

# 神经网络

# Neural Networks

## 第三章

---

# 误差反向传播网络

史忠植

中国科学院计算技术研究所  
<http://www.intsci.ac.cn/>

---

# 内容提要

---

- 3.1 概述
- 3.2 反向传播网络的结构
- 3.3 反向传播算法
- 3.4 反向传播算法性能分析
- 3.5 反向传播算法的改进
- 3.6 反向传播网络学习程序

# BP算法的出现

1974年，Werbos已提出了该方法

1982年，Paker就完成了相似的工作

Rumelhart, McClelland于1985年提出了BP网络的误差反向后传BP(Back Propagation)学习算法

David  
Rumelhart



J.  
McClelland



- BP算法基本原理

- 利用输出后的误差来估计输出层的直接前导层的误差，再用这个误差估计更前一层的误差，如此一层一层的反传下去，就获得了所有其他各层的误差估计。

# BP算法

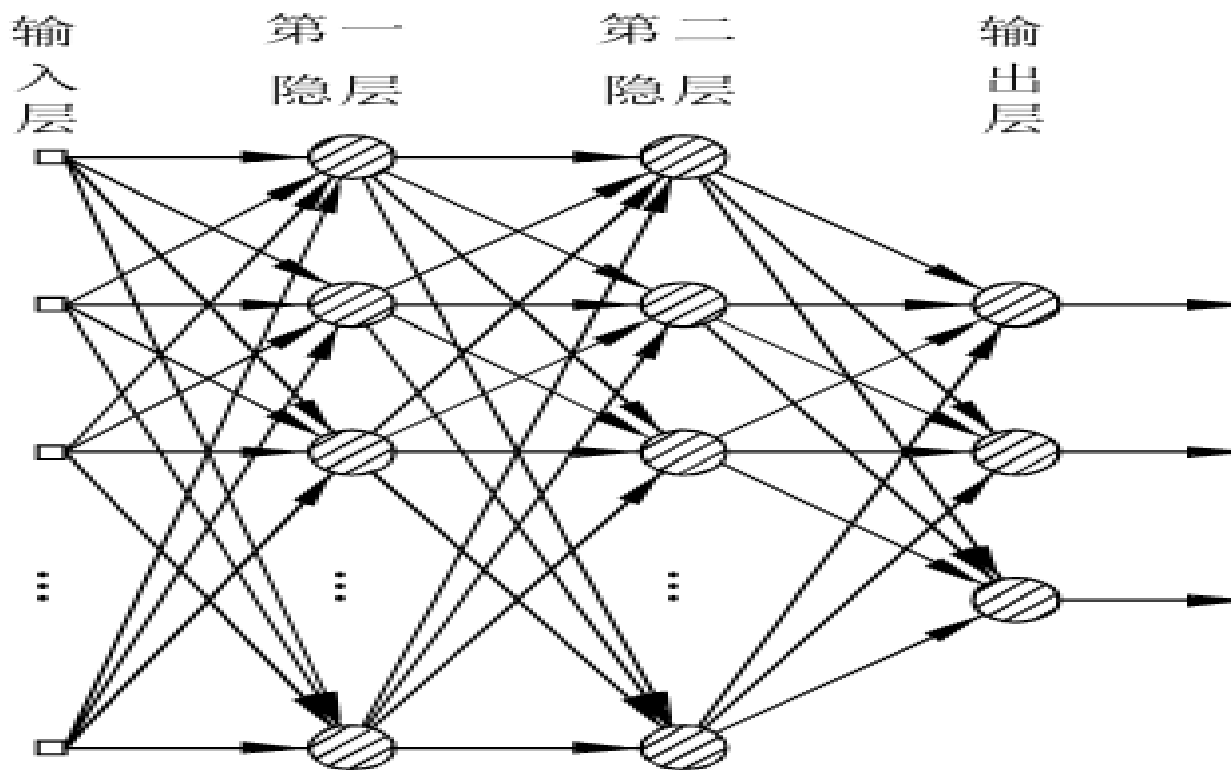
---

**优点：**广泛的适应性和有效性。

**弱点：**训练速度非常慢、局部极小点的逃离问题、算法不一定收敛。

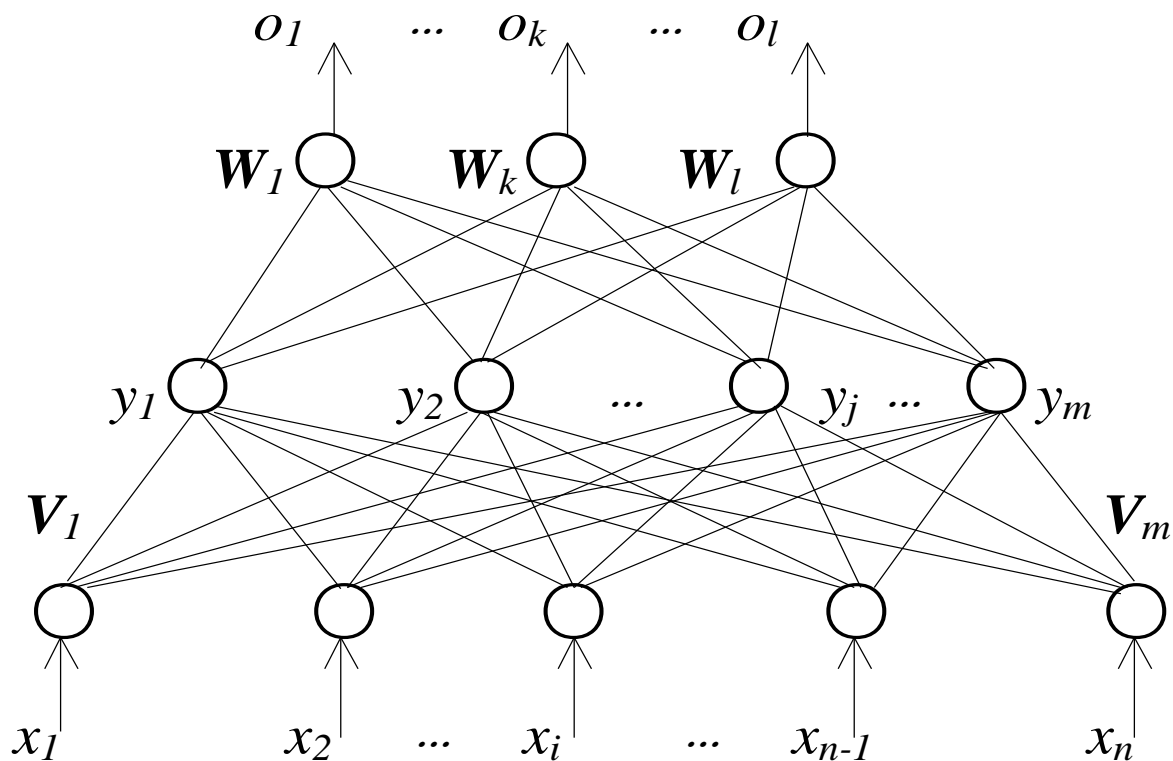
# BP神经网络模型

- BP网络



# 误差反向传播 (BP) 网路

基于BP算法的多层前馈网络模型



# 误差反向传播 (BP) 网路

基于BP算法的多层前馈网络模型

