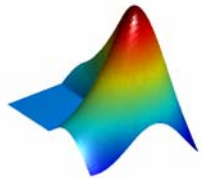


MATLAB与数字信号处理

神经网络

6、神经网络技术

6.1 神经网络概述

6.2 感知机模型

6.3 自适应线性元件

6.4 多层前向神经网络模型与BP算法

6.5 Hopfield网络

6.6 竞争神经网络

6.7 深度神经网络和深度学习

6.1 神经网络概述

- 人工神经网络: **Atificial Neural Network (ANN)**

  是由大量处理单元广泛互连而成的网络, 是对人脑的抽象、简化和模拟，反映人脑的基本特征。在现代生物学研究人脑所取得的成果基础上提出的, 用于模拟人类大脑神经网络的结构和行为特性, 解决实际工程问题.

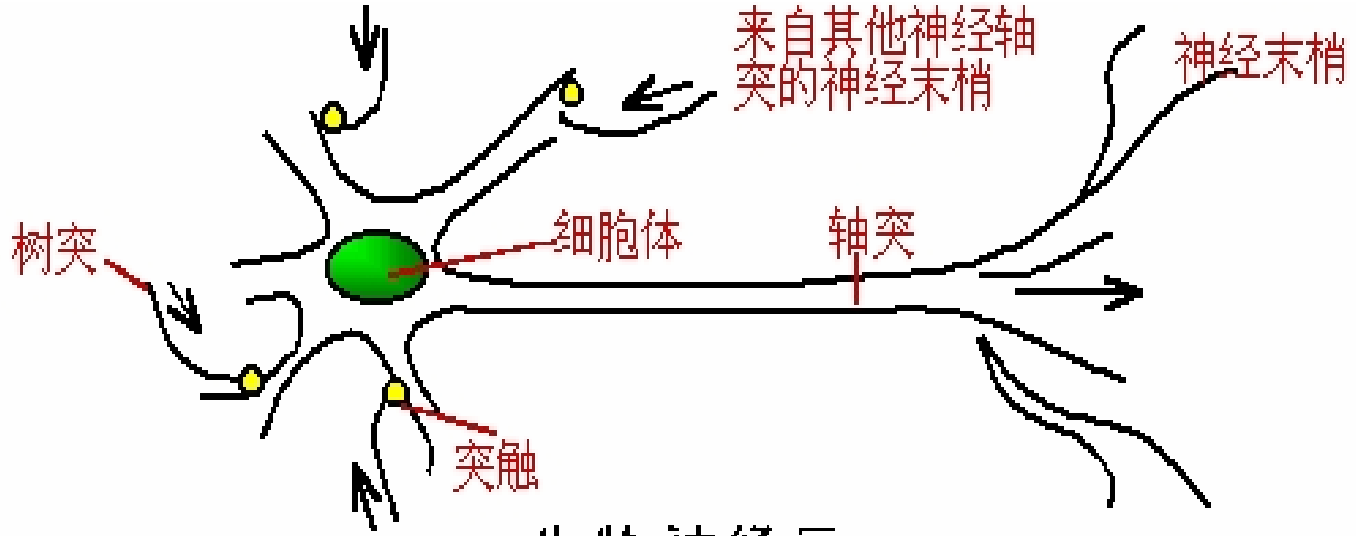
- 三要素（神经元、网络拓扑、学习算法）

6.1 神经网络概述

一、神经元

1)生物神经元: 是生物神经系统的基本单元。

- 人脑: 约 100亿神经元组成的巨系统。
- 神经元组成: 细胞体,轴突,树突和突触



生物神经元

- 轴突:由细胞体向外延伸的最长一条分支，

  输出电缆　　传出细胞体的电信号。
- 树突:由细胞体向外延伸其他许多较短分支，

  输入端　接收来自四面八方的神经冲动.
- 突触: 轴突和树突接触　　功能性接触
- 细胞体:神经元主体,由细胞核,细胞质,细胞膜

  处理器：对来自其它神经元的神经冲动进行处理，产生一神经输出信号。

怎样处理？

细胞膜将细胞体分为内外两部分，膜电位

阈值　　兴奋状态　　抑制状态

说明：随着脑科学和生物控制论的发展，人们对生物神经元的结构和功能有了进一步认识，神经元并不是一个简单的双稳态逻辑元件，是超级的微生物信息处理机或控制机。

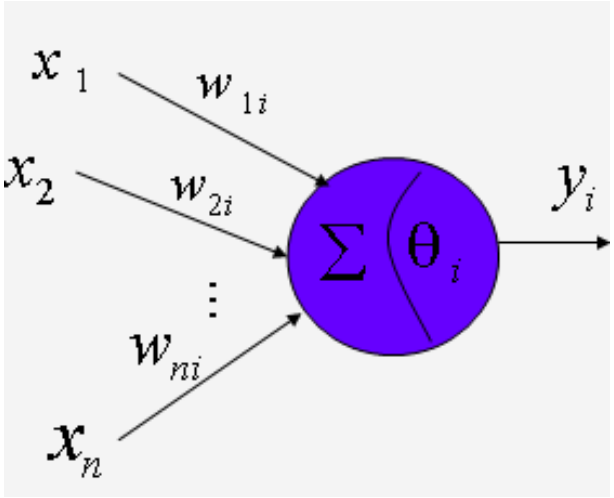
6.1 神经网络概述

2）形式神经元：生物神经元的抽象和模拟。即人工神经网络的基本处理单元，在网络中称为节点或网点。

$$y_i = f(A_i)$$

作用函数

$$A_i = \sum_{j=1}^{n} w_{ij} x_j + b_i$$



把若干个输入加权求和，并将这个加权和非线性处理后输出。

6.1 神经网络概述

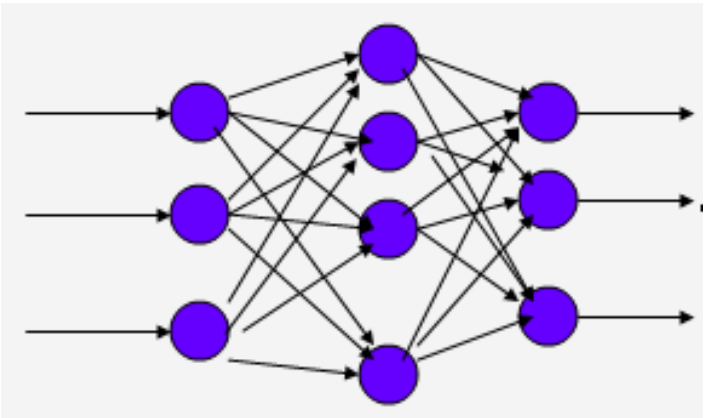
二、网络拓扑：指网络的结构以及神经元之间的联接方式。

分层网络：将神经元按功能分为若干层

输入层：接收外部输入模式，传递给隐层

隐层：内部处理

输出层：产生输出模式



相互连接型网络（反馈网络）：

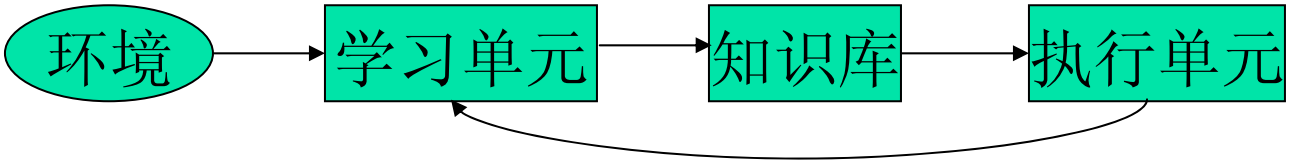
任意两个神经元之间都是可达的。

三、学习

1）学习　　与人脑惊人的相似特性。

- 学习：（概念的认识是不断变化的）

　　系统为了适应环境而产生的某种长远变化，这种变化使得系统能够更有效地在下一次完成同一或同类工作。



- 学习往往离不开训练

　课程学习

　学习过程是对一系列符号的学习，存储及以后的提取和应用的过程。

2)记忆 （学习和记忆是分不开）

记忆是大脑对过去经历中发生过的事情的反映，是获 得知识和经验的保存。

大脑是如何记忆？

记忆存取机制的研究，形成两大学派

- 化学学派：人脑经学习获得的信息是记录在某些生物大分子。如蛋白质，核糖核酸等等。
- 突触修正学派：人脑经学习获得的信息分布存储 在神经元之间的突触连接上。 **Hebb**

突触修正假说 是**NN**学习记忆机制的理论基础。

联想记忆：大脑记忆的一个重要特征

自联想记忆和异联想记忆

6.1 神经网络概述

3) 学习算法：网络的学习算法是以满足网络所需的性能为目标，决定联接各神经元的初始权值及在训练中调整权值的方法。

学习方法可分为监督学习与非监督学习：

监督学习（有导师学习）：训练时，同时向网络提供输入模式及输出的样板模式（导师），在不断输入不同训练模式的同时调整权值，从而使输出模式尽量接近样板模式；

非监督学习（无导师学习）：它是一种自动聚类过程，通过输入训练模式的加入，不断调整权值以使输出能够反映输入训练模式的分布特点。

6.1 神经网络概述

- 学习规则

1）Hebb学习规则：

Hebb学习规则是最著名的学习规则，是为了纪念神经心理学家**Hebb（1949）**而命名的。

- Hebb学习规则的基本思想是：

如果神经元u_i接收来自另一神经元u_j的输出，则当这两个神经元同时兴奋时，从u_i到w_{ij}的权值u_j就得到加强。

具体到前述的神经元模型，可以将Hebb规则表现为如下的算法形式：

$$\Delta w_i = \eta y x_i$$

公式中Δw_i是对第i个权值的修正值，η是控制学习速度的系数。

6.1 神经网络概述

2）纠错学习（误差修正学习规则）：

基本思想:

设某神经网络的输出层中的一个神经元i，

实际输出 $y_i(n)$

期望输出（目标输出） $d(n)$

实际输出与期望输出之间存在着误差 $e(n)$

$$e(n)=d(n)-y_i(n)$$

调整突触权值，使误差信号减少。

代价函数 $$E(n)=\frac{1}{2}e^2(n)$$

反复调整突触权值使代价函数达到最小或使系统达到一个稳定状态，就完成了学习过程。

6.1 神经网络概述

δ学习规则的步骤：

(1).选择 一组初始权值

(2).计算某一输入模式对应期望输出与实际输出的误差

(3).更新权值

$$w_{ji}(t+1) = w_{ji}(t) + \eta[d_j - y_j(t)]x_i(t)$$

(4).返回步骤(2)，直到对所有训练模式，网络输出都满足

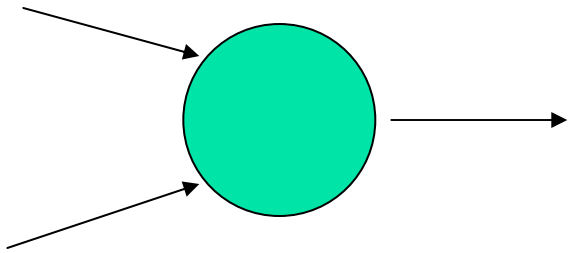
例：通过学习用2输入M-P模型神经元，

实现下列功能:

x1: 0 0 1 1

x2: 0 1 0 1

d: 1 1 0 0



过程：

假设：初始权值**w1=0.2，w2=-0.5，θ=0.1**；

计算某一输入模式对应的实际输出模式；

- **M-P模型的数学描述：**

$$y_i = f(A_i) = \begin{cases} 1 \\ 0 \end{cases}$$

$$A_i = \sum_{j=1}^{n} w_{ji} x_j - \theta_i$$

- 激活函数为：

  A=w1*x1+w2*x2- θ

  =0.2*x1 - 0.5*x2 - 0.1

- 对应四组输入模式

  输出**y**分别为：**0　0　1　0**

- 修正：δ学习算法　η**(0，1)**

  学习因子 假设取**0.1** 也可如下：

$$\eta = \frac{1}{2}\left(\left|\sum_{i=1}^{2} w_i(t)x_i(t) - \theta(t)\right| + \alpha\right)$$

模式对 **0 0 1**

$$w_1(1) = w_1(0) + \eta(d_j - y_j(0))x_1 = 0.2 + 0.1[1-0]*0 = 0.2$$

$$w_2(1) = -0.5$$

$$\theta(1) = \theta(0) + \eta(d_i - y_i(0))x_0 = 0.1 + 0.1[1-0](-1) = 0$$

输入模式 **0 1** 对应的实际输出为 **0**

$$w_1(2) = w_1(1) + \eta(d_j - y_j(0))x_1 = 0.2 + 0.1[1-0]*0 = 0.2$$

$$w_2(2) = -0.5 + 0.1(1-0)*1 = -0.4$$

$$\theta(2) = 0 + 0.1[1-0](-1) = -0.1$$

重复这个过程直到所有模式都满足要求，学习结束

学习：权值修正的过程

- 应用：

  智能控制，模式识别，信号处理，计算机视觉，优化计算，知识处理，生物医学工程等。

- 神经网络的特点

  - 学习和归纳能力

    归纳指神经网络在学习（训练）过程中能为新的输入产生合理的输出。

  - 大量的并行分布结构
  - 容错性
  - 非线性特性

6.2 感知机模型

1. 基本感知机（二层网络）

输入层单元：接收外部输入模式，并传给输出层单元

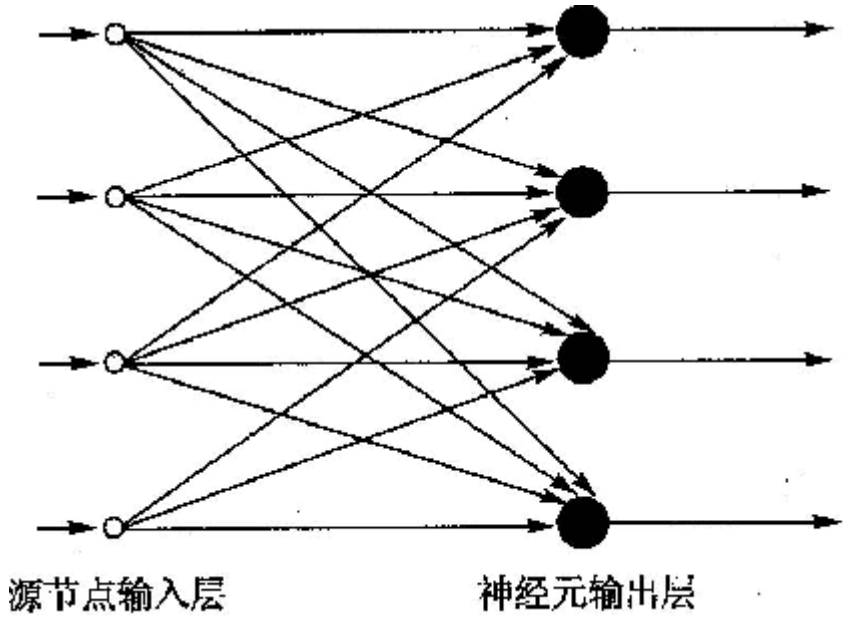
输出层单元：对输入模式处理，产生输出模式。

输出为二进制(阈值函数)，两层间全互连，权值可调

有导师学习

最简单的感知机：

M--P模型



2. MATLAB 实现

例： 采用单一感知器神经元解决一个简单的分类问题：将四个输入矢量分为两类，其中两个矢量对应的目标值为1，另两个矢量对应的目标值为0，即

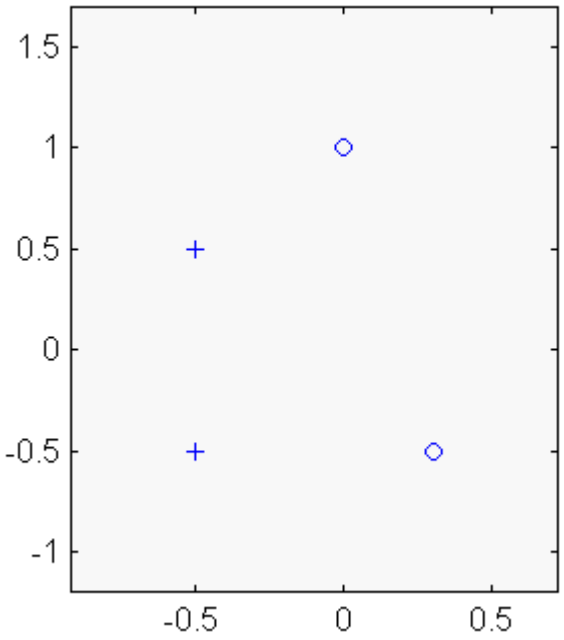
输入矢量：P=[-0.5 –0.5 0.3 0.0;
-0.5 0.5 -0.5 1.0]

