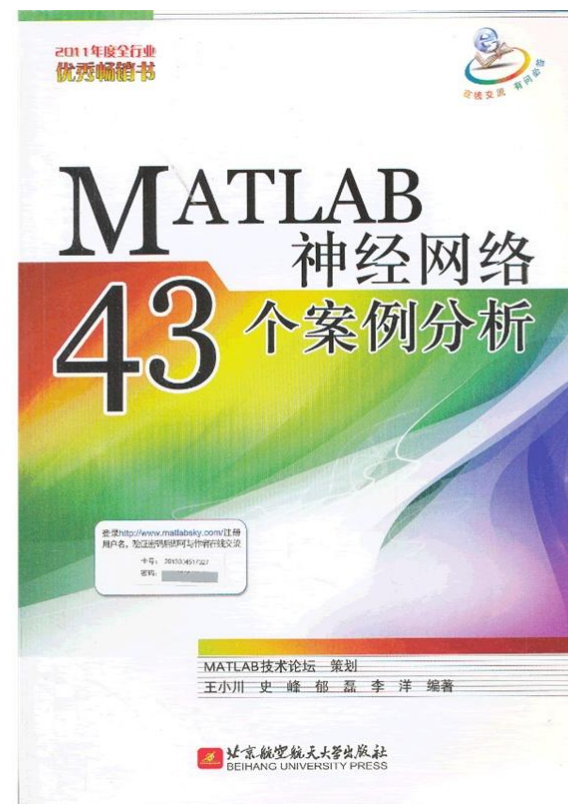
学习率的选择

- 学习率太大，容易造成权值计算不稳定
- 学习率太小，不能充分体现对权值的修正，迭代次数很多
- 比喻（显微镜对焦）

单层感知器的局限

- 无泛化能力
- 结构简单，激活函数只能是符号函数
- 只对线性可分问题收敛
- 如果存在离群点，则需要花费较多的训练时间

- MATLAB=matrix+laboratory，是由美国mathworks公司发布的主要面对科学计算、可视化以及交互式程序设计的高科技计算环境。
- MATLAB和Mathematica、Maple并称为三大数学软件。它在数学类科技应用软件中在数值计算方面首屈一指。MATLAB可以进行矩阵运算、绘制函数和数据、实现算法、创建用户界面、连接其他编程语言的程序等，主要应用于工程计算、控制设计、信号处理与通讯、图像处理、信号检测、金融建模设计与分析等领域。
- 具有功能完备强大的神经网络包



www.mathworks.cn/products/matlab/whatsnew.html?s_tid=main_release_ML_rp



Google



Accelerating the pace of engineering and science

中国 | 联系我们 | 如何购买

创建帐户 | 登录

产品和服务 解决方案 教育 支持 用户中心 活动 公司

产品和服务 > MATLAB > 新特性

MATLAB

0.1270	0.5469	0.9575	0.4683	0.8007
0.9134	0.9575	0.4683	0.8007	0.1270
0.6324	0.9649	0.8007	0.1270	0.5469

分享

弹出命令历史记录
使用弹出命令历史记录查看，筛选和搜索最近使用的命令。

观看视频

```
clc
plot(f,2*abs(Y(1:NFFT/2+1)))
Fs = 1000;
T = 1/Fs;
t = (0:L-1)*T;
L = 1000;
clc
3x plot(f,2*abs(Y(1:NFFT/2+1)))
fx >> plot(f,2*abs(Y(1:NFFT/2+1)))
```

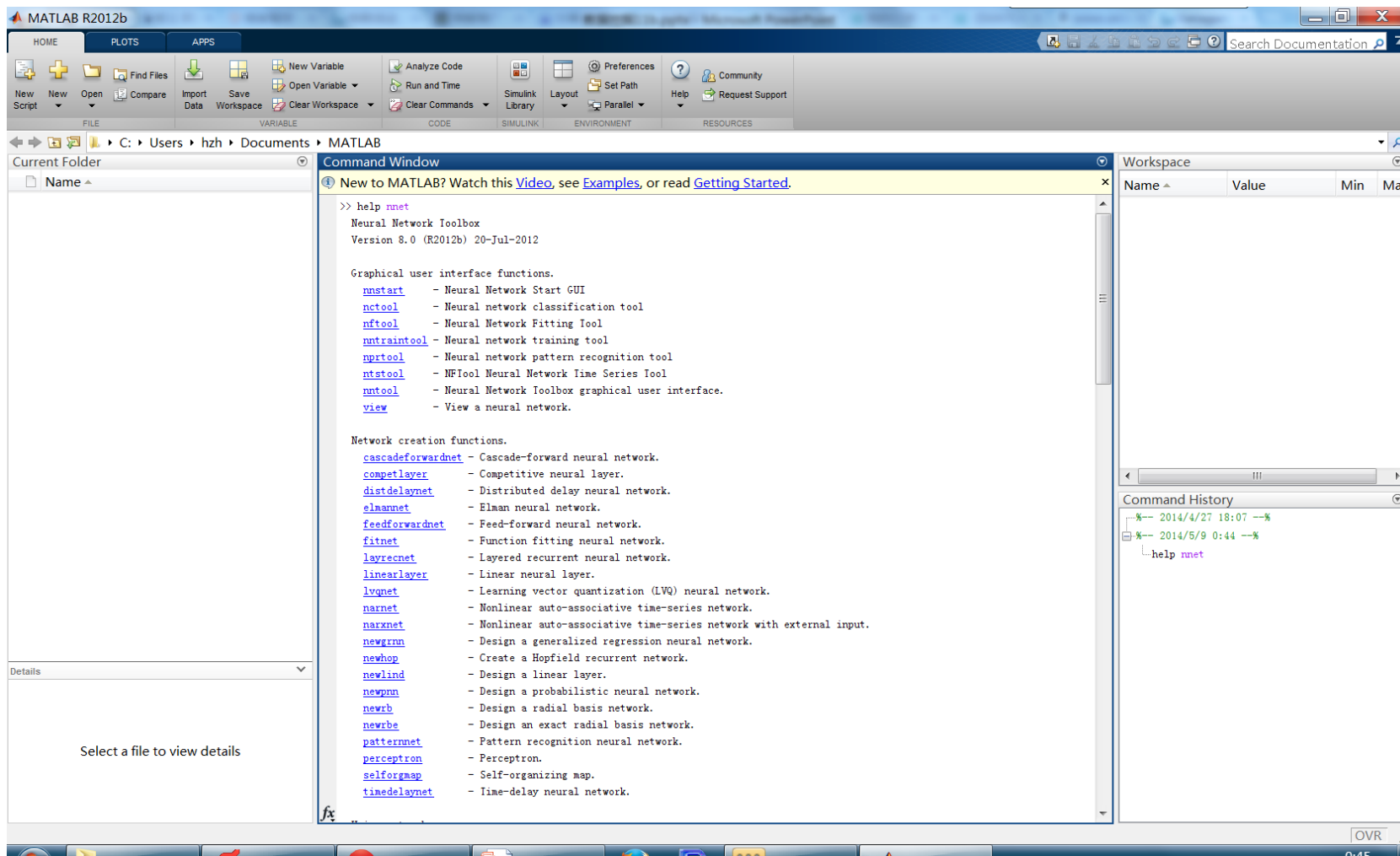
马上下载 R2014a

续订软件维护服务

试用软件

在线购买

» See release highlights for all products



Command Window

🔔 New to MATLAB? Watch this [Video](#), see [Examples](#), or read [Getting Started](#).

```
>>  
>>  
>> nntool  
>> nntool  
>> P=[1, 1, 1, 1, 0, 0, 0, 0; 0, 1, 1, 0, 1, 1, 0, 0; 1, 0, 1, 1, 0, 1, 1, 0, 1, 0];  
>> I=[-1, 1, 1, 1, -1, -1, 1, -1];  
>> P
```

