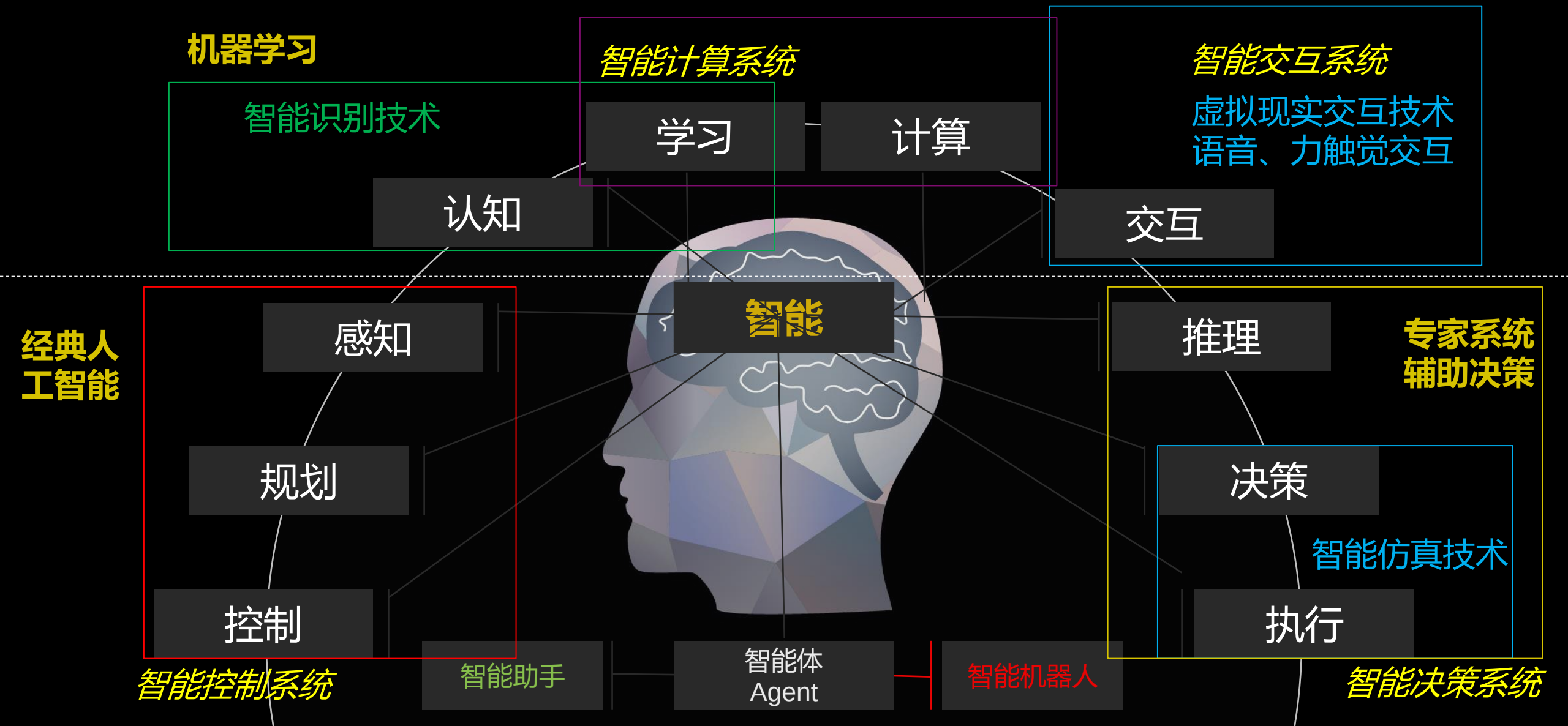
人工智能技术
Artificial Intelligence

——人工智能: 经典智能+计算智能+机器学习
AI: Classical Intelligence + Computing Intelligence + Machine Learning

王鸿鹏、杜月、王润花、许丽

南开大学人工智能学院

“人工智能技术”知识体系

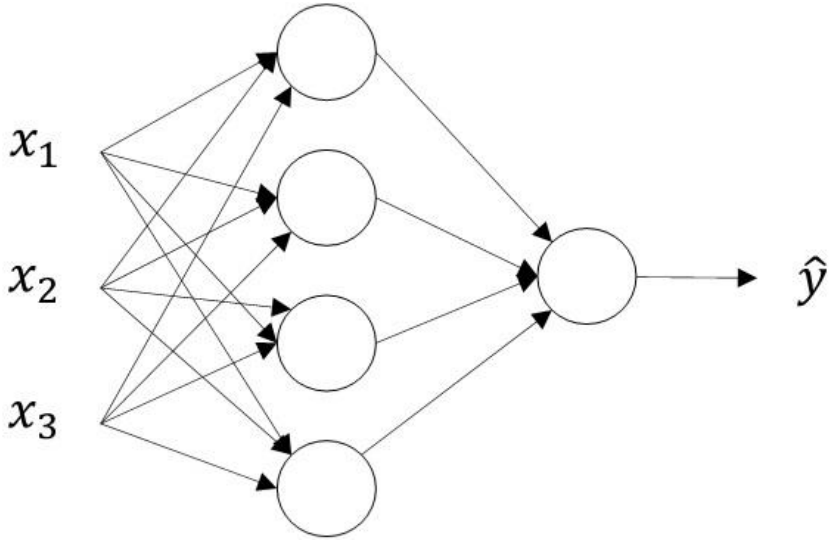

机器学习
智能识别技术
智能计算系统
学习
计算
认知
智能交互系统
虚拟现实交互技术
语音、力触觉交互
交互
智能
经典人工智能
感知
规划
控制
推理
专家系统
辅助决策
决策
智能仿真技术
执行
智能控制系统
智能助手
智能体
Agent
智能机器人
智能决策系统

第三部分 计算智能
Computational Intelligence

——第八章：概述、人工神经网络
Chapter 8: Introduction & ANN

人工神经网络

Artificial Neural Networks



神经计算是以神经网络为基础的计算

人工神经网络

人工神经网络（Artificial neural network, ANN）是一个将大量简单处理单元广泛连接而组成的人工网络（计算系统），是对人脑或生物神经网络若干基本特性的抽象和模拟。

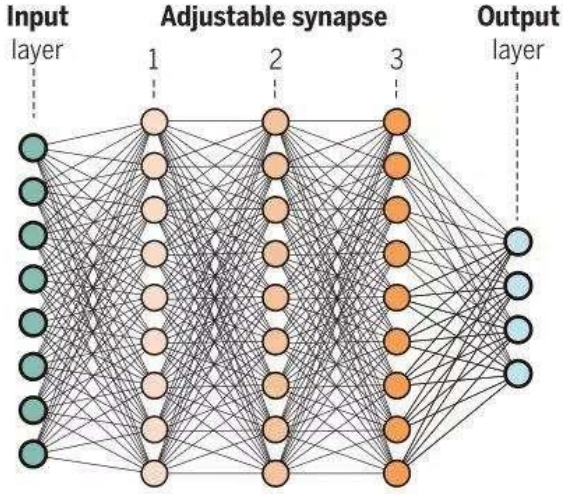
ANN目前已在模式识别、机器视觉、语音识别、机器翻译、图像处理、联想记忆、自动控制、信号处理、决策分析、智能计算、组合优化问题求解、数据挖掘等方面获得成功应用。

大量的神经网络模型和算法

BP神经网络

Hopfield神经网络及其应用

卷积神经网络、生成对抗网络……

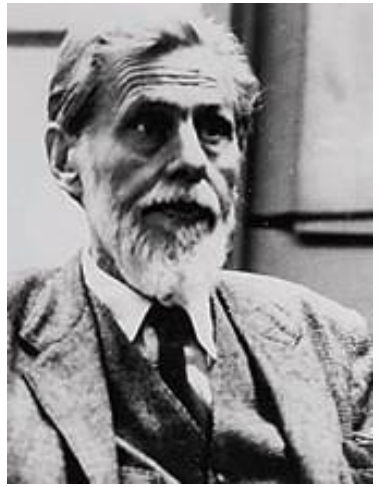


ANN研究历史

McCulloch & Pitts (1943) 被公认为第一个人工神经网络的设计者 (MP model)

他们使用阈值以及用多个简单单元结合在一起以提高计算能力的思想到今天仍被广泛使用

- 试图通过用计算模型模拟生物神经系统的方法来理解生物神经系统的工作模式
- 高度的并行性使得其计算效率非常高
- 有助于理解神经表示的“分布式”特征



麦克洛奇(McCulloch)

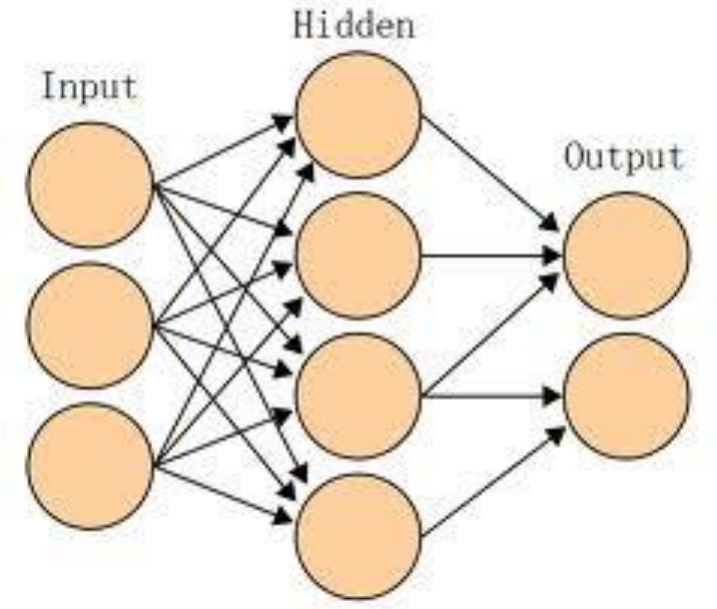
皮茨 (Pitts)

ANN研究历史

- 1949年Hebb提出了第一个神经网络的学习规则
- 1958年Rosenblatt提出感知器 (perceptron)模型，1950’ s 和1960’ s, 许多研究者致力于研究该模型
- 1969年 Minsky和Papert展示了感知器模型的局限性，从此神经网络的研究陷入低潮，沉寂了大概15年
- 1980’ s，神经网络重新成为研究热点
  - 1982 Hopfield: 递归神经网络(recurrent network model, RNN)
  - 1982 Kohonen:自组织神经网络(self-organizing maps, SOFM)
  - 1986 Rumelhart: 反向传播网络(backpropagation)

ANN的特性

- 并行分布处理
- 非线性映射
- 通过训练进行学习
- 适应与集成
- 硬件实现



这些特性使得人工神经网络具有应用于各种智能系统的巨大潜力

生物神经元模型

生物神经元结构Cell structures

Cell body(细胞体)

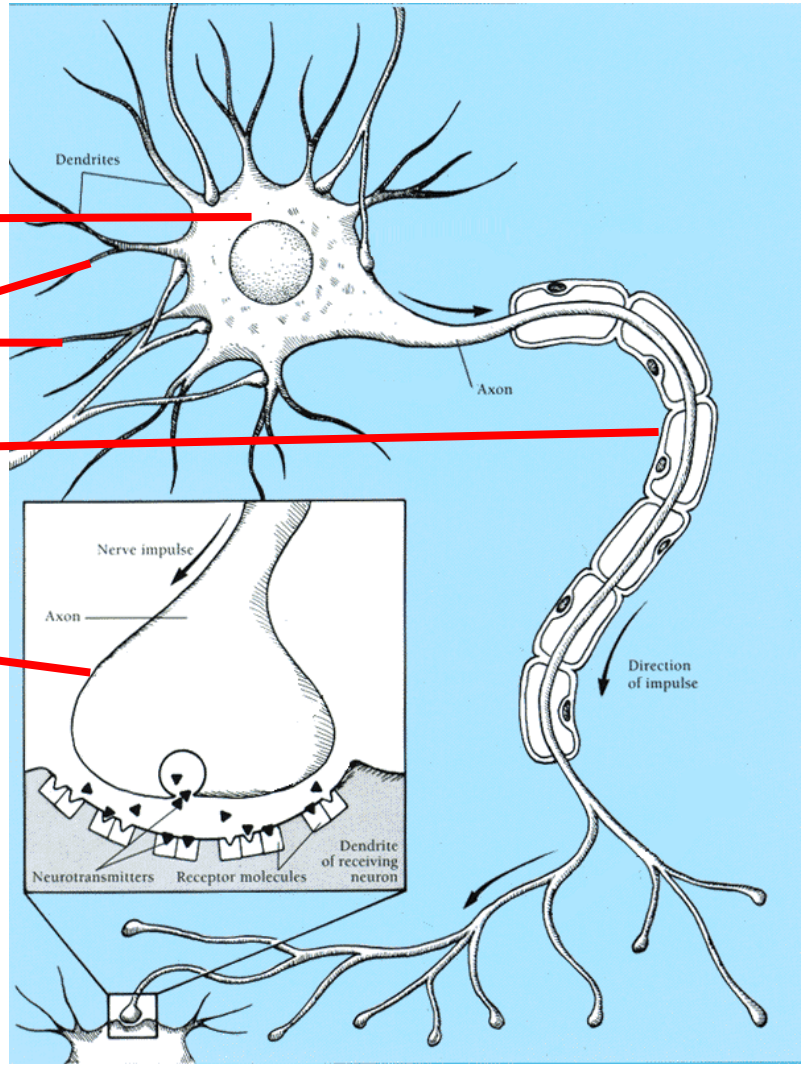
Dendrites(树突)