目标分类矢量：T=[1.0 1.0 0.0 0.0]

解　首先定义输入矢量及相应的目标矢量：

```
P=[-0.5 –0.5  0.3  0.0;
   -0.5   0.5  -0.5 1.0];
T=[1.0 1.0 0.0 0.0];
```

输入矢量可以用图来描述，对应于目标值0的输入矢量用符号‘0’表示，对应于目标值1的输入矢量符号‘+’表示。

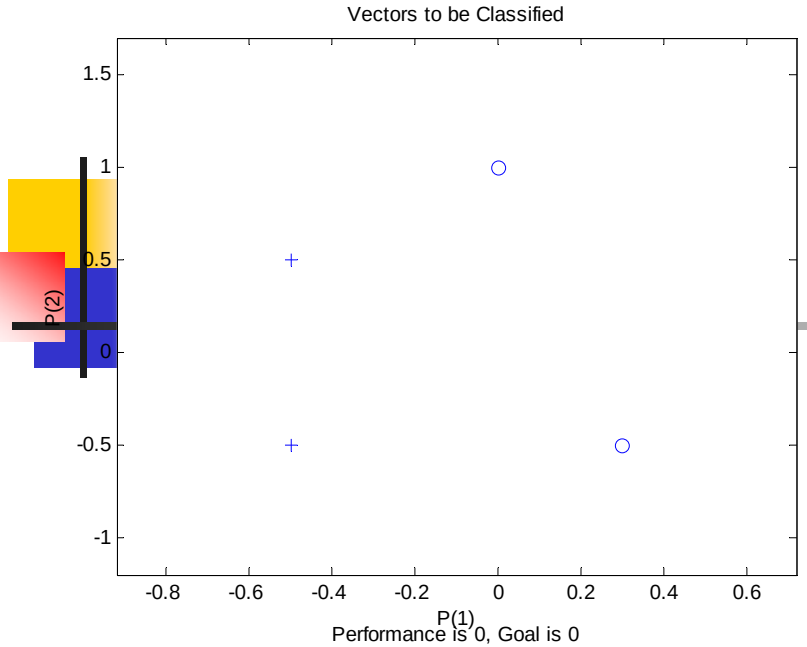


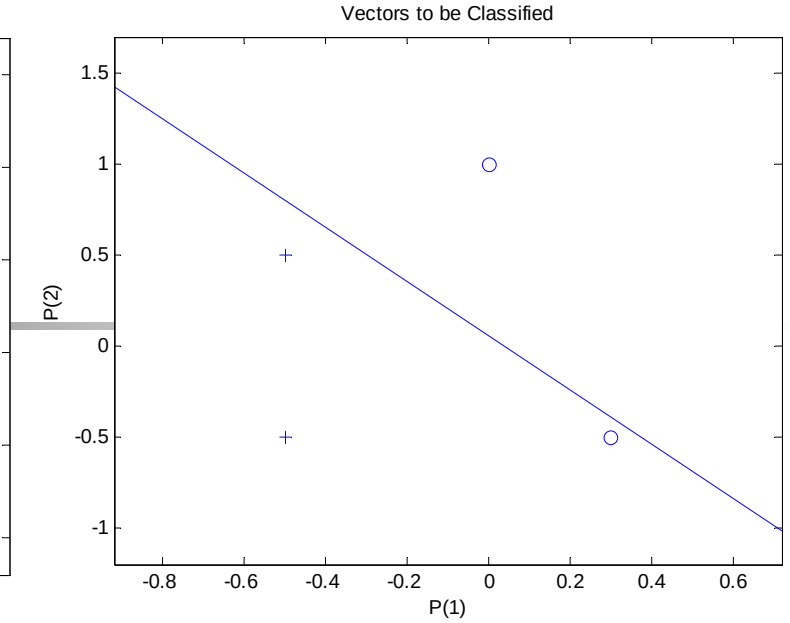
下面给出本例的MATLAB程序

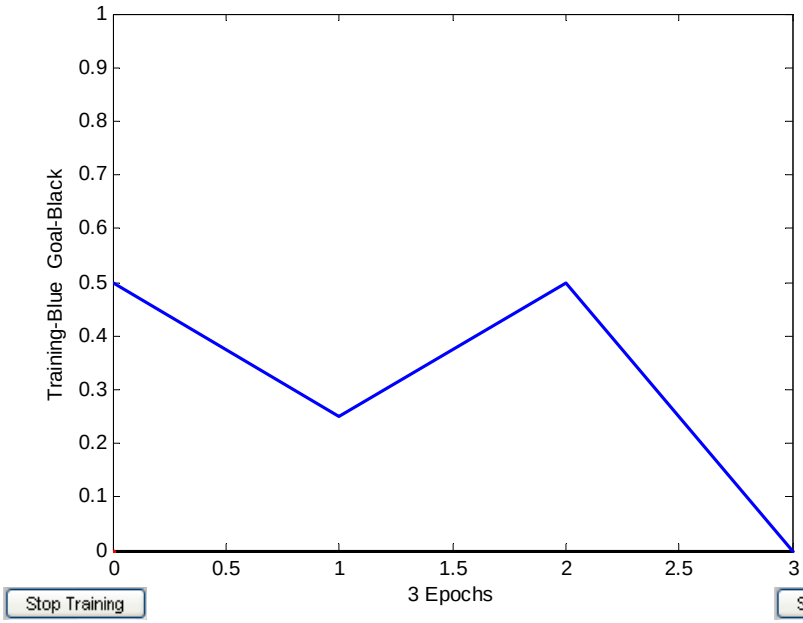
```
% Example
%NEWP —— 建立一个感知器神经元
%INIT —— 对感知器神经元初始化
%TRAIN —— 训练感知器神经元
%SIM —— 对感知器神经元仿真
clc
% P为输入矢量
P = [-0.5 -0.5 +0.3 +0.0;
     -0.5 +0.5 -0.5 1.0];
% T为目标矢量
T = [1 1 0 0];
pause
```

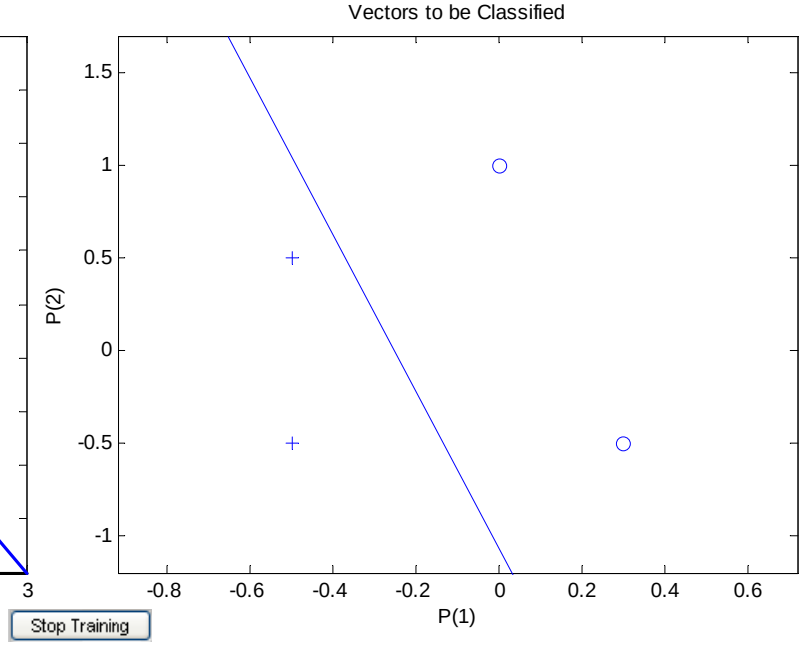
```
% 绘制输入矢量图
plotpv(P,T);
pause
% 定义感知器神经元并对其初始化
net=newp([-0.5 0.3;-0.5 1],1);
net.initFcn='initlay';
net.layers{1}.initFcn='initwb';
net.inputWeights{1,1}.initFcn='rands';
net.layerWeights{1,1}.initFcn='rands';
net.biases{1}.initFcn='rands';
net=init(net);
plotpc(net.iw{1,1},net.b{1})
pause
```

```
% 训练感知器神经元
net=train(net,P,T);
pause
% 绘制结果分类曲线
plotpv(P,T)
plotpc(net.iw{1,1},net.b{1});
pause
% 利用训练完的感知器神经元分类
p = [-0.5; 0];
a = sim(net,p)
echo off
```


Vectors to be Classified
P(2)
P(1)


Vectors to be Classified
P(2)
P(1)


Performance is 0, Goal is 0
Training-Blue Goal-Black
3 Epochs
Stop Training


Vectors to be Classified
P(2)
P(1)
Stop Training

训练结束后得到的分类结果，分类线将两类输入矢量分开，其相应的训练误差的变化图看出，经过几步训练后，就达到了误差指标的要求。

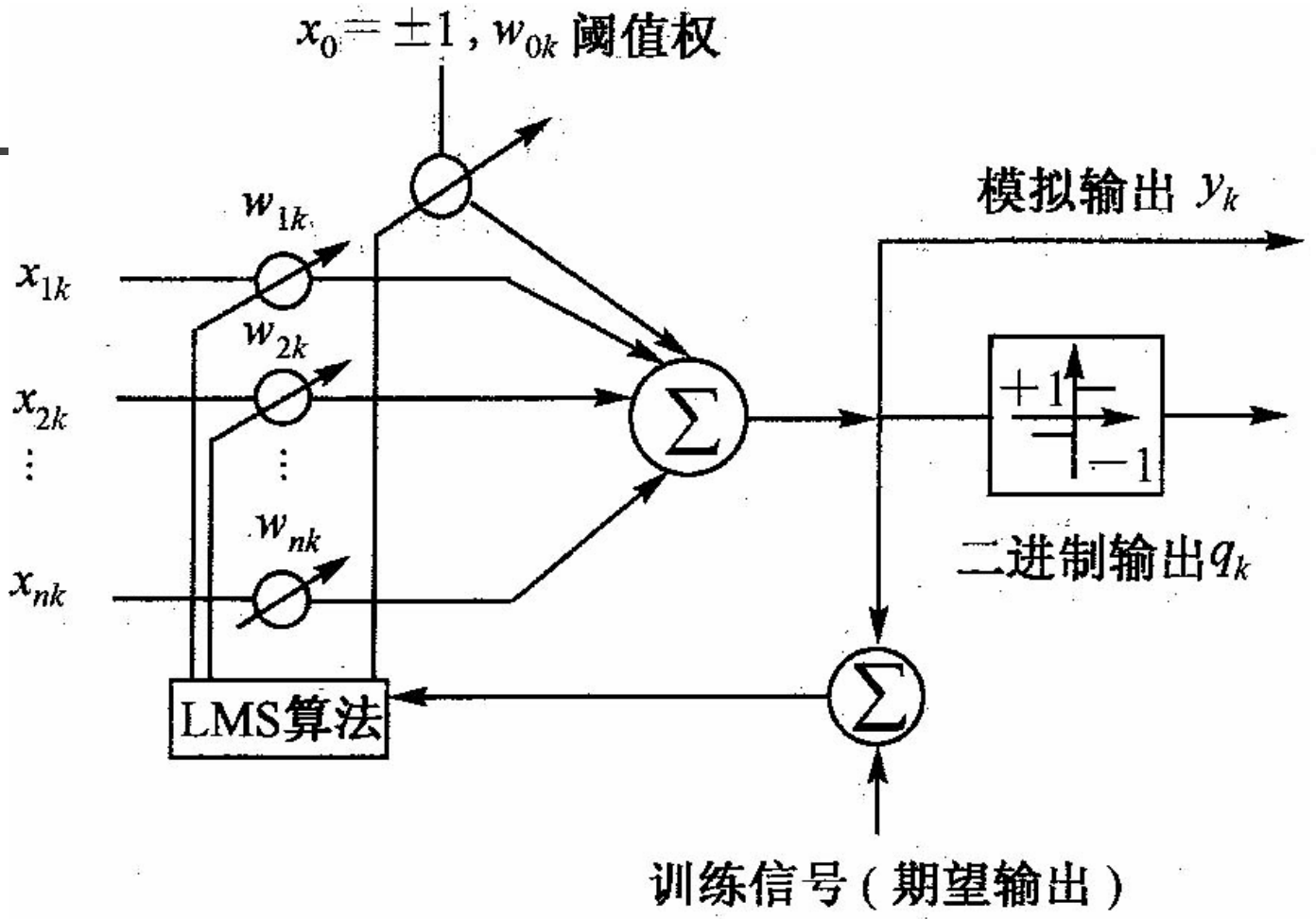
```
p = [-0.5; 0];
a = sim(net,p)
```

结果：

```
a=1
```

§6.3 自适应线性元件

- 自适应线性元件模型是由美国斯坦福大学的**Widrow**和**Hoff**提出的。
- **LMS**学习规则（**Least Mean Square**）
- 有导师学习（监督学习）
- 在信号处理中：自适应滤波、预测、噪声消除


$x_0 = \pm 1$，w_{0k} 阈值权
模拟输出 y_k
w_{1k}
x_{1k}
w_{2k}
x_{2k}
w_{nk}
x_{nk}
Σ
+1
−1
二进制输出q_k
LMS算法
Σ
训练信号（期望输出）

例： 设计自适应线性元件

实现输入矢量到输出矢量的变换关系。

输入矢量**P=[1.0 -1.2]**

输出矢量**T=[0.5 1.0]**

err_goal=0.001

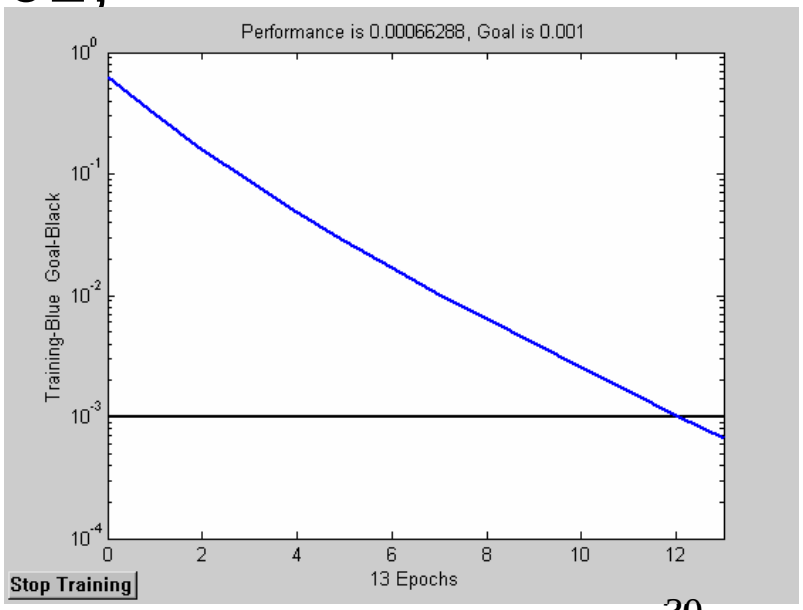
Max_epoch=200

学习因子：**0.1**

```
net = newlin([-1.2 1],1,[0 1],0.1);
net = init(net);
P1 = [1.0 -1.2];
T1 = [0.5  1.0];
net.trainParam.epochs = 200;
net.trainParam.goal = 0.001;
net = train(net,P1,T1);
Y = sim(net,P1)
mse=(sum((Y-T1).^2)/2)
```