P =

1	1	1	1	0	0	0	0
0	0	1	1	0	1	1	0
0	1	0	1	1	0	1	0

```
>> I
```

I =

-1	1	1	1	-1	-1	1	-1
----	---	---	---	----	----	---	----

```
>> p=[0,1;0,1;0,1]

p =

     0     1
     0     1
     0     1

>> t=1

t =

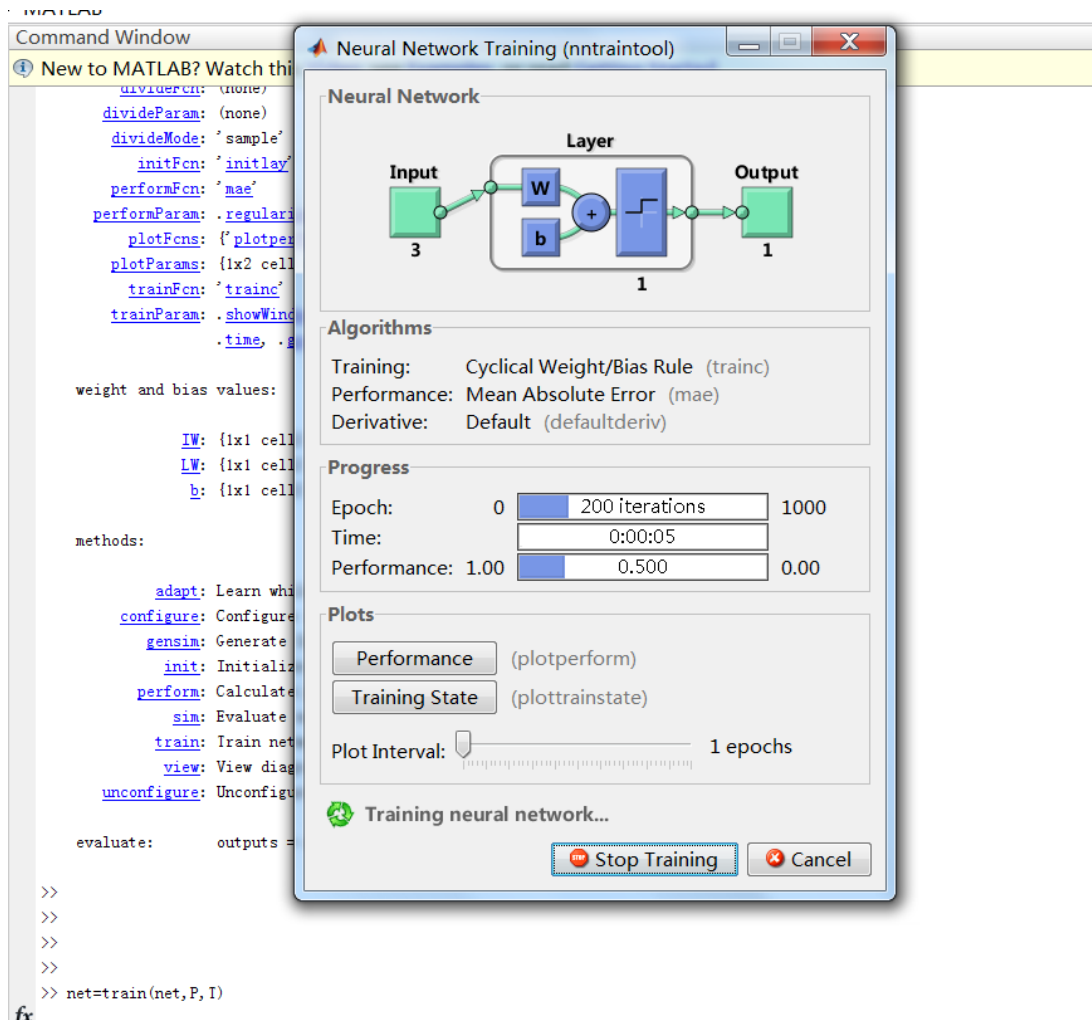
     1

>> net=newp(p,t)

net =

    Neural Network

        name: 'Custom Neural Network'
    efficiency: .cacheDelayedInputs, .flattenTime,
               .memoryReduction
        userdata: (your custom info)
```

