

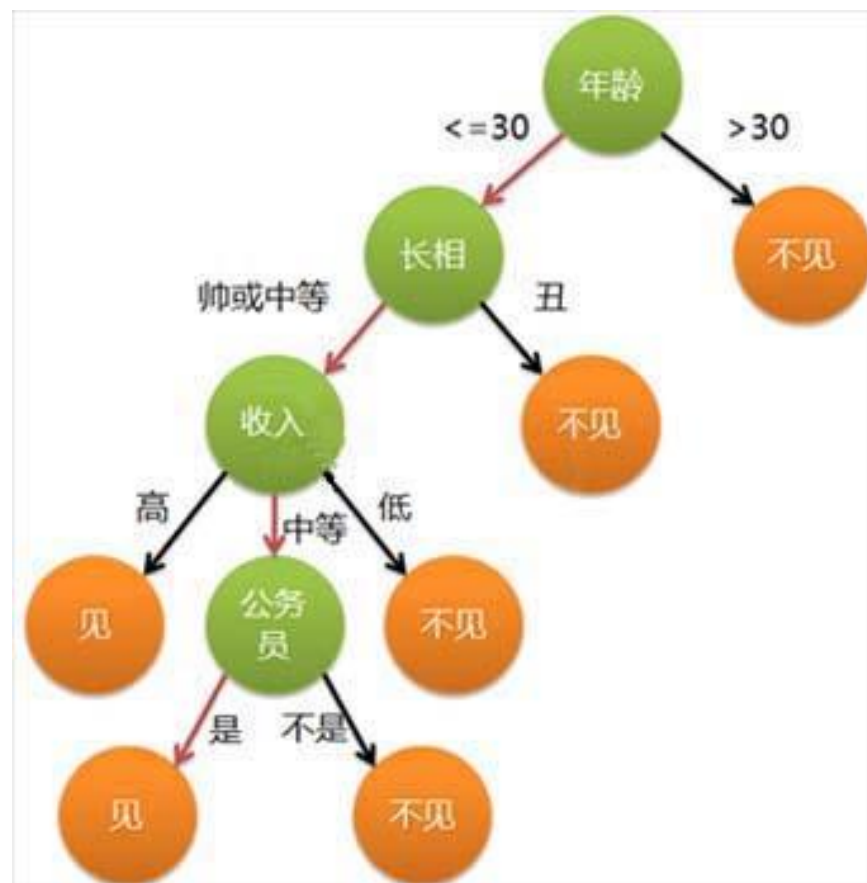
## 数据分析与R语言 第8周

2012.7.10

# 分类：分类的意义

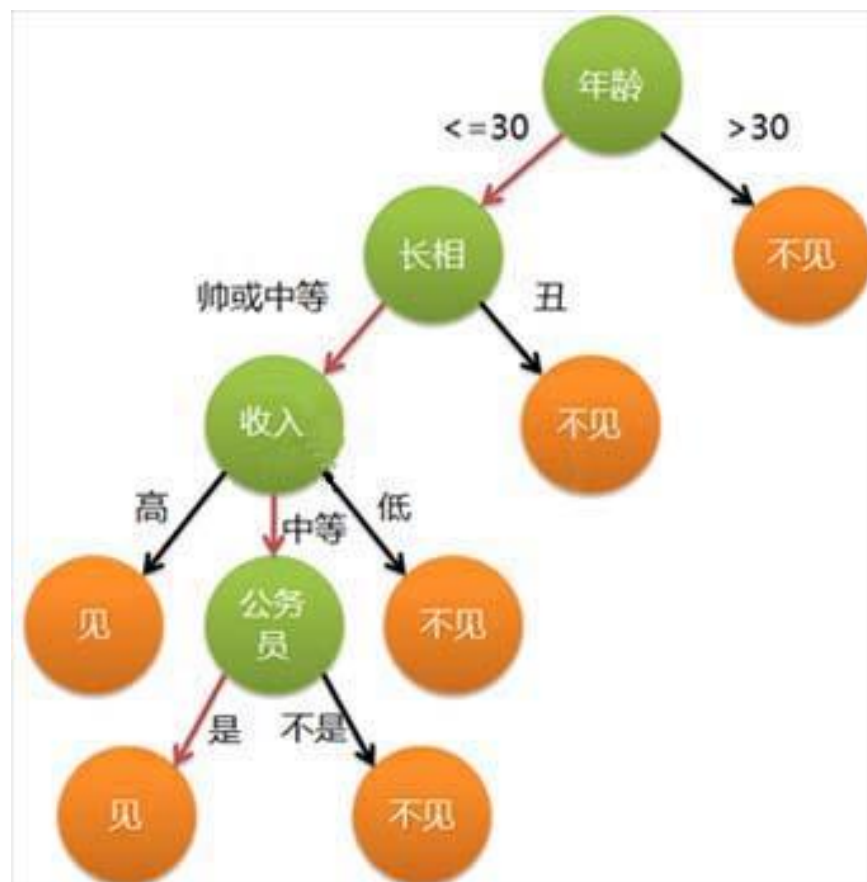
- 传统意义下的分类：生物物种
- 预测：天气预报
- 决策：yes or no
- 分类的传统模型
- 分类（判别分析）与聚类有什么差别？

- 线性判别法
- 距离判别法
- 贝叶斯分类器
- 决策树
- 支持向量机(SVM)
- 神经网络



# 决策树 decision tree

- 什么是决策树
- 输入：学习集
- 输出：分类规则（决策树）



- 用SNS社区中不真实账号检测的例子说明如何使用ID3算法构造决策树。为了简单起见，我们假设训练集合包含10个元素。其中s、m和l分别表示小、中和大。

日志密度	好友密度	是否使用真实头像	账号是否真实
s	s	no	no
s	l	yes	yes
l	m	yes	yes
m	m	yes	yes
l	m	yes	yes
m	l	no	yes
m	s	no	no
l	m	no	yes
m	s	no	yes
s	s	yes	no