输入向量:  $X=(x_1, x_2, \dots, x_j, \dots, x_n)^T$

隐层输出向量:  $Y=(y_1, y_2, \dots, y_j, \dots, y_m)^T$

输出层输出向量:  $O=(o_1, o_2, \dots, o_k, \dots, o_l)^T$

期望输出向量:  $d=(d_1, d_2, \dots, d_k, \dots, d_l)^T$

输入层到隐层之间的权值矩阵:  $V=(V_1, V_2, \dots, V_j, \dots, V_m)$

隐层到输出层之间的权值矩阵:  $W=(W_1, W_2, \dots, W_k, \dots, W_l)$

# 误差反向传播 (BP) 网路

## 3.4.1 基于BP算法的多层前馈网络模型

对于输出层:  $o_k = f(net_k) \quad k=1,2,\dots,l$

$$net_k = \sum_{j=0}^m w_{jk} y_j \quad k=1,2,\dots,l$$

对于隐层:  $y_j = f(net_j) \quad j=1,2,\dots,m$

$$net_j = \sum_{i=0}^n v_{ij} x_i \quad j=1,2,\dots,m$$



# 误差反向传播（BP）网路

## 3.4.1 基于BP算法的多层前馈网络模型

单极性Sigmoid函数：

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

双极性Sigmoid函数：