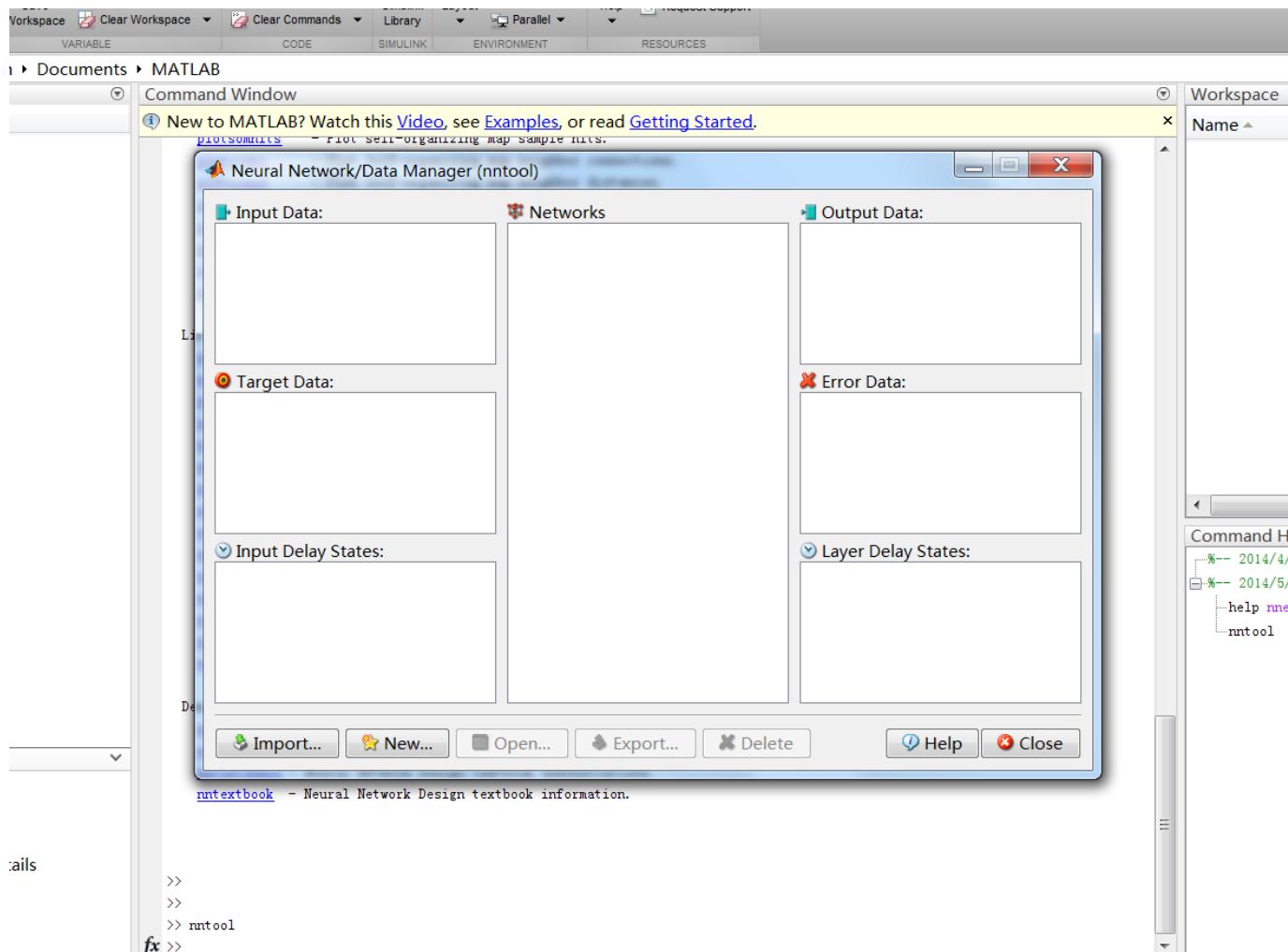


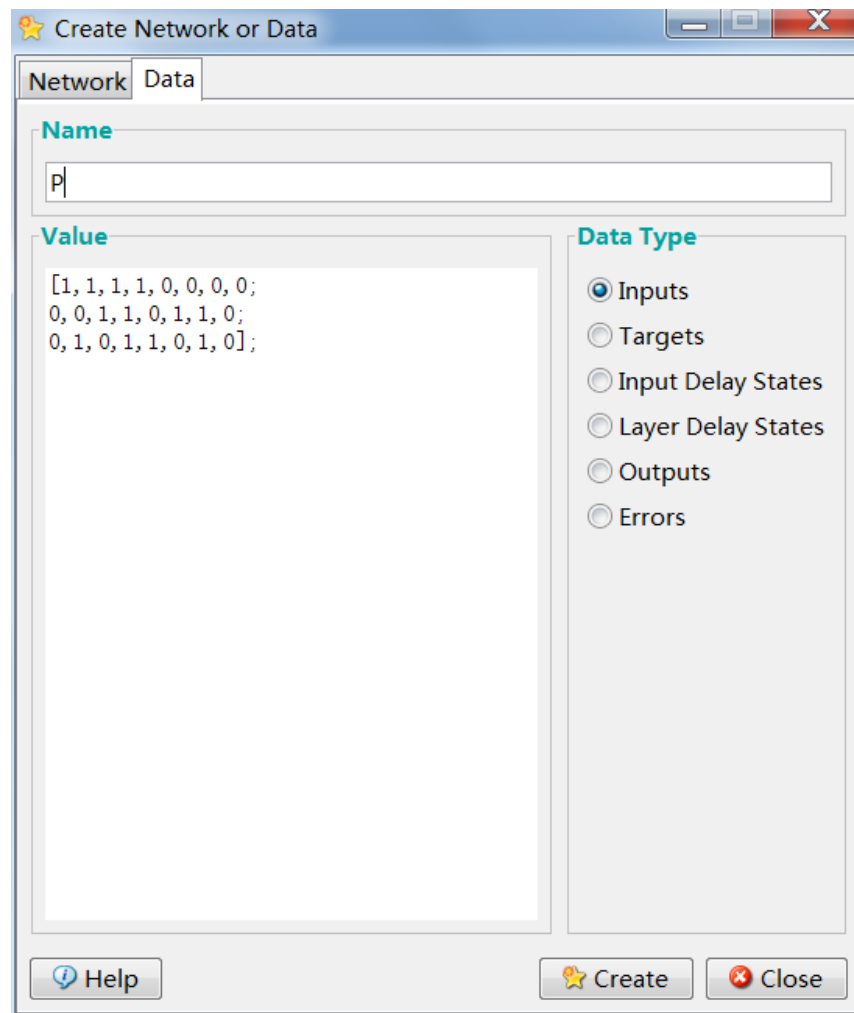

```
>> newP=[0, 0, 1]';  
>> newI=sim(net,newP)
```