- 设L、F、H和R表示日志密度、好友密度、是否使用真实头像和账号是否真实，下面计算各属性的信息增益。

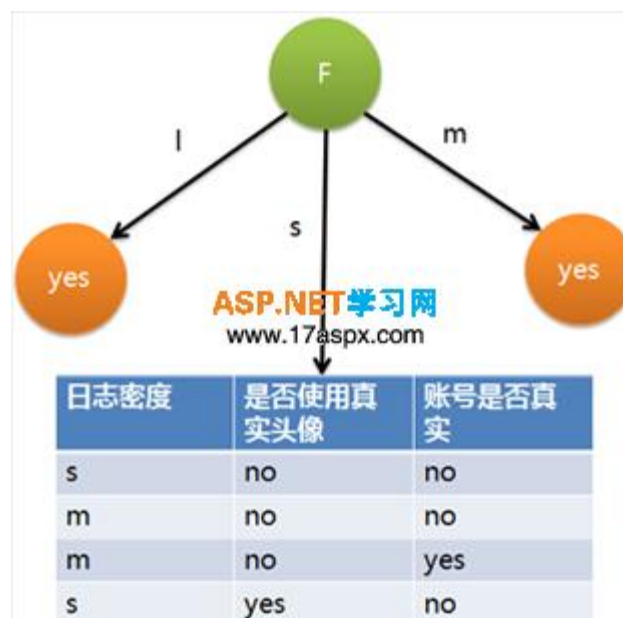
$$info(D) = -0.7\log_2 0.7 - 0.3\log_2 0.3 = 0.7 * 0.51 + 0.3 * 1.74 = 0.879$$

$$info_L(D) = 0.3 * \left(-\frac{0}{3}\log_2 \frac{0}{3} - \frac{3}{3}\log_2 \frac{3}{3}\right) + 0.4 * \left(-\frac{1}{4}\log_2 \frac{1}{4} - \frac{3}{4}\log_2 \frac{3}{4}\right) + 0.3 * \left(-\frac{1}{3}\log_2 \frac{1}{3} - \frac{2}{3}\log_2 \frac{2}{3}\right) = 0 + 0.326 + 0.277 = 0.603$$

$$gain(L) = 0.879 - 0.603 = 0.276$$

## 根据信息增益选择分裂属性

- 因此日志密度的信息增益是0.276。用同样方法得到H和F的信息增益分别为0.033和0.553。因为F具有最大的信息增益，所以第一次分裂选择F为分裂属性，分裂后的结果如下图所示：



## 递归+分而治之

- 在上图的基础上，再递归使用这个方法计算子节点的分裂属性，最终就可以得到整个决策树。
- 这个方法称为ID3算法，还有其它的算法也可以产生决策树
- 对于特征属性为**连续值**，可以如此使用ID3算法：先将D中元素按照特征属性排序，则每两个相邻元素的中间点可以看做潜在分裂点，从第一个潜在分裂点开始，分裂D并计算两个集合的期望信息，具有最小期望信息的点称为这个属性的最佳分裂点，其信息期望作为此属性的信息期望。



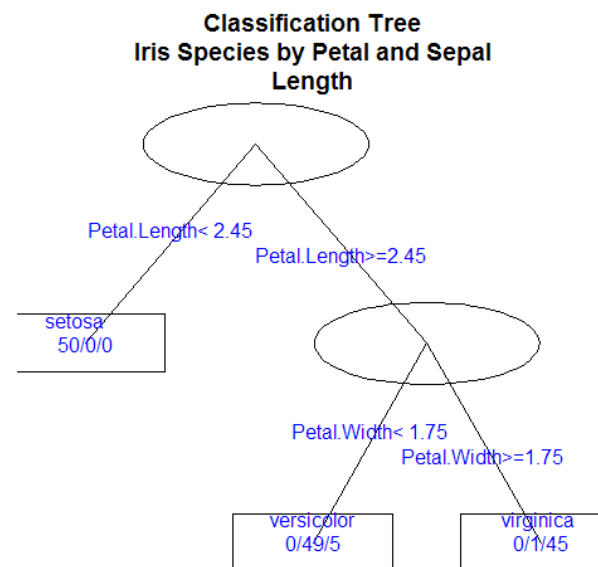
# R语言实现决策树：rpart扩展包

- 以鸢尾花数据集作为算例说明

```
iris.rp = rpart(Species~., data=iris,  
method="class")
```

```
plot(iris.rp, uniform=T, branch=0,  
margin=0.1, main= " Classification  
Tree\nIris Species by Petal and Sepal  
Length")
```

```
text(iris.rp, use.n=T, fancy=T, col="blue")
```