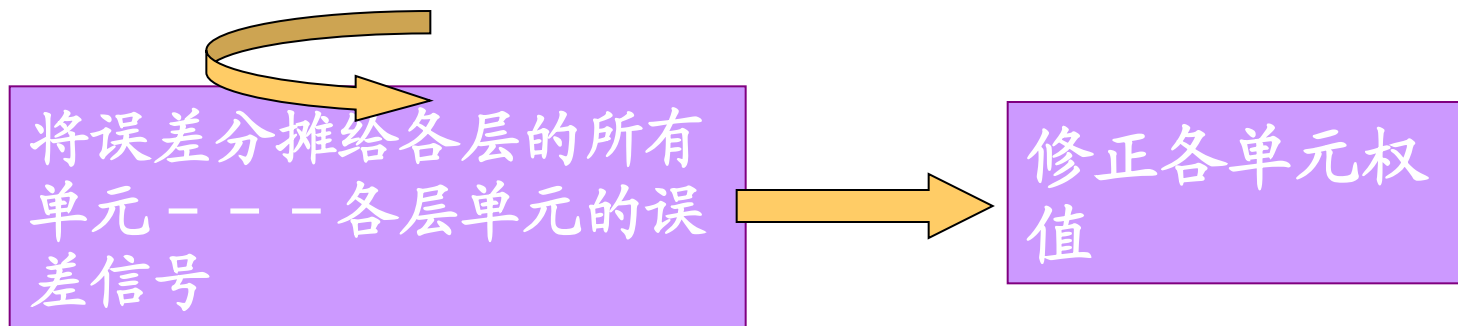
$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$


# BP网络的标准学习算法

- 学习的过程：
  - 神经网络在外界输入样本的刺激下不断改变网络的连接权值,以使网络的输出不断地接近期望的输出。
- 学习的本质：
  - 对各连接权值的动态调整
- 学习规则：
  - 权值调整规则，即在学习过程中网络中各神经元的连接权变化所依据的一定的调整规则。

# BP网络的标准学习算法-算法思想

- 学习的类型：有导师学习
- 核心思想：
  - 将输出误差以某种形式通过隐层向输入层逐层反传



- 学习的过程：
  - 信号的正向传播  误差的反向传播

# BP网络的标准学习算法-学习过程

- 正向传播：
  - 输入样本——输入层——各隐层——输出层
- 判断是否转入反向传播阶段：
  - 若输出层的实际输出与期望的输出（教师信号）不符
- 误差反传
  - 误差以某种形式在各层表示——修正各层单元的权值

- 网络输出的误差减少到可接受的程度  
进行到预先设定的学习次数为止

# BP学习算法

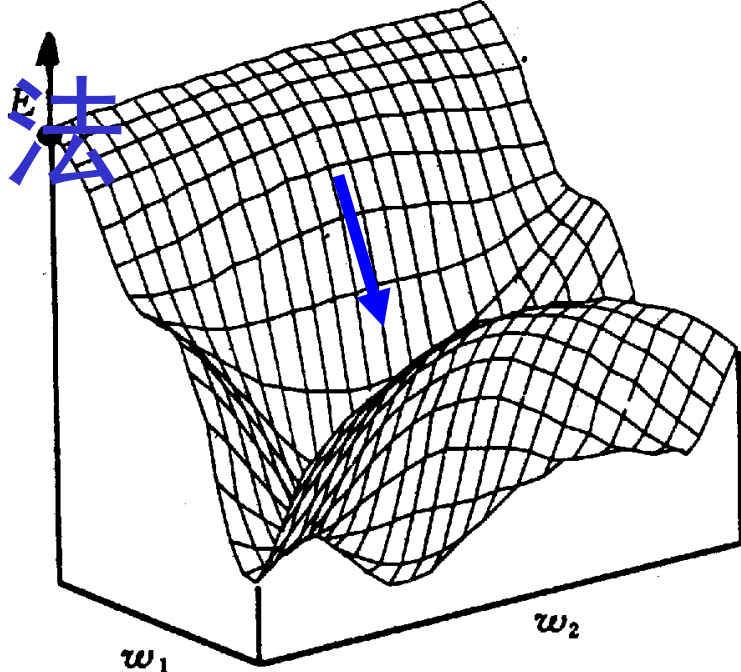
## 一、网络误差 定义与权值调整思路

输出误差 $E$ 定义: 
$$E = \frac{1}{2}(\mathbf{d} - \mathbf{O})^2 = \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2$$