Y ： 0.4666 0.9712

mse： 9.7514e-004



6.4 多层前向神经网络模型

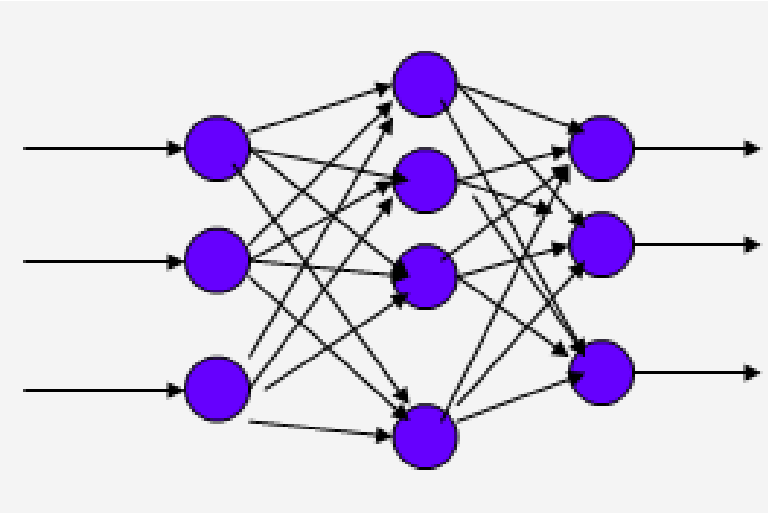
- 多层前向神经网络模型（多层感知机）

输入层：接收外部输入模式，传递给隐层

隐层：内部处理

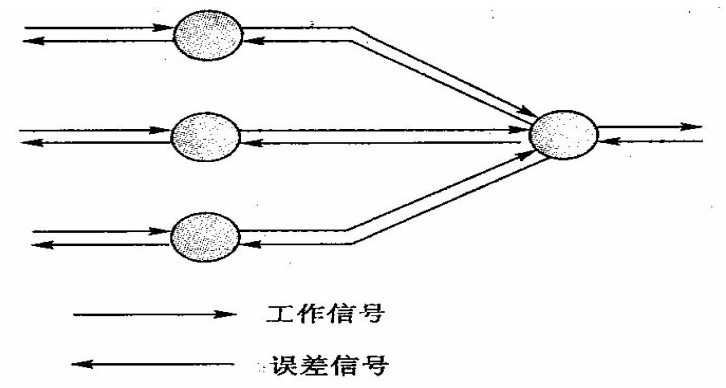
输出层：产生输出模式

即信号从输入层输入，经隐层传给输出层，由输出层得到输出信号。



- **BP实现多层网络学习过程：**
  正向传播和反向传播组成

  - 正向传播：对一给定的输入模式，由输入层传到隐层单元，经隐层单元的逐层处理后，传到输出层产生输出模式。若输出层不能得到期望的输出，转入反向传播。
  - 反向传播：将误差信号延原来的连接通路返回，通过修改各神经元的权值，使得误差信号减小。

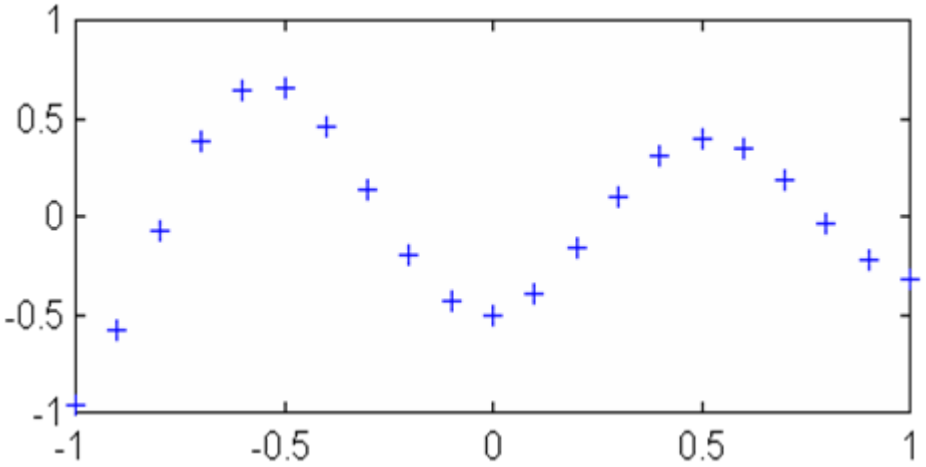


- 有导师学习

  对简单δ规则的推广。梯度最速下降法

例:函数逼近

应用三层**BP**网络来完成函数逼近的任务，其中隐层的神经元个数选为**5**个。



解 首先定义输入样本和目标矢量

```
P=-1:.1:1;
T=[-.9602  -.5770  -.0729  .3771
    .6405  .6600  .4609   .1336
   -.2013  -.4344  -.5000 -.3930
   -.1647  .0988 .3072  .3960
   .3449  .1816 -.0312  -.2189  -.3201];
```

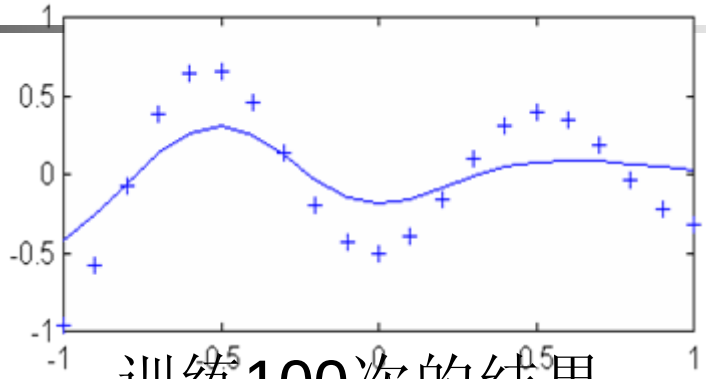
建立网络：

net=newff(PR,[S1 S2…SNl],{TF1 TF2…TFNl},BTF,BLF,PF)

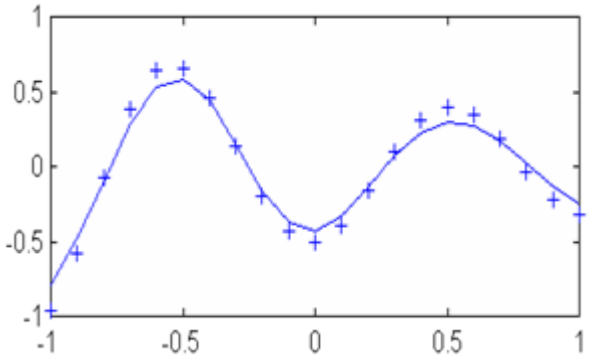
进行训练： net=train(net,P,T);

测试网络： Y=sim(net,Pt,T); 泛化能力

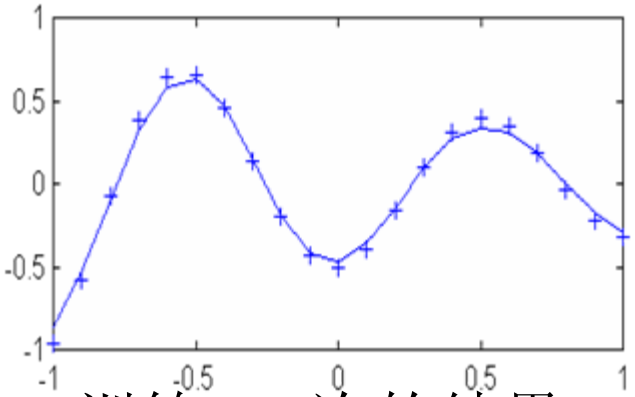
下图给出了网络输出值随训练次数的增加而变化的过程，并给出了454次训练后的最终网络结果，以及网络的误差纪录。



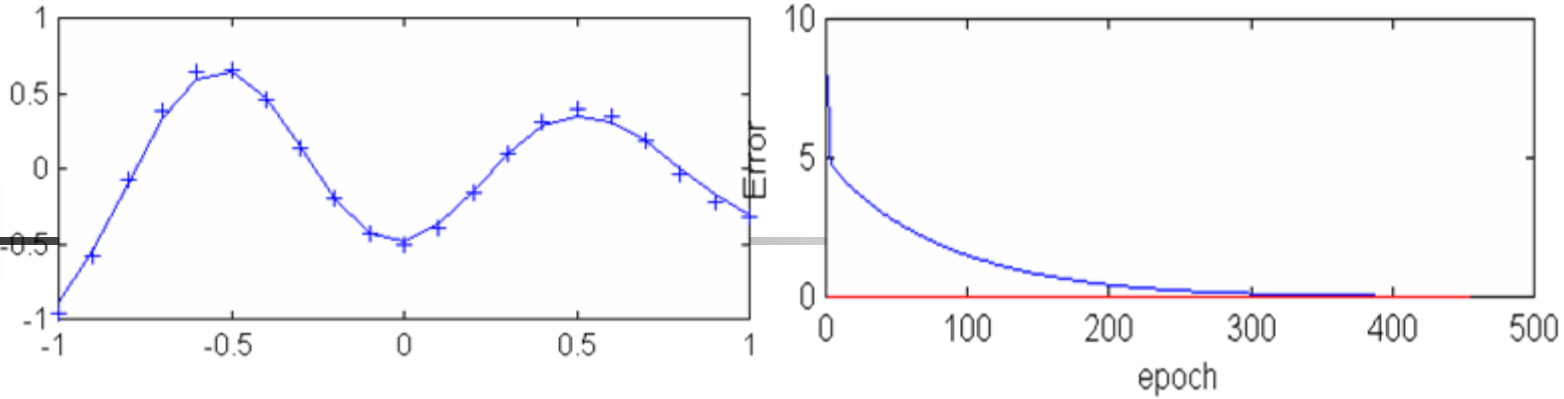
训练100次的结果



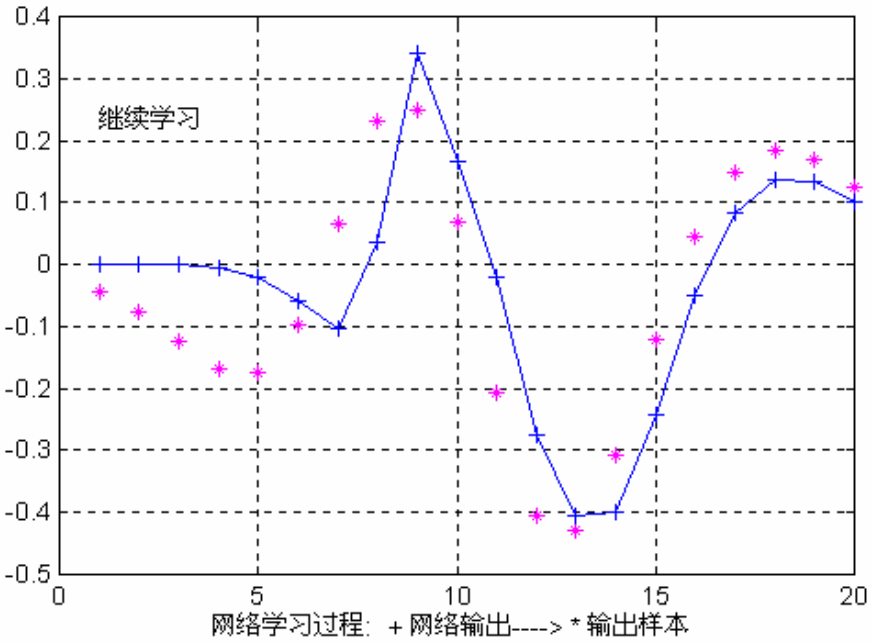
训练300次的结果



训练400次的结果

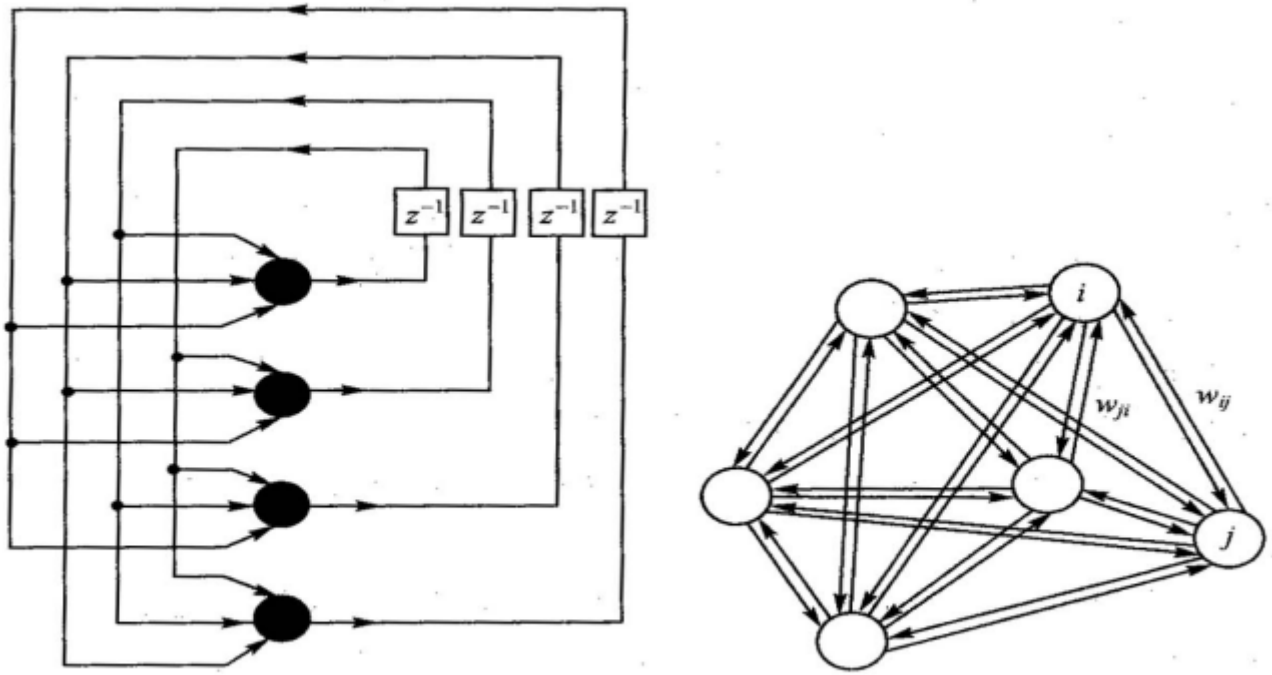


训练结束后的网络输出与误差结果



6.5 Hopfield网络

- 单层全互连含有对称突触连接的反馈网络



Hopfield 神经网络结构

例　含有两个神经元的**Hofield**网络设计

稳定点：　　$T=[1\ \ -1;\ -1\ \ 1]$

解：

建立：net = newhop(T);

测试：[Y,Pf,Af,E,perf]=sim(net,P,Pi,Ai,T)

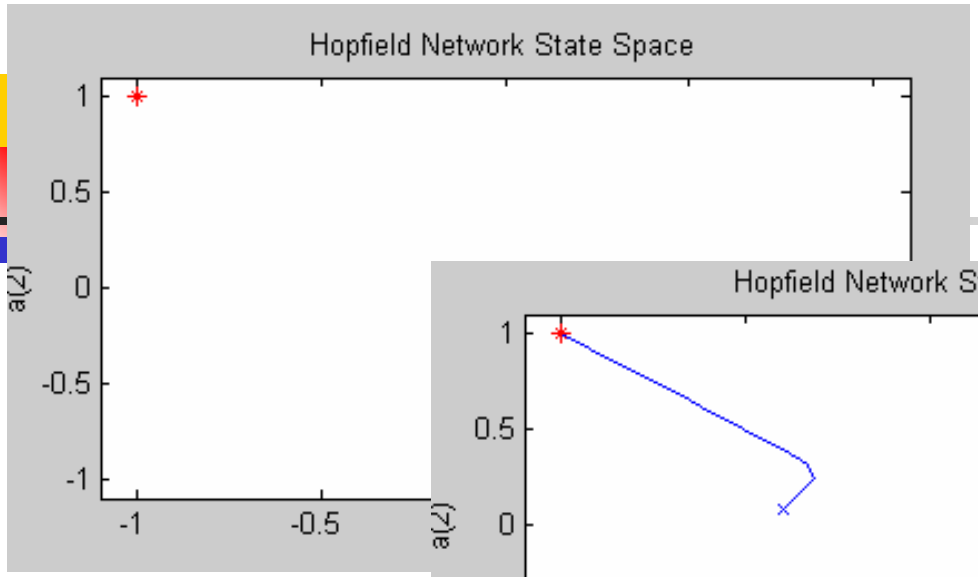
[Y,Pf,Af,E,perf]=sim(net,{Q Ts},Pi,Ai,T)