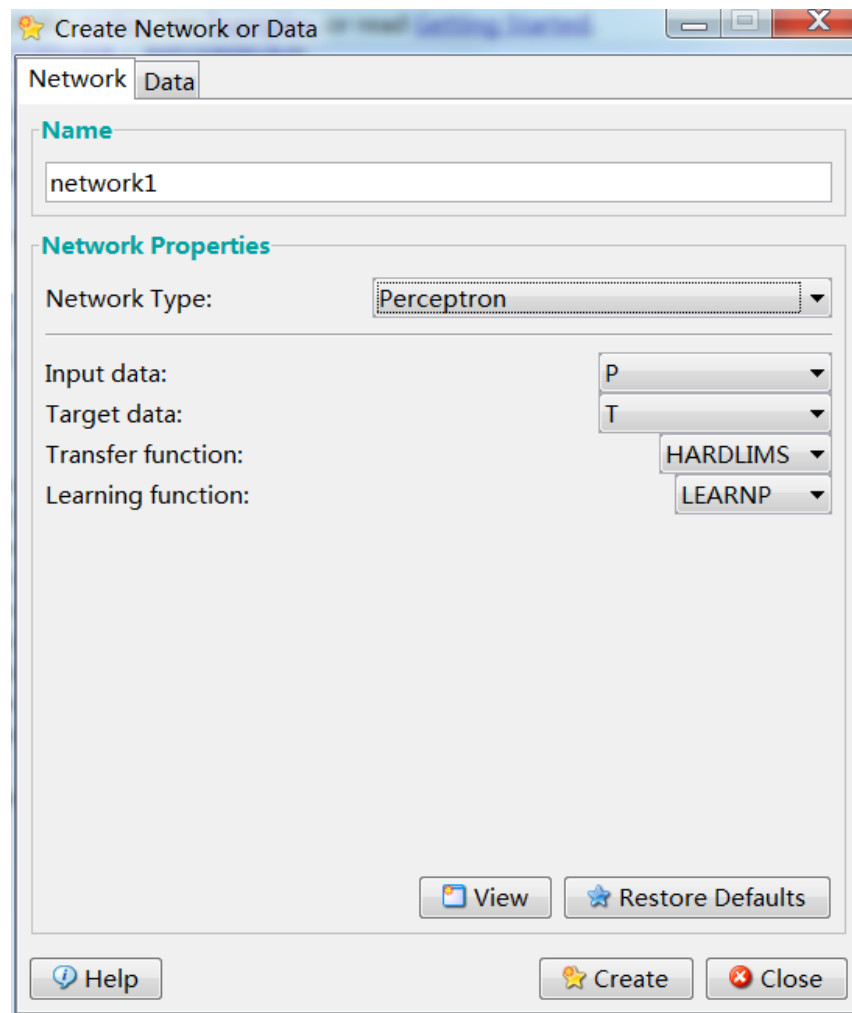
```
newI =
```

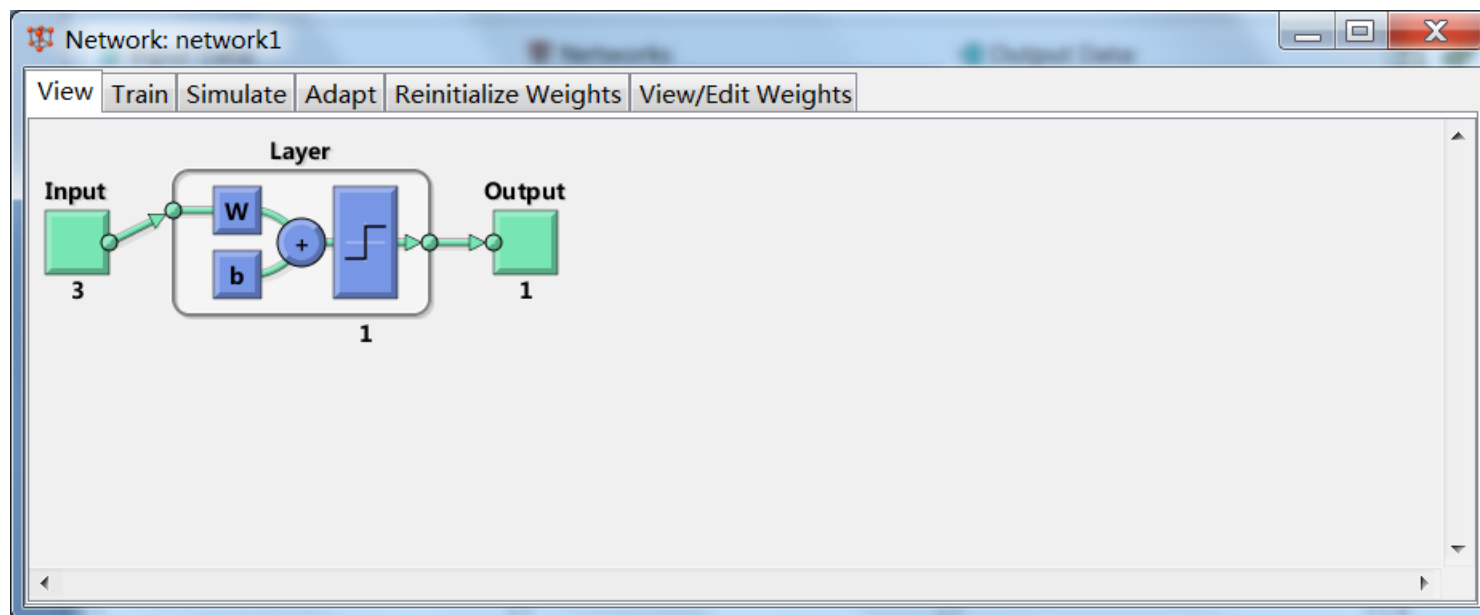
```
    -1
```

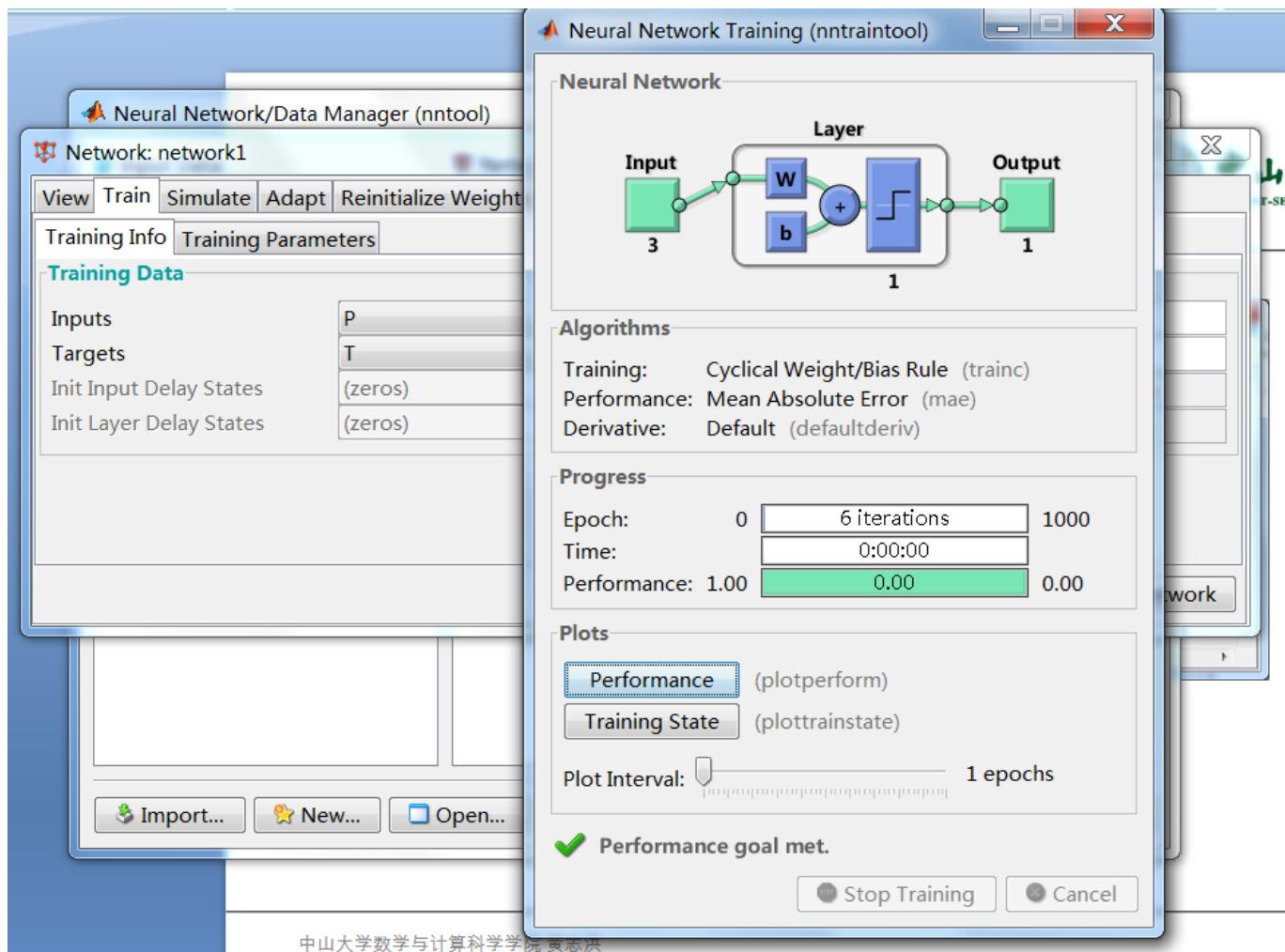
图形界面操作：nntool



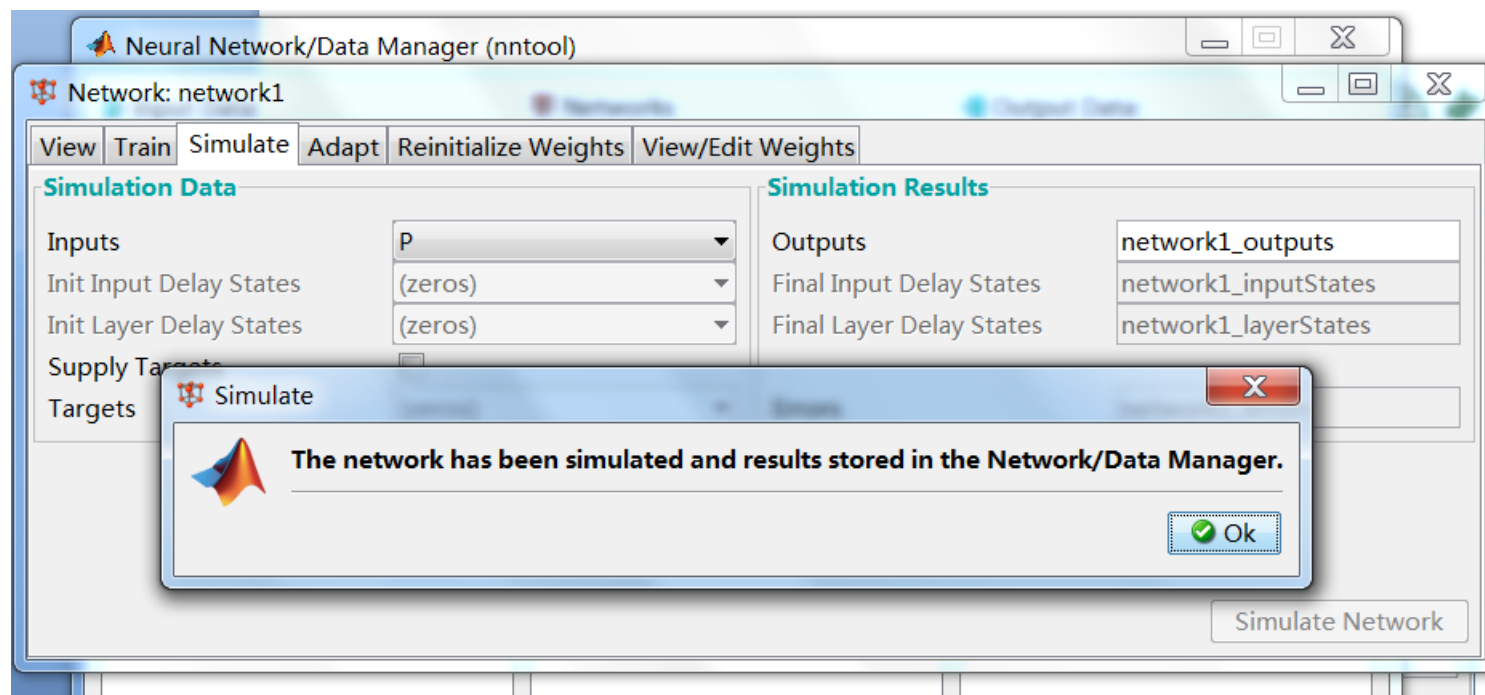


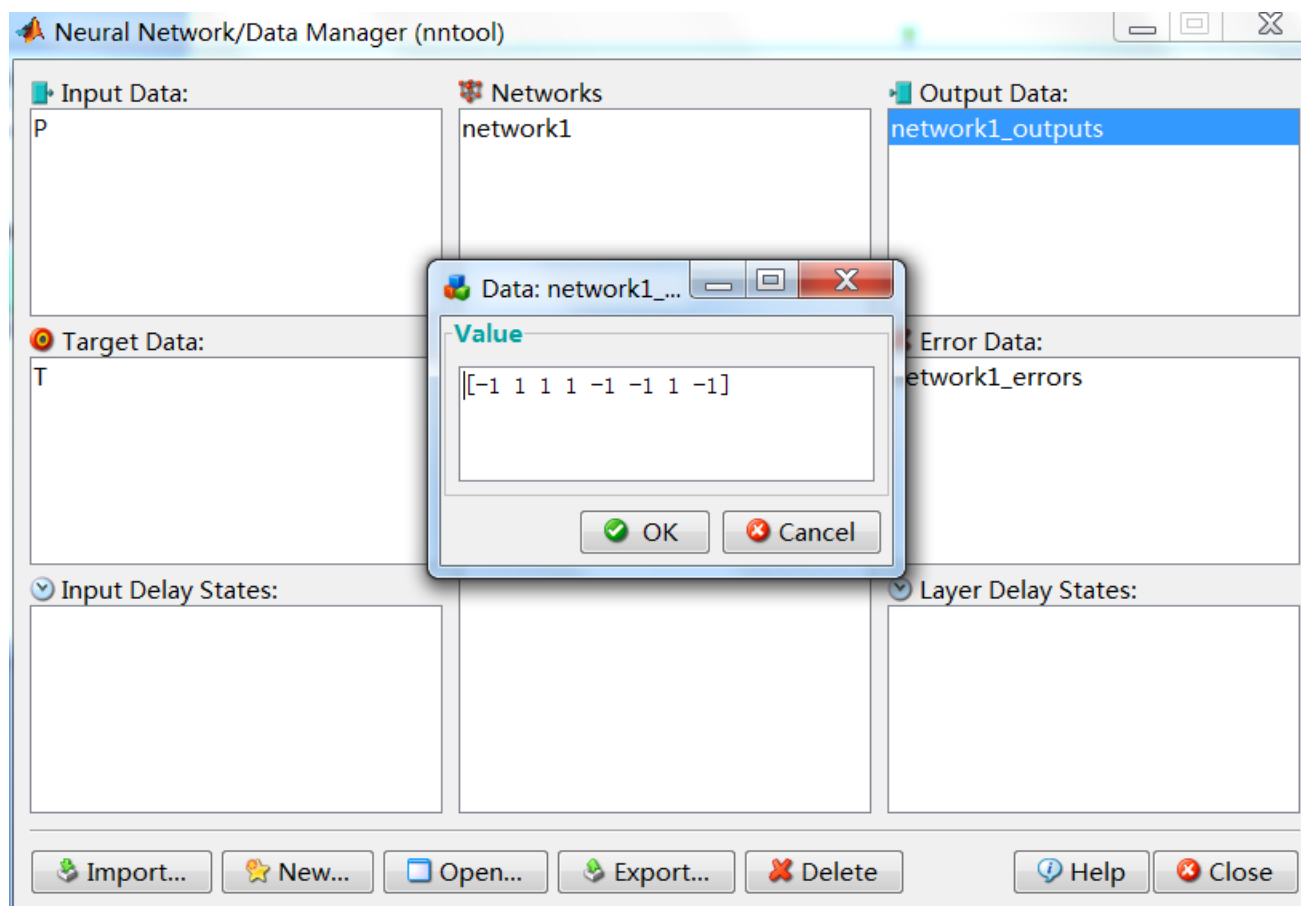




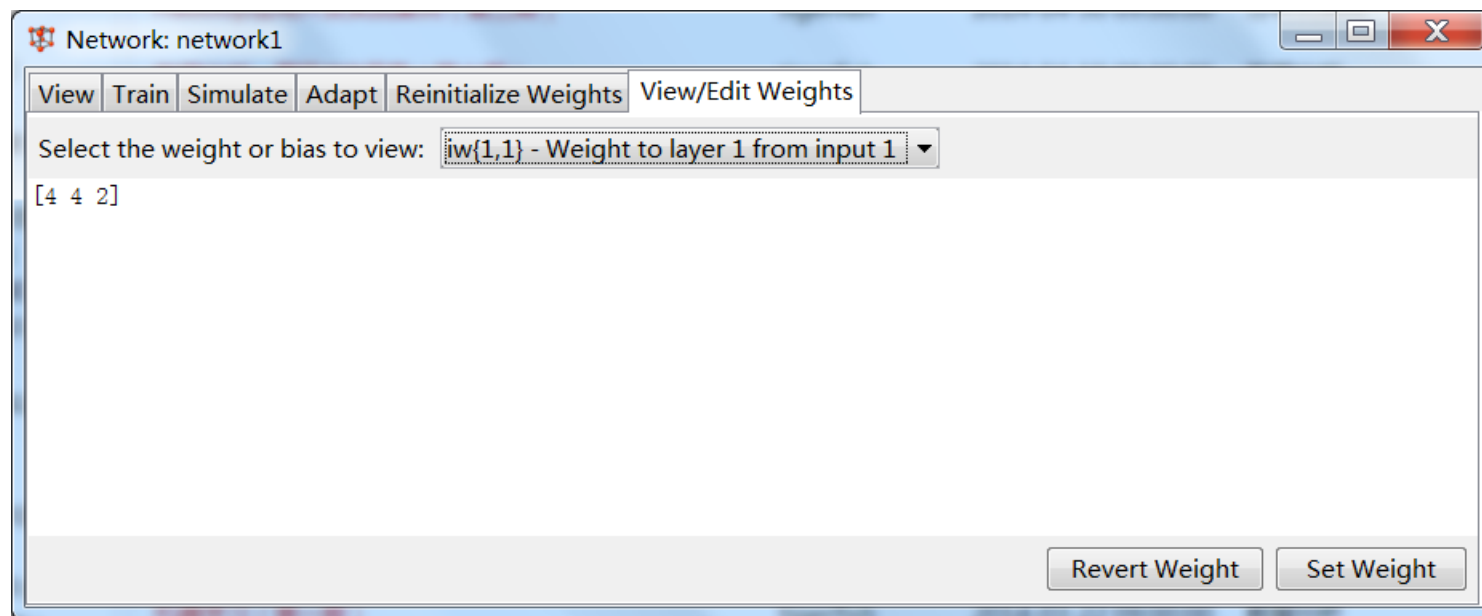


中山大学数学与计算科学学院 黄志洪

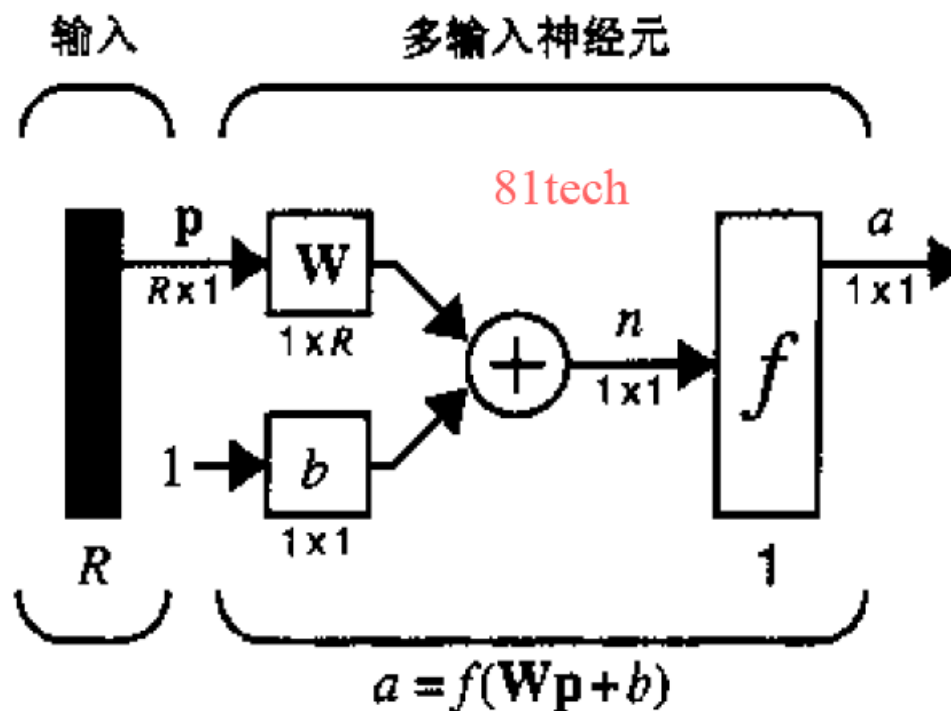




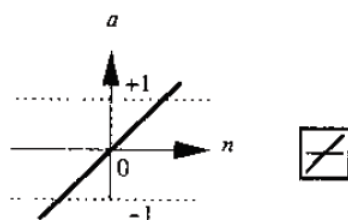
观察训练出来的权重和偏移



- 把偏置值也看成权重之一，对应的变量输入值恒为1

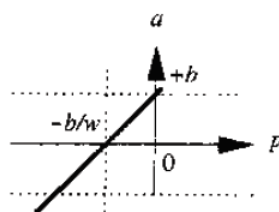


- MAE , 平均绝对误差 (误差的绝对值的平均值)
- MSE , 均方误差 (误差平方的平均值)
- SSE , 误差平方和



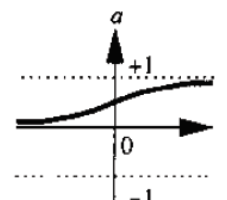
$$a = \text{purelin}(n)$$

线性传输函数