将以上误差定义式展开至隐层:

$$E = \frac{1}{2} \sum_{k=1}^l [d_k - f(\text{net}_k)]^2 = \frac{1}{2} \sum_{k=1}^l [d_k - f(\sum_{j=0}^m w_{jk} y_j)]^2$$

# BP学习算法



## 一、网络误差与权值调整

进一步展开至输入层:

$$E = \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m w_{jk} f(\text{net}_j)]\}^2$$
$$= \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m \underline{w_{jk}} f(\sum_{i=0}^n \underline{v_{ij}} x_i)]\}^2$$

# BP学习算法

BP学习算法

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \quad j=0,1,2,\dots,m; k=1,2,\dots,l$$

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} \quad i=0,1,2,\dots,n; j=1,2,\dots,m$$

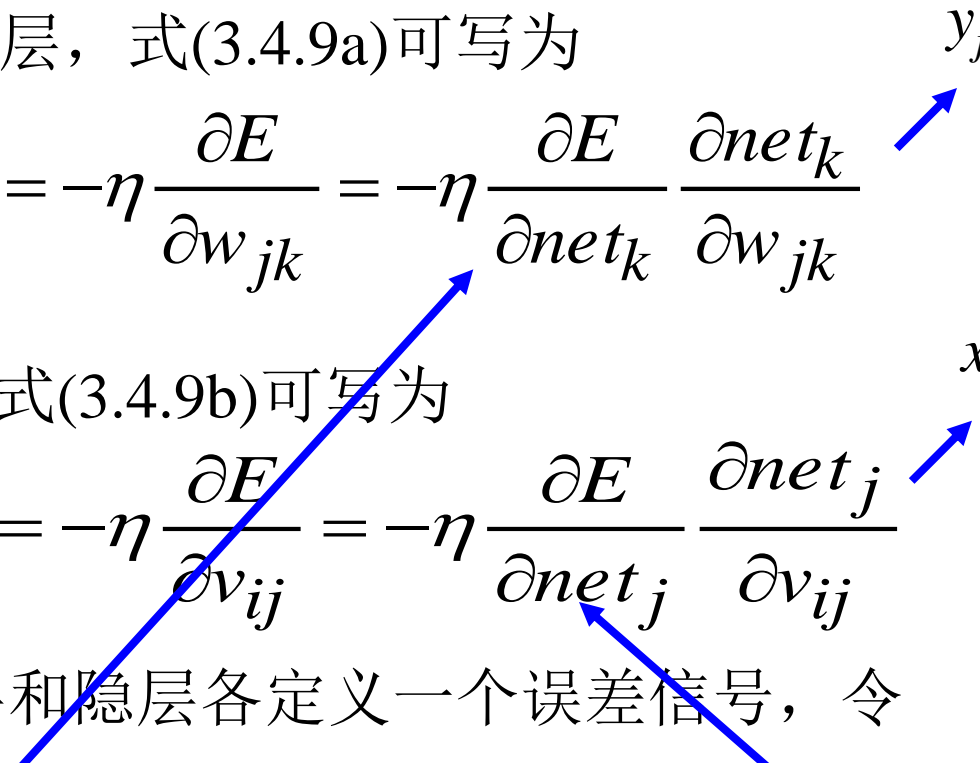
式中负号表示梯度下降，常数 $\eta \in (0,1)$ 表示比例系数。

在全部推导过程中，对输出层有 $j=0,1,2,\dots,m; k=1,2,\dots,l$

对隐层有  $i=0,1,2,\dots,n; j=1,2,\dots,m$

# BP算法推导

对于输出层，式(3.4.9a)可写为

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}}$$


对隐层，式(3.4.9b)可写为

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} = -\eta \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial v_{ij}}$$

对输出层和隐层各定义一个误差信号，令

$$\delta_k^o = -\frac{\partial E}{\partial net_k} \quad (3.4.11a) \quad \delta_j^y = -\frac{\partial E}{\partial net_j}$$