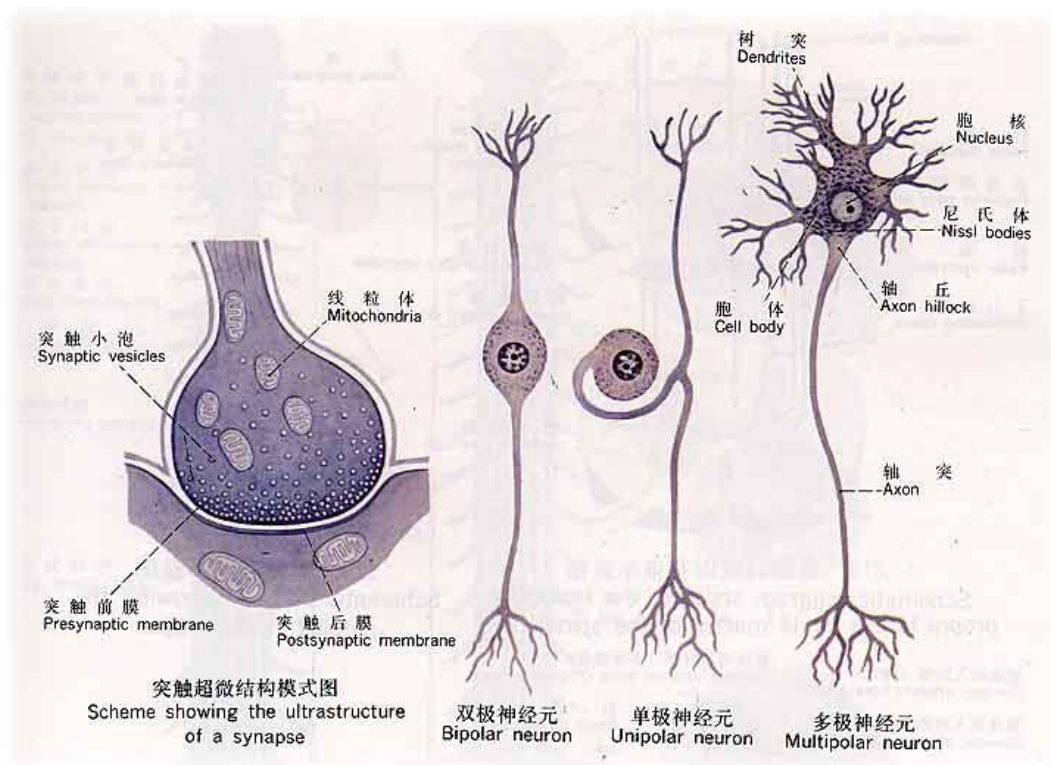


Rule 1: if  $\text{Petal.Length} \geq 2.45 \& \text{Petal.Width} < 1.75$ , then it is versicolor(0/49/5)  
Rule2: if  $\text{Petal.Length} \geq 2.45 \& \text{Petal.Width} \geq 1.75$ , then it is virginica (0/1/45)  
Rule 3: if  $\text{Petal.Length} < 2.45$ , then it is setosa (50/0/0)

- 算法主要思想：

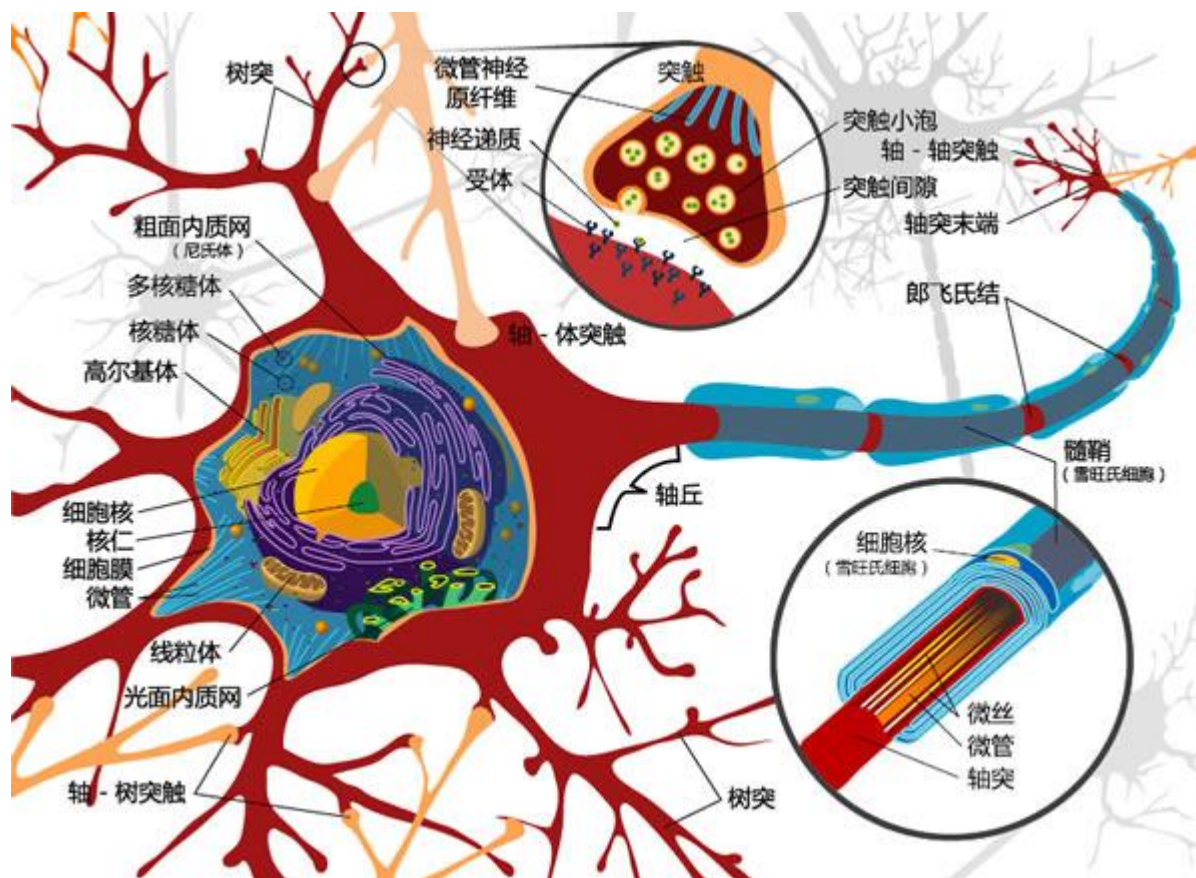
- 1 选取**k个**和待分类点**距离**最近的样本点
- 2 看1中的样本点的分类情况，**投票**决定待分类点所属的类