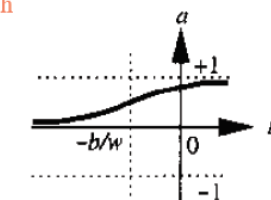
$$a = \text{purelin}(wp + b)$$

单输入 *purelin* 神经元



$$a = \text{logsig}(n)$$

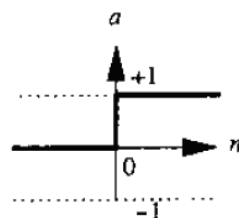
Log-Sigmoid 传输函数



$$a = \text{logsig}(wp + b)$$

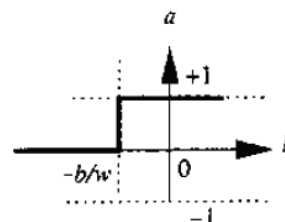
单输入 *logsig* 神经元

81tech








$$a = \text{hardlim}(n)$$






硬极限传输函数



$$a = \text{hardlim}(wp + b)$$

单输入 *hardlim* 神经元

硬极限函数	$a = 0, n < 0$ $a = 1, n \geq 0$		hardlim
对称硬极限函数	$a = -1, n < 0$ $a = +1, n \geq 0$		hardlims
线性函数	$a = n$		purelin
饱和线性函数	$a = 0, n < 0$ $a = n, 0 \leq n \leq 1$ $a = 1, n > 1$		satlin
对称饱和线性函数	$a = -1, n < -1$ $a = n, -1 \leq n \leq 1$ $a = 1, n > 1$		satlins