# BP算法的程序实现

(1)初始化;

(2)输入训练样本对 $X \leftarrow X^p$ 、 $d \leftarrow d^p$   
计算各层输出;

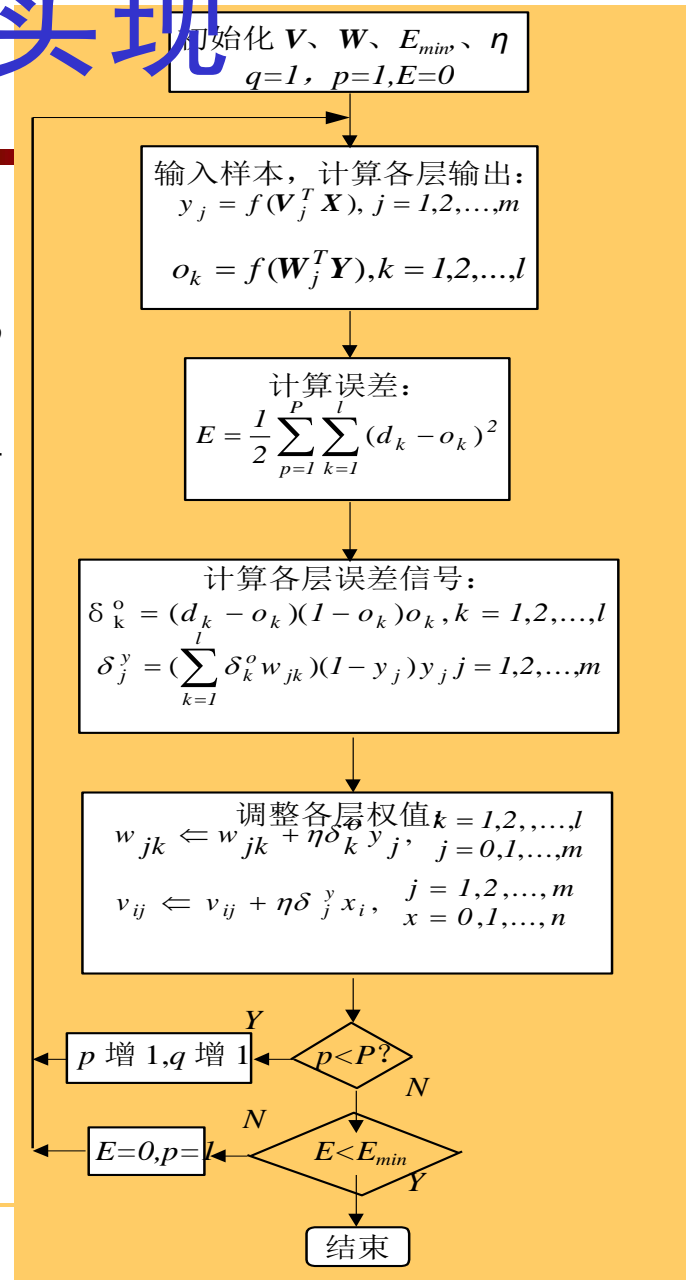
(3)计算网络输出误差; $E_{RME} = \sqrt{\frac{1}{P} \sum_{p=1}^P E^p}$

(4)计算各层误差信号;

(5)调整各层权值;

(6)检查是否对所有样本完成一次  
轮训;