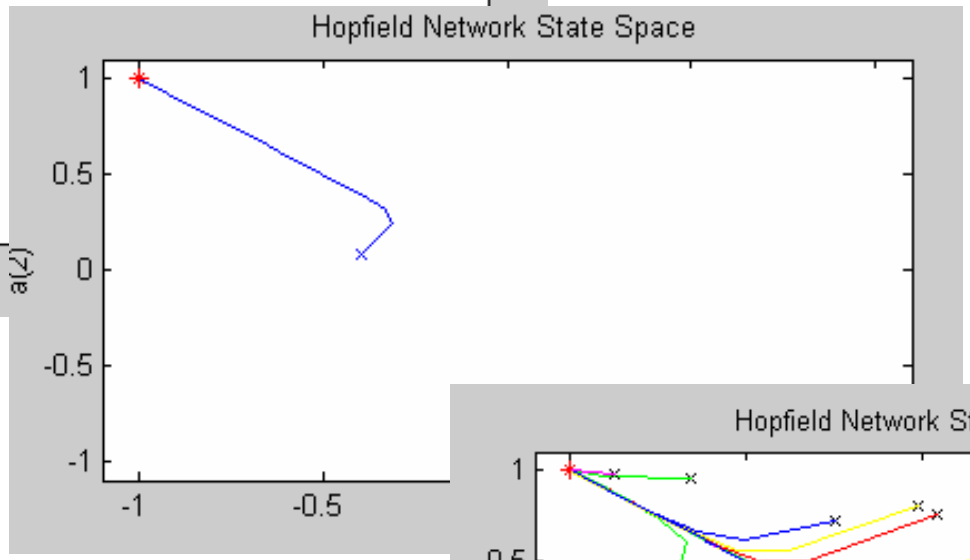
记忆样本　　$T=[1\ \ -1;\ -1\ \ 1]$

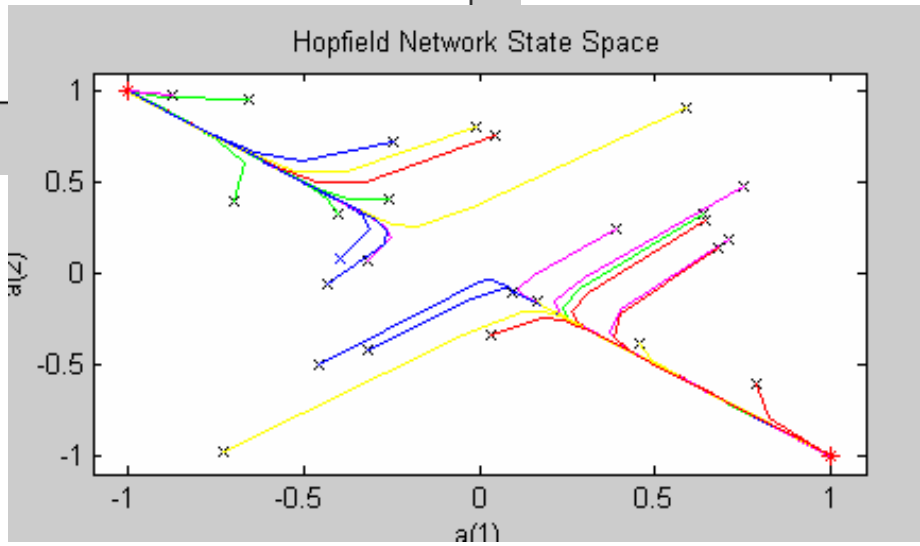
随机产生矢量　P=randn(2,1)

随机产生25矢量

```
T=[1,-1;-1 1];
plot(T(1,:),T(2,:),'*');
axis([-1.1 1.1 -1.1 1.1]);
net = newhop(T);
for i=1:25
   a={randn(2,1)};
   [Y,Pf,Af] = sim(net,{1 20},{ },a);
   record=[cell2mat(a) cell2mat(Y)];
  start=cell2mat(a);
  hold on
 plot(start(1,1),start(2,1),'+',record(1,:),record(2,:),‘o-');
end
```


Hopfield Network State Space
1
0.5
0
-0.5
-1
a(2)
-1
-0.5


Hopfield Network State Space
1
0.5
0
-0.5
-1
a(2)
-1
-0.5


Hopfield Network State Space
1
0.5
0
-0.5
-1
a(2)
-1
-0.5
0
0.5
1
a(1)

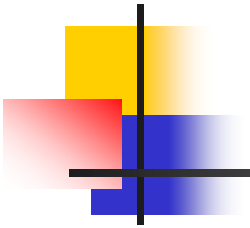
6.6竞争神经网络

竞争神经网络：更接近于人脑工作特性

特点：竞争层，

竞争层神经元相互竞争以确定胜者。

思路：竞争单元争相响应输入模式，胜者表示输入模式的所属类别。



无导师学习：

学习时只需给定一个输入模式集作为训练集，网络自组织训练模式，并将其分成不同类型。

竞争层：许多网络的重要组成部分。

Hamming, SOM，LVQ，CPN，ART

自组织特征映射（SOFM）模型： Kohonen提出。

- 网络模型

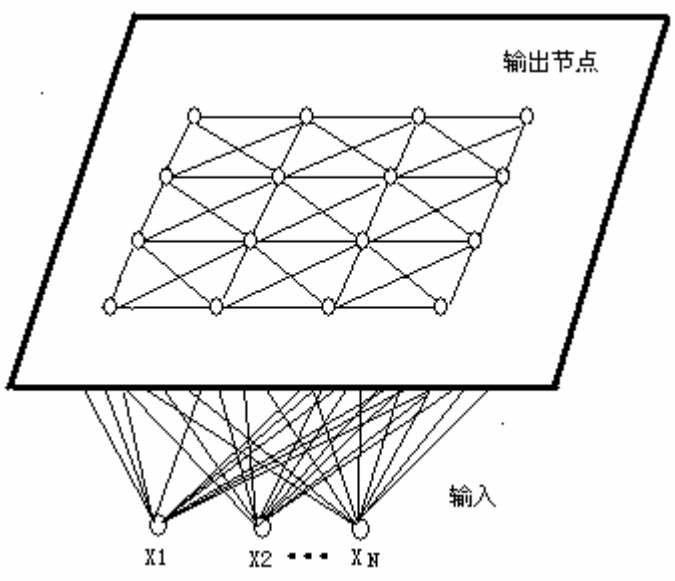
第一层：输入层

由接收输入模式的处理单元构成。

第二层：竞争层

竞争单元争相响应输入模式，胜者表示输入模式的所属类别。

输入单元到竞争单元的连接为全互连的。



- 学习算法：竞争、合作和更新

1）竞争：计算神经元的输入总和

$$u_i = \sum_j w_{ij} x_j = W_i \cdot X$$

竞争原则：具有最高输入总和的单元获胜，

2）合作过程：

确定获胜神经元的加强中心（邻域）。

3）更新过程（修正）：

只修正获胜神经元及邻域的权值。

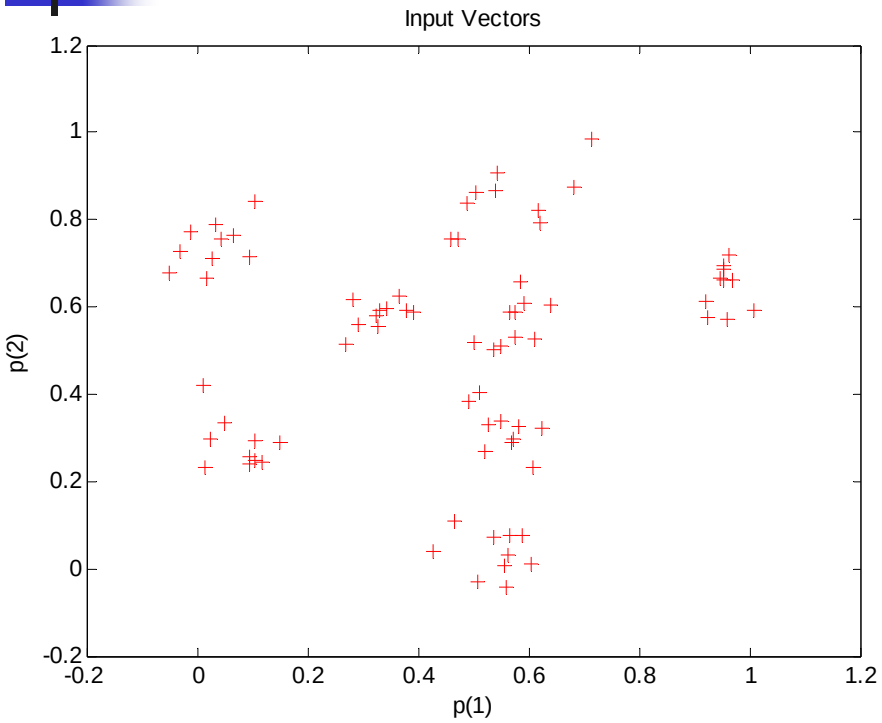
$$W_j(n+1) = W_j(n) + \eta(n)(X - W_j(n)) \qquad n = 0,1,2,\cdots,N$$

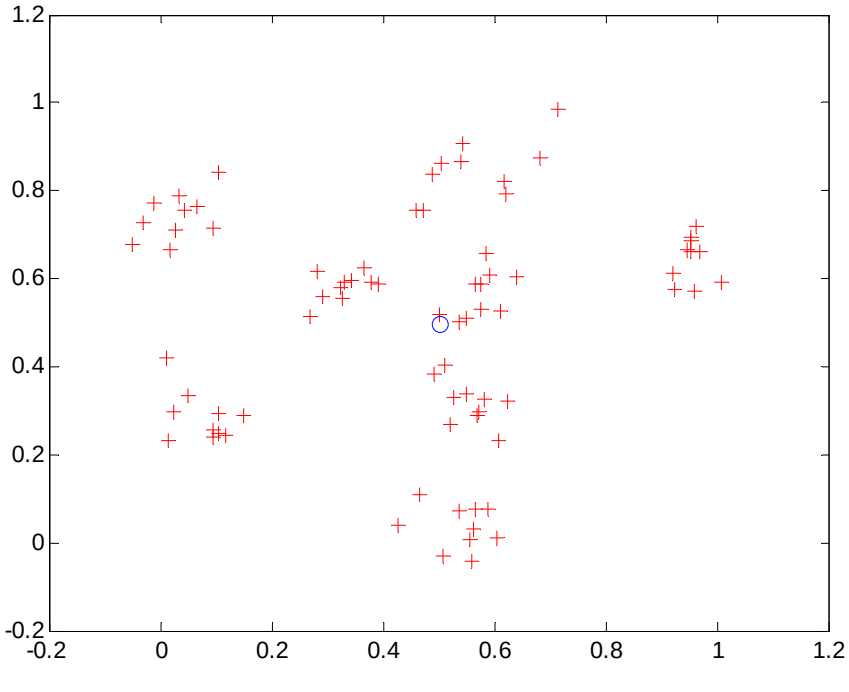
获胜单元的权值越来越接近于输入模式。

例：竞争网络的聚类实验

随机产生呈聚类分布的测试数据**80**个，用竞争网络分类

```
X=[0,1;0,1]; clusters=8;
points=10; std_dev=0.05
P = nngenc(X,clusters,points,std_dev);
Plot(P(1,:),P(2,:),'+r');
net = newc([0 1;0 1],8,.1);
net.trainParam.epochs = 10;
net = train(net,P);
```

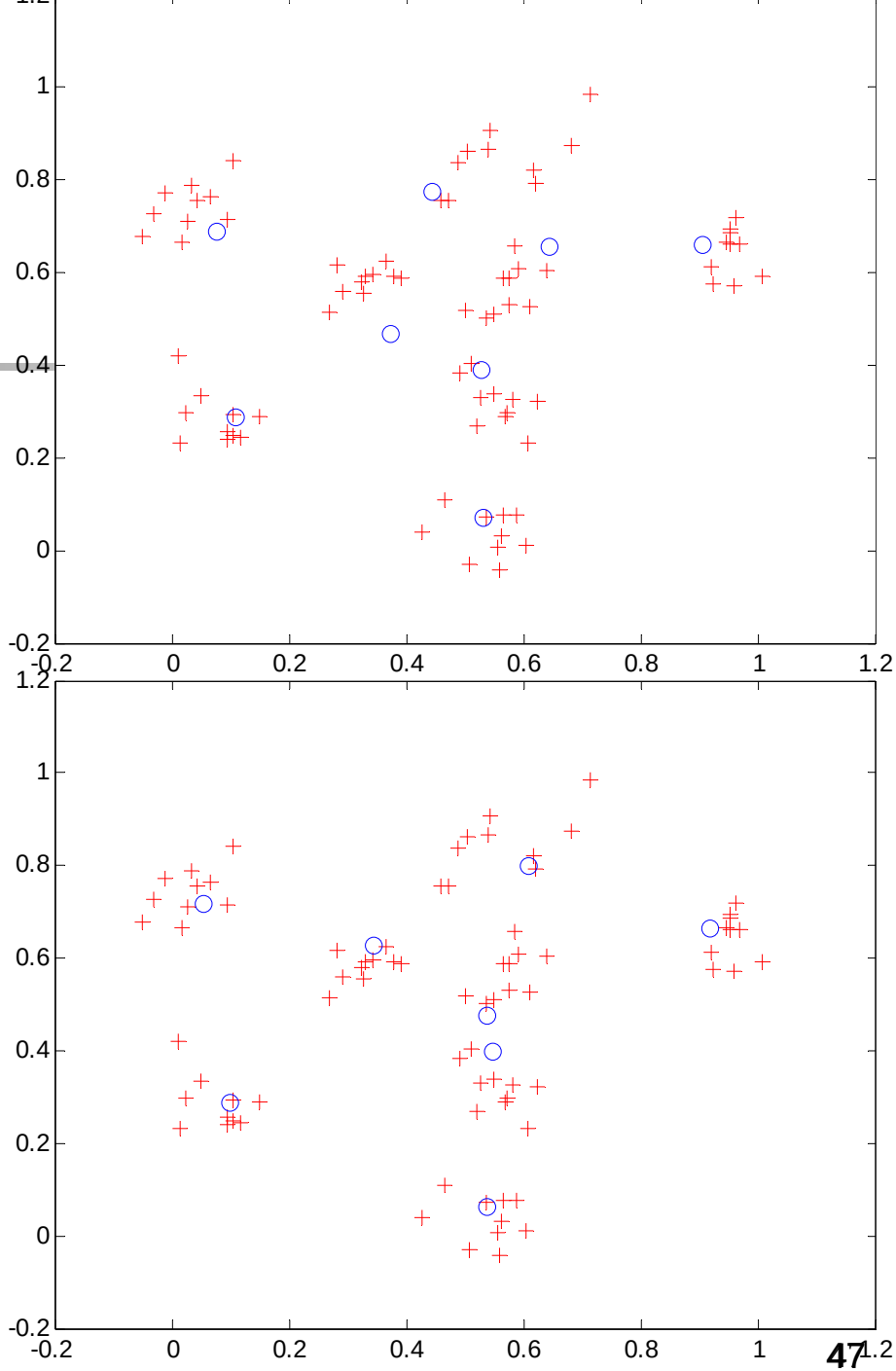

Input Vectors
p(2)
p(1)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.2
0
0.2
0.4
0.6
0.8
1
1.2