对称饱和线性函数	$a = -1, n < -1$ $a = n, -1 \leq n \leq 1$ $a = 1, n > 1$		satlins
对数-S形函数	$a = \frac{1}{1 + e^{-n}}$		logsig
双曲正切 S 形函数	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
正线性函数	$a = 0, n < 0$ $a = n, n \geq 0$		poslin
竞争函数	$a = 1, \text{具有最大 } n \text{ 的神经元}$ $a = 0, \text{所有其他神经元}$		compet

■ 安装AMORE包。AMORE文档中的一段样例 (p12)

```
library(AMORE)

# P is the input vector

P <- matrix(sample(seq(-1,1,length=1000), 1000, replace=FALSE), ncol=1)

# The network will try to approximate the target  $P^2$ 

target <- P^2

# We create a feedforward network, with two hidden layers.

# The first hidden layer has three neurons and the second has two neurons.

# The hidden layers have got Tansig activation functions and the output layer is Purelin.

net <- newff(n.neurons=c(1,3,2,1), learning.rate.global=1e-2, momentum.global=0.5,
error.criterium="LMS", Stao=NA, hidden.layer="tansig",
output.layer="purelin", method="ADAPTgdwm")

result <- train(net, P, target, error.criterium="LMS", report=TRUE, show.step=100, n.shows=5 )

y <- sim(result$net, P)

plot(P,y, col="blue", pch="+")

points(P,target, col="red", pch="x")
```

- 改造样例代码，解决之前的问题

```
P=cbind(x1,x2,x3)
```

```
target=y
```

```
net <- newff(n.neurons=c(3,1,1), learning.rate.global=1e-2,  
            momentum.global=0.4,
```

```
error.criterium="LMS", Stao=NA, hidden.layer="tansig",
```

```
output.layer="purelin", method="ADAPTgdwm")
```

```
result <- train(net, P, target, error.criterium="LMS", report=TRUE, show.step=100,  
               n.shows=5 )
```

```
z <- sim(result$net, P)
```

```
z
```

```
y
```

```
> result <- train(net, P, target, error.criterium="LMS", report=TRUE, show
index.show: 1 LMS 0.218461167551626
index.show: 2 LMS 0.207110702685202
index.show: 3 LMS 0.195167206269104
index.show: 4 LMS 0.180648885193377
index.show: 5 LMS 0.164384874021575
> z <- sim(result$net, P)
> z
      [,1]
[1,] -0.5975882
[2,]  0.6585419
[3,]  0.6599574
[4,]  1.3632631
[5,] -0.5929691
[6,] -0.5912290
[7,]  0.6637112
[8,] -1.5857499
> y
[1] -1  1  1  1 -1 -1  1 -1
> |
```

- 层次型：单纯层次型，输出层到输入层有连接的层次型，层内有互连的层次性
- 互连型：全互连型，局部互连型，稀疏连接型

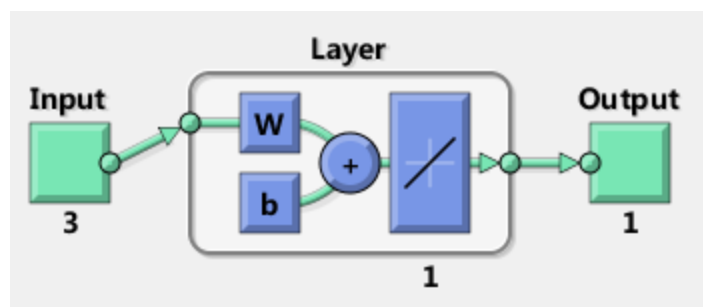
- 前馈型网络
- 反馈性网络

- Hebb学习规则
- Perceptron学习规则
- Delta学习规则
- LMS学习规则
- Outstar学习规则

- 赫布（1904-1985）是加拿大著名生理心理学家。在1949年出版的《行为的组织》中，赫布提出了其神经心理学理论。赫布认为神经网络的学习过程最终是发生在神经元之间的突触部位，突触的联结强度随着突触前后神经元的活动而变化，变化的量与两个神经元的活性之和成正比。
- Hebb学习规则是一个无监督学习规则，这种学习的结果是使网络能够提取训练集的统计特性，从而把输入信息按照它们的相似性程度划分为若干类。这一点与人类观察和认识世界的过程非常吻合，人类观察和认识世界在相当程度上就是在根据事物的统计特征进行分类。Hebb学习规则只根据神经元连接间的激活水平改变权值，因此这种方法又称为相关学习或并联学习。

- 选择适当的神经网络类型
- 对权重赋予初始值
- 选择一定的学习规则对模型进行迭代训练
- 最终收敛到合适的权重，确定模型
- 模型泛化
- 神经网络可以用于哪些场景？
- 神经网络的本质是一种通用的逼近器（类比于数学分析中的幂函数展开，傅里叶展开）

- 与单层感知器结构相似
- 增加支持purelin传输函数
- 除了二值输出外还可以支持模拟输出，因此除了充当分类器还可以实现类似回归的效果
- 使用多个输出（Madaline网络）可以变相解决线性不可分问题
- 引入非线性成分（升维）可以一定程度上解决非线性问题



线性网络能解决非线性问题吗

- 可以把变量的非线性项作为新输入变量来处理
- 《Matlab神经网络原理与实例精解》第130页

- 一种利用梯度下降法的一般性的学习规则
- 1986年，心理学家McClelland和Rumelhart引入
- 目标函数：最小平方误差条件（MSE）
- 学习规则：权值变化量（delta）正比于负梯度（比例系数即为学习率）
- 一个算例：《人工神经网络理论、设计及应用》第34页

- <http://zh.wikipedia.org/wiki/%E6%9C%80%E9%80%9F%E4%B8%8B%E9%99%8D%E6%B3%95>

梯度下降法，基于这样的观察：如果实值函数 $F(\mathbf{x})$ 在点 \mathbf{a} 处可微且有定义，那么函数 $F(\mathbf{x})$ 在 \mathbf{a} 点沿着梯度相反的方向 $-\nabla F(\mathbf{a})$ 下降最快。因而，如果

$$\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$$

对于 $\gamma > 0$ 为一个够小数值时成立，那么 $F(\mathbf{a}) \geq F(\mathbf{b})$ 。

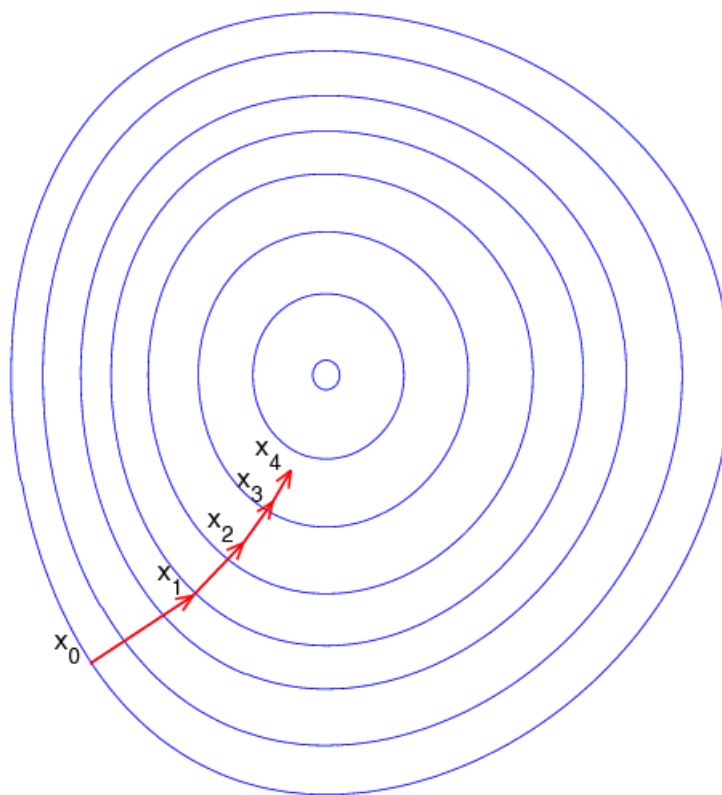
考虑到这一点，我们可以从函数 F 的局部极小值的初始估计 \mathbf{x}_0 出发，并考虑如下序列 $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ 使得