## ■ 人类神经系统原理



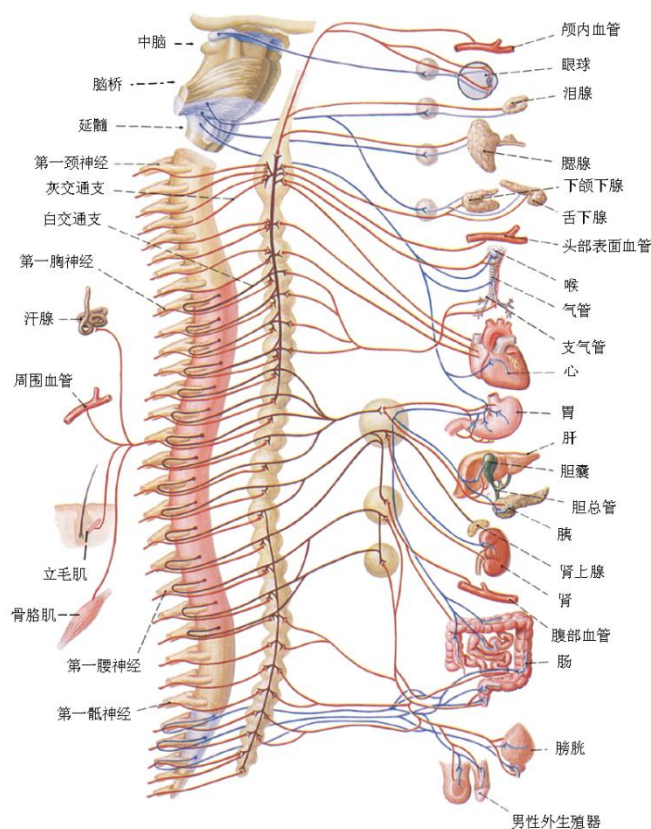
2012.7.10

## ■ 人类神经系统原理



2012.7.10

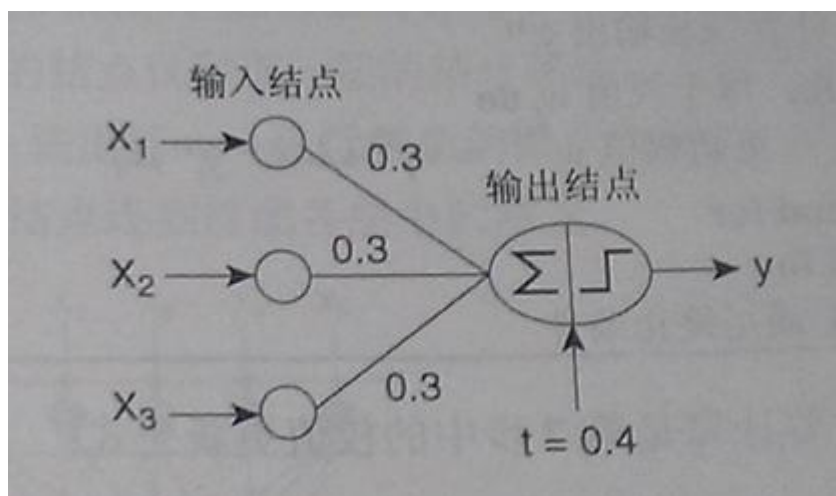
## ■ 人类神经系统



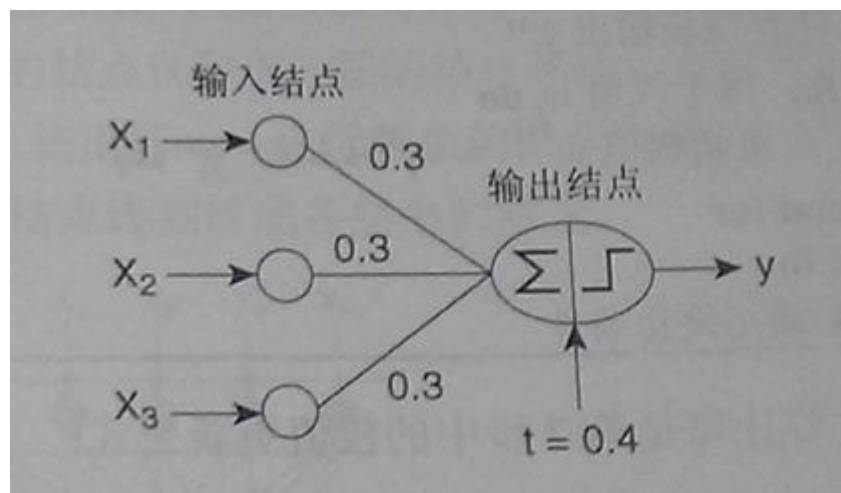
自主神经系概观

2012.7.10

- ANN=Artificial Neural Networks , 人工神经网络
- 神经元 —— 感知器



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





# 例子

## 建立数据

$x1=c(1,1,1,1,0,$   
 $0,0,0)$

$x2=c(0,0,1,1,0,$   
 $1,1,0)$

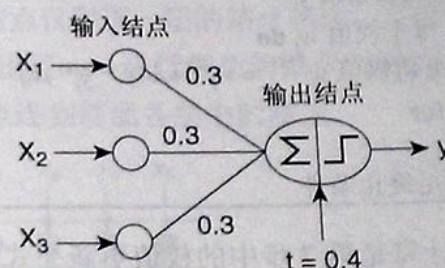
$x3=c(0,1,0,1,1,$   
 $0,1,0)$

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

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

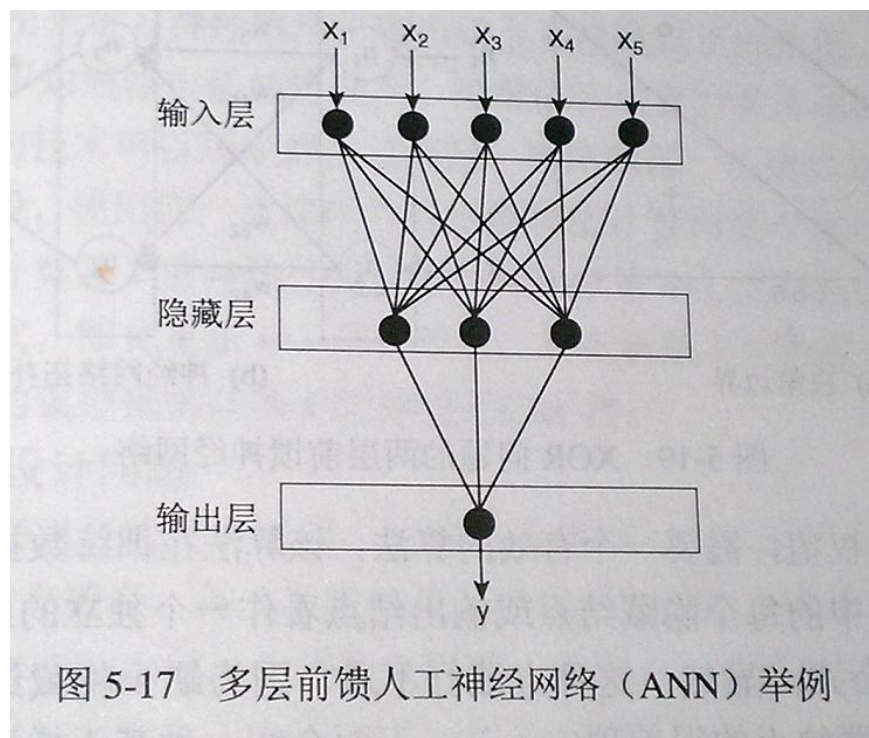
$$\hat{y} = \begin{cases} 1 & \text{如果 } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0 \\ -1 & \text{如果 } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 < 0 \end{cases}$$