(7)检查网络总误差是否达到精  
度要求。

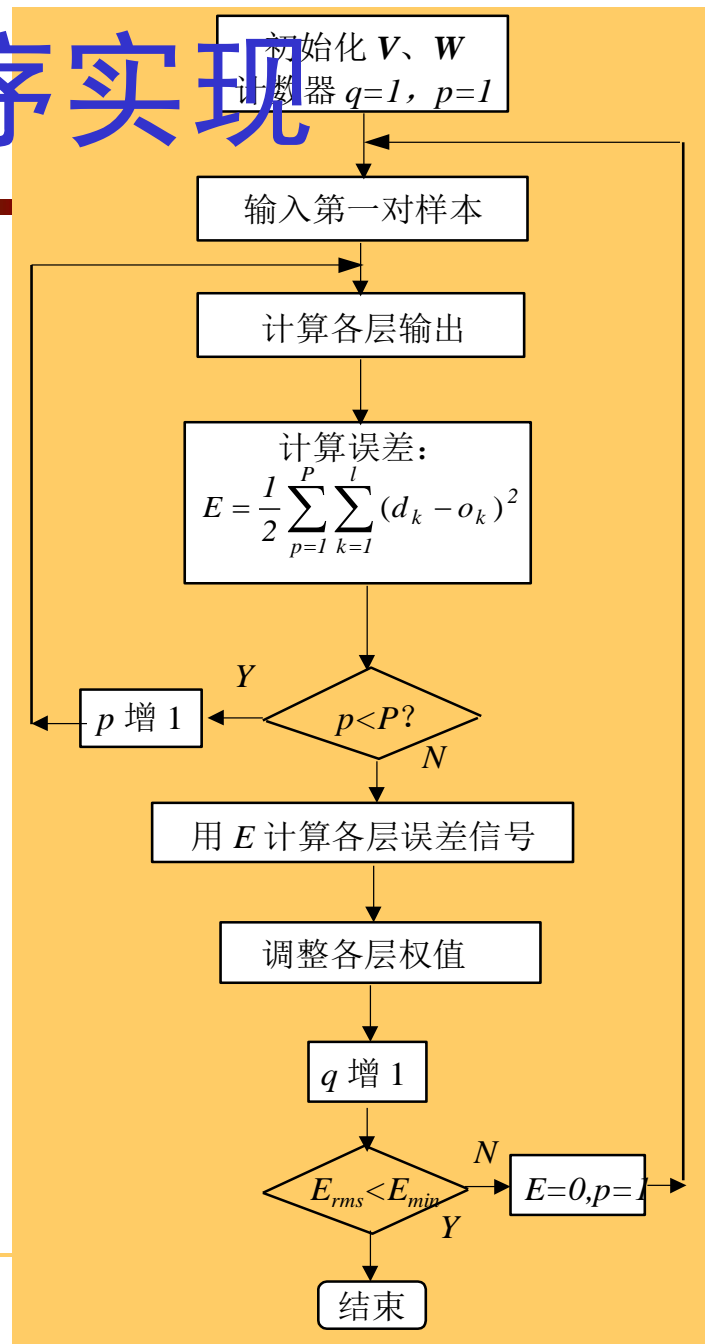


# BP算法的程序实现

另一种方法是在所有样本输入之后，计算网络的总误差：

$$E_{\text{总}} = \frac{1}{2} \sum_{p=1}^P \sum_{k=1}^l (d_k^p - o_k^p)^2$$

然后根据总误差计算各层的误差信号并调整权值。



# BP神经网络学习算法的MATLAB实现

- MATLAB中BP神经网络的重要函数和基本功能

函数名	功能
newff()	生成一个前馈BP网络
tansig()	双曲正切S型(Tan-Sigmoid)传输函数
logsig()	对数S型(Log-Sigmoid)传输函数
traingd()	梯度下降BP训练函数

# BP神经网络学习算法的MATLAB实现

- MATLAB中BP神经网络的重要函数和基本功能
  - newff()
    - 功能 建立一个前向BP网络
    - 格式 `net = newff(PR, [S1 S2...SN1], {TF1 TF2...TFN1}, BTF, BLF, PF)`
    - 说明 `net`为创建的新BP神经网络；`PR`为网络输入向量取值范围的矩阵；`[S1 S2...SN1]`表示网络隐含层和输出层神经元的个数；`{TF1 TF2...TFN1}`表示网络隐含层和输出层的传输函数，默认为‘tansig’；`BTF`表示网络的训练函数，默认为‘trainlm’；`BLF`表示网络的权值学习函数，默认为‘learngdm’；`PF`表示性能数，默认为‘mse’。