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$

因此可得到

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \dots,$$

如果顺利的话序列 (\mathbf{x}_n) 收敛到期望的极值。注意每次迭代步长 γ 可以改变。

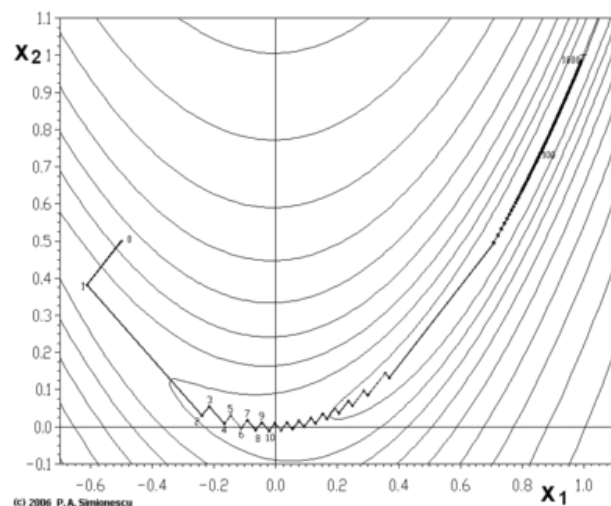


- 目标函数必须可微
- 学习率难以选取，太大会产生“之字形”震荡，太小迭代次数太多，前进很慢
- 容易陷入局部最优

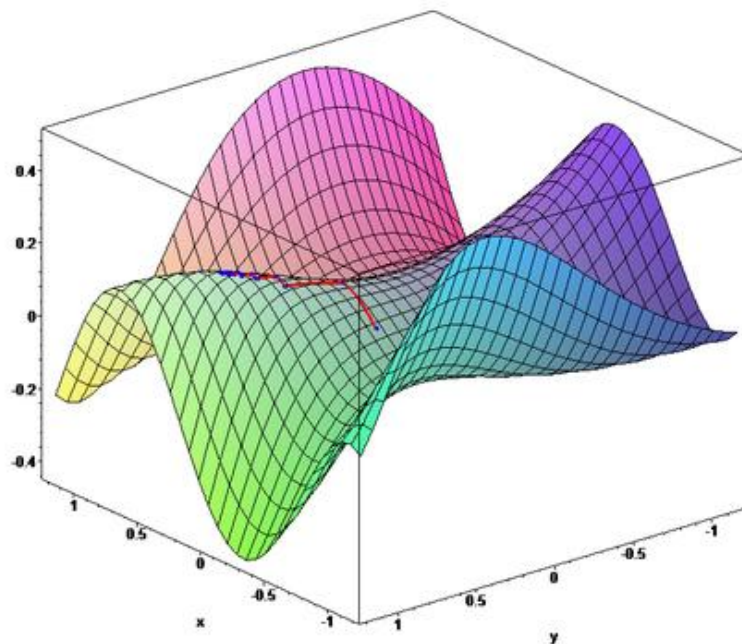
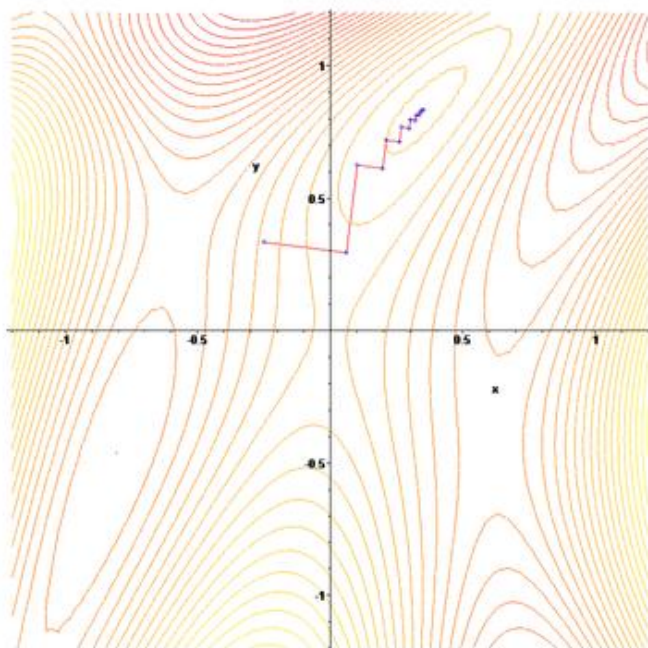
梯度下降法处理一些复杂的非线性函数会出现问题，例如Rosenbrock函数

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2.$$

其最小值在 $(x, y) = (1, 1)$ 处，数值为 $f(x, y) = 0$ 。但是此函数具有狭窄弯曲的山谷，最小值 $(x, y) = (1, 1)$ 就在这些山谷之中，并且谷底很平。优化过程是之字形的向极小值点靠近，速度非常缓慢。



下面这个例子也鲜明的示例了"之字"的下降，这个例子用梯度下降法求 $F(x, y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right) \cos(2x + 1 - e^y)$ 的极小值。



- Least Mean Square , Widrow和Hoff在1960提出
- 可以理解为Delta学习规则的特殊情况（激活函数使用purelin）
- 和感知器学习规则神似
- 仅用于训练单层神经网络（多层经过转化为单层，事实上也能应用）