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.2
0
0.2
0.4
0.6
0.8
1
1.2

训练3次
训练结束（10次）

测试:

```
p = [0.1; 0.2];
a = sim(net,p)
p = [0.9; 0.8];
a = sim(net,p)
p = [0.5; 0.5];
a = sim(net,p)
a =
   (1,1)        1
a =
   (2,1)        1
a =
   (5,1)        1
```



6.7深度神经网络和深度学习（DL)

1 概述

2 深度学习思路

3 深度学习模型或方法

4 应用

1 概述

深度神经网络：具有多个隐层，五层至10层，甚至更多的神经网络。与传统的前向神经网络具有相似的结构。

- 深度学习的概念源于**ANN**的研究及大脑研究

  NN使用中存在问题：局部极小，梯度消失，特征提取等

- 机器学习领域泰斗**Hinton**等人于**2006**年在《科学》发表论文提出

两个主要观点：

1）多隐层的人工神经网络具有优异的特征学习能力，学习得到的特征对数据有更本质的刻画，从而有利于可视化或分类；

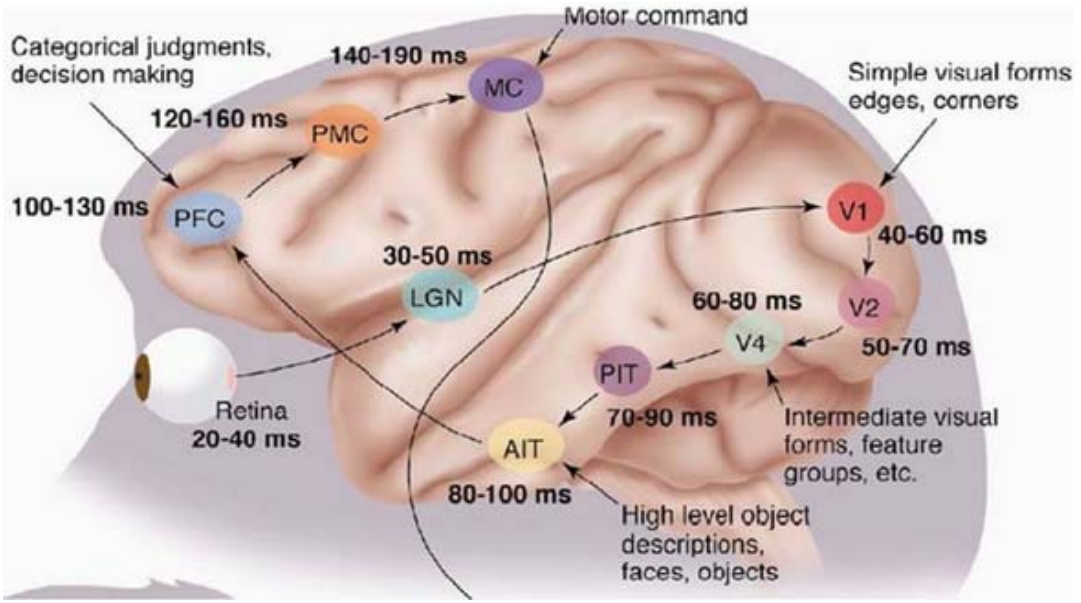
2）深度神经网络在训练上的难度，可以通过“逐层初始化”（layer-wise pre-training）来有效克服，在这篇文章中，逐层初始化是通过无监督学习实现的。

2　深度学习思路

人脑视觉机理

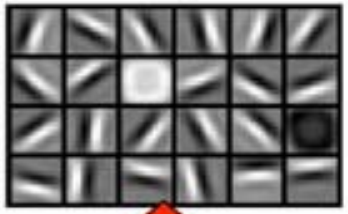
1981 年的诺贝尔医学奖，颁发给了 David Hubel 和TorstenWiesel，以及 Roger Sperry。

主要贡献：发现了
视觉系统的信息处理
是分级的，即
可视皮层是分级的。



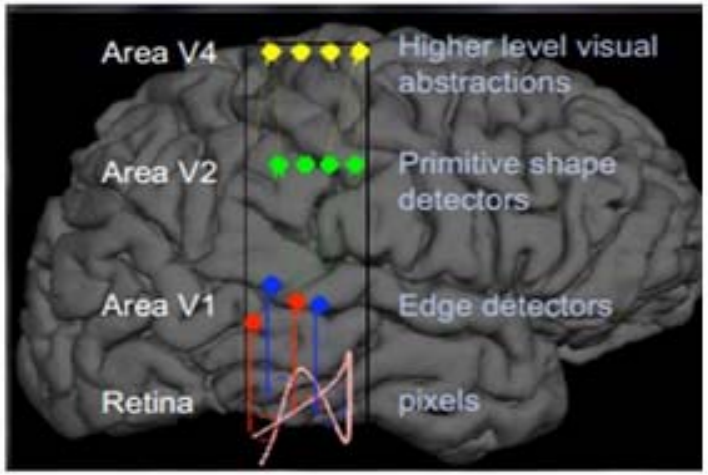
object models

object parts (combination of edges)

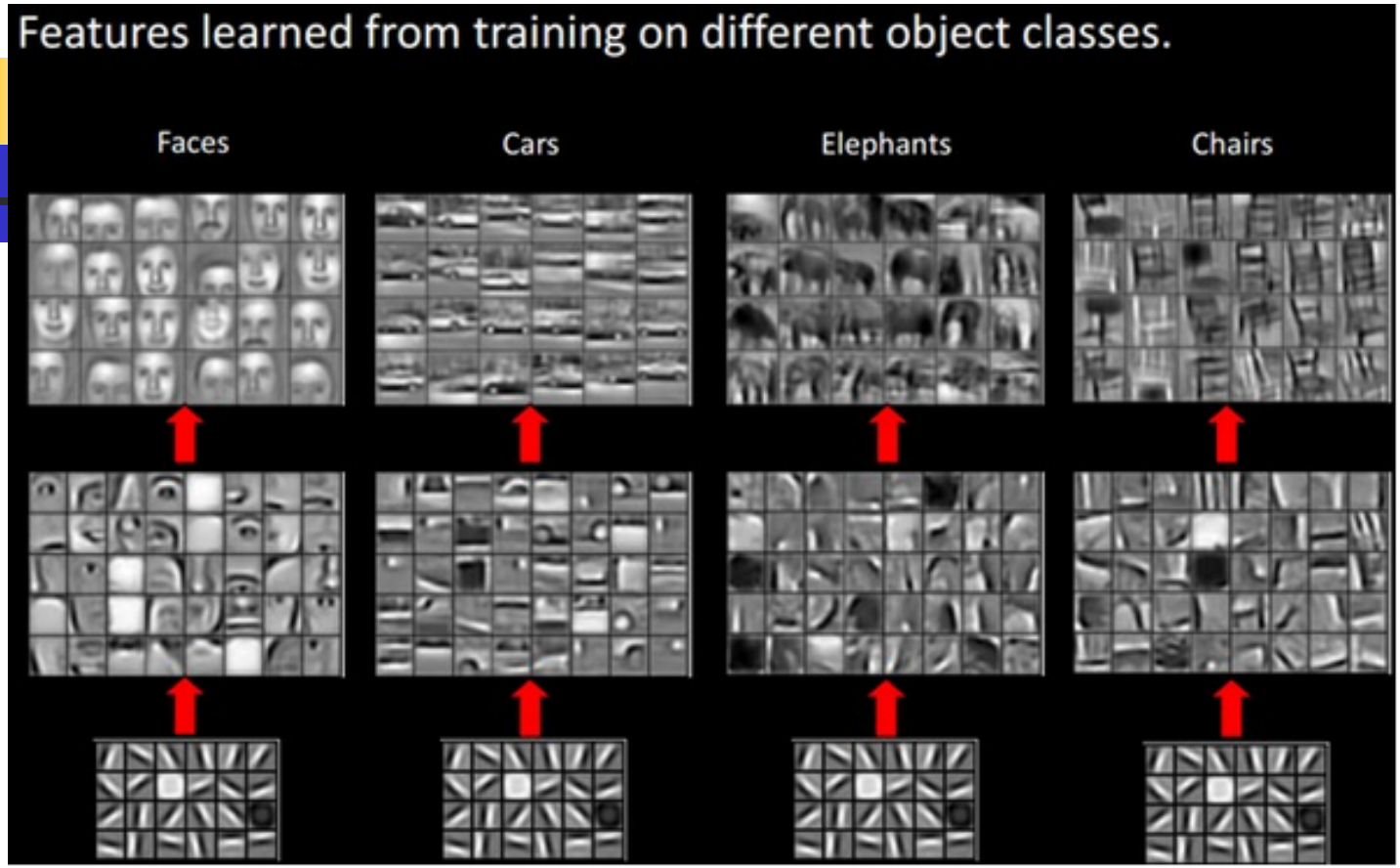


edges

pixels



关键词有两个，一个是抽象，一个是迭代。从原始信号，做低级抽象，逐渐向高级抽象迭代。人类的逻辑思维，经常使用高度抽象的概念。


Features learned from training on different object classes.
Faces
Cars
Elephants
Chairs

高层特征更容易识别

Deep Learning的基本思想

- 假设一个系统S，有n层（S1,...Sn），输入是I，输出是0，

即 I =>S1=>S2=>.....=>Sn => 0，

如果输出0等于输入I，即输入I经过这个系统变化之后没有任何的信息损失，这意味着输入I经过每一层Si都没有任何的信息损失，即在任何一层Si，它都是原有信息（即输入I）的另外一种表示。

Deep Learning，需要自动地学习特征，若有一堆输入I（如一堆图像或者文本），假设设计了一个系统S（有n层），通过调整系统中参数，使得它的输出仍然是输入I，那么就可以自动地获取得到输入I的一系列层次特征，即S1，…，Sn。

- Lecun等人提出的卷积神经网络

deep learning训练思路

hinton等提出了在非监督数据上建立多层神经网络的一个有效方法，分为两步，一是利用无监督学习对每层进行逐层训练，二是最上层采用监督学习进行微调。

Wake-Sleep算法

1）wake阶段：认知过程，通过外界的特征和向上的权重（认知权重）产生每一层的抽象表示（结点状态），并且使用梯度下降法修改层间的权重。

2）sleep阶段：生成过程，通过顶层表示（醒时学得的概念）和向下权重，生成底层的状态，同时修改层间向上的权重。

deep learning训练过程

1）使用自下向上非监督学习

采用无标定数据（有标定数据也可）分层训练各层参数，这一步可以看作是一个无监督训练过程，是和传统神经网络区别最大的部分（这个过程可以看作是feature learning过程）：

具体的，先用无标定数据训练第一层，训练时先学习第一层的参数（这一层可以看作是得到一个使得输出和输入差别最小的三层神经网络的隐层），由于模型capacity的限制以及稀疏性约束，使得得到的模型能够学习到数据本身的结构，从而得到比输入更具有表示能力的特征；在学习得到第n-1层后，将n-1层的输出作为第n层的输入，训练第n层，由此分别得到各层的参数；

2）自顶向下的监督学习

通过带标签的数据去训练，误差自顶向下传输，对网络进行微调

基于第一步得到的各层参数进一步fine-tune整个多层模型的参数，这一步是一个有监督训练过程;

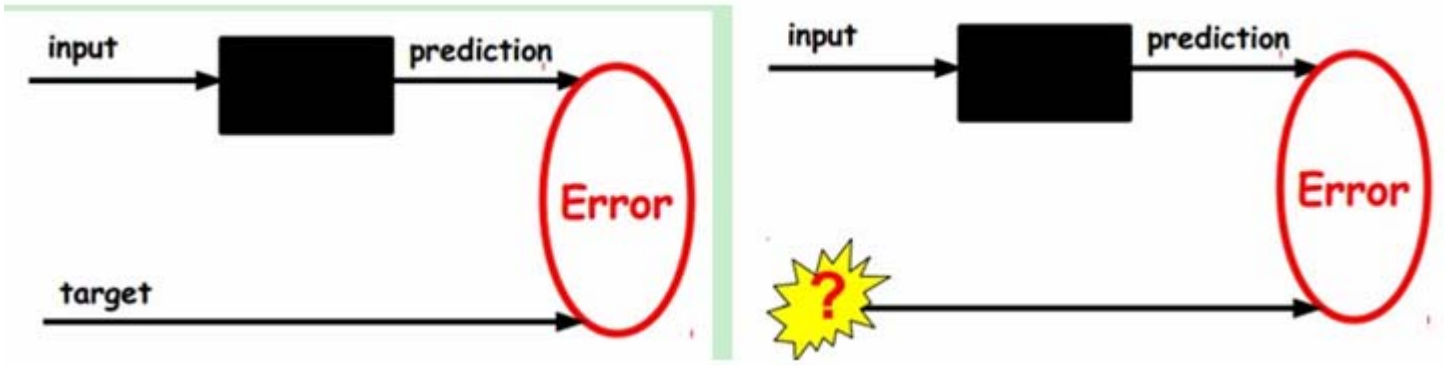
第一步类似神经网络的随机初始化初值过程，由于DL的第一步不是随机初始化，而是通过学习输入数据的结构得到的，因而这个初值更接近全局最优，从而能够取得更好的效果；所以DL效果好很大程度上归功于第一步的feature