Axon(轴突)

Synapse(突触)

10^{11} - 10^{12} neurons in human brain

Each neuron connected to 10^4 others on average



ANN的结构

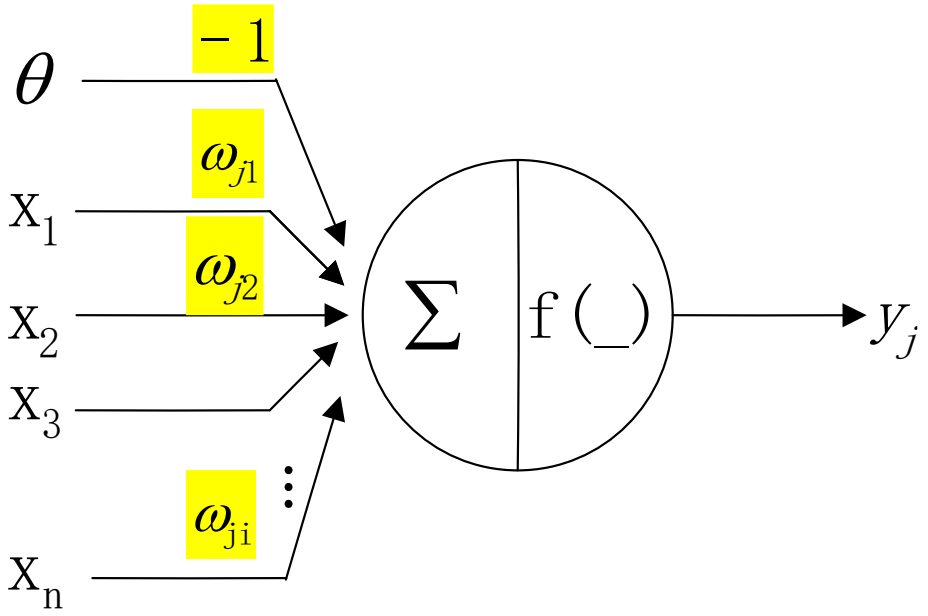
神经元及其特性(MP模型)：非线性阈值元件模型

In (Net)

$$u_j(t) = \sum_{i=1}^{n} \omega_{ji} x_i - \theta_j$$

Response

$$y_j(t) = f\left(\sum_{i=1}^{n} \omega_{ji} x_i - \theta_j\right)$$



θ_j	神经元单位的偏置(bias)或阈值(threshold)
ω_{ji}	连接强度/连接权系数(对于激发状态， ω_{ji}取正值，对于抑制状态， ω_{ji}取负值)
n	输入信号数目
y_j	神经元输出
t	时间
$f(\_)$	输出变换函数(transfer function)，或激活函数(activation function)

ANN的结构

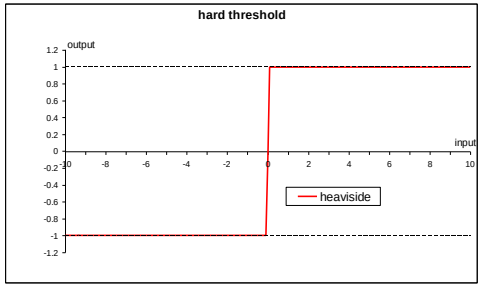
输出变换函数(激励函数)

输出变换函数往往采用0和1这种二值函数或S形函数

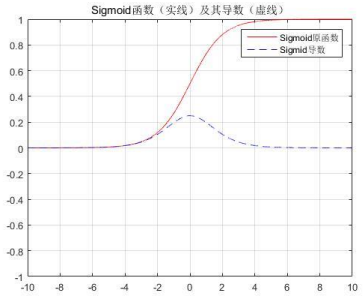
$$f(x)=\begin{cases}1, & x \geq x_0\\ 0, & x < x_0\end{cases}$$

$$f(x)=\frac{1}{1+e^{-ax}}, 0<f(x)<1$$

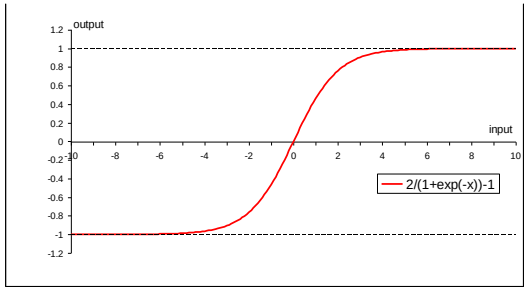
$$f(x)=\frac{1-e^{-ax}}{1+e^{-ax}}, -1<f(x)<1$$



(a) 二值函数/阶跃函数



(b) Sigmoid形函数



(c) 双曲正切函数

图 神经元中的某些变换(激励)函数

ANN的结构

人工神经网络由神经元模型构成，这种由许多神经元组成的信息处理网络具有并行分布结构。每个神经元具有单一输出，并且能够与其他神经元连接；存在许多(多重输出连接方式)，每种连接方法对应于一个连接权系数。

严格地说，人工神经网络是一种具有下列特征的有向图：

① 对于每个节点i存在一个状态变量x_i；

② 从节点i至节点j，存在一个连接权系数ω_{ij}；

③ 对于每个节点i，存在一个阈值θ_i；

④ 对于每个节点i，定义一个变换函数$f_i(x_j, \omega_{ij}, \theta_i)$，$i \neq j$；对于最一般的情况，此函数取$f_i(\sum_i \omega_{ij} x_j - \theta_i)$形式。

ANN的学习

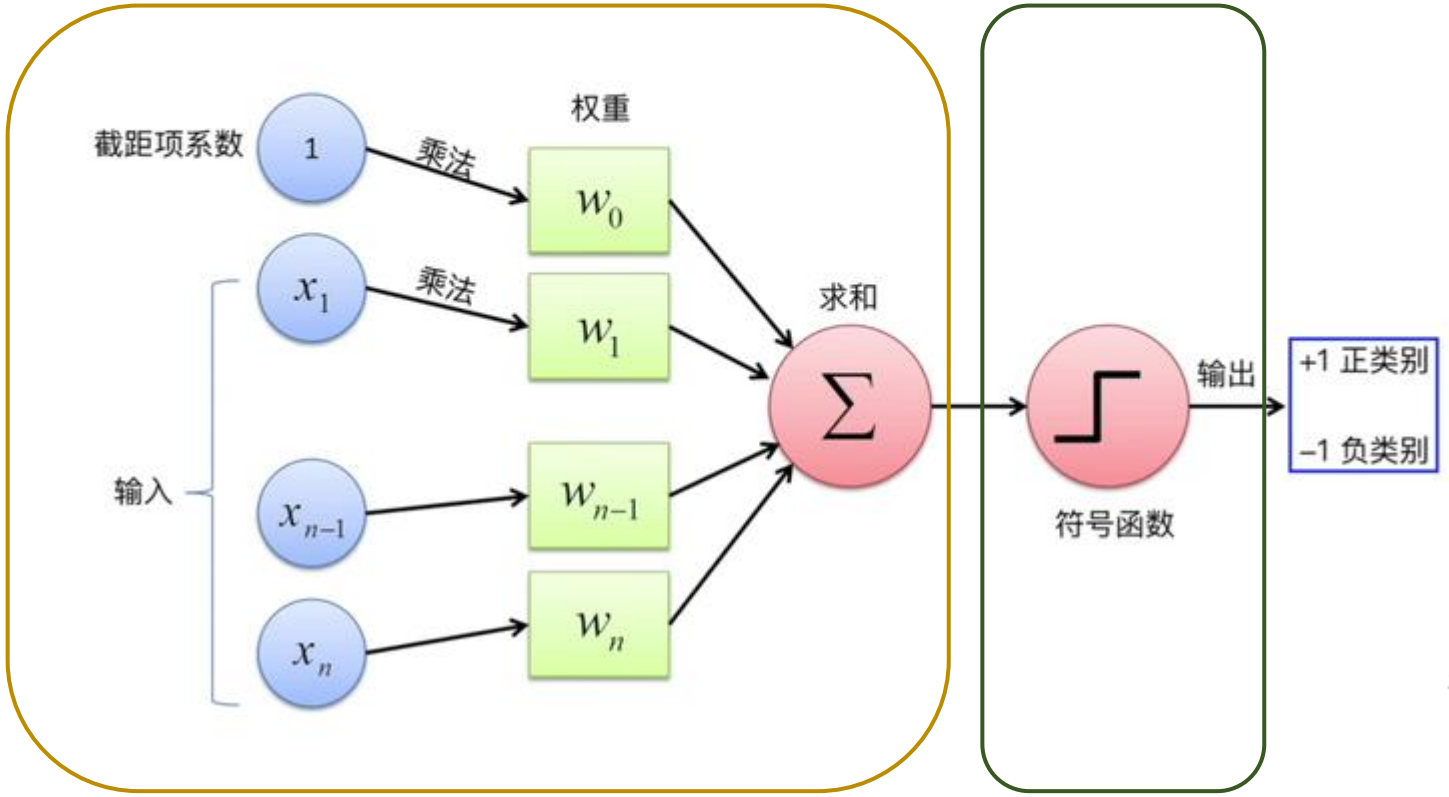
神经网络方法是一种知识表示和推理方法。

神经网络知识表示是一种隐式的表示方法，将某一问题的若干知识通过学习表示在同一网络中。

神经网络的学习是指调整神经网络的连接权值或者结构，使输入输出具有需要的特性。

从神经元到感知机

$$t = f(\sum_{i=1}^{n} w_i x_i + b) = f(\mathbf{w}^T \mathbf{x})$$

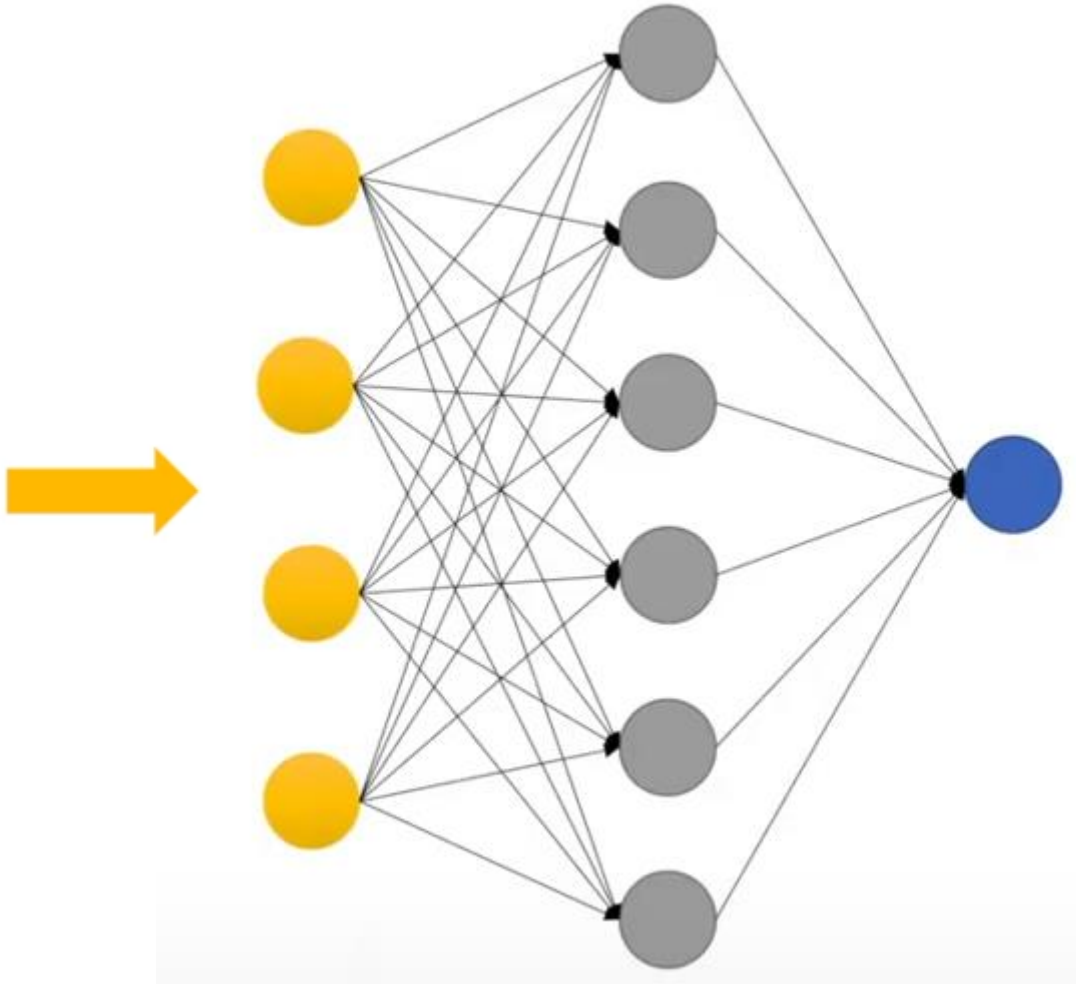


线性函数　　激活函数

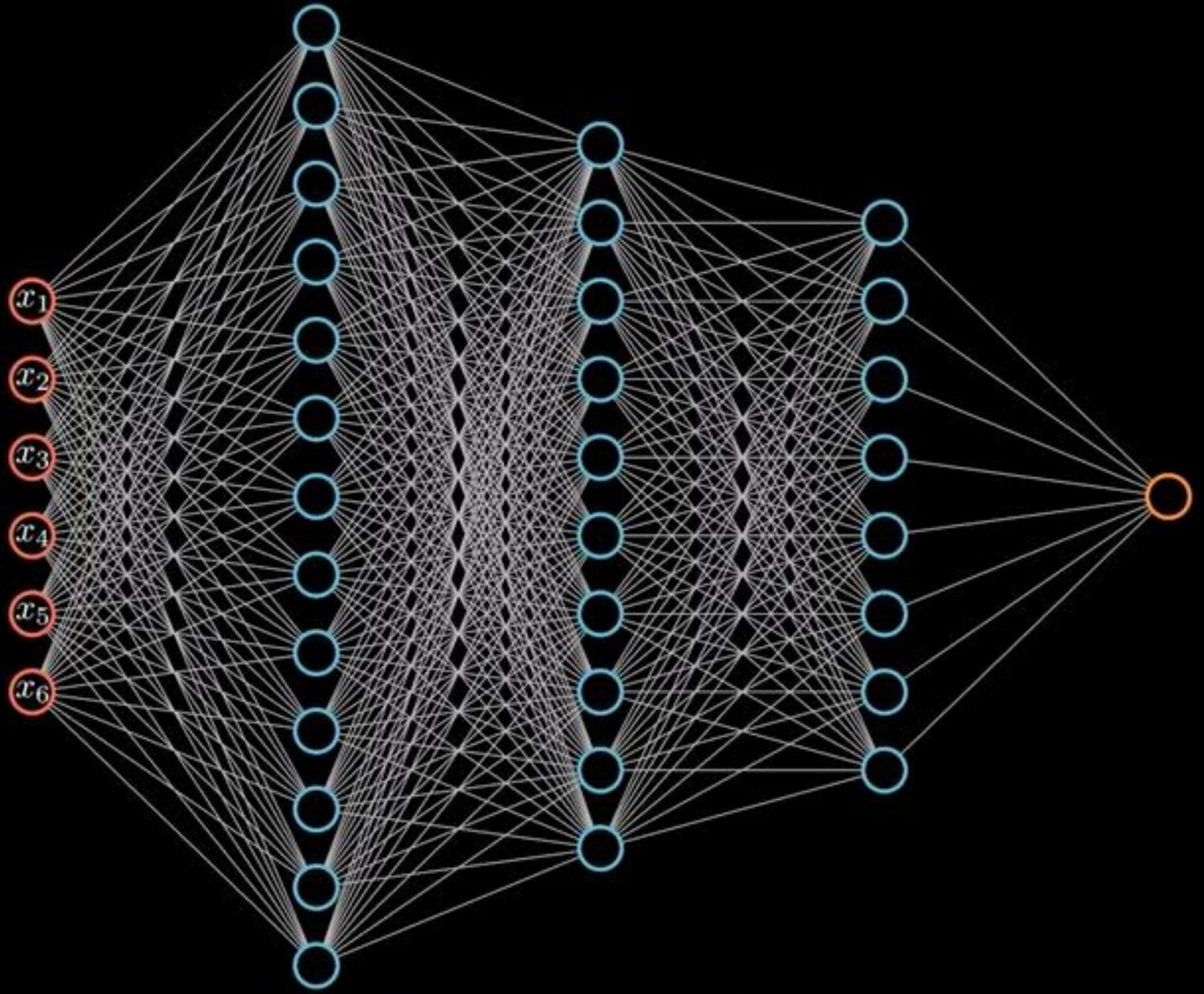
ANN的结构

OR

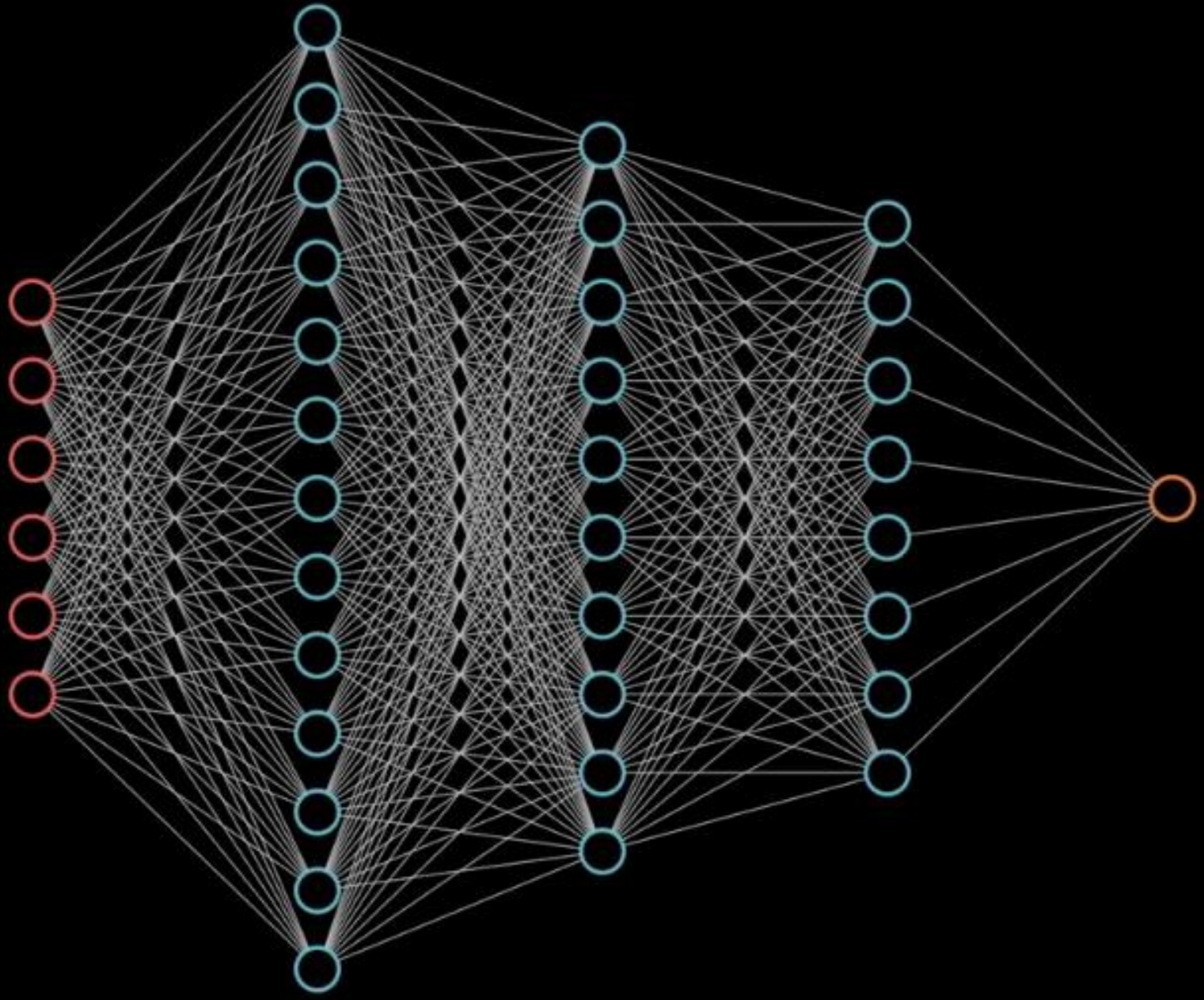
耳朵大小	毛的长度	胡子长短	脸圆
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....



这是猫



数据通过这一层输入到神经网络

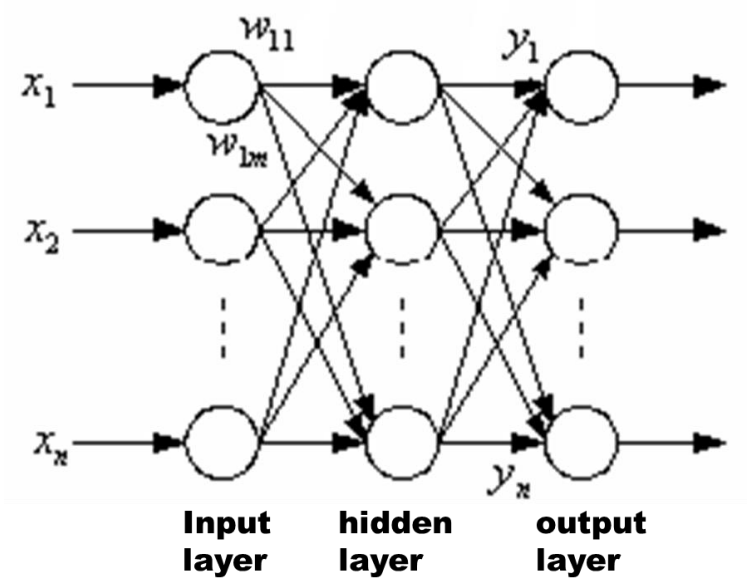


科学声音

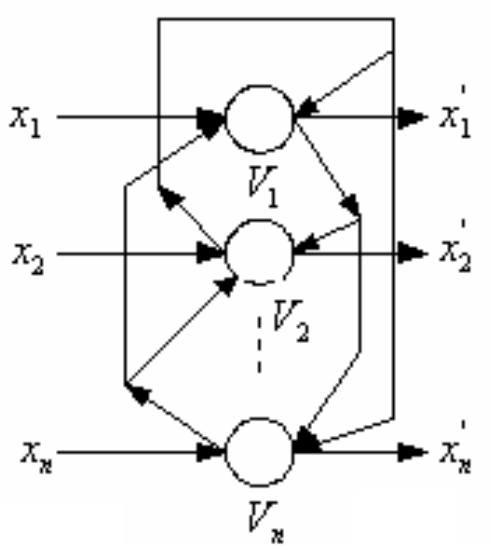
ANN的结构

基本分类

人工神经网络的结构基本上分为两类，即递归(反馈)网络和前馈网络：



(a)前馈网络

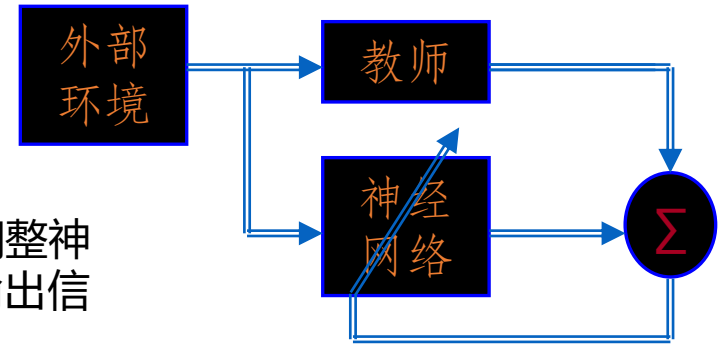


(b)递归(反馈)网络

ANN的主要学习算法

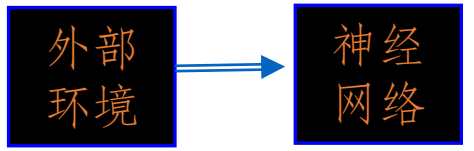
人工神经网络主要通过两种学习算法进行训练，即指导式(有师)学习算法和非指导式(无师)学习算法。此外，还存在第三种学习算法，即强化学习算法，可把它看做是有师学习的一种特例。

- 有师学习、监督学习(Supervised Learning)



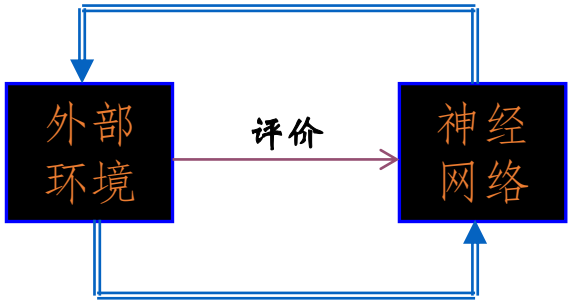
有师学习算法能够根据期望和实际的网络输出（对应于给定输入）之间的差来调整神经元间连接的强度或权。因此，有师学习需要老师或导师来提供期望目标或目标输出信号。

- 无师学习、非监督学习(Unsupervised Learning)



无师学习算法不需要知道期望输出。在训练过程中，只要向神经网络提供输入模式，神经网络就能自动地适应连接权，以便按相似特征把输入模式分组聚集

- 再激励学习、增强学习、强化学习(Reinforcement Learning)



增强（强化）学习算法是有师学习的特例，它不需要老师给出目标输出。增强学习算法采用要给“评论员”来评价与给定输入相对应的神经网络输出的优度(质量因数)。增强学习算法的一个例子是遗传算法(GA)

ANN工作原理

网络学习的准则：

如果网络作出错误的判决，则通过网络的学习，应使得网络减少下次犯同样错误的可能性。将模式分布地记忆在网络的各个连接权值上。当网络再次遇到其中任何一个模式时，能够作出迅速、准确的判断和识别。

神经网络的工作过程主要由两个阶段组成：

- 一个阶段是工作期，此时各连接权值固定，计算单元的状态变化，以求达到稳定状态，
- 另一阶段是学习期(自适应期或设计期)，此时各计算单元状态不变，各连接权值可修改(通过学习样本或其他方法)。

ANN学习规则

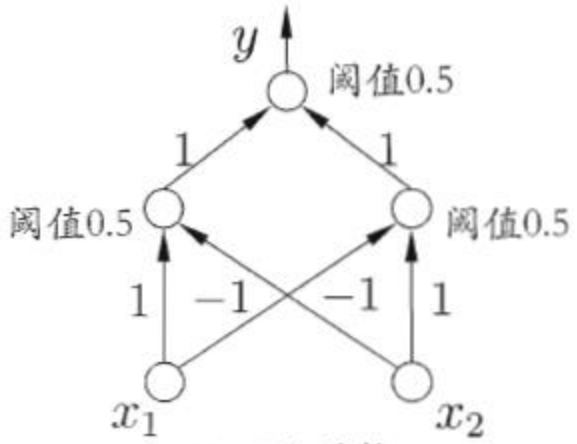
神经网络的学习过程：

首先设定初始权值，如果无先验知识，初始权值可设定为随机值。接着输入样本数据进行学习，参照评价标准进行评判。如果达到要求，就停止学习，否则按照给定的学习法则调整权值，继续进行学习，直到取得满意的结果为止。

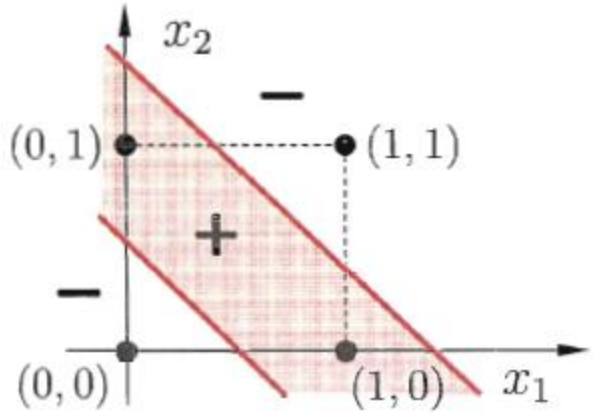
神经网络的学习规则(以Hebb规则为基础)，主要包括误差传播式学习、联想学习、竞争学习和基于知识的学习。

7.3.2 感知机与多层网络

- 非线性可分问题
  - 使用多层功能神经元
  - 输入层与输出层之间的神经元，称为隐层或隐含层（hidden layer）
  - 隐含层和输出层神经元都具有激活函数的功能神经元



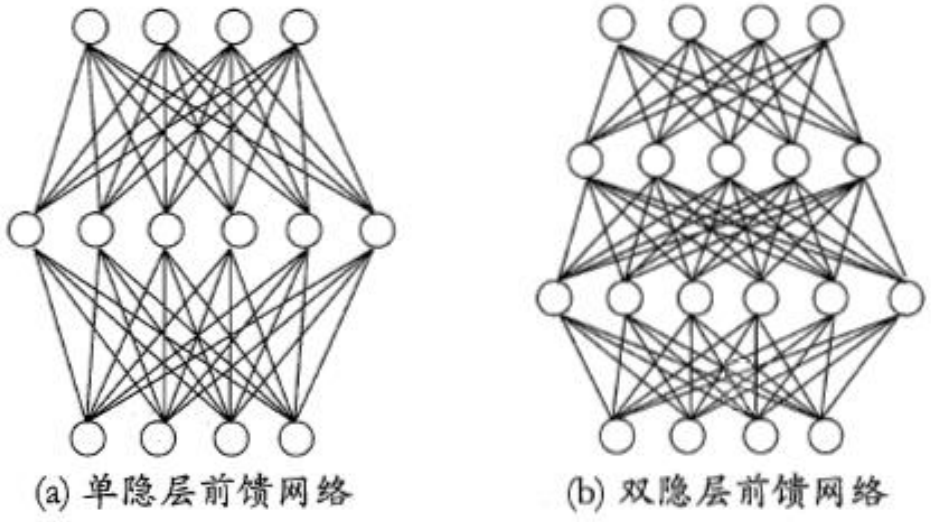
(a) 网络结构



(b) 分类区域

7.3.2 感知机与多层网络

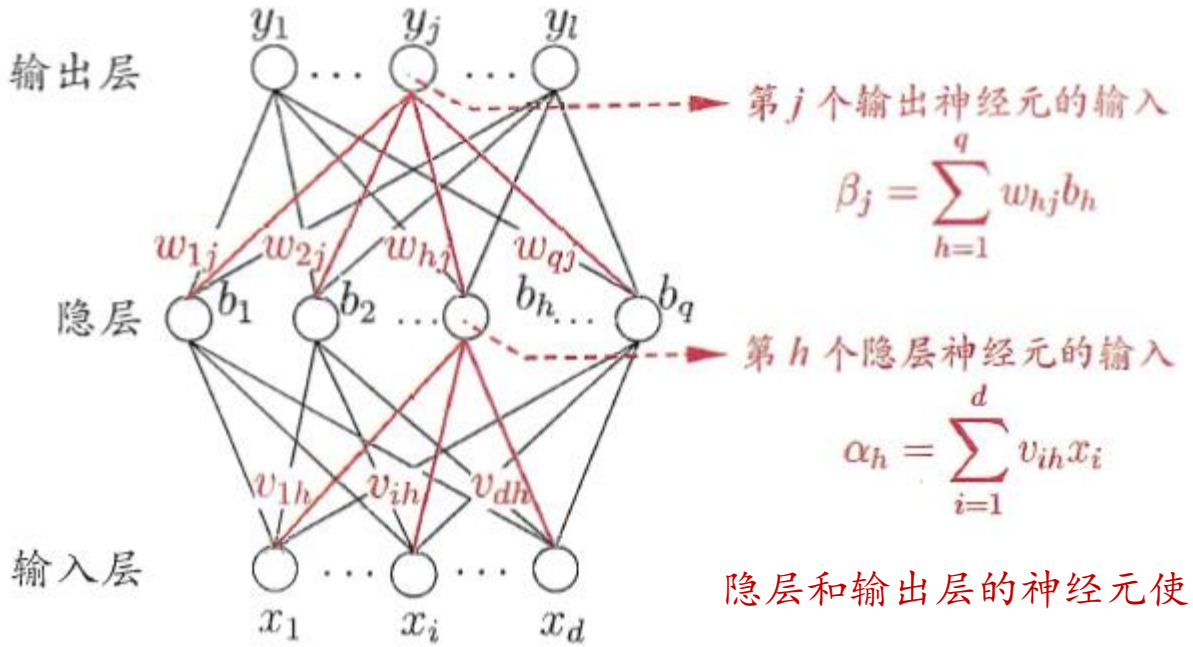
- 多层前馈神经网络（multi-layer feedforward neural network）
  - 输入层：接受外界输入
  - 隐层与输出层：包含功能神经元，进行函数处理
  - 同层神经元不互连，每层神经元与下一次神经元全互连，不存在



(a) 单隐层前馈网络　　(b) 双隐层前馈网络

7.3.3 误差逆传播算法

- 利用误差逆传播（error BackPropagation，BP）算法实现多层网络训练
- 训练集 $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^l$



隐层和输出层的神经元使用Sigmoid函数

7.3.3 误差逆传播算法

- BP算法是一个迭代学习算法，在迭代每一轮采用广义的感知机学习规则对参数进行更新
- 对于训练例(x_k, y_k)，假定神经网络输出为 即

$$\hat{\boldsymbol{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$$

$$\hat{y}_j^k = f(\beta_j - \theta_j)$$

- 神经网络均方差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$$

- 参数更新公式

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$$

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$

$$v \leftarrow v + \Delta v$$

$$\Delta w_{hj} = \eta g_j b_h$$

$$\Delta \theta_j = -\eta g_j$$

$$\Delta v_{ih} = \eta e_h x_i$$

$$\Delta \gamma_h = -\eta e_h$$

$$g_j = \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k)$$

$$e_h = b_h(1 - b_h)\sum_{j=1}^{l} w_{hj} g_j$$

7.3.3 误差逆传播算法

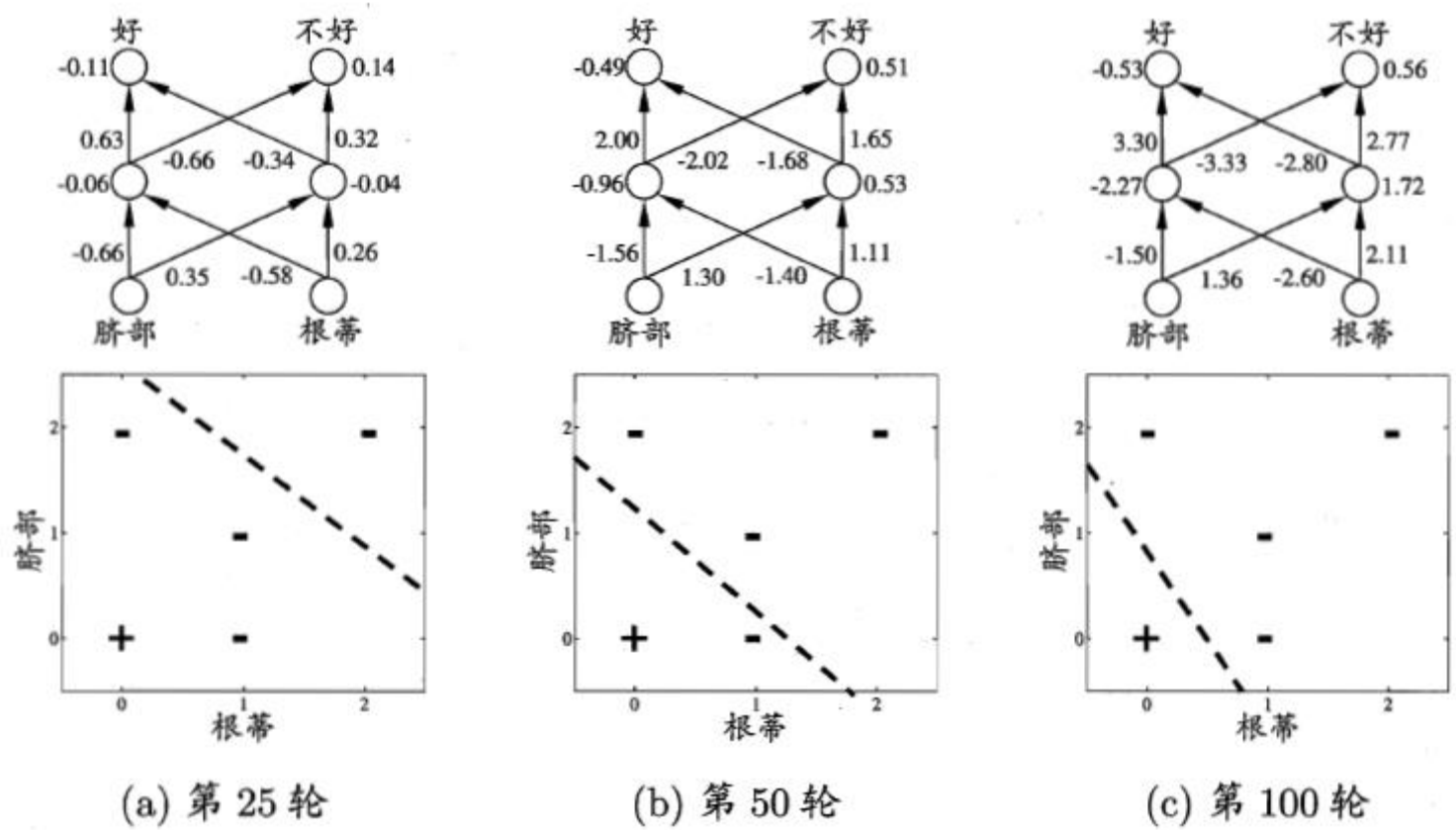
输入: 训练集 $D = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^{m}$;
学习率 η.

过程:

在$(0,1)$范围内随机初始化网络中所有连接权和阈值
repeat
  for all $(\boldsymbol{x}_k, \boldsymbol{y}_k) \in D$ **do**
    根据当前参数和式(5.3) 计算当前样本的输出 $\hat{\boldsymbol{y}}_k$;
    根据式(5.10) 计算输出层神经元的梯度项 g_j;
    根据式(5.15) 计算隐层神经元的梯度项 e_h;
    根据式(5.11)-(5.14) 更新连接权 w_{hj}, v_{ih} 与阈值 θ_j, γ_h
  end for
until 达到停止条件

输出: 连接权与阈值确定的多层前馈神经网络

7.3.3 误差逆传播算法



(a) 第 25 轮　(b) 第 50 轮　(c) 第 100 轮

7.3.3 误差逆传播算法

- 标准BP算法

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$$

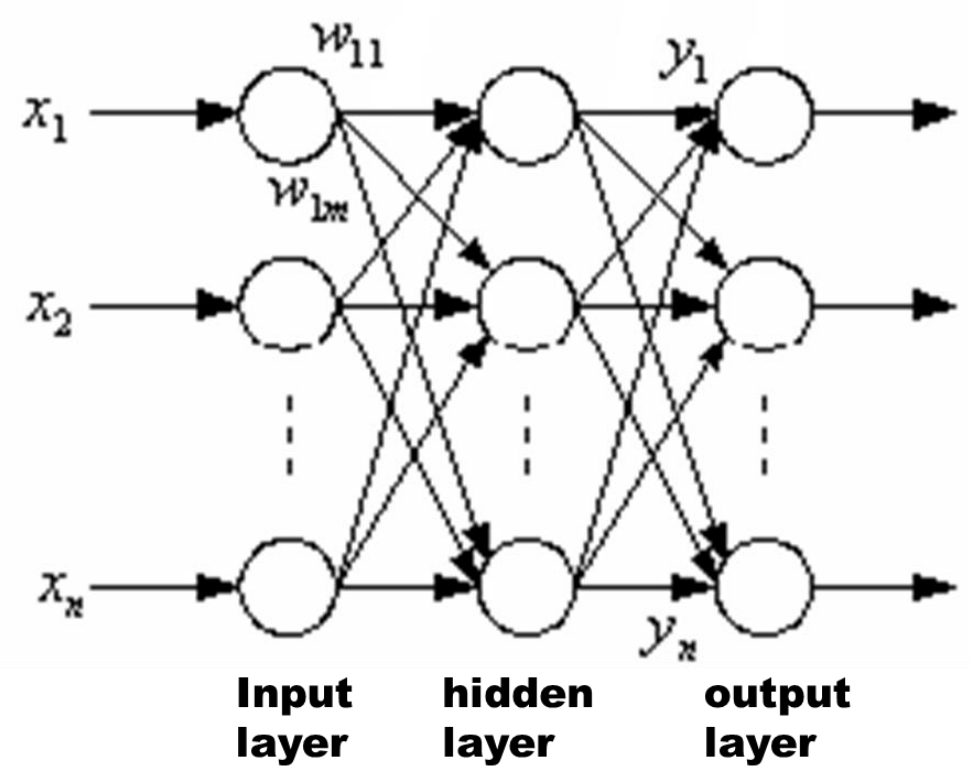
- 累计（accumulate error backpropagation）误差逆传播算法

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k$$

BP网络(Back propagation neural network)

- 正向传播：
  - 输入样本——输入层——隐含层——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



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，就获得了所有其它各层的误差估计。

BP网络

- BP算法是一个迭代学习算法，在迭代每一轮采用广义的感知机学习规则对参数进行更新
- 对于训练例(x_k, y_k)，假定神经网络输出为 即

$$\hat{\boldsymbol{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$$

$$\hat{y}_j^k = f(\beta_j - \theta_j)$$

- 神经网络均方差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$$

- 参数更新公式

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$$

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$

$$v \leftarrow v + \Delta v$$

$$\Delta w_{hj} = \eta g_j b_h$$

$$\Delta \theta_j = -\eta g_j$$

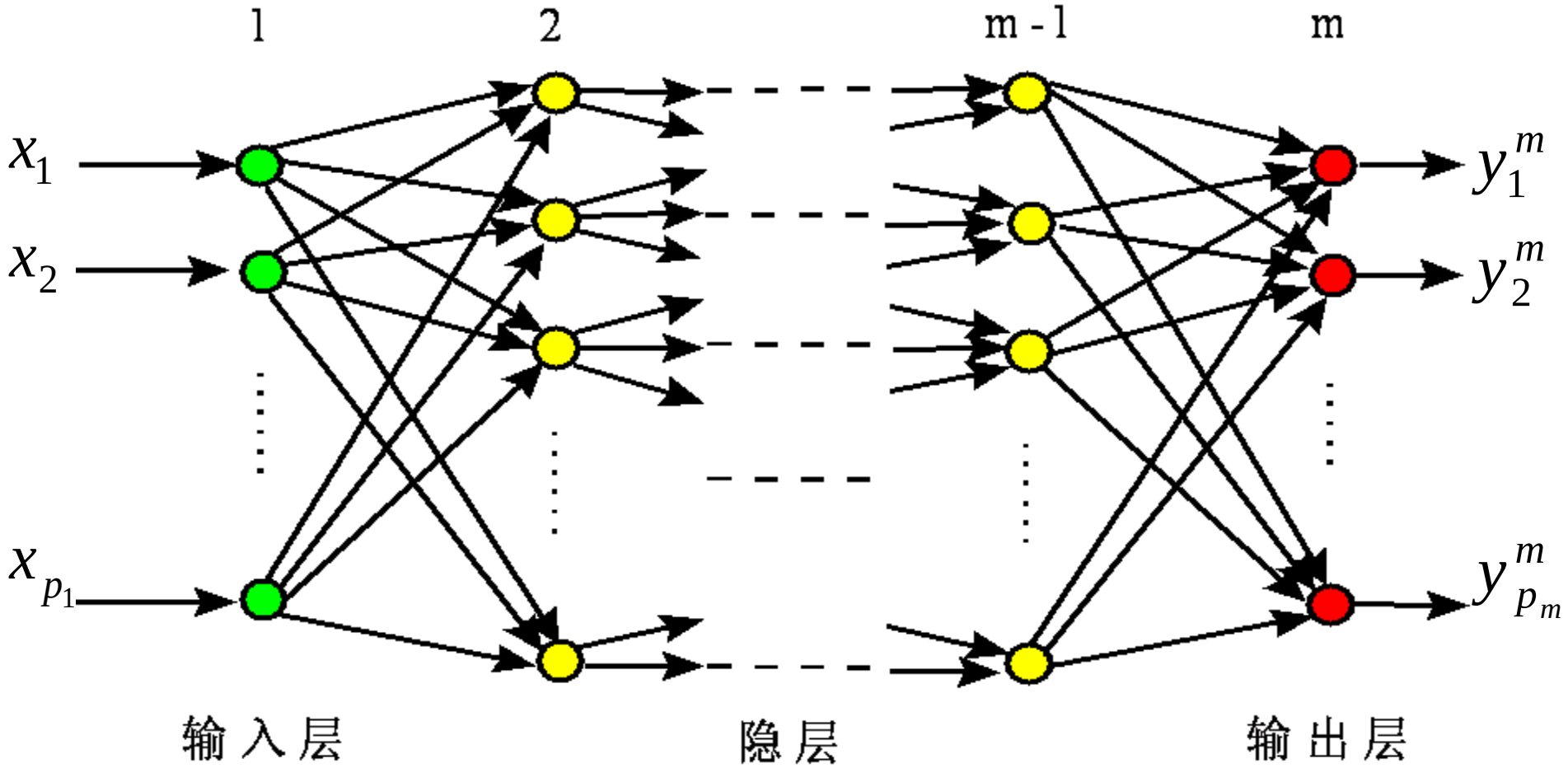
$$\Delta v_{ih} = \eta e_h x_i$$

$$\Delta \gamma_h = -\eta e_h$$

$$g_j = \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k)$$

$$e_h = b_h(1 - b_h)\sum_{j=1}^{l} w_{hj} g_j$$

BP神经网络的结构



$$X = [x_1 \quad x_2 \quad \cdots \quad x_{p_1}]^T$$

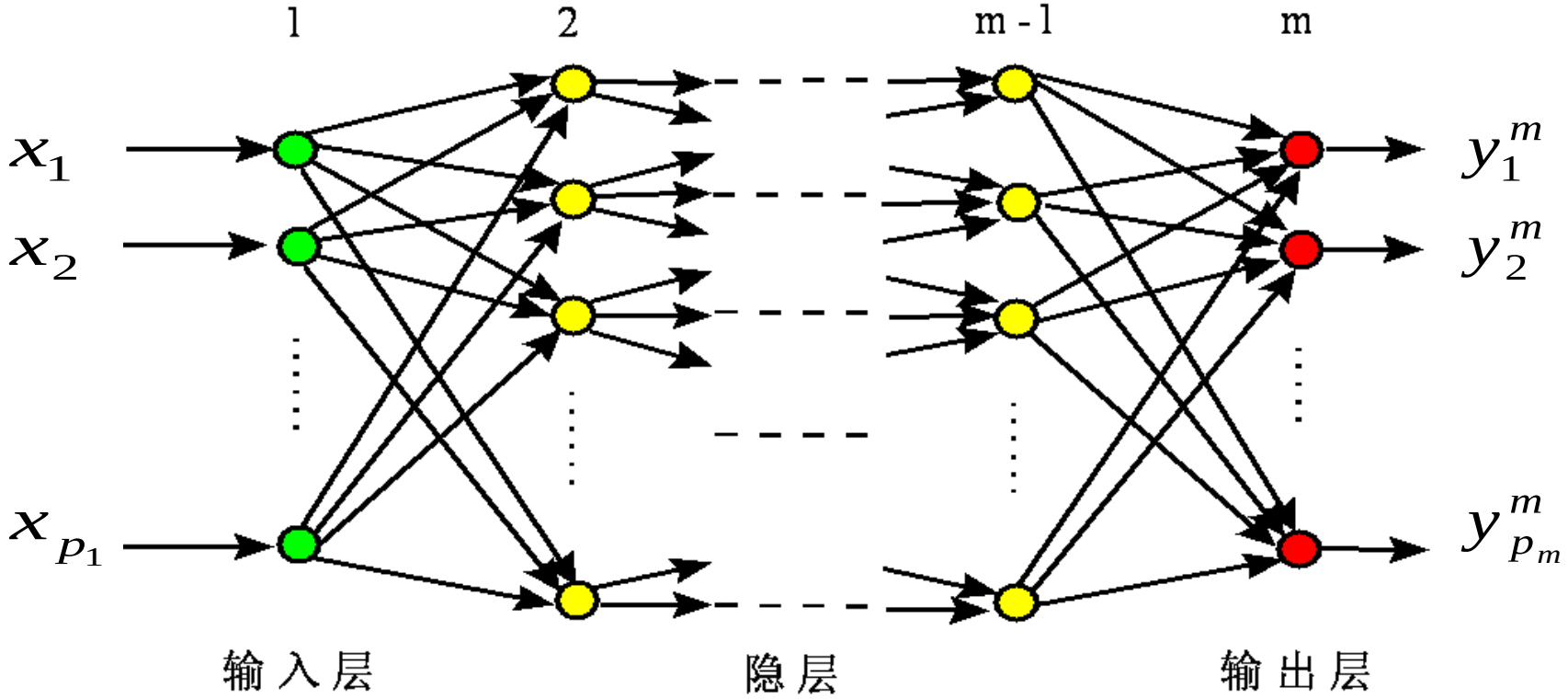
$$Y = [y_1^m \quad y_2^m \quad \cdots \quad y_{p_m}^m]^T$$

BP学习算法

2、学习算法

正向传播：输入信息由输入层传至隐层，最终在输出层输出。

反向传播：修改各层神经元的权值，使误差信号最小。



BP网络

输入： 训练集 $D = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^{m}$；
学习率 η.

过程：

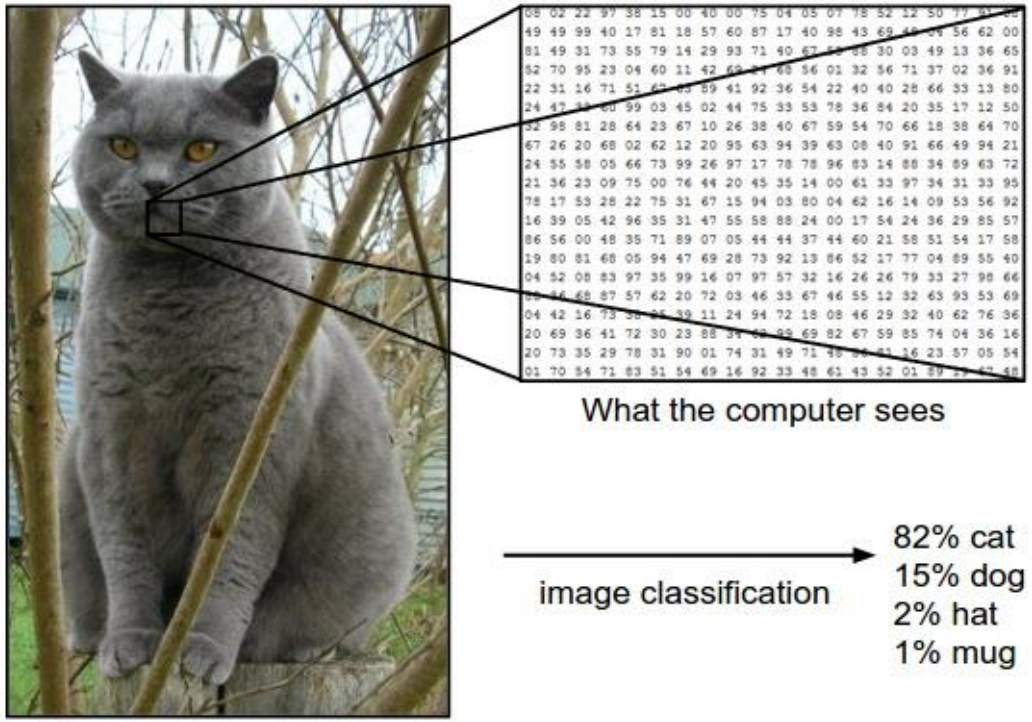
在$(0,1)$范围内随机初始化网络中所有连接权和阈值
repeat
  for all $(\boldsymbol{x}_k, \boldsymbol{y}_k) \in D$ **do**
    根据当前参数和式(5.3) 计算当前样本的输出 $\hat{\boldsymbol{y}}_k$;
    根据式(5.10) 计算输出层神经元的梯度项 g_j;
    根据式(5.15) 计算隐层神经元的梯度项 e_h;
    根据式(5.11)-(5.14) 更新连接权 w_{hj}, v_{ih} 与阈值 θ_j, γ_h
  end for
until 达到停止条件

输出： 连接权与阈值确定的多层前馈神经网络

BP网络应用——图像分类

什么是图像分类?

所谓图像分类问题，就是已有固定的分类标签集合，然后对于输入的图像，从分类标签集合中找出一个分类标签，最后把分类标签分配给该输入图像。



BP网络应用——图像分类

图像分类流程：

输入

制作数据集：将图像数据处理成网络模型能读取的形式。

训练

用训练集训练分类器或称学习一个模型。

预测

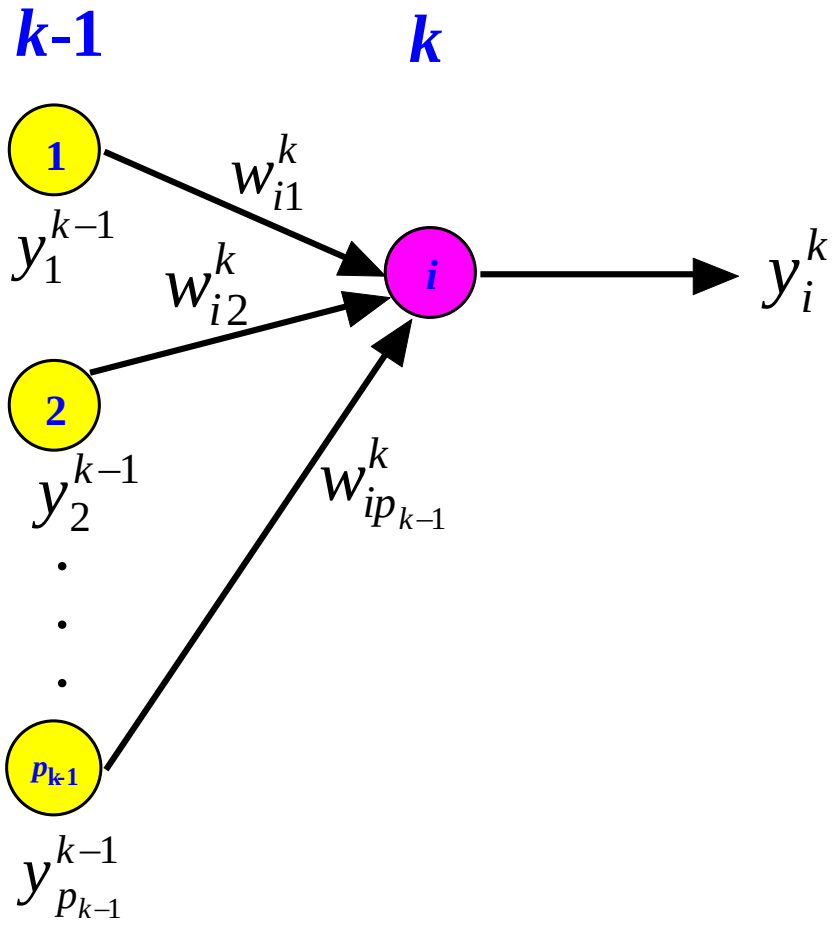
让分类器来预测它未曾见过的图像的分类标签，把分类器预测的标签和图像真正的分类标签对比并以此来评价分类器的质量。

BP网络应用——图像分类

note：图像以3维数组表示，数组中的元素是亮度值。一旦元素值发生变化，同样方法提取到的特征就会随之改变。因此计算机视觉算法在图像识别方面有以下困难和挑战：

BP神经网络的结构

输入输出变换关系



$$u_i^k = \sum_{j=1}^{p_{k-1}} w_{ij}^k y_j^{k-1} - \theta_i = \sum_{j=0}^{p_{k-1}} w_{ij}^k y_j^{k-1}$$

$$(y_0^{k-1} = \theta_i, w_{i0}^k = -1)$$

$$y_i^k = f(u_i^k) = \frac{1}{1+e^{-s_i^k}}$$

$$i = 1,2,\ldots, p_k$$

$$k = 1,2,\ldots,m$$

BP神经网络的结构

工作过程

- 第一阶段或网络训练阶段：
  - N 组输入输出样本：$x_i=[x_{i1}, x_{i2},..., x_{ip1}]^T$

    $d_i=[d_{i1}, d_{i2},...,d_{ipm}]^T$

    $i=1, 2,..., N$
  - 对网络的连接权进行学习和调整，以使该网络实现给定样本的输入输出映射关系。
- 第二阶段或称工作阶段：把实验数据或实际数据输入到网络，网络在误差范围内预测计算出结果。

BP学习算法

两个问题

（1）是否存在一个BP神经网络能够逼近给定的样本或者函数。

> Kolmogorov 定理：给定任意 $\varepsilon > 0$，对于任意的 L_2 型连续函数 $f:[0,1]^n \rightarrow R^m$，存在一个三层 BP 神经网络，其输入层有 n 个神经元，中间层有 $2n+1$ 个神经元，输出层有 m 个神经元，它可以在任意 ε 平方误差精度内逼近 f。

（ 2）如何调整BP神经网络的连接权，使网络的输入与输出与给定的样本相同。

1986年，鲁梅尔哈特（D. Rumelhart）等提出BP学习算法。

BP学习算法

1、基本思想

目标函数：

约束条件：

$$u_i^k = \sum_j w_{ij}^{k-1} y_j^{k-1} \qquad i = 1,2,\ldots, p_k$$

$$y_i^k = f_k(u_i^k) \qquad k = 1,2,\ldots,m$$

连接权值的修正量：

$$\Delta w_{ij}^{k-1} = -\varepsilon \frac{\partial J}{\partial w_{ij}^{k-1}} \qquad j = 1,2,\ldots, p_{k-1}$$

BP学习算法

先求 $$\frac{\partial J}{\partial w_{ij}^{k-1}}=\frac{\partial J}{\partial u_i^k}\frac{\partial u_i^k}{\partial w_{ij}^{k-1}}=\frac{\partial J}{\partial u_i^k}\frac{\partial}{\partial w_{ij}^{k-1}}\left(\sum_j w_{ij}^{k-1}y_j^{k-1}\right)=\frac{\partial J}{\partial u_i^k}y_j^{k-1}$$

记 $$d_i^k=\frac{\partial J}{\partial u_i^k}=\frac{\partial J}{\partial y_i^k}\frac{\partial y_i^k}{\partial u_i^k}=\frac{\partial J}{\partial y_i^k}f_k'(u_i^k)$$

(1) 对输出层的神经元 $$\frac{\partial J}{\partial y_i^k}=\frac{\partial J}{\partial y_i^m}=y_i^m-y_{si}$$

$$d_i^m=(y_i^m-y_{si})f_m'(u_i^m)$$

(2) 对隐单元层，则有 $$\frac{\partial J}{\partial y_i^k}=\sum_l\frac{\partial J}{\partial u_l^{k+1}}\frac{\partial u_l^{k+1}}{\partial y_i^k}=\sum_l d_l^{k+1}w_{li}^k$$

$$d_i^k=f_k'(u_i^k)\sum_l d_l^{k+1}w_{li}^k$$

$$\Delta w_{ij}^{k-1}=-\varepsilon d_i^k y_j^{k-1}$$

BP学习算法

2、学习算法

$$\Delta w_{ij}^{k-1} = -\varepsilon d_i^k y_j^{k-1}$$

$$d_i^m = (y_i^m - y_{si}) f_m'(u_i^m)$$

——输出层连接权调整公式

$$d_i^k = f_k'(u_i^k) \sum_l d_l^{k+1} w_{li}^k$$

——隐层连接权调整公式

BP学习算法

2、学习算法

$$当 y_i^k = \frac{1}{1+e^{-u_i^k}} 时$$

$$\Delta w_{ij}^{k-1} = -\varepsilon d_i^k y_j^{k-1}$$

$$d_i^m = y_i^m (1-y_i^m)(y_i^m - y_i)$$

——输出层连接权调整公 式

$$d_i^k = y_i^k (1-y_i^k) \sum_{l=1}^{p_{k+1}} w_{li}^{k+1} d_l^{k+1}$$

——隐层连接权调整公式

BP算法的实现

1、BP算法的设计

- 隐层数及隐层神经元数的确定：目前尚无理论指导。
- 初始权值的设置：一般以一个均值为0的随机分布设置网络的初始权值。
- 训练数据预处理：线性的特征比例变换，将所有的特征变换到[0，1]或者[-1，1]区间内，使得在每个训练集上，每个特征的均值为0，并且具有相同的方差。
- 后处理过程：当应用神经网络进行分类操作时，通常将输出值编码成所谓的名义变量，具体的值对应类别标号。

BP算法的实现

2、BP算法的计算机实现流程

（1）初始化：对所有连接权和阈值赋以随机任意小值；

$$w_{ij}^{k}(t),\quad \theta_{i}^{k}(t),(k=1,\ldots,m;i=1,\ldots,p_k;j=1,\ldots,p_{k-1};t=0)$$

（2） 从 N 组输入输出样本中取一组样本：$x=[x_1, x_2,\ldots, x_{p1}]^T$，$d=[d_1, d_2,\ldots,d_{pm}]^T$，把输入信息$x=[x_1, x_2,\ldots, x_{p1}]^T$输入到BP网络中

（3）正向传播：计算各层节点的输出：

$$y_i^k \qquad (i=1,\ldots,p_k;\ \ k=1,\ldots,m)$$

（4）计算网络的实际输出与期望输出的误差：

$$e_i=y_i-y_i^m \qquad (i=1,\ldots,p_m)$$

BP算法的实现

2、BP算法的计算机实现流程

（5）反向传播：从输出层方向计算到第一个隐层，按连接权值修正公式向减小误差方向调整网络的各个连接权值。

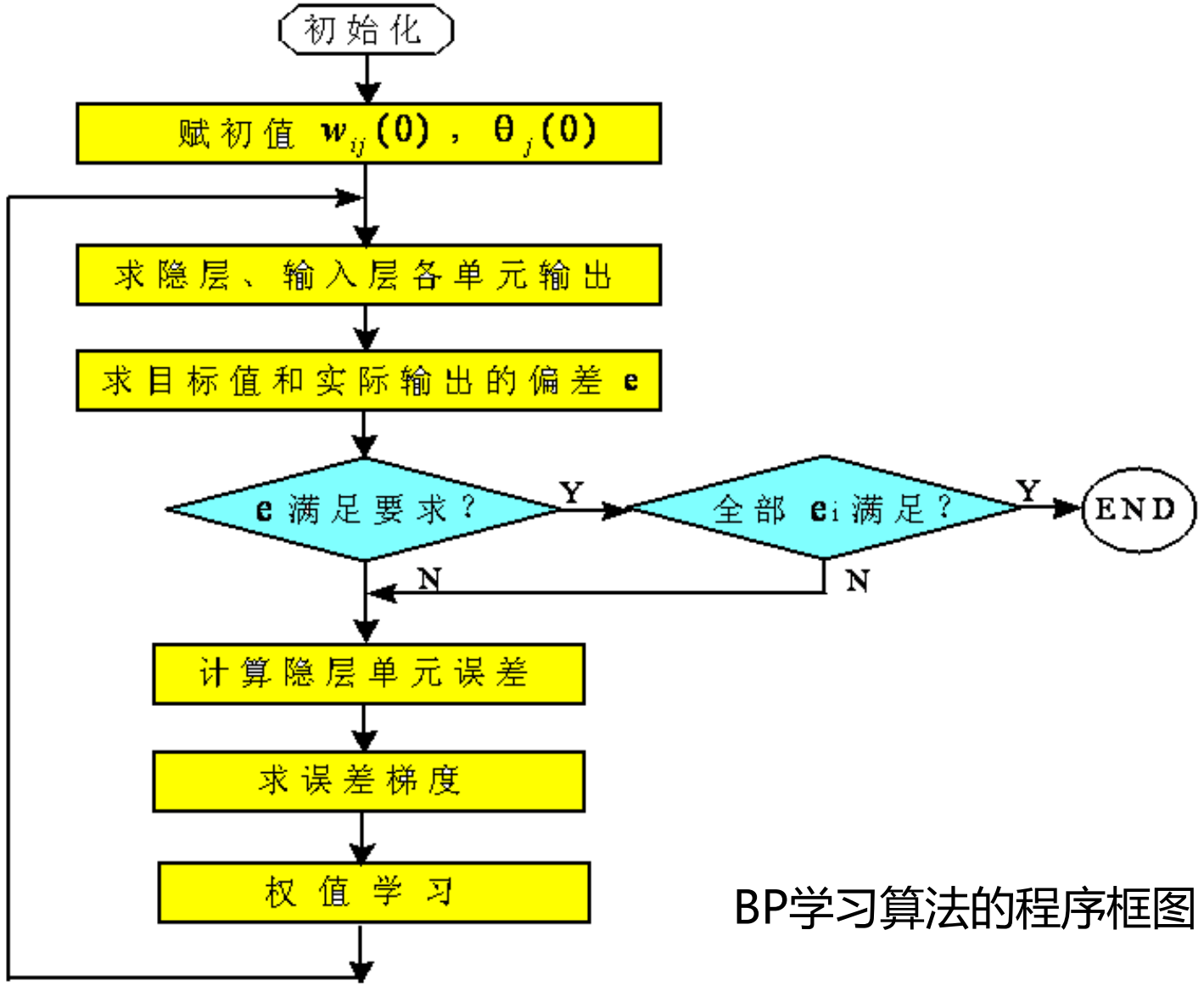
$$\Delta w_{ij} = -\alpha d_i^k y_j^{k-1}$$

$$d_i^m = y_i^m(1-y_i^m)(y_i^m - y_i) \quad \text{——输出层连接权调整公式}$$

$$d_i^k = y_i^k(1-y_i^k)\sum_{l=1}^{p_{k+1}} w_{li}^{k+1} d_l^{k+1} \quad \text{——隐层连接权调整公式}$$

（6）让$t+1 \to t$，取出另一组样本重复（2）－（5），直到 N 组输入输出样本的误差达到要求时为止。

BP学习算法



BP学习算法的程序框图

BP算法的特点分析

1. 特点

- BP网络：多层前向网络（输入层、隐层、输出层）。
- 连接权值：通过Delta学习算法进行修正。
- 神经元传输函数：S形函数。
- 学习算法：正向传播、反向传播。
- 层与层的连接是单向的，信息的传播是双向的。

BP算法的特点分析

2. BP网络的主要优缺点

优点

- 很好的逼近特性。
- 具有较强的泛化能力。
- 具有较好的容错性。

缺点

- 收敛速度慢。
- 局部极值。
- 难以确定隐层和隐层结点的数目。

BP神经网络在模式识别中的应用

模式识别研究用计算机模拟生物、人的感知，对模式信息，如图像、文字、语音等，进行识别和分类。

传统人工智能的研究部分地显示了人脑的归纳、推理等智能。但是，对于人类底层的智能，如视觉、听觉、触觉等方面，现代计算机系统的信息处理能力还不如一个幼儿园的孩子。

神经网络模型模拟了人脑神经系统的特点：处理单元的广泛连接；并行分布式信息储存、处理；自适应学习能力等。

神经网络模式识别方法具有较强的容错能力、自适应学习能力、并行信息处理能力。

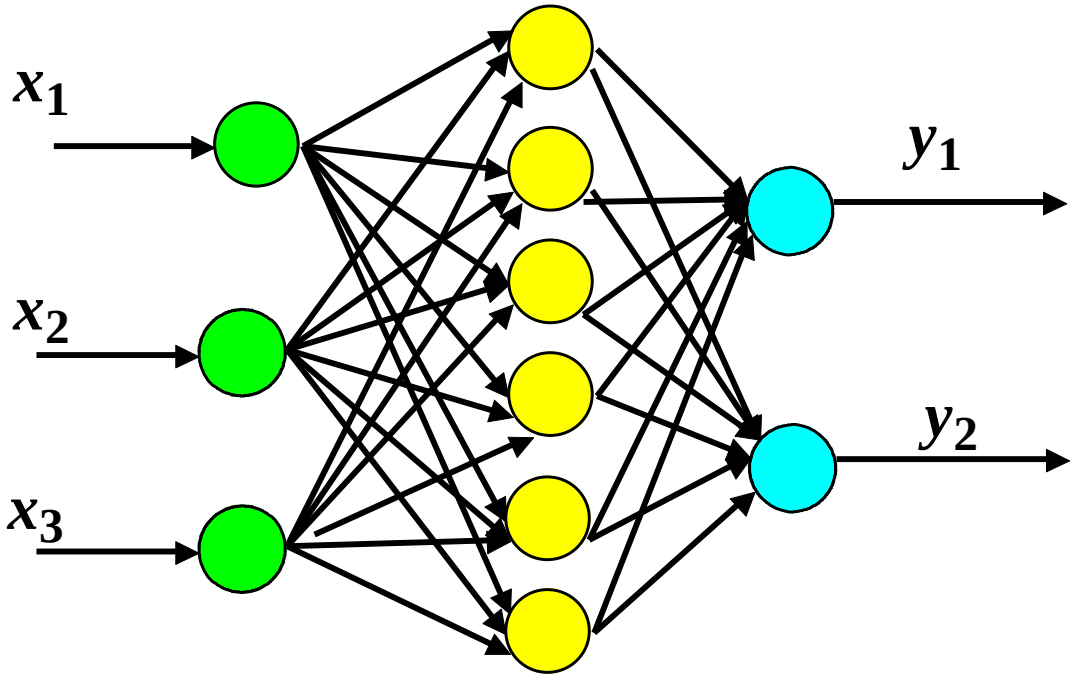
BP神经网络在模式识别中的应用

- 例 输入输出样本：

输入			输出	
1	0	0	1	0
0	1	0	0	0.5
0	0	1	0	1

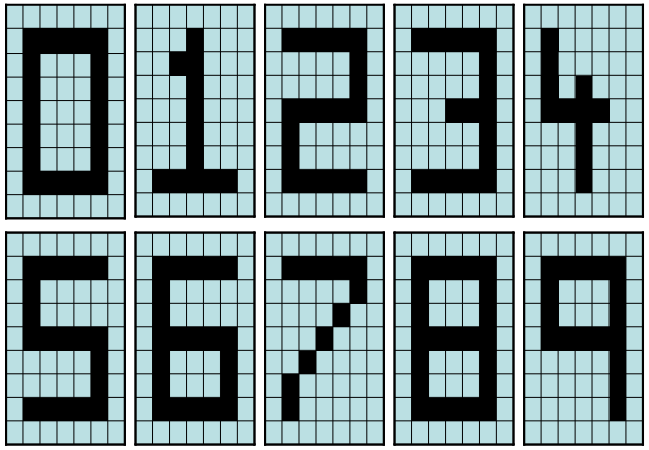
- 测试数据：

输入		
0.97	0.001	0.001
0	0.98	0
0.002	0	1.04
0.5	0.5	0.5
1	0	0
0	1	0
0	0	1



BP神经网络在模式识别中的应用

例 设计一个三层BP网络对数字0至9进行分类。



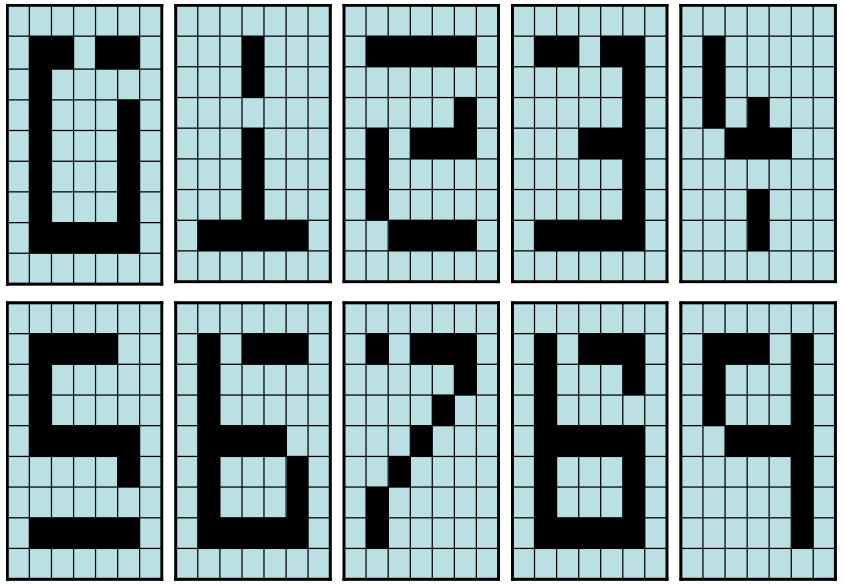
每个数字用9×7的网格表示，灰色像素代表0，黑色像素代表1。将每个网格表示为0，1的长位串。位映射由左上角开始向下直到网格的整个一列，然后重复其他列。

选择BP网络结构为63-6-9。97个输入结点，对应上述网格的映射。9个输出结点对应10种分类。

使用的学习步长为0.3。训练600个周期，如果输出结点的值大于0.9，则取为ON，如果输出结点的值小于0.1，则取为OFF。

BP神经网络在模式识别中的应用

当训练成功后，对如图所示测试数据进行测试。测试数据都有一个或者多个位丢失。



测试结果表明：除了8以外，所有被测的数字都能够被正确地识别。

对于数字8，神经网络的第6个结点的输出值为0.53，第8个结点的输出值为0.41，表明第8个样本是模糊的，可能是数字6，也可能是数字8，但也不完全确信是两者之一。

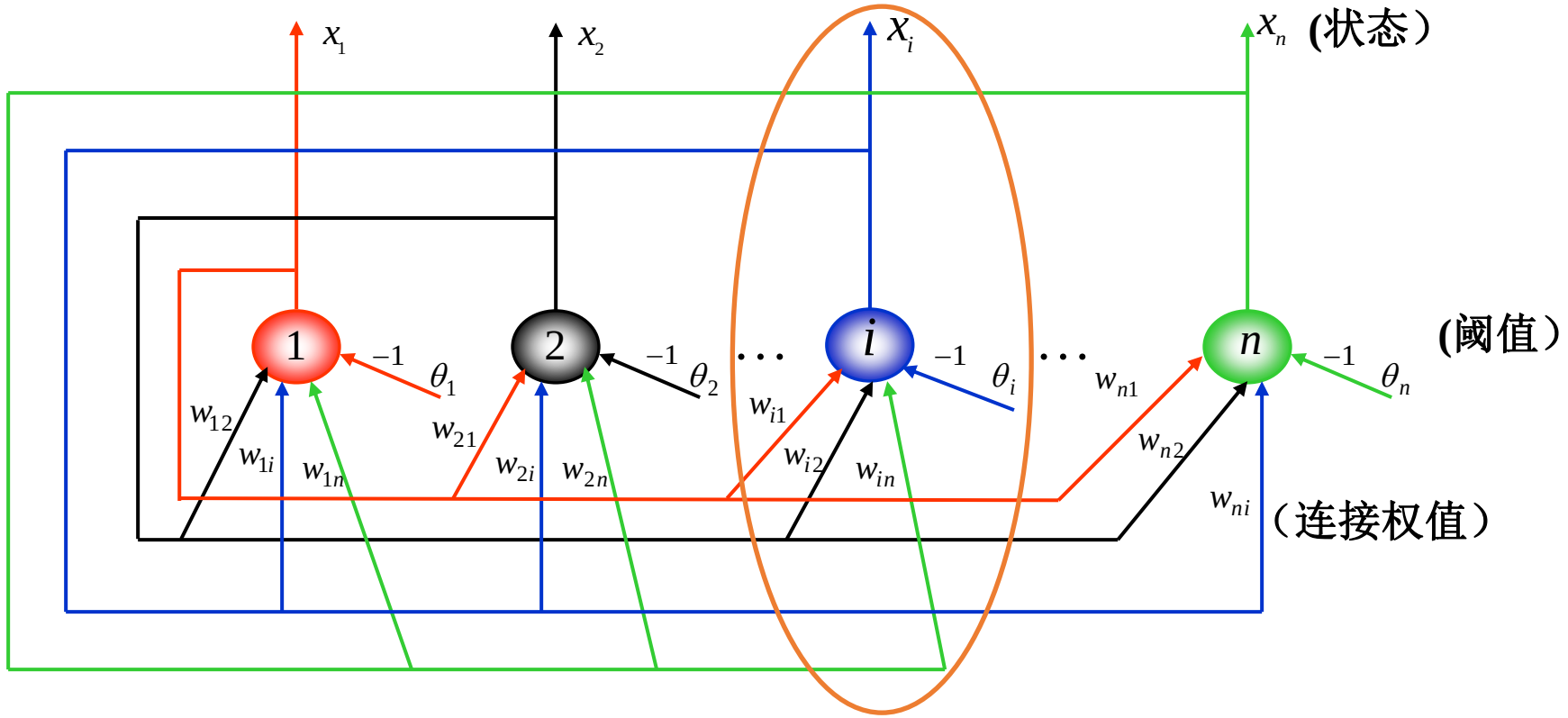
Hopfield 神经网络及其改进

- 离散型Hopfield神经网络
- 连续型Hopfield神经网络及其VLSI实现
- 随机神经网络
- 混沌神经网络

离散Hopfield神经网络

1. 离散Hopfield神经网络模型

- 网络结构：



离散Hopfield神经网络结构图

离散Hopfield神经网络

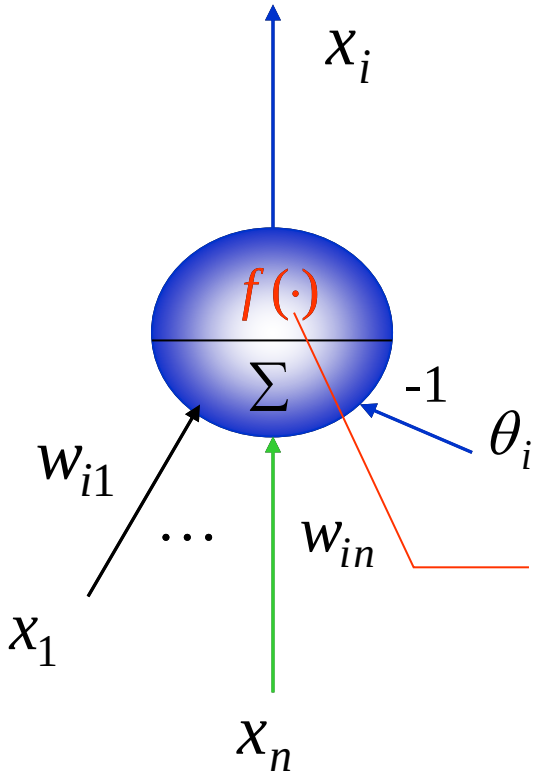
1. 离散Hopfield神经网络模型

- 输入输出关系：

$$x(k+1) = f(W_x(k) - \theta), \forall i$$

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T \quad \boldsymbol{\theta} = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_n]^T$$

$$\mathbf{W} = [w_{ij}]_{n\times n} \quad \mathbf{f}(s) = [f(s_1) \quad f(s_2) \quad \cdots \quad f(s_n)]^T$$



$$x_i(k+1) = f\left(\sum_{j=1}^{n} w_{ij} x_j(k) - \theta_i\right)$$

注：$w_{ii=0}$

$$f(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases} \quad \text{或} \quad f(s) = \begin{cases} 1 & s \geq 0 \\ 0 & s < 0 \end{cases}$$

离散Hopfield神经网络

1. 离散Hopfield神经网络模型

- 工作方式：

异步（串行）方式：

$$\begin{cases} x_i(k+1) = f\left(\sum_{j=1}^{n} w_{ij}x_j(k) - \theta_i\right) \\ x_j(k+1) = x_j(k), \qquad j \neq i \end{cases}$$

同步（并行）方式：

$$x_i(k+1) = f\left(\sum_{j=1}^{n} w_{ij}x_j(k) - \theta_i\right), \forall i$$

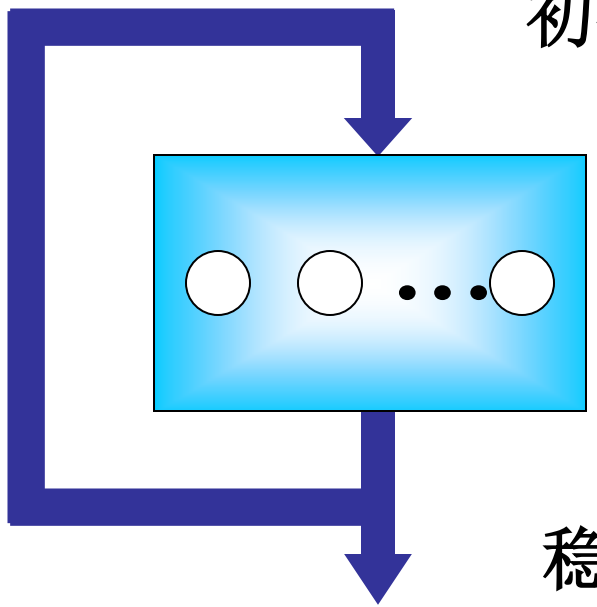
$$\Downarrow$$

$$x(k+1) = f(W_x(k) - \theta), \forall i$$

离散Hopfield神经网络

1. 离散Hopfield神经网络模型

- 工作过程：



初态：$x(0)=[x_1(0),\quad x_2(0),\quad \cdots,\quad x_n(0)]^{\mathrm{T}}$

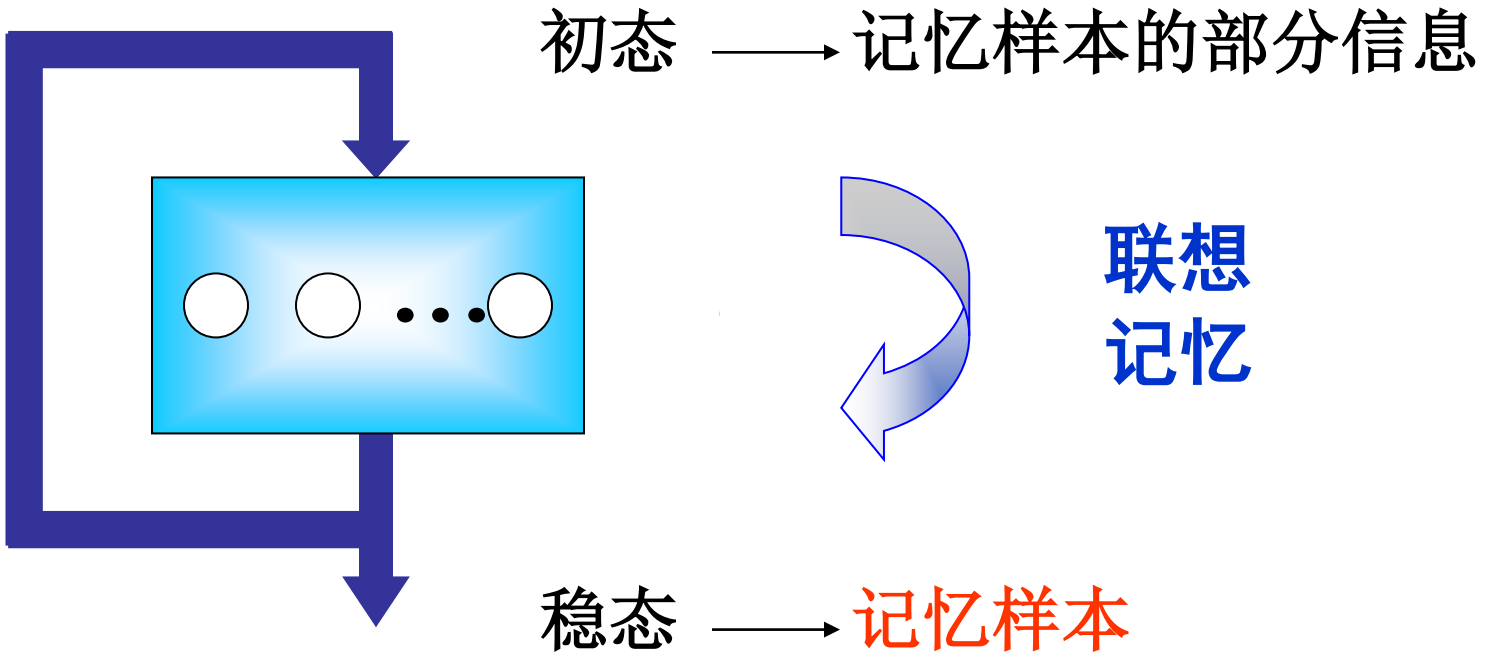
（异步或同步方式）

稳态：$\lim\limits_{k\to\infty} x(k)$

离散Hopfield神经网络

1. 离散Hopfield神经网络模型

- 工作过程：



离散Hopfield神经网络

2. 网络的稳定性

- 稳定性定义：

- 若从某一时刻开始，网络中所有神经元的状态不再改变，即 $x = f(w_x - \theta)$， 则称该网络是稳定的，x 为网络的稳定点或吸引子 。 $x(k) = f(W_x(k) - \theta) = x(k+1)$

- **Hopfield**神经网络是高维非线性系统，可能有许多稳定优态。从任何初始状态开始运动，总可以到某个稳定状态。这些稳定状态可以通过改变网络参数得到。

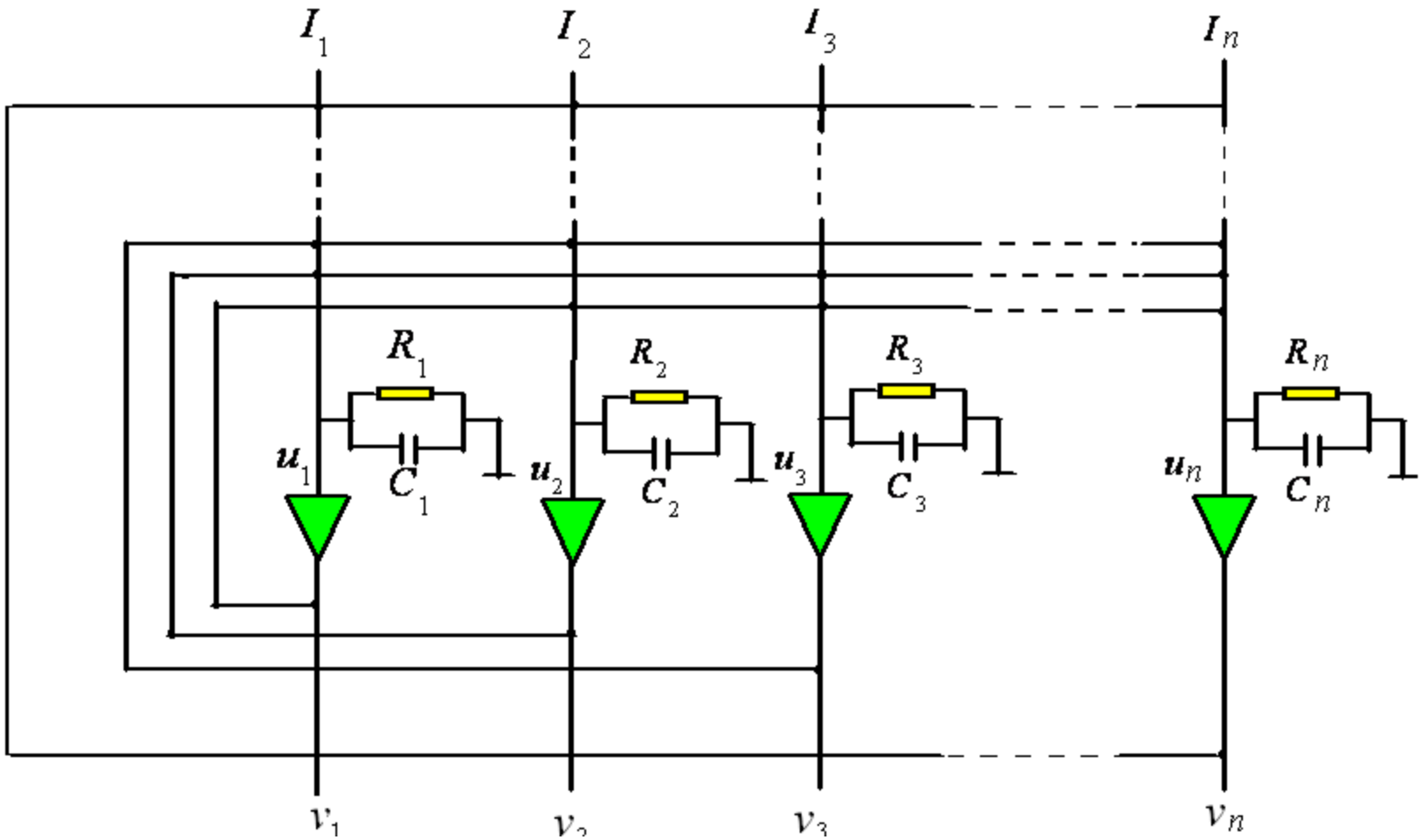
离散Hopfield神经网络

2. 网络的稳定性

- 稳定性定理证明：1983年，科恩（Cohen）、葛劳斯伯格（S. Grossberg）。
- 稳定性定理（Hopfield）
  - 串行稳定性 —— W：对称阵
  - 并行稳定性 —— W：非负定对称阵

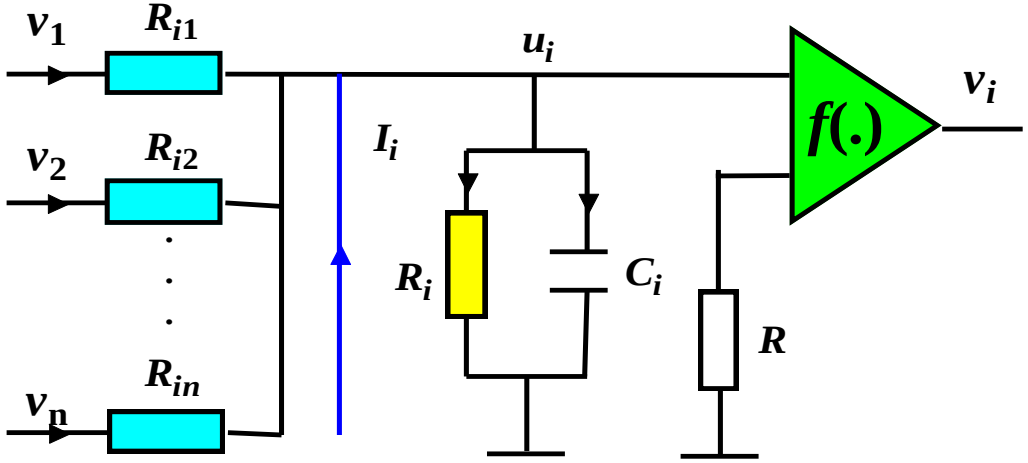
连续型Hopfield神经网络及其VLSI实现

1. 连续Hopfield神经网络模型



连续型Hopfield神经网络及其VLSI实现

1. 连续Hopfield神经网络模型



$$\begin{cases} \dfrac{u_i}{R_i} + C_i \dfrac{du_i}{dt} = \sum_{j=1}^{n} \dfrac{v_j - u_i}{R_{ij}} + I_i \\ v_i = f(u_i) \end{cases}$$

$$\frac{1}{R'_i} = \frac{1}{R_i} + \sum_{j=1}^{n} \frac{1}{R_{ij}}$$

$$C_i \frac{du_i}{dt} = -\frac{u_i}{R'_i} + \sum_{j=1}^{n} w_{ij} v_j + I_i$$

$$w_{ij} = \frac{1}{R_{ij}}$$

$$v_i = f(u_i) = \frac{1}{1 + e^{-\frac{2u_i}{u_0}}} = \frac{1}{2}[1 + \tanh(\frac{u_i}{u_0})]$$

连续型Hopfield神经网络及其VLSI实现

2. 网络的稳定性

- 计算能量函数：

$$E=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}v_iv_j-\sum_{i=1}^{n}v_iI_i+\sum_{i=1}^{n}\frac{1}{R'_i}\int_0^{v_i}f^{-1}(v)dv$$

- 定理：对于连续型 **Hopfield** 神经网络，若 $f^{-1}(\cdot)$ 为单调递增的连续函数，$C_i>0$，$w_{ij}=w_{ji}$，则 $\frac{dE}{dt}\le 0$；当且仅当 $\frac{dv_i}{dt}=0,\quad (1\le i\le n)$时，$\frac{dE}{dt}=0$

Q&A

THANKS!