# BP神经网络学习算法的MATLAB实现

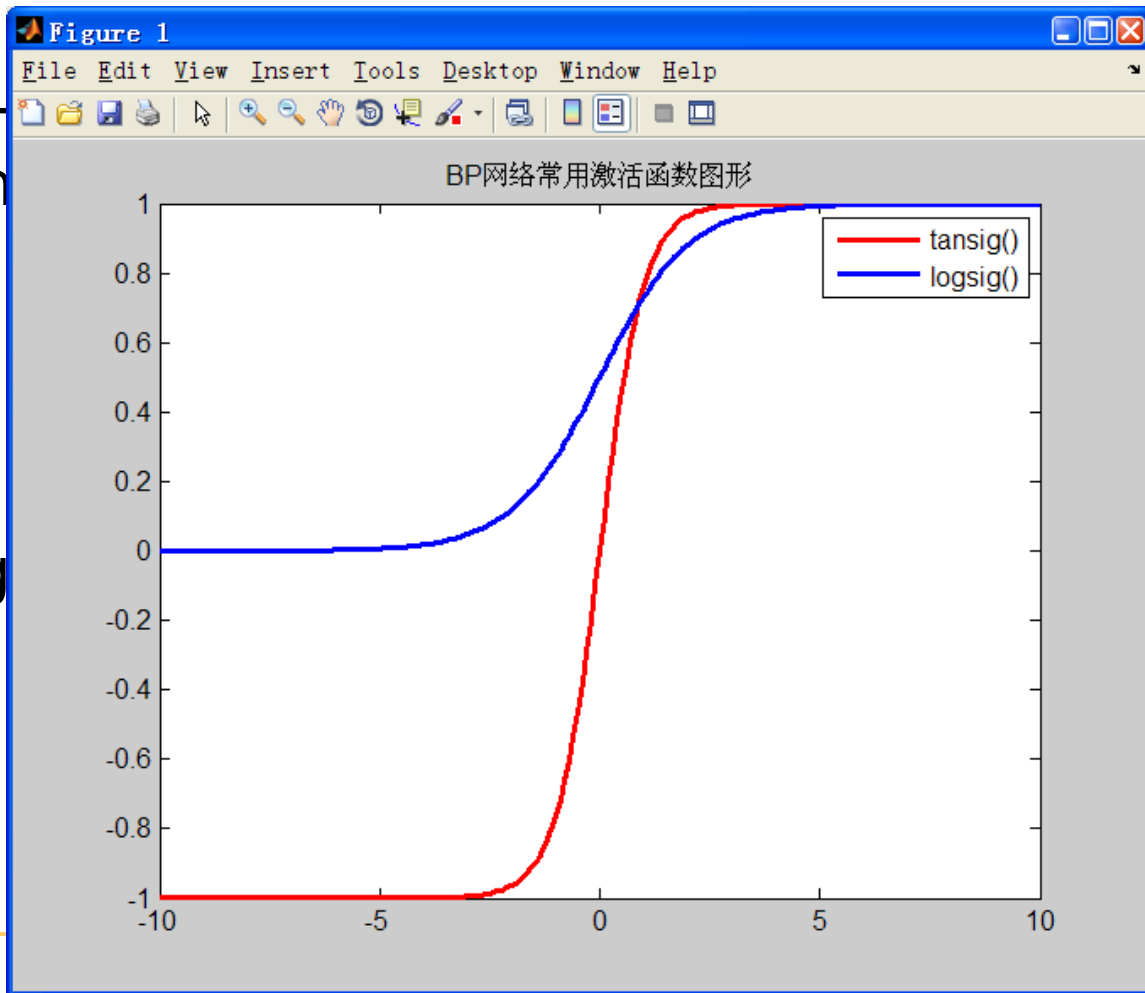
- MATLAB

- tan

- - 
    -

- log

- - 
    -



从 $(-\infty, +\infty)$   
神经元。

$+\infty$ )映射  
元。

# BP神经网络学习算法的MATLAB实现

例2:

```
net=newff([0 10],[5 1],{'tansig','purelin'},'trainlm',  
    'learngdm','mse');  
p=[0 1 2 3 4 5 6 7 8 9 10];  
t=[0 1 2 3 4 3 2 1 2 3 4];  
net.trainParam.epochs=200;  
net.trainParam.goal=0.001;  
net=train(net,p,t);  
figure  
y=sim(net,p)  
plot(P,t,P,y,'o')
```

# BP神经网络学习算法的MATLAB实现

- 例3, 神经网络对销量, 预测第四个月的销量, 复直至

输入			输出
2056	2395	2600	2298
2395	2600	2298	1634
2600	2298	1634	1600
2298	1634	1600	1873
1634	1600	1873	1478
1600	1873	1478	1900

BP神经网络对销量, 预测第四个月的销量, 复直至

月份	1	2	3	4	5	6
销量	2056	2395	2600	2298	1634	1600
月份	7	8	9	10	11	12
销量	1873	1478	1900	1500	2046	1556

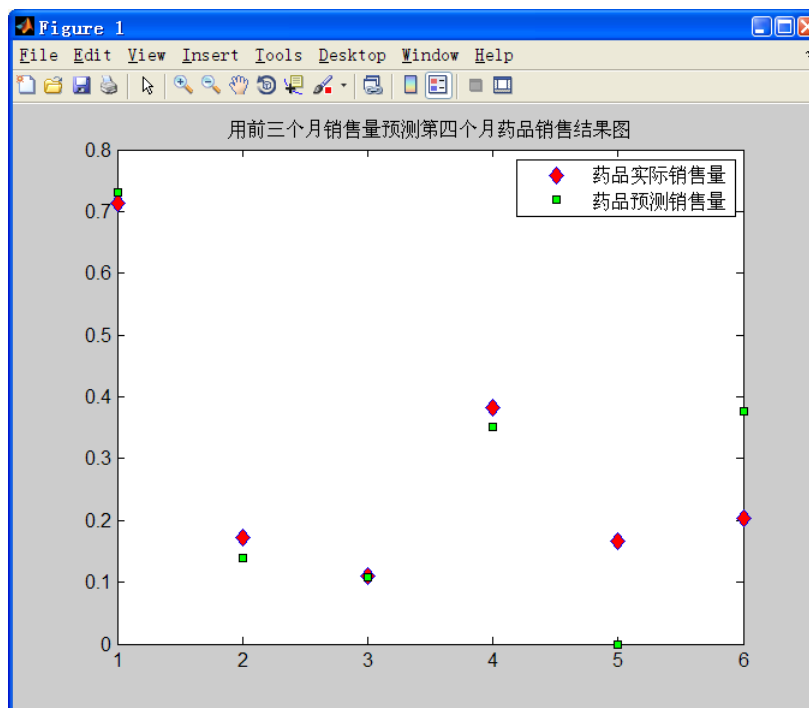
# BP神经网络学习算法的MATLAB实现

- %以每三个月的销售量经归一化处理后作为输入
- `P=[0.5152 0.8173 1.0000 ;  
0.8173 1.0000 0.7308;  
1.0000 0.7308 0.1390;  
0.7308 0.1390 0.1087;  
0.1390 0.1087 0.3520;  
0.1087 0.3520 0.0000];`
- %以第四个月的销售量归一化处理后作为目标向量
- `T=[0.7308 0.1390 0.1087 0.3520 0.0000 0.3761];`
- %创建一个BP神经网络，每一个输入向量的取值范围为[0,1]，隐含层有5个神经元，输出层有一个神经元，隐含层的激活函数为tansig，输出层的激活函数为logsig，训练函数为梯度下降函数，即2.3.2节中所描述的标准学习算法
- `net=newff([0 1;0 1;0 1],[5,1],{'tansig','logsig'},'traingd');`
- `net.trainParam.epochs=1500;`
- `net.trainParam.goal=0.01;`
  - %设置学习速率为0.1
- `LP.lr=0.1;`
- `net=train(net,P,T);`



# BP神经网络学习算法的MATLAB实现

- BP网络应用于药品预测对比图



- 由对比图可以看出预测效果与实际存在一定误差，此误差可以通过增加运行步数和提高预设误差精度进一步缩小

# BP神经网络的特点

- 非线性映射能力

- 能学习和存贮大量输入-输出模式映射关系，而无需事先了解描述这种映射关系的数学方程。只要能提供足够多的样本模式对供网络进行学习训练，它便能完成由 $n$ 维输入空间到 $m$ 维输出空间的非线性映射。

- 泛化能力

- 当向网络输入训练时未曾见过的非样本数据时，网络也能完成由输入空间向输出空间的正确映射。这种能力称为泛化能力。

- 容错能力

- 输入样本中带有较大的误差甚至个别错误对网络的输入输出规律影响很小。

# Thank You

---

Question!

Intelligence Science

<http://www.intsci.ac.cn/>

