



河海大学
HOHAI UNIVERSITY

第八章 神经网络基础

目录

Contents

01

人工神经网络的基本概念与发展历史

02

神经元数学模型与激活函数

03

前馈神经网络

04

BP神经网络的应用



- 人工神经网络是对人脑或生物神经网络若干基本特性的抽象和模拟。为机器学习等许多问题的研究提供了一条新的思路，目前已经在模式识别、机器视觉、联想记忆、自动控制、信号处理、软测量、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。
- 本章着重介绍人工神经网络的发展历史及基本概念，神经元模型，以及最基本、最典型、应用最广泛的BP神经网络。



01 / PART ONE

人工神经网络的基本概念与发展历史

传统机器学习模型适用场景：

- 数据量小且特征明确
- 可解释性要求高
- 计算资源有限



嵌入式设备中的轻量级模型

血常规

①先看三大指标有无异常

代号	项目	检验结果	单位	参考值
WBC	白细胞计数	4.90	10 ⁹ /L	4—10
NEUT#	中性粒细胞#	3.31	10 ⁹ /L	2—7.5
LYMPH#	淋巴细胞#	1.30	10 ⁹ /L	0.8—4
MONO#	单核细胞#	0.25	10 ⁹ /L	0.16—1.2
EO#	嗜酸性粒细胞#	0.04	10 ⁹ /L	0.02—0.5
BASO#	嗜碱性粒细胞#	0	10 ⁹ /L	0—0.1
NEUT%	中性粒细胞%	67.64	%	50—75
LYMPH%	淋巴细胞%	26.50	%	20—40
MONO%	单核细胞%	5.10	%	4—12
EO%	嗜酸性粒细胞%	0.84	%	0.5—5
BASO%	嗜碱性粒细胞%	0.04	%	0—1
RBC	红细胞计数	4.1	10 ¹² /L	4—5.5
HGB	血红蛋白	117	g/L	120—160
HCT	红细胞压积	44	%	37—47
MCV	平均红细胞体积	90	fL	80—100
MCH	平均红细胞血红蛋白量	28	pg	27—32
MCHC	平均红细胞血红蛋白浓度	313	g/L	316—348
RDW-SD	红细胞分布宽度SD	41	fL	35—56
RDW-CV	红细胞分布宽度CV	11.1	%	11.6—14.8
PLT	血小板计数	111	10 ⁹ /L	100—400
PDW	血小板分布宽度	11.1	%	9—13
MPV	平均血小板体积	1.1	fL	0.7—1.2
PCT	血小板压积	0.12	%	0.1—0.3
P-LCR	大型血小板比率	3.7	%	0—10

血常规也称

医疗诊断中的指标分类

金融风控模型需解释拒贷原因

可解释性

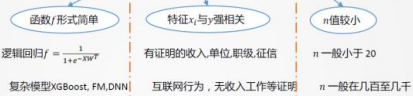
风险模型：

$$P(y = 1|X) = f(x_1, x_2, x_3, \dots, x_n), X \in R^n, y \in \{0,1\}$$

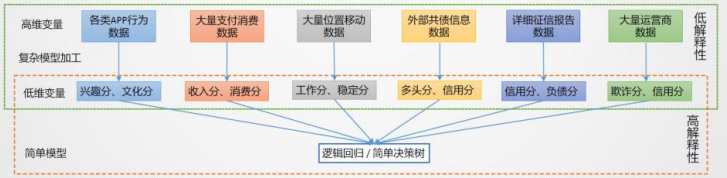
满足可解释性要求：

传统金融模型：

互联网金融模型：



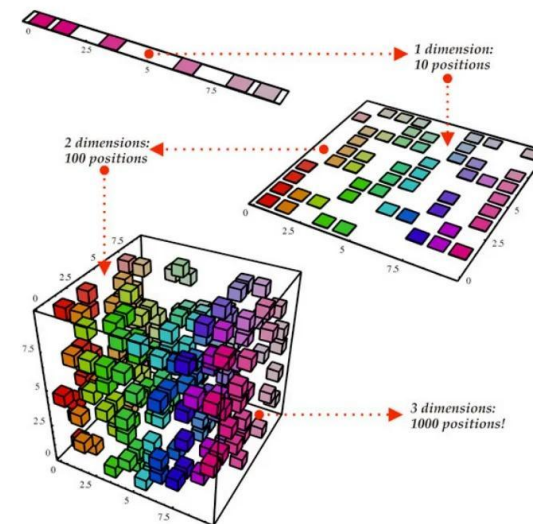
互联网金融模型的可解释性实现



处理复杂问题时是否受限？

传统机器学习模型处理复杂问题时的局限性：

- 模型表达能力的局限性：线性假设与简单结构
 - 线性模型限制（如逻辑回归）
 - 非线性处理瓶颈（如支持向量机，“维度灾难”）
- 特征工程的依赖性：人工设计难以覆盖复杂交互
 - 手动特征提取的低效性
 - 海量特征问题（基因数据、高光谱图像）
- 计算效率与数据规模的不匹配
 - 大规模数据下的性能瓶颈

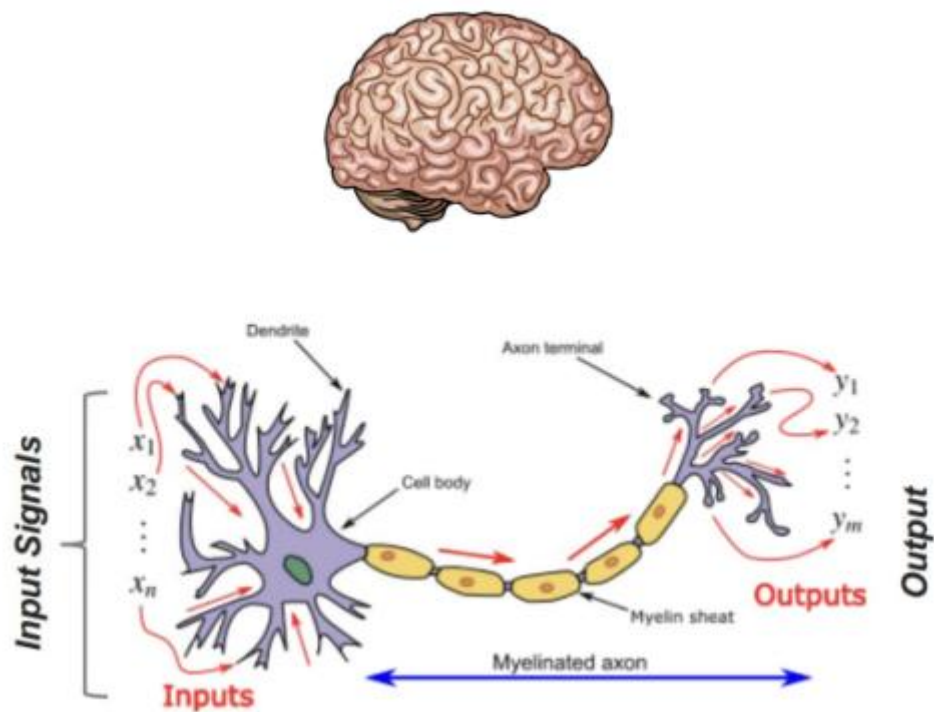


- 泛化能力与复杂模式的矛盾
 - 模型复杂度与泛化的权衡困难
 - 对非结构化数据的处理不足
- 端到端学习能力的缺失

使用传统机器学习模型（线性回归，逻辑回归等）在遇到复杂问题时性能堪忧。

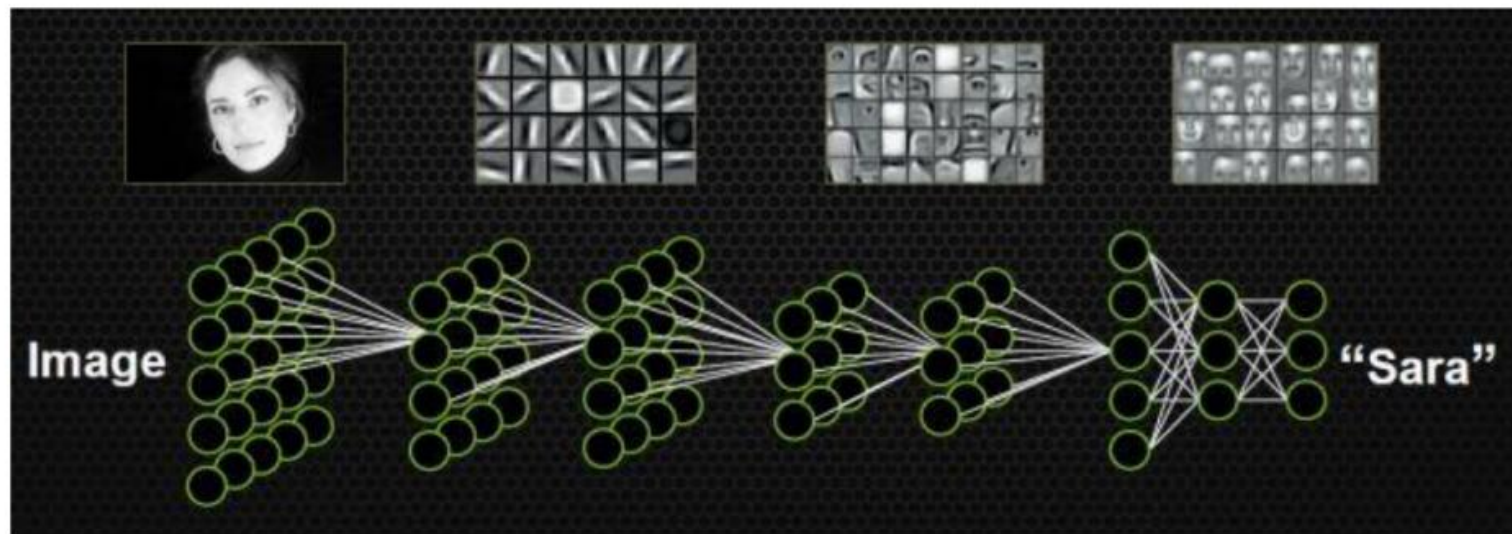
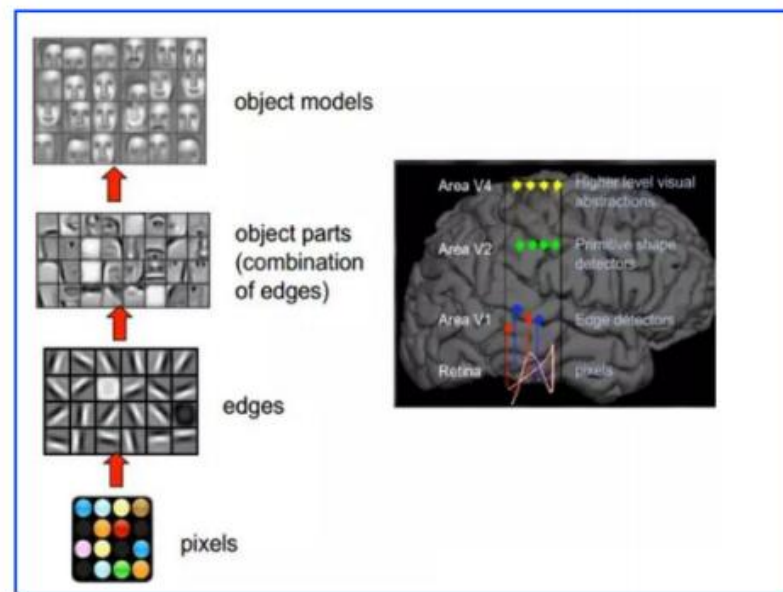


设想：借鉴仿生学，让计算机模拟人类大脑信息处理的方式。



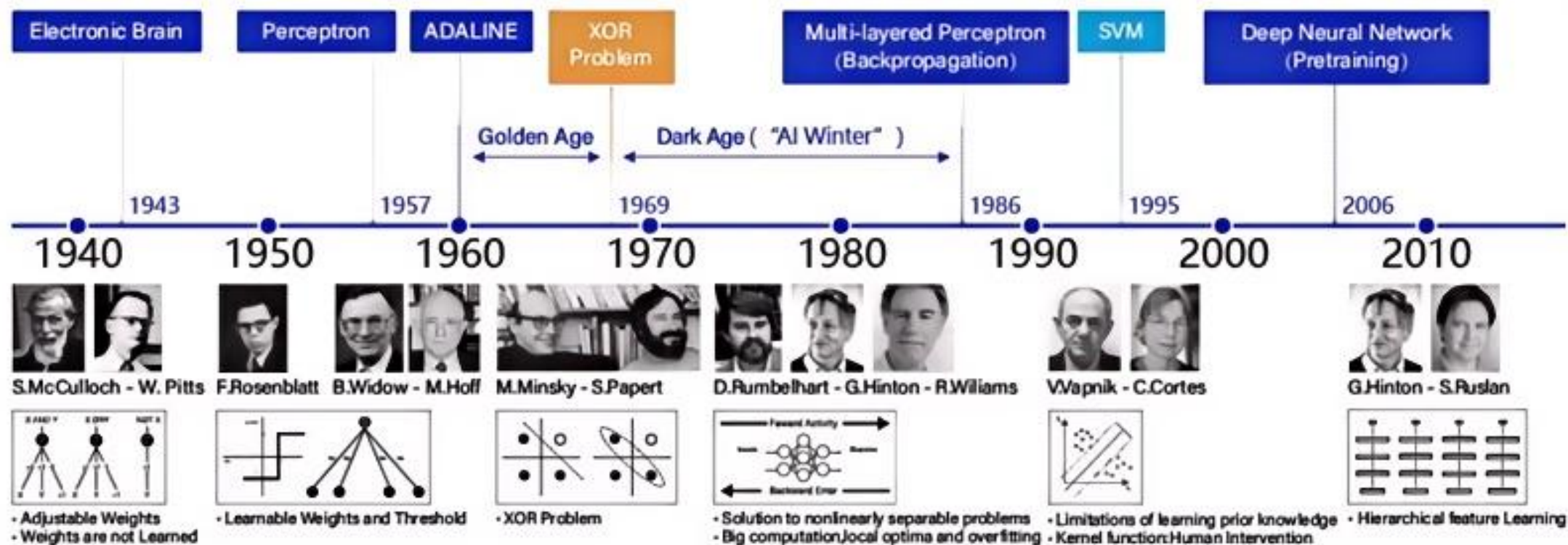
● 人类识别人脸的机理

1. 从原始信号摄入 (瞳孔摄入像素 Pixels)
2. 初步处理 (大脑皮层某些细胞发现边缘和方向)
3. 抽象 (大脑判定眼前的物体轮廓, 是圆形的)
4. 进一步抽象 (大脑进一步判定该物体是人脸)

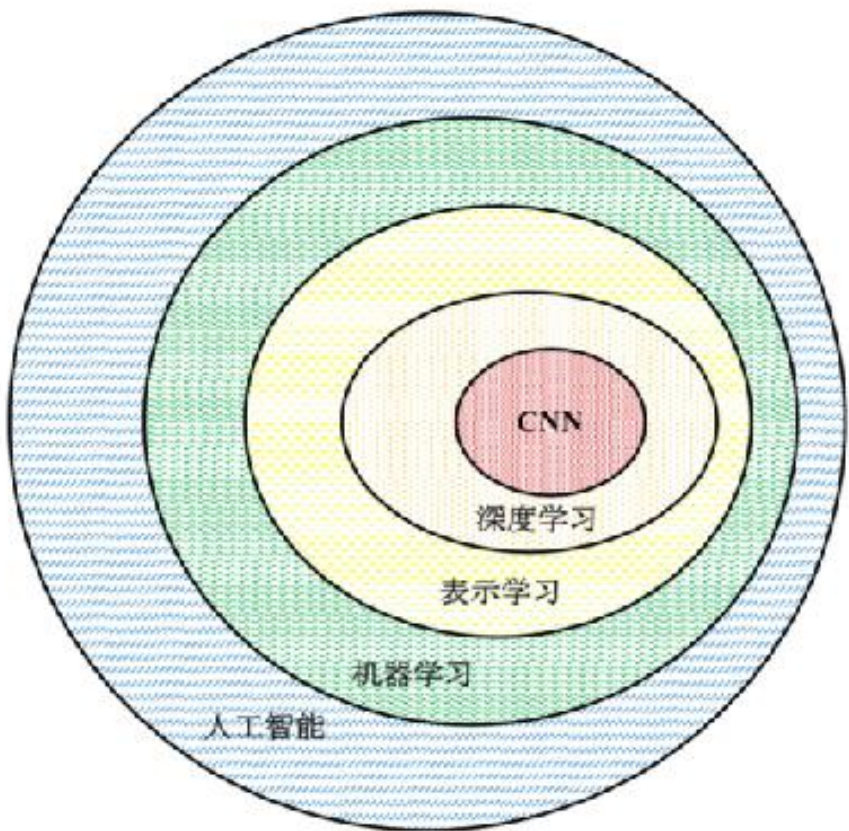


- **生物神经网络**(natural neural network, NNN): 由中枢神经系统（脑和脊髓）及周围神经系统（感觉神经、运动神经等）所构成的错综复杂的神经网络，其中最重要的是**脑神经系统**。
- **人工神经网络**(artificial neural networks, ANN): 模拟**人脑神经系统的**结构和功能，运用大量简单处理单元经广泛连接而组成的人工网络系统。

神经网络方法：**隐式的**
知识表示方法



深度学习焕发生机的三大原因：大数据涌现、硬件发展和算法优化。



深度学习是机器学习的子集：

- **机器学习**研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。
- **深度学习**通过多层神经网络结构（如 CNN、RNN、Transformer 等）实现对数据的自动特征提取和复杂模式建模。

- “2:0! Dota2世界冠军OG, 被OpenAI按在地上摩擦”
- “《星际争霸2人类1:10输给AI! DeepMind “AlphaStar”进化神速》”
- “柯洁:感谢AlphaGo击败我 我现在看到AI也比较痛苦”



□ 2007年台北国际发明暨技术交易展览上，第三代智能机器人DOC现场表演下棋。



□ 2016年围棋李世石大战ALPHAGO





百度 AI 实力担当
百度人工智能的诚意之作，好用、好听、好玩





图像检索

无人超市



智慧城市——安全帽检测



在施工现场，现场往来和工作人员的安全问题极为重要，安全帽是保障现场人员安全的第一道防线。

智慧城市



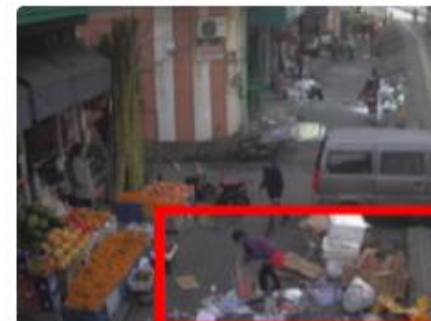
跨门店经营



沿街曝晒



违章停车



暴露垃圾



无照游商

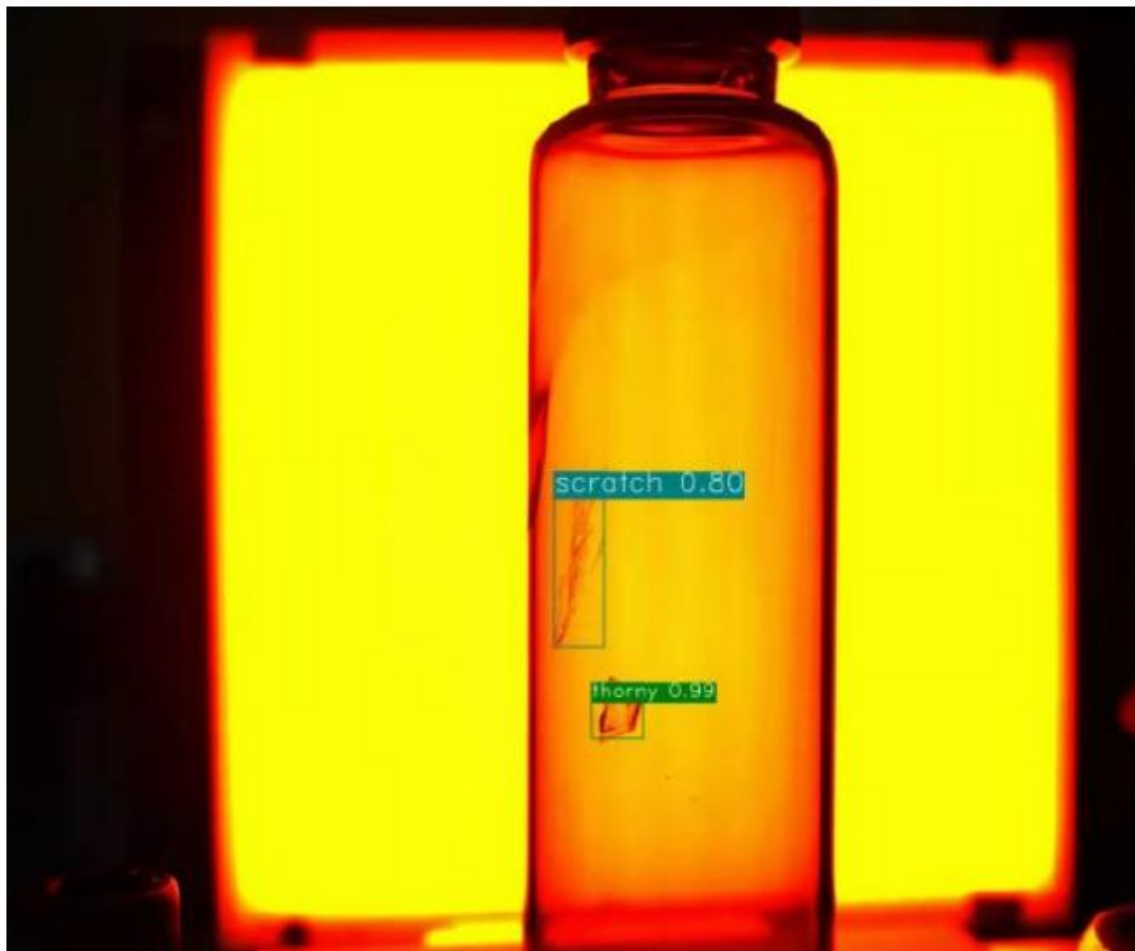


堆物堆料



积存垃圾渣土

智能制造——医用药品表面缺陷检测