1 设置变量和参量：

$X(n)$ 为输入向量，或称为训练样本

$W(n)$ 为权值向量

$b(n)$ 为偏差

$d(n)$ 为期望输出

$y(n)$ 为实际输出

η 为学习速率

n 为迭代次数

2 初始化，赋给 $w(0)$ 各一个较小的随机非零值，令 $n=0$

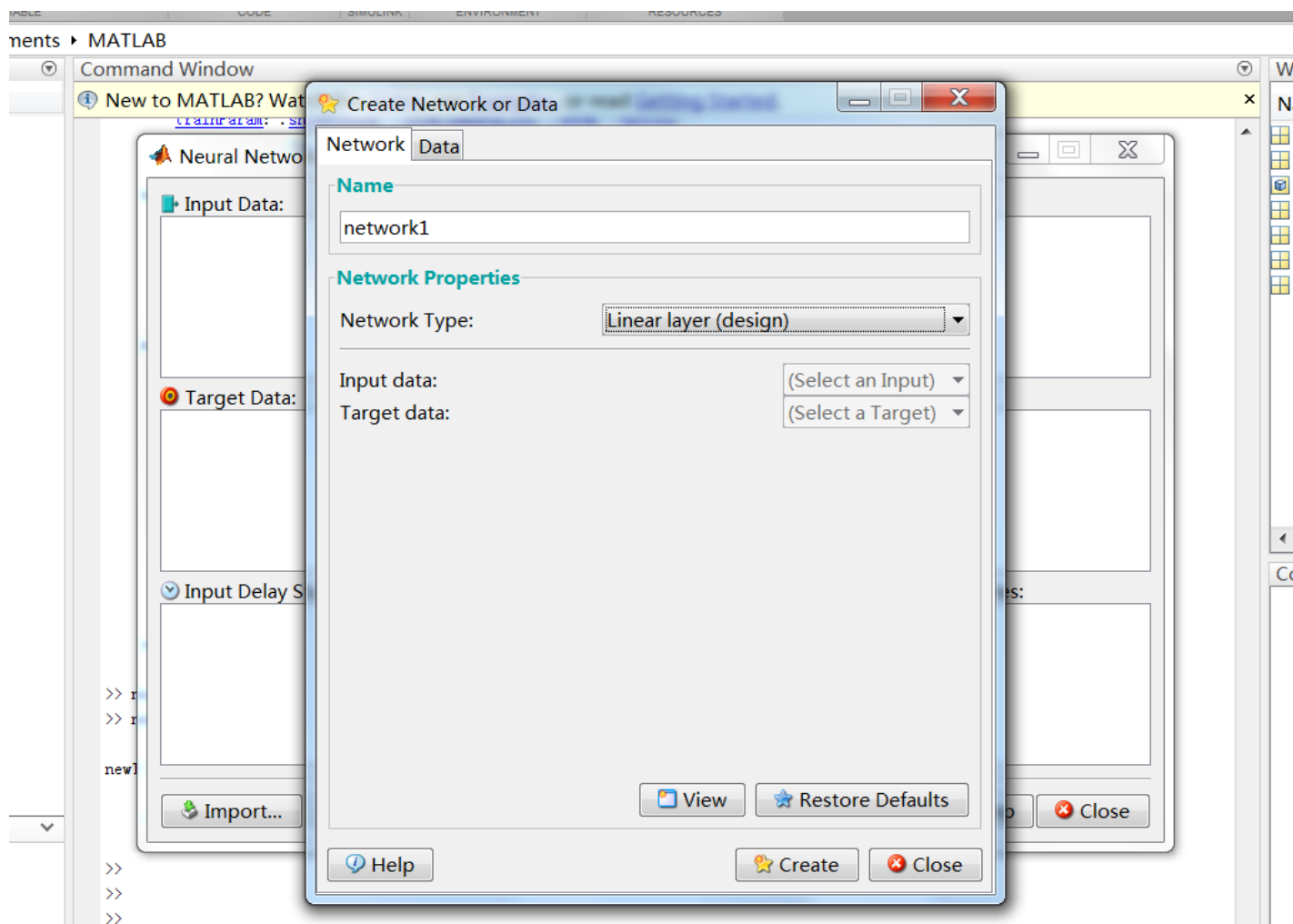
3 对于一组输入样本 $x(n)$ 和对应的期望输出 d ，计算

$$e(n) = d(n) - X(n)W(n)$$

$$W(n+1) = W(n) + \eta X(n)e(n)$$

4 判断是否满足条件，若满足算法结束，若否 n 增加1，转入第3步继续执行。

- R 为输入样本 X 的自相关矩阵， a 是 R 的最大特征根
- 可以证明，只要选择学习率在0和 $2/a$ 之间，LMS算法收敛（Haykin，1996）
- 由于 a 一般难以计算，通常用 R 的迹 t 代替，因为 $2/t < 2/a$ ，所以学习率如果小于 $2/t$ ，那么它必定小于 $2/a$
- LMS也有类似多重共线性的情况，对 R 中的条件数敏感
- 学习率逐渐下降的算法（《Matlab神经网络原理与实例精解》第133页）



观看学习算法核心代码

The image shows the MATLAB R2012b interface. The Command Window displays the source code for the `learnwh` function, which implements the Widrow-Hoff weight/bias learning function. The Workspace panel is empty. The Command History panel shows the sequence of commands executed, including `runtool` and `type learnwh`.

```
>> runtool
>> type learnwh

function [out1,out2] = learnwh(varargin)
%LEARNWH Widrow-Hoff weight/bias learning function.
%
% Syntax
%
%   [dW,LS] = learnwh(W,P,Z,N,A,I,E,gW,gA,D,LP,LS)
%   [db,LS] = learnwh(b,ones(1,Q),Z,N,A,I,E,gW,gA,D,LP,LS)
%   info = learnwh(code)
%
% Description
%
%   LEARNWH is the Widrow-Hoff weight/bias learning function,
%   and is also known as the delta or least mean squared (LMS) rule.
%
%   LEARNWH(W,P,Z,N,A,I,E,gW,gA,D,LP,LS) takes several inputs,
%   W - SxR weight matrix (or b, an Sx1 bias vector).
%   P - RxQ input vectors (or ones(1,Q)).
%   Z - SxQ weighted input vectors.
%   N - SxQ net input vectors.
%   A - SxQ output vectors.
%   I - SxQ layer target vectors.
%   E - SxQ layer error vectors.
%   gW - SxR gradient with respect to performance.
%   gA - SxQ output gradient with respect to performance.
%   D - SxS neuron distances.
%   LP - Learning parameters, none, LP = [].
%   LS - Learning state, initially should be = [].
% and returns,
%   dW - SxR weight (or bias) change matrix.
%   LS - New learning state.
%
% Learning occurs according to LEARNWH's learning parameter,
% shown here with its default value.
% LP.lr - 0.01 - Learning rate
```

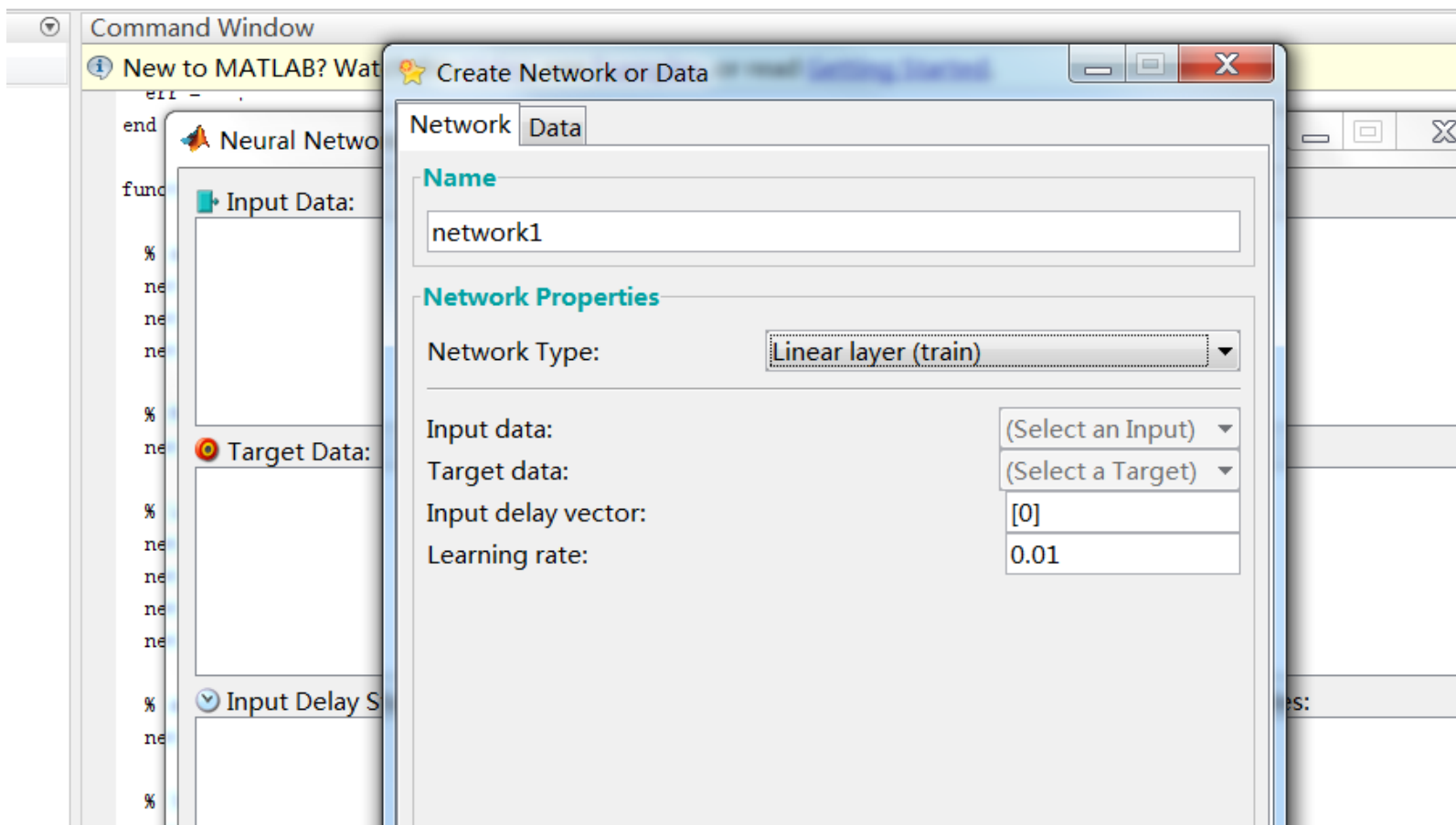
Workspace

Name	Value	Min	Max
------	-------	-----	-----

Command History

```
%-- 2014/5/9 11:45 --%
P=[1,1,1,1,0,0,0,0,0,0,1,1,0,1,1,0,0,1,0,
P
I=[-1,1,1,1,-1,-1,-1,-1,-1,-1]
p=[0,1:0,1:0,1]
t=1
net=newp(p,t,'hardlims')
net=train(net,P,I)
newP=[0,0,1]';
newI=sim(net,newP)
runtool
runtools
runtool
%-- 2014/5/13 10:14 --%
runtool
type learnwh
```


- 可以用于处理时间序列数据



- **Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。**
- **关于逆向收费式网络的详情，请看我们的培训网站 <http://edu.dataguru.cn>**



Thanks

FAQ时间