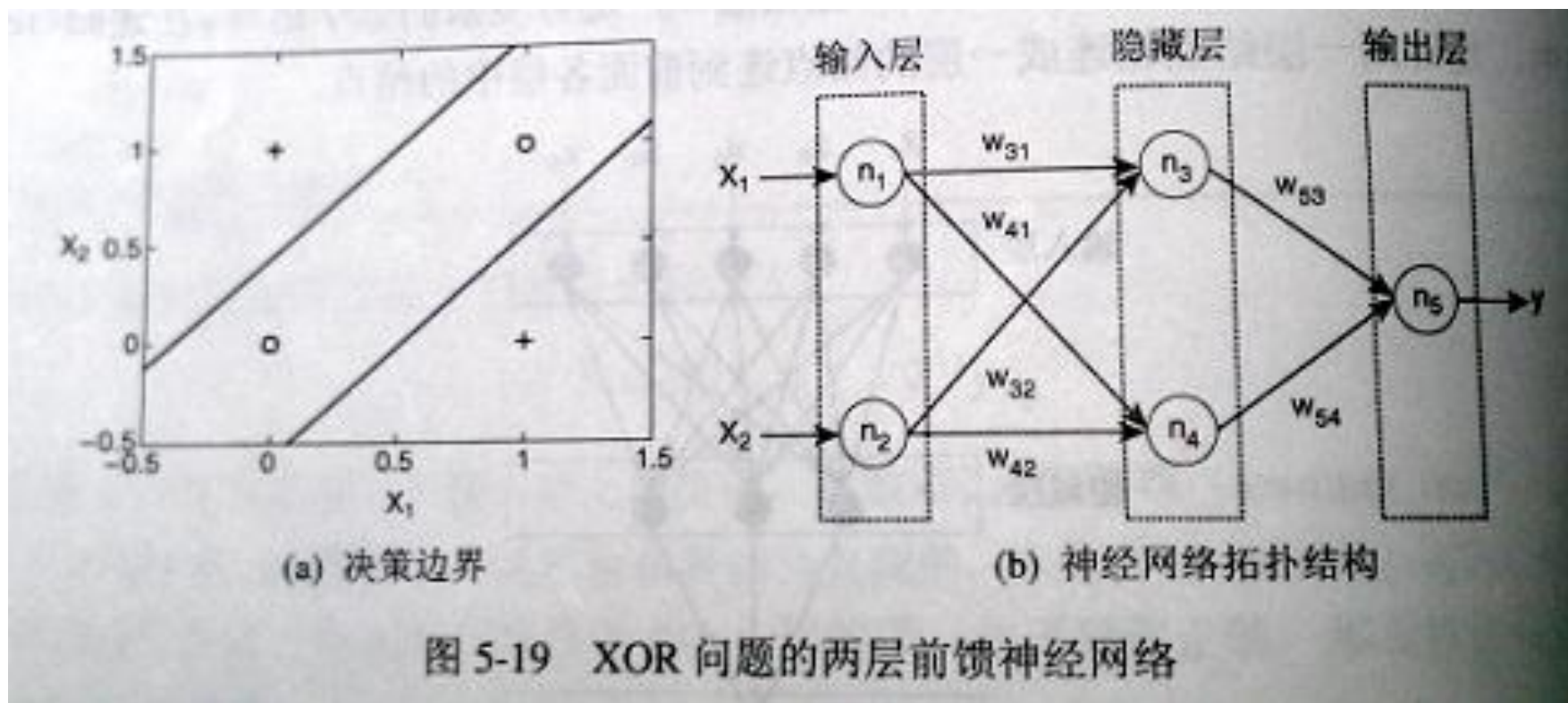
## 算法 5.4 感知器学习算法

- 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

- 隐藏层与隐藏节点
- 前馈 —— 每一层的节点仅和下一层节点相连

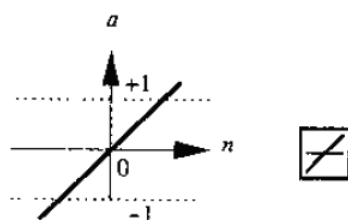


# 单个感应器无法解决的问题



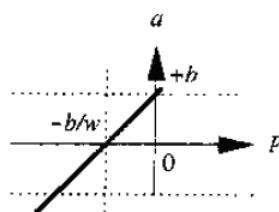
2012.7.10

# 各种激活函数



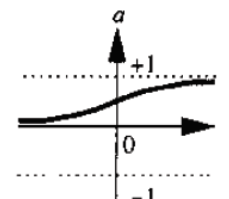
$$a = \text{purelin}(n)$$

线性传输函数



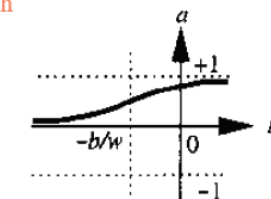
$$a = \text{purelin}(wp + b)$$

单输入 *purelin* 神经元



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

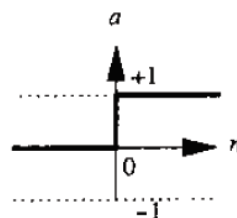
Log-Sigmoid 传输函数



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

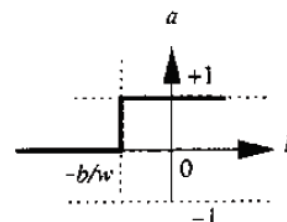
单输入 *logsig* 神经元

81tech