3 深度学习模型或方法

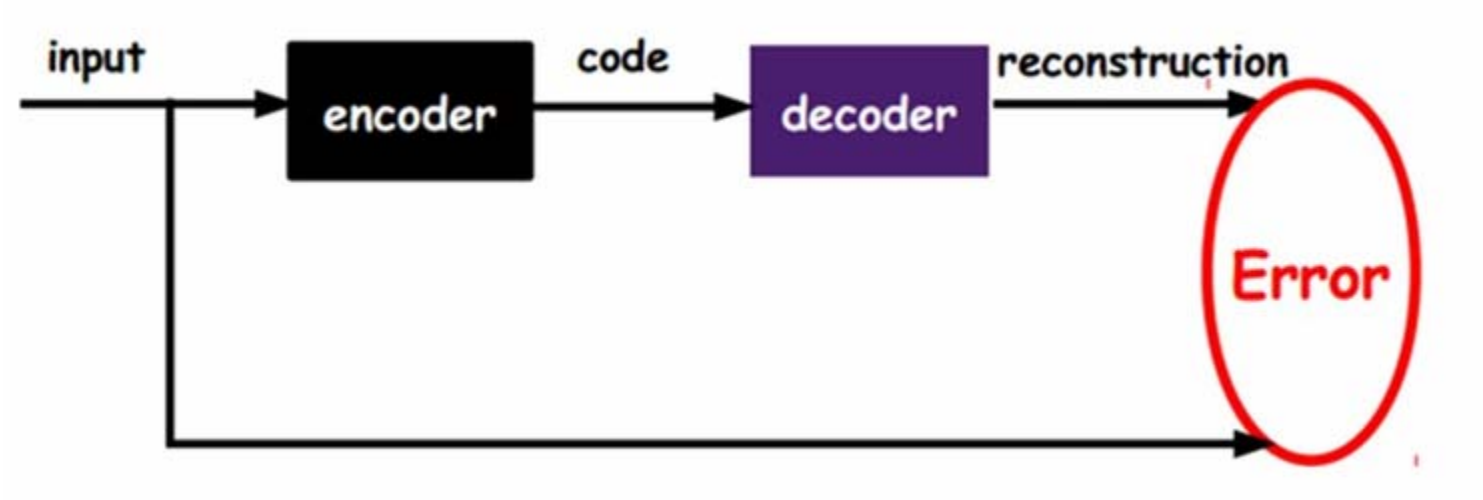
AutoEncoder自动编码器

假设其输出与输入是相同的，然后训练调整其参数，得到每一层中的权重

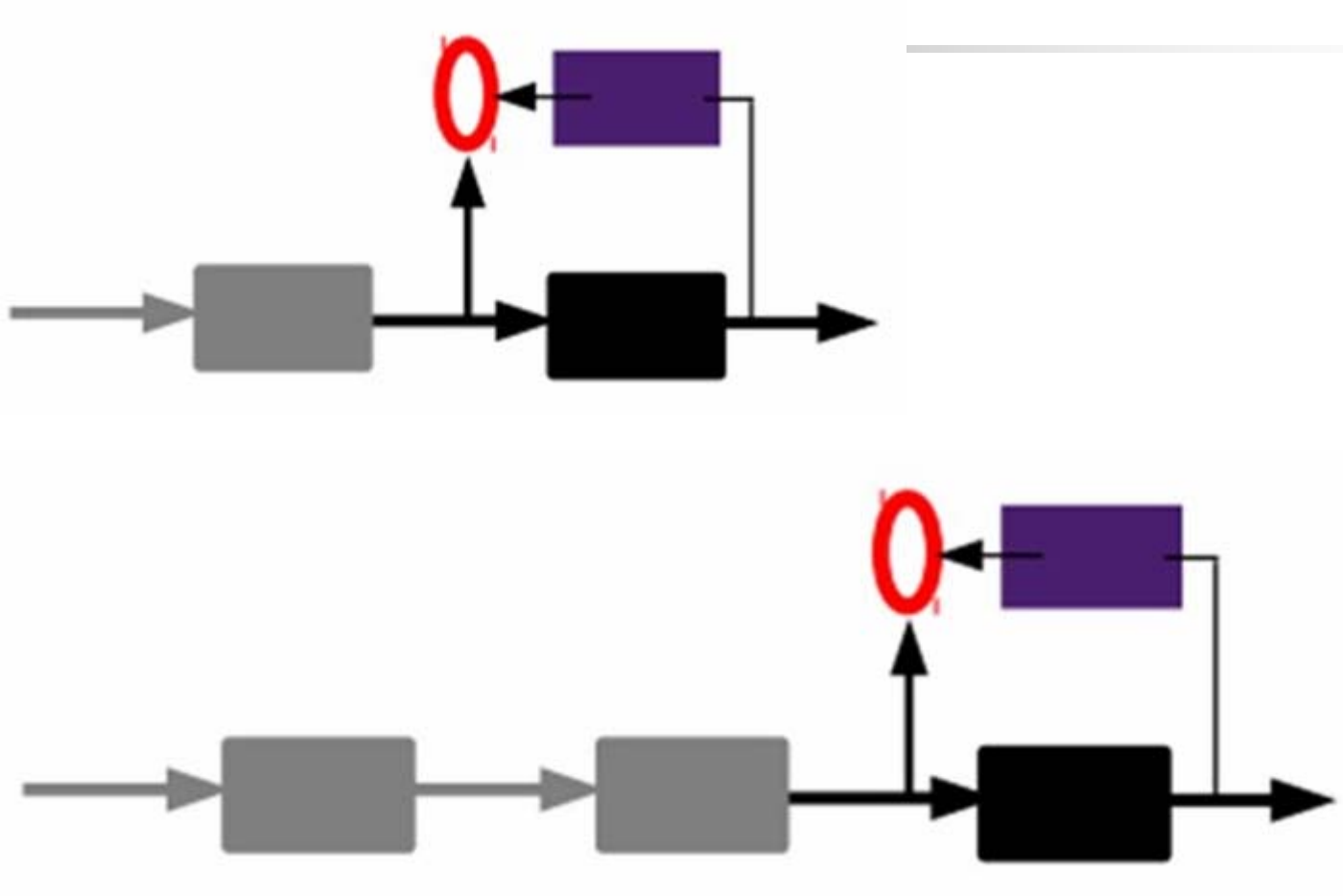
1）给定无标签数据，用非监督学习学习特征：



code 是否是input表示呢？加一个decoder解码器，decoder就会输出一个信息，那么如果输出的这个信息和一开始的输入信号input是很像的（理想情况下就是一样的）。通过调整encoder和decoder的参数，使得重构误差最小，这时候就得到了输入input信号的第一个表示了

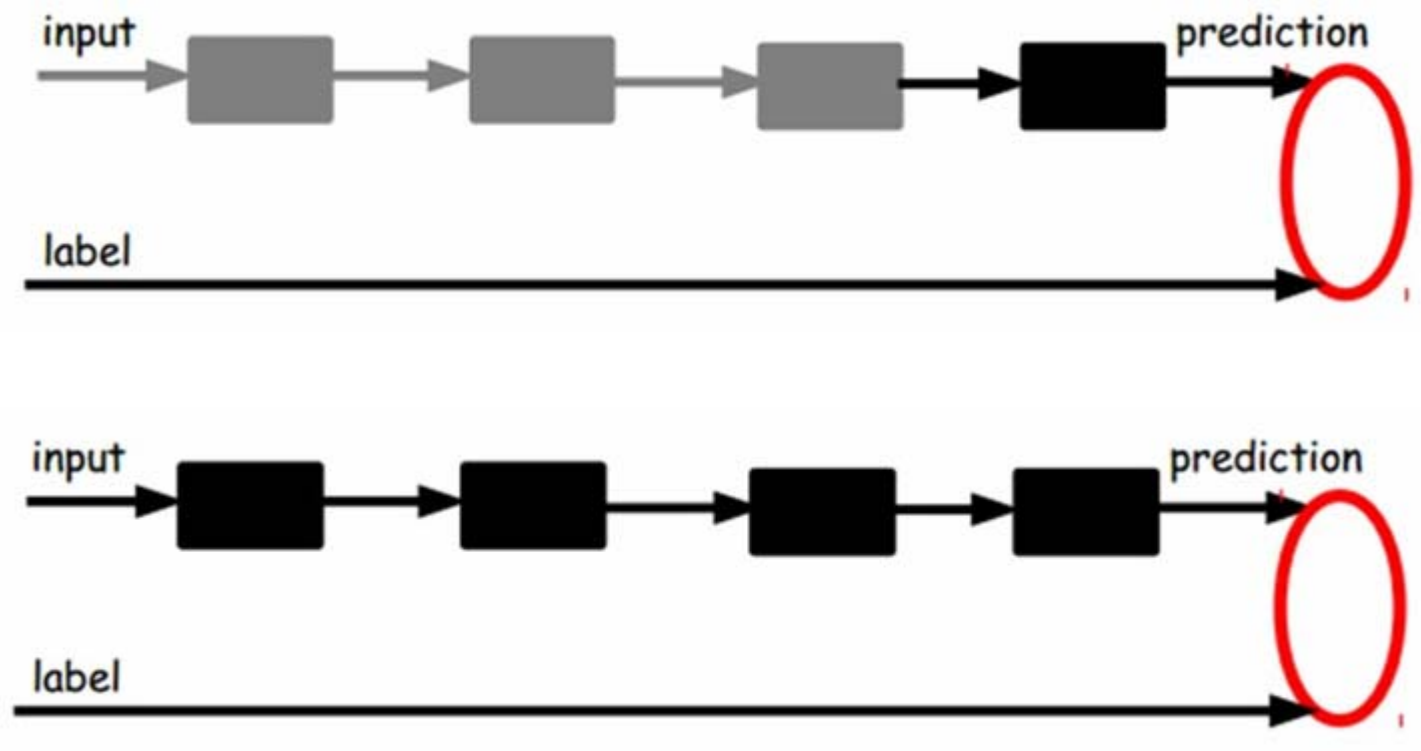


2）通过编码器产生特征，训练下一层，逐层训练



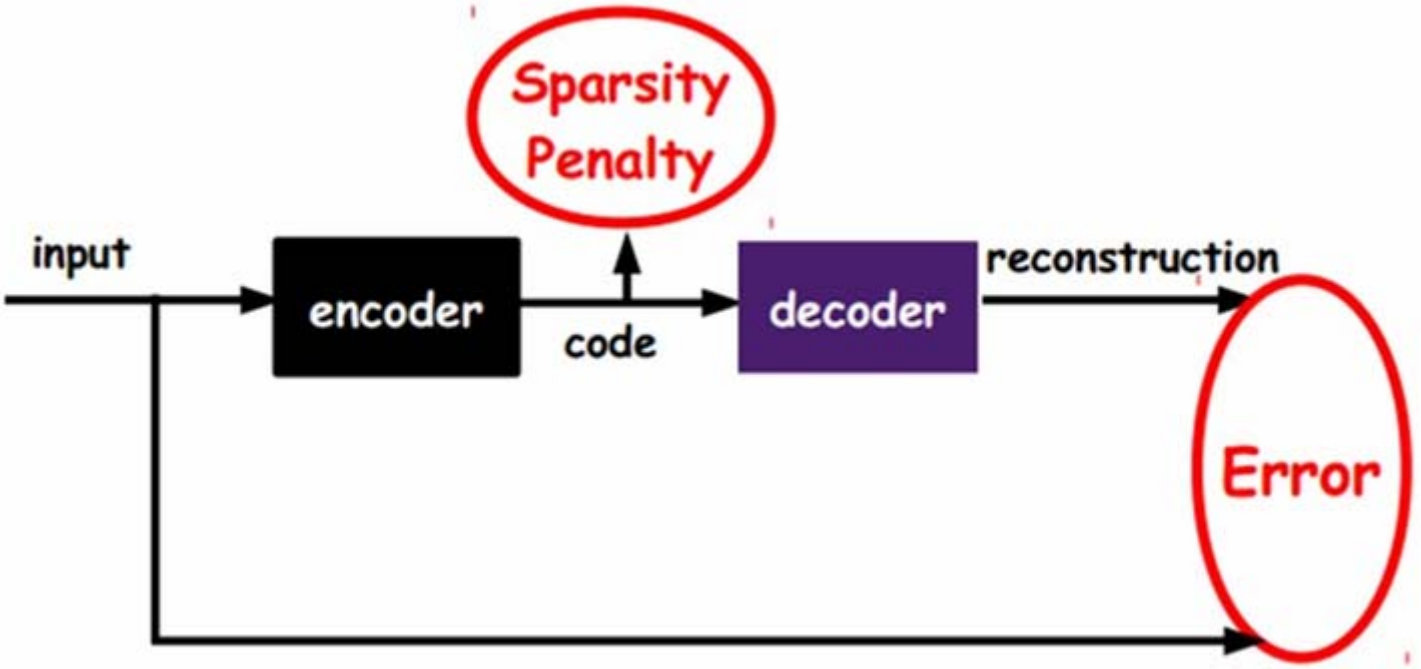
3）有监督微调

实现分类，在AutoEncoder的最顶的编码层添加一个分类器，通过标准的多层神经网络的监督训练方法（梯度下降法）去训练。



Sparse AutoEncoder稀疏自动编码器

增加约束条件得到新的Deep Learning方法 ， L1范数



- input: X code: $h = W^T X$

- loss: $L(X;W) = \|Wh - X\|^2 + \lambda \sum_j |h_j|$

Sparse Coding稀疏编码

- 输入输出相等的限制放松，利用基的概念，

  O = a1*Φ1 + a2*Φ2+....+ an*Φn，Φi是基，ai是系数，

  优化　　Min |I - O|，其中I表示输入，O表示输出。

- 求得系数ai和基Φi，系数和基是输入的另外一种近似表达
- 加上L1的Regularity限制，得到:
- Min |I - O| + u*(|a1| + |a2| + ... + |an |)
- 稀疏表达或稀疏编码（无监督学习方法 ）

- input: X code: $h = W^T X$

- loss: $L(X;W) = \|Wh - X\|^2 + \lambda \sum_j |h_j|$

- **Sparse coding**过程分为两个部分：
- **1）Training**阶段：

给定一系列的样本[x1，x 2，...]，学习得到一组基[Φ1，Φ2，...]，即字典。训练过程就是一个重复迭代的过程

$$\min_{a,\phi} \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{k} a_{i,j}\phi_j \right\|^2 + \lambda \sum_{i=1}^{m} \sum_{j=1}^{k} |a_{i,j}|$$

a）固定字典Φ[k]，然后调整a[k]，使得上式，即目标函数最小

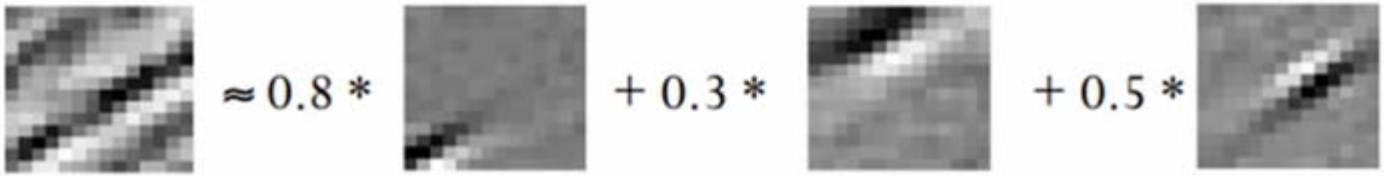
b）然后固定a [k]，调整Φ [k]，使得上式，即目标函数最小

2）Coding阶段：

给定一个新的图片x，由上面得到的字典，得到稀疏向量a。这个稀疏向量就是这个输入向量x的一个稀疏表达

$$\min_{a} \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{k} a_{i,j}\phi_j \right\|^2 + \lambda \sum_{i=1}^{m} \sum_{j=1}^{k} |a_{i,j}|$$

例如：

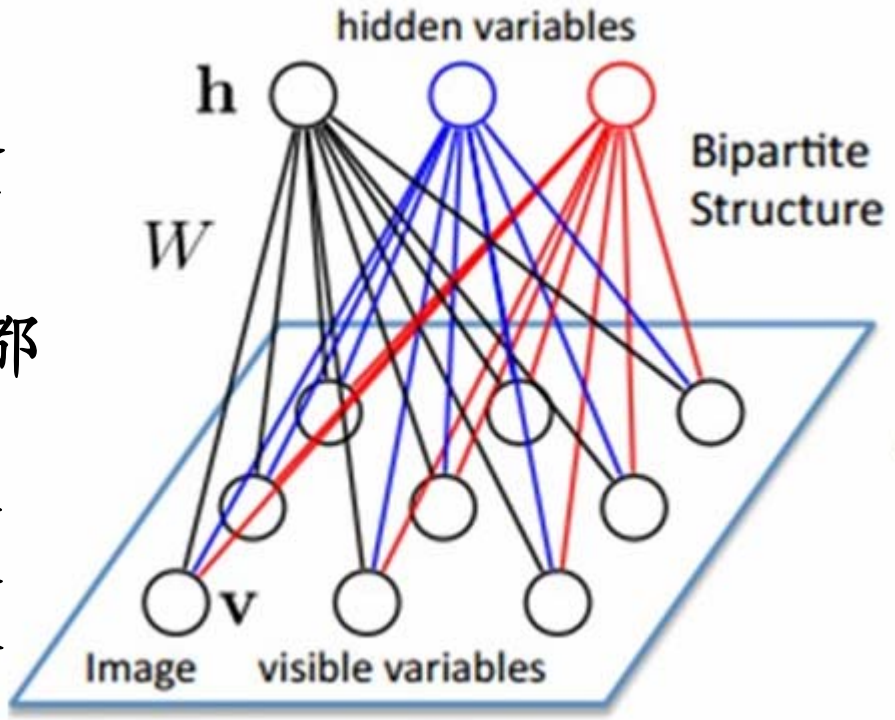


Represent x_i as: a_i = [0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, …]

Restricted Boltzmann Machine (RBM)

- 概率神经网络

  每一层的节点之间没有链接，一层是可视层，即输入数据层（v），一层是隐藏层(h)，若所有的节点都是随机二值变量节点（只能取0或者1值），同时假设全概率分布p(v,h)满足Boltzmann 分布，称受限玻尔兹曼机 (RBM)。