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






硬极限传输函数








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

单输入 *hardlim* 神经元

# 各种激活函数

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

# 各种激活函数

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.criterion="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
> |
```

## ■ 用BP神经网络处理非线性拟合问题

本章拟合的非线性函数为

$$y = x_1^2 + x_2^2$$

该函数的图形如图 2-1 所示。

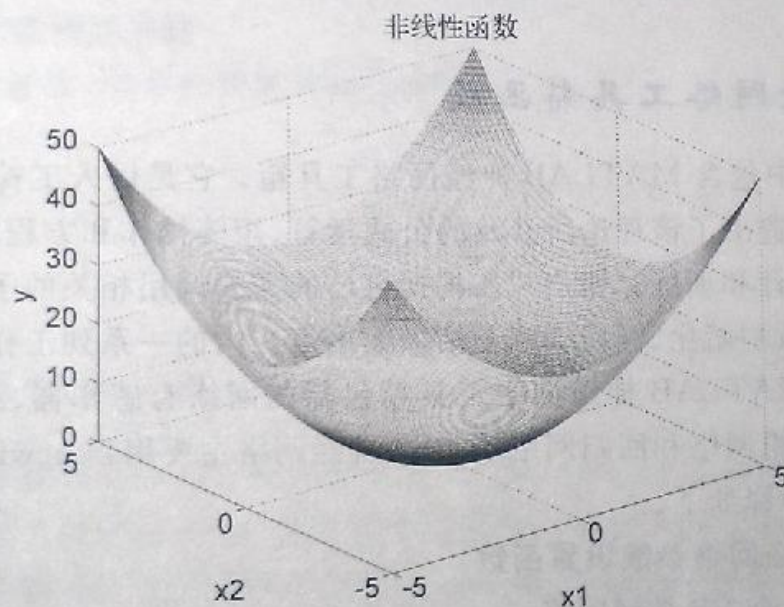


图 2-1 非线性函数图形

2012.7.10

# 人工神经网络应用举例

- 随机抽选2000个样本。1900个作为学习集，100个作为验证集
- 先使用2-5-1类型的BP神经网络进行训练和拟合
- 建立神经网络模型并用学习集进行训练

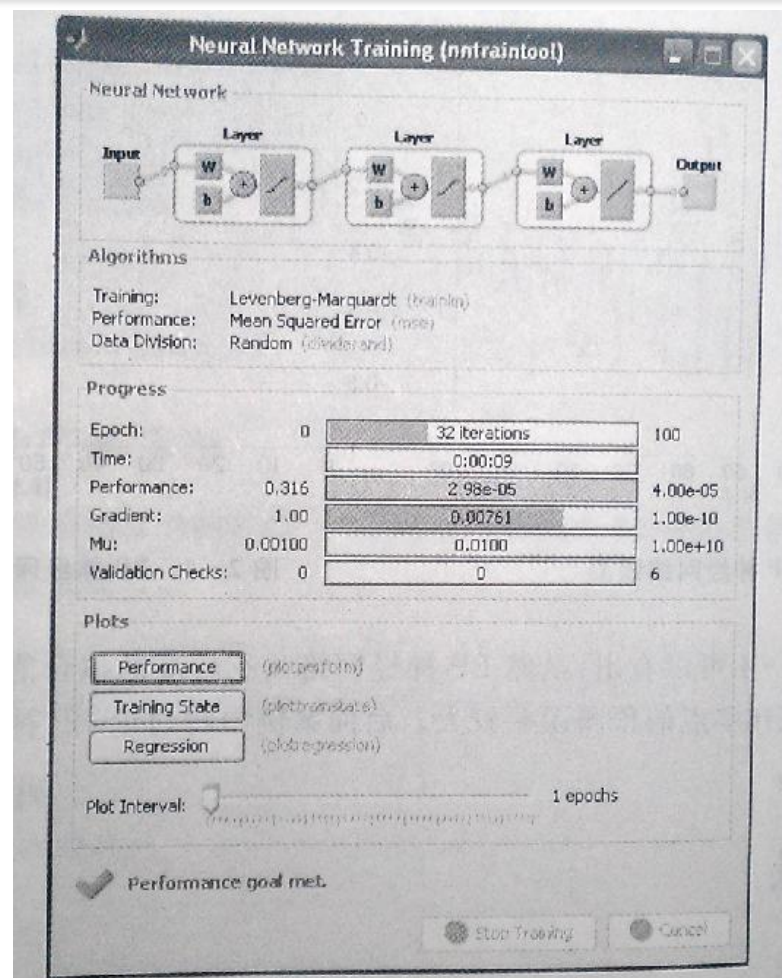
```
% BP 神经网络构建
net = newff(inputn,outputn,5);

% 网络参数配置(迭代次数,学习率,目标)
net.trainParam.epochs = 100;
net.trainParam.lr = 0.1;
net.trainParam.goal = 0.00004;

% BP 神经网络训练
net = train(net,inputn,outputn);
```

# 人工神经网络应用举例

- 存在较大误差（拟合不足？）
- 需要结构更加复杂的神经网络
- 使用双隐含层神经网络，每层5节点



2012.7.10

- 训练样本数量
- 隐含层数与每层节点数。层数和节点太少，不能建立复杂的映射关系，预测误差较大。但层数和节点数过多，学习时间增加，还会产生“**过度拟合**”的可能。预测误差随节点数呈现先减少后增加的趋势。
- 激活函数的影响

表 2-4 不同转移函数对应预测误差

隐含层函数	输出层函数	误差百分比	均方误差
logsig	tansig	40.63%	0.902 5
logsig	purelin	0.08%	0.000 1
logsig	logsig	352.65%	181.251 1
tansig	tansig	31.90%	1.173 3
tansig	logsig	340.90%	162.969 8
tansig	purelin	1.70%	0.010 7
purelin	logsig	343.36%	143.763 34
purelin	tansig	120.08%	113.028 1
purelin	purelin	196.49%	99.012 1

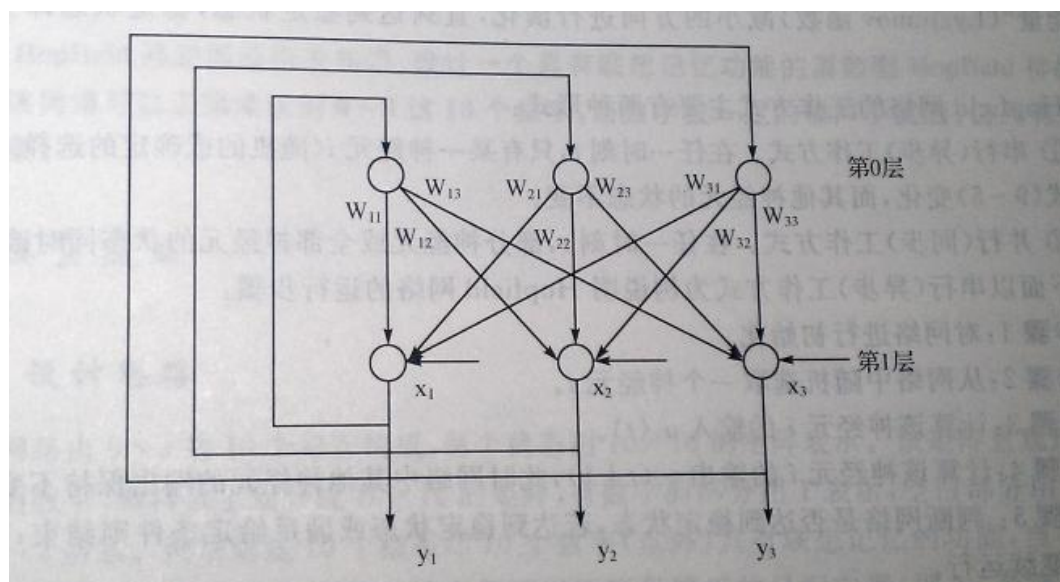
2012.7.10

- 作业：把上述例子用R语言实现
- 构想中的数据挖掘比赛

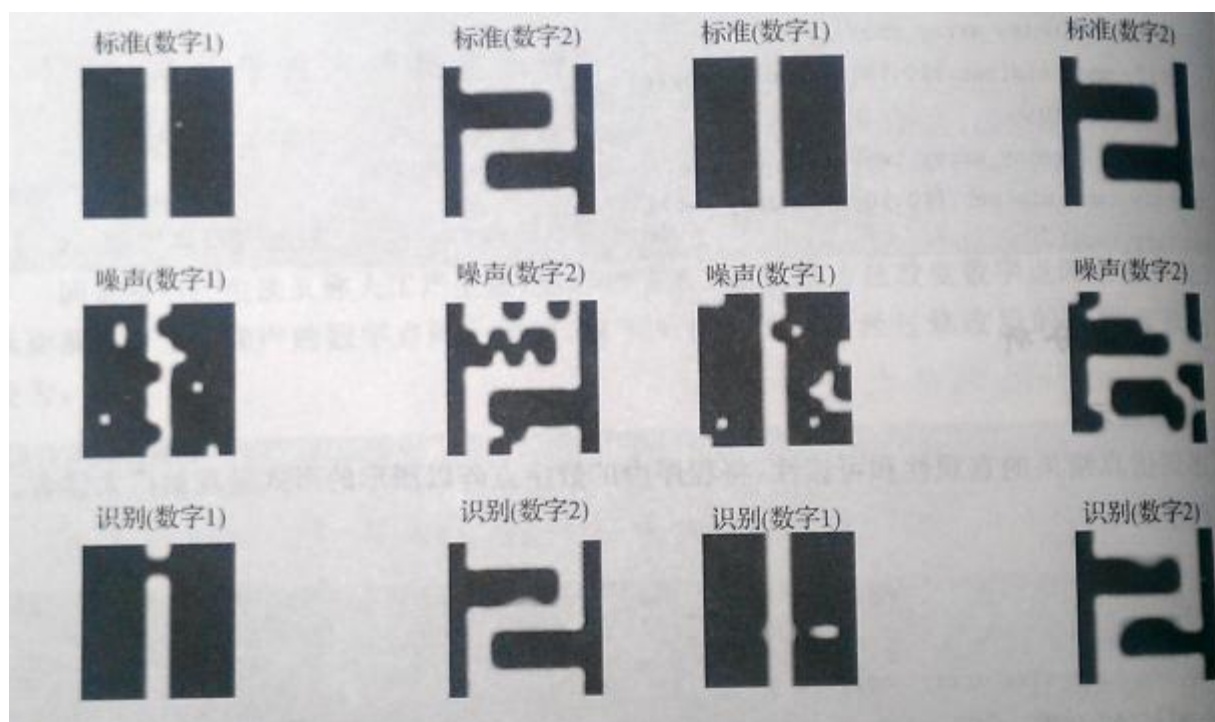


# Hopfield神经网络

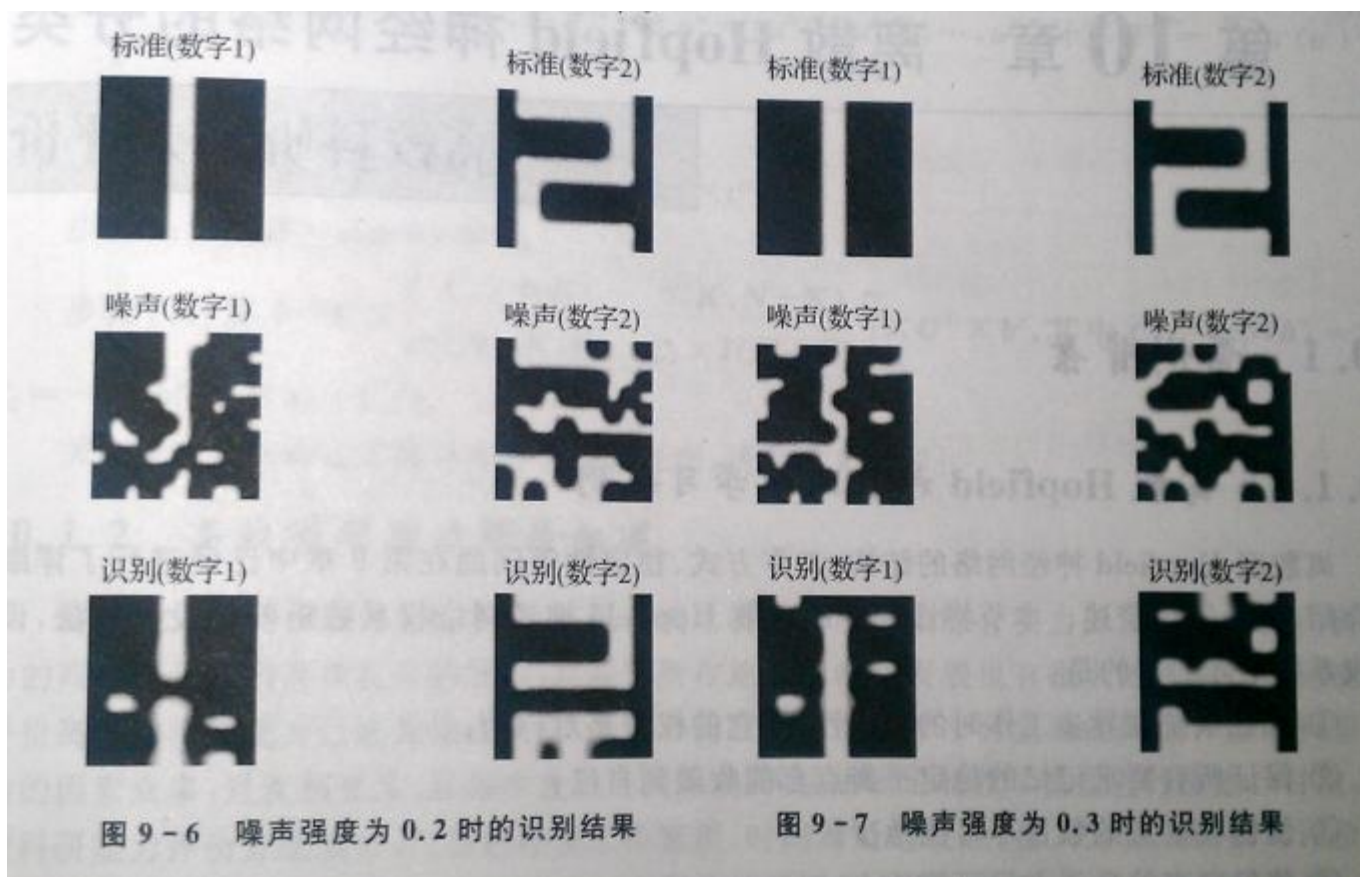
- 人类的联想记忆能力
- Hopfield人工神经网络能模拟联想记忆功能，曾被誉为人工神经网络方法的以此革命和再生
- Hopfield人工神经网络按动力学方式运行



2012.7.10







# OCR的思路

- 把图像信息数字化为1和-1二值矩阵
- 标准图样生成的矩阵作为Hopfield网络的目标向量
- 生成Hopfield网络
- 使用带噪音的矩阵测试
- 输出已经降噪，再和标准目标矩阵（向量）比对，找出最接近者

# 神经网络方法的优缺点

- 可以用统一的模式去处理高度复杂问题
- 便于元器件化，形成物理机器
- 中间过程无法从业务角度进行解释
- 容易出现过度拟合问题

## 参考书

- 《神经网络设计》，机械工业出版社
- 《神经网络与机器学习》，机械工业出版社
- 《人工神经网络理论、设计及应用》，化学工业出版社
- 《MATLAB神经网络30个案例分析》，北京航空航天大学出版社



# Thanks

## FAQ时间