双向的、层内无连接、 v和h满足Boltzmann 分布。

已知v的情况下，所有的隐藏节点之间是条件独立的即p(h|v)=p(h1|v)…p(hn|v)。同理，在已知隐藏层h的情况下，所有的可视节点都是条件独立的。

当输入v的时候，通过p(h|v) 可以得到隐藏层h，而得到隐藏层h之后，通过p(v|h)又能得到可视层，

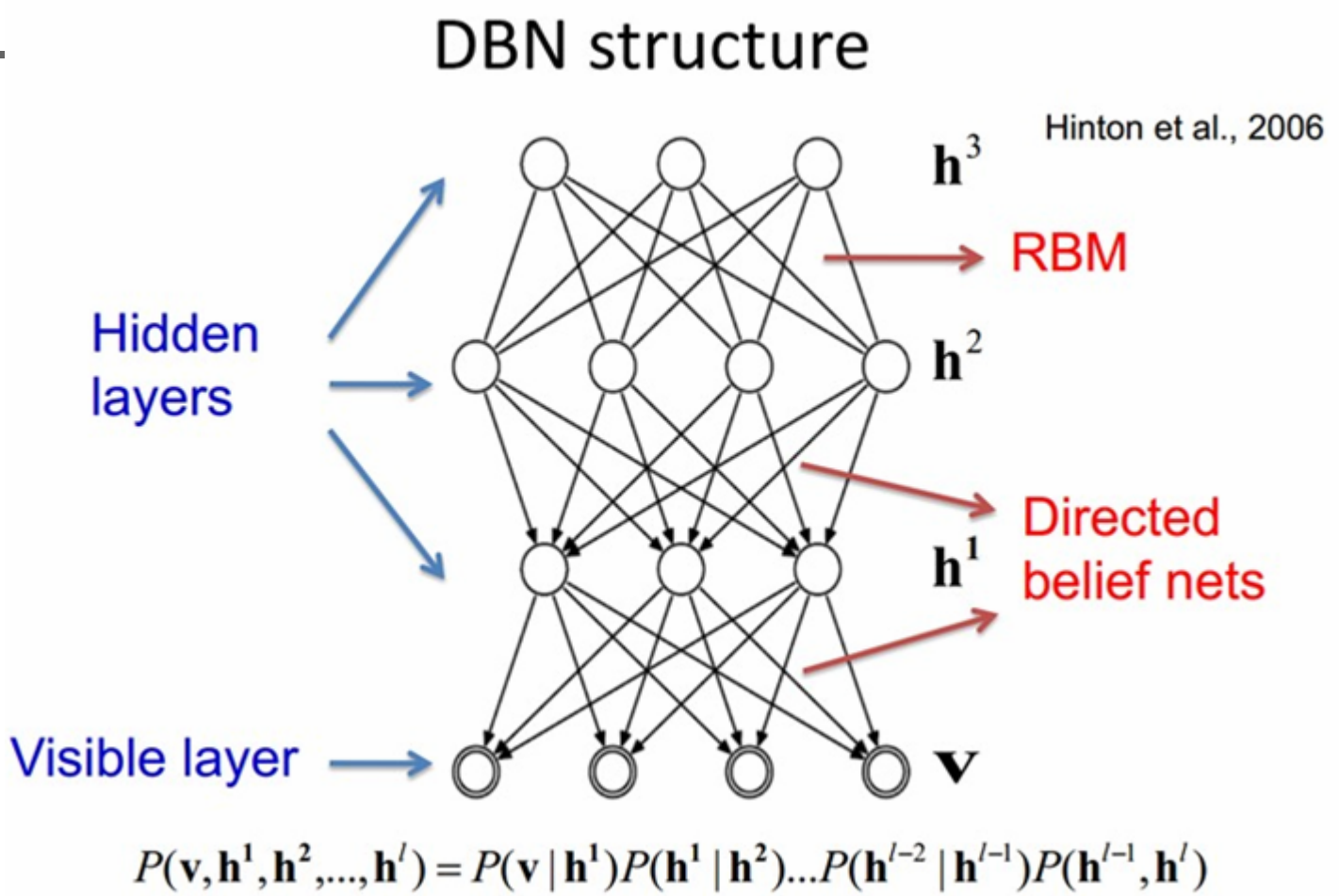
通过调整参数，使得从隐藏层得到的可视层v1与原来的可视层v靠近

若一样那么得到的隐藏层就是可视层另外一种表达，因此隐藏层可以作为可视层输入数据的特征

增加网络的深度：DBM

Deep Belief Networks深信度网络

- 靠近可视层的部分使用贝叶斯信念网络（即有向图模型，且依然限制层中节点之间没有链接）
- 远离可视层的部分使用RBM



Networks（CNN Convolutional Neural ）

1）概述

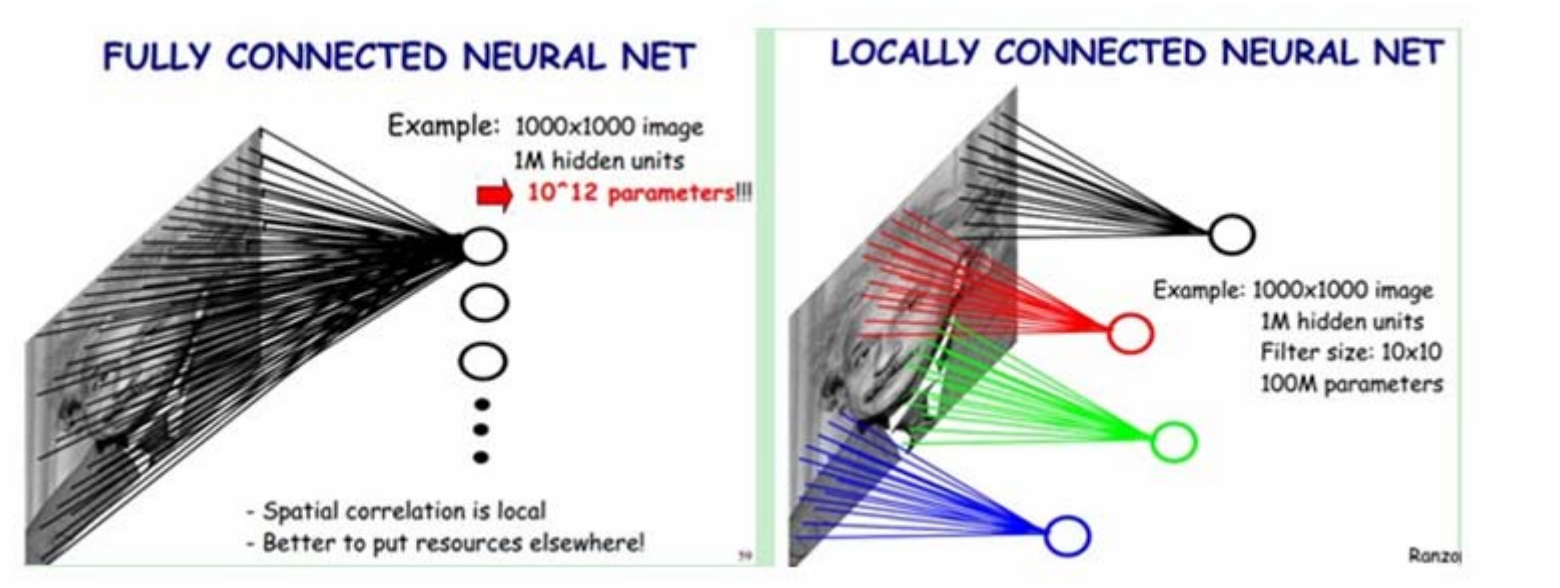
卷积神经网络是一种人工神经网络；计算机视觉广泛采用

权值共享网络结构使之更类似于生物神经网络，降低了网络模型的复杂度，减少了权值的数量。该优点在网络的输入是多维图像时表现的更为明显，使图像可以直接作为网络的输入，避免了传统识别算法中复杂的特征提取和数据重建过程。

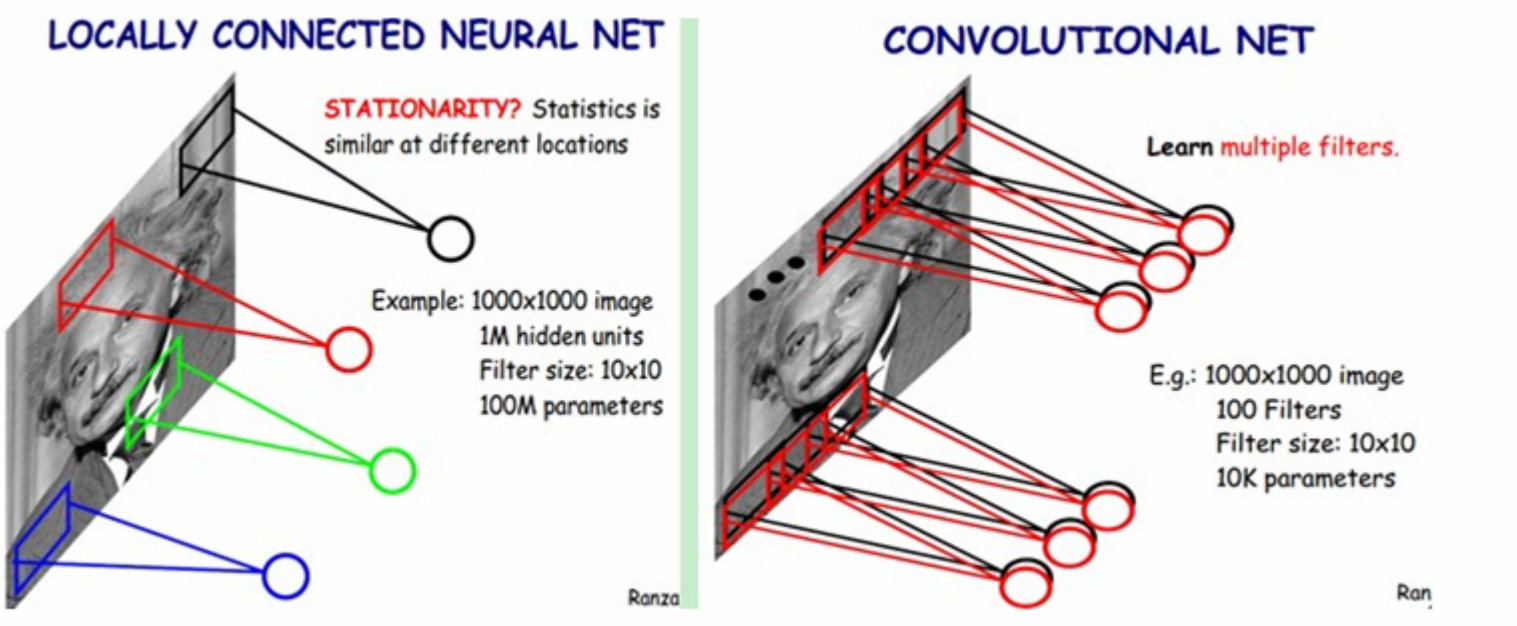
卷积网络是为识别二维形状而特殊设计的一个多层感知器，这种网络结构对平移、比例缩放、倾斜或者其他形式的变形具有高度不变性。

2）卷积神经网络的网络结构

多层的神经网络，每层由多个二维平面组成，而每个平面由多个独立神经元组成。局部连接、权值共享、池化



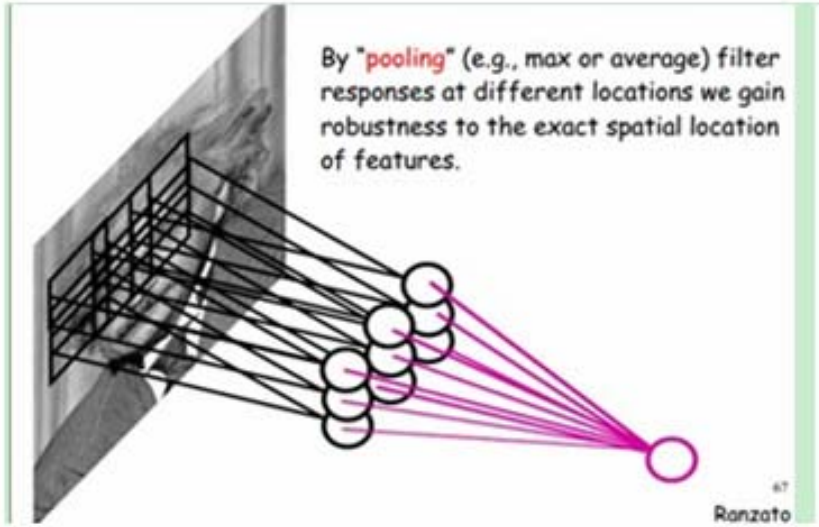
只提取了一种特征？多加几种滤波器

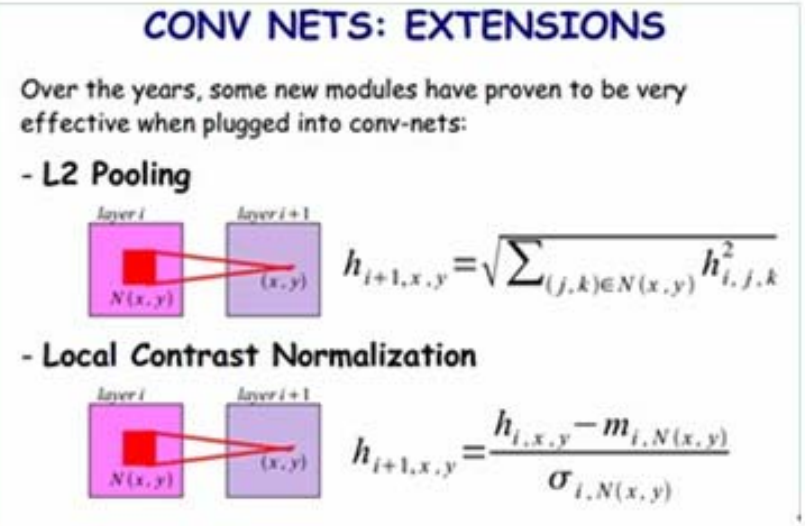


隐层的参数个数和隐层的神经元个数无关，只和滤波器的大小和滤波器种类的多少有关

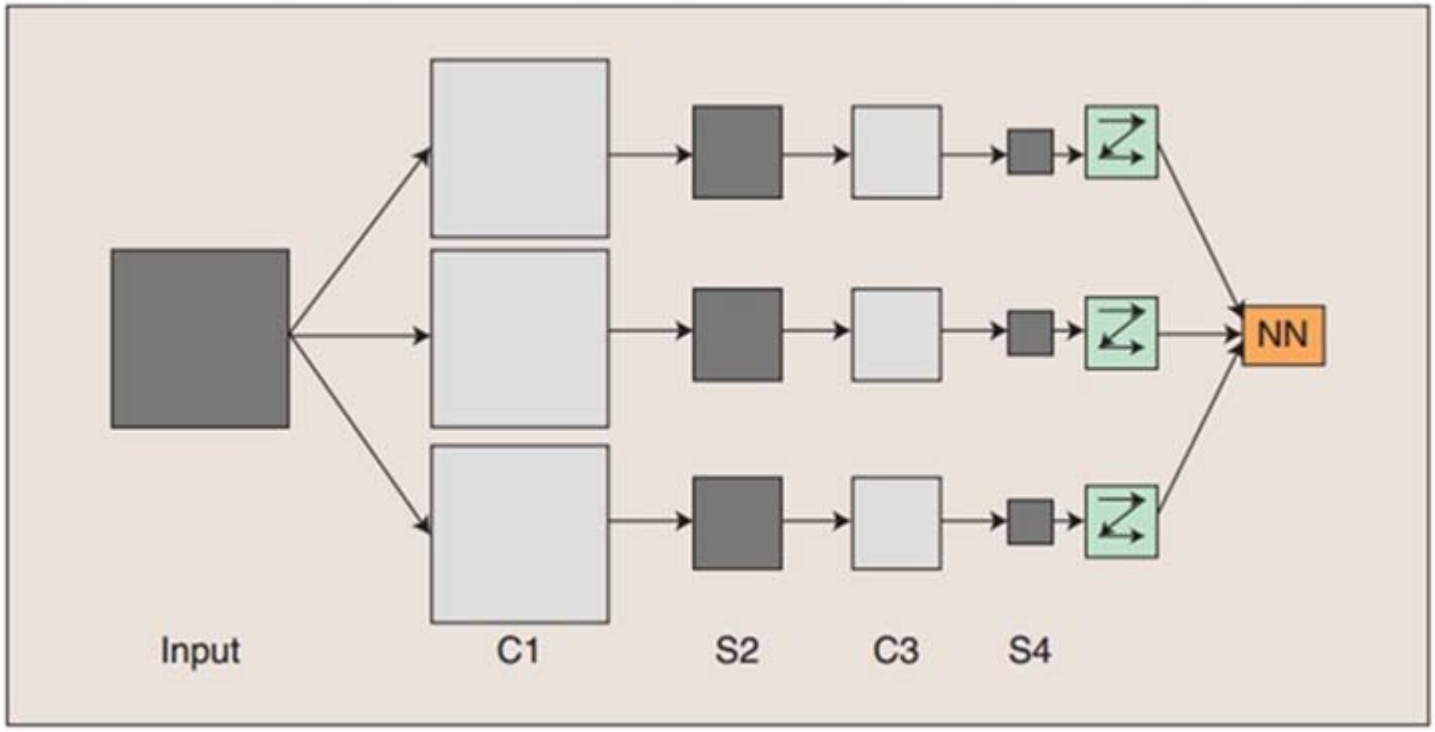
池化：下采样层过程

利用图像局部相关性的原理，对图像进行子抽样，可以减少数据处理量同时保留有用信息





CNN结构框图



三个滤波器和偏置进行卷积，C1层产生三个特征映射图；

特征映射图中每组的四个像素再进行求和，加权值，加偏置，通过Sigmoid或ReLU得到三个S2层的特征映射图；

3）训练过程 有导师训练

第一阶段，向前传播阶段:

a）从样本集中取一个样本(X,Yp)，将X输入网络;

b）计算相应的实际输出Op。

第二阶段，向后传播阶段

a）算实际输出Op与相应的理想输出Yp的差;

b）按极小化误差的方法反向传播调整权矩阵。

4）卷积神经网络的优点

a）特征提取和模式分类同时进行，直接输入图像;

b）输入图像的局部特性，且适应性更强;

c）权重共享可以减少网络的训练参数，且便于并行实现。

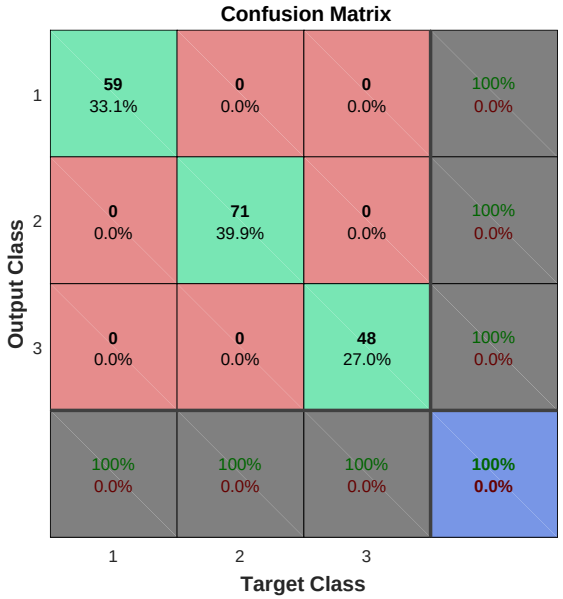
4 实现

下载：Deep Learn Toolbox

matlab R2016b,nn toolbox：DL (CNN,Autoencoders)

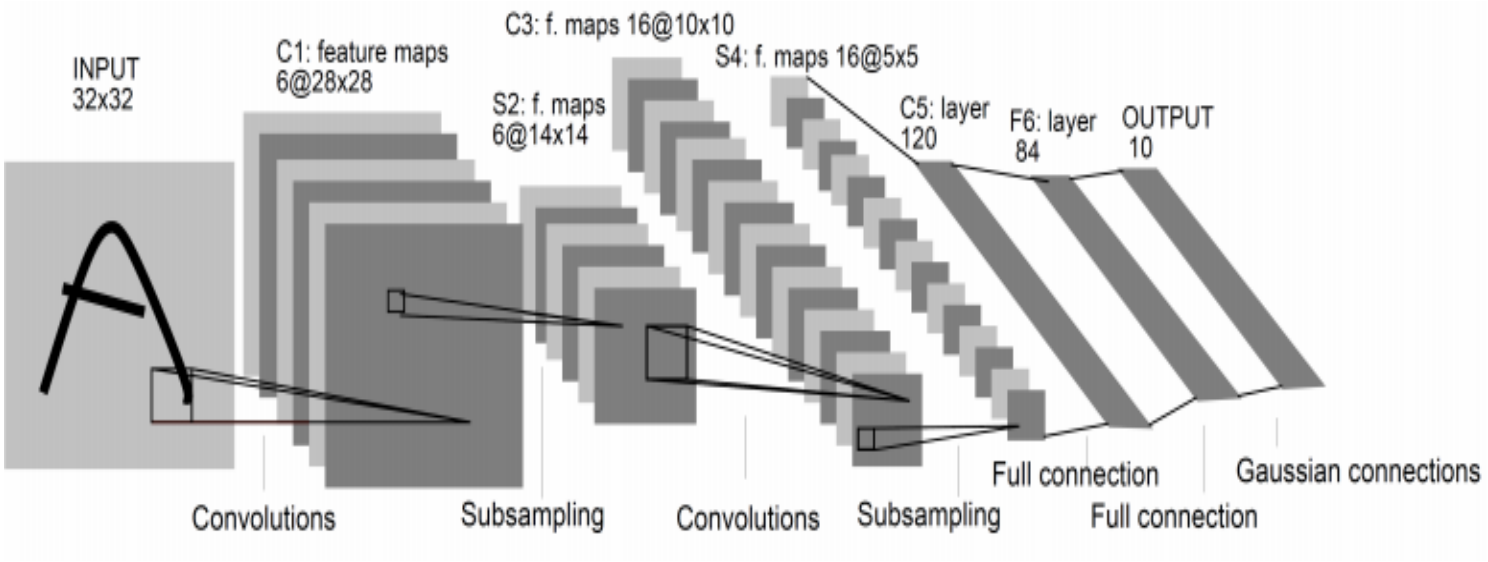
(1)利用自编码提取特征

```
[X,T] = wine_dataset;
hiddenSize = 10;
autoenc1 = trainAutoencoder(X,hiddenSize,...
    'L2WeightRegularization',0.001,...
    'SparsityRegularization',4,...
    'SparsityProportion',0.05,...
    'DecoderTransferFunction','purelin');
features1 = encode(autoenc1,X);
hiddenSize = 10;
autoenc2 = trainAutoencoder(features1,hiddenSize,...
    'L2WeightRegularization',0.001,...
    'SparsityRegularization',4,...
    'SparsityProportion',0.05,...
    'DecoderTransferFunction','purelin',...
    'ScaleData',false);
features2 = encode(autoenc2,features1);
softnet =
trainSoftmaxLayer(features2,T,'LossFunction','crossentropy');
deepnet = stack(autoenc1,autoenc2,softnet);
deepnet = train(deepnet,X,T);
wine_type = deepnet(X);
plotconfusion(T,wine_type);
```



(2)CNN在图像处理中的应用

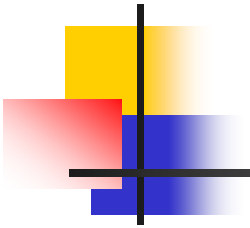
LeNet-5 虽然提出时间比较早，但是是一个非常成功的CNN神经网络模型。基于 LeNet-5 的手写数字识别系统在 90 年代被美国很多银行使用，用来识别支票上面的手写数字。LeNet-5 共有 7 层。



LeNet-5 网络结构

- 输入层：输入图像大小为 32 × 32 = 1024。
- C1 层：这一层是卷积层。滤波器的大小是 5×5 = 25，共有 6 个滤波器。得到 6 组大小为 28 × 28 = 784 的特征映射。因此，C1 层的神经元个数为 6 × 784 = 4,704。可训练参数个数为6 × 25 + 6 = 156。连接数为 156 × 784 = 122,304

- S2 层：这一层为子采样层。由 C1 层每组特征映射中的 2×2 邻域点次采样为 1 个点，也就是 4 个数的平均。这一层的神经元个数为14× 14 = 196。可训练参数个数为 6×（1 + 1）= 12。连接数为6×196×(4 + 1) = 122,304



- C3 层：这一层是卷积层。由于 S2 层也有多组特征映射，需要一个连接表来定义不同层特征映射之间的依赖关系。LeNet-5 的连接表如图所示，共有60个滤波器，大小是5×5=25。
- 得到 16 组大小为 10×10 = 100 的特征映射。C3 层的神经元个数为16×100 = 1,600。可训练参数个数为 60×25+16 = 1,516。连接数为1,516×100 = 151,600。

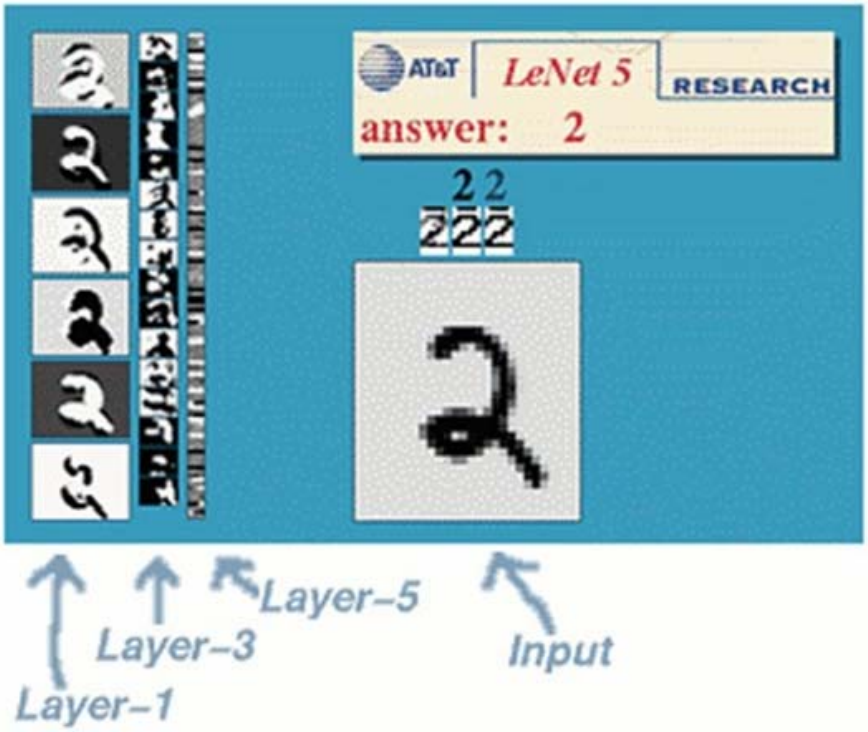
- S4 层：这一层是一个子采样层，由 2×2 邻域点次采样为 1 个点，得到 16 组 5×5 大小的特征映射。可训练参数个数为 16×2 = 32。连接数为 16×(4 +1) = 80。

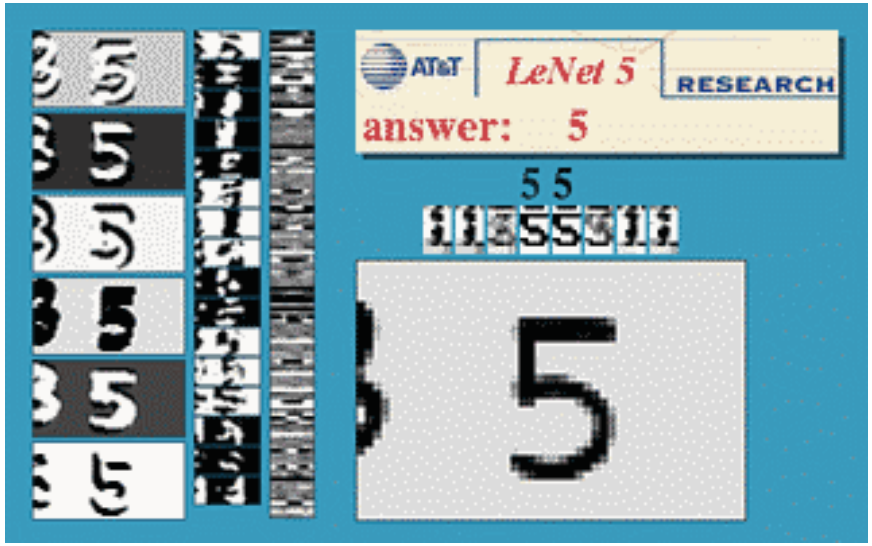
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

LeNet-5 中C3层的连接表

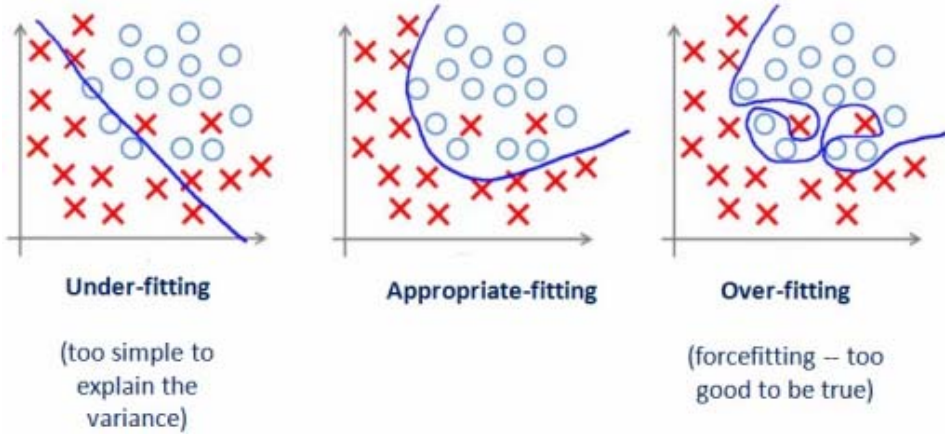
- C3 层的最开始的 6 个特征映射依赖于 S2 层的特征映射的每 3 个连续子集。
- 接下来的 6 个特征映射依赖于 S2 层的特征映射的每 4 个连续子集。
- 再接下来的 3 个特征映射依赖于 S2 层的特征映射的每 4 个不连续子集。
- 最后一个特征映射依赖于 S2 层的所有特征映射。

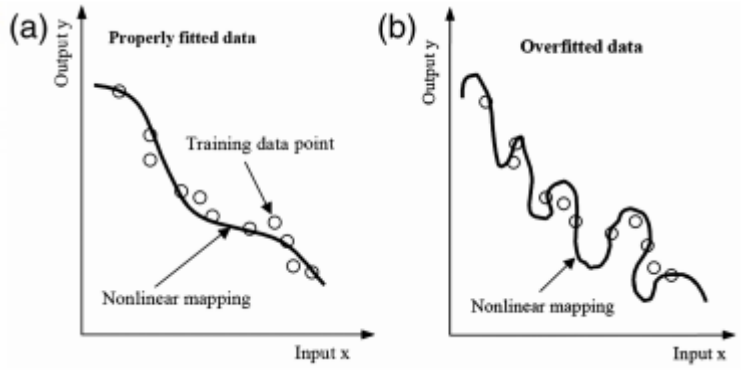
- C5 层：是一个卷积层，得到 120 组大小为 1×1 的特征映射。每个特征映射与 S4 层的全部特征映射相连。有120×16 = 1,920 个滤波器，大小是 5×5 = 25。C5 层的神经元个数为120，可训练参数个数为 1,920×25+120 = 48,120。连接数为120×(16×25+1) = 48,120。
- F6 层：是一个全连接层，有 84 个神经元，可训练参数个数为84×(120+1) = 10,164。连接数和可训练参数个数相同，为10,164。
- 输出层：输出层由 10 个欧氏径向基函数（Radial Basis Function，RBF）函数组成。


AT&T
LeNet 5
RESEARCH
answer: 2
Layer–1
Layer–3
Layer–5
Input


AT&T
LeNet 5
RESEARCH
answer: 5

过拟合 overfitting





深度学习框架（Framework）

Caffe(M)、tensorflow、 torch/pytorch、 mxnet(M)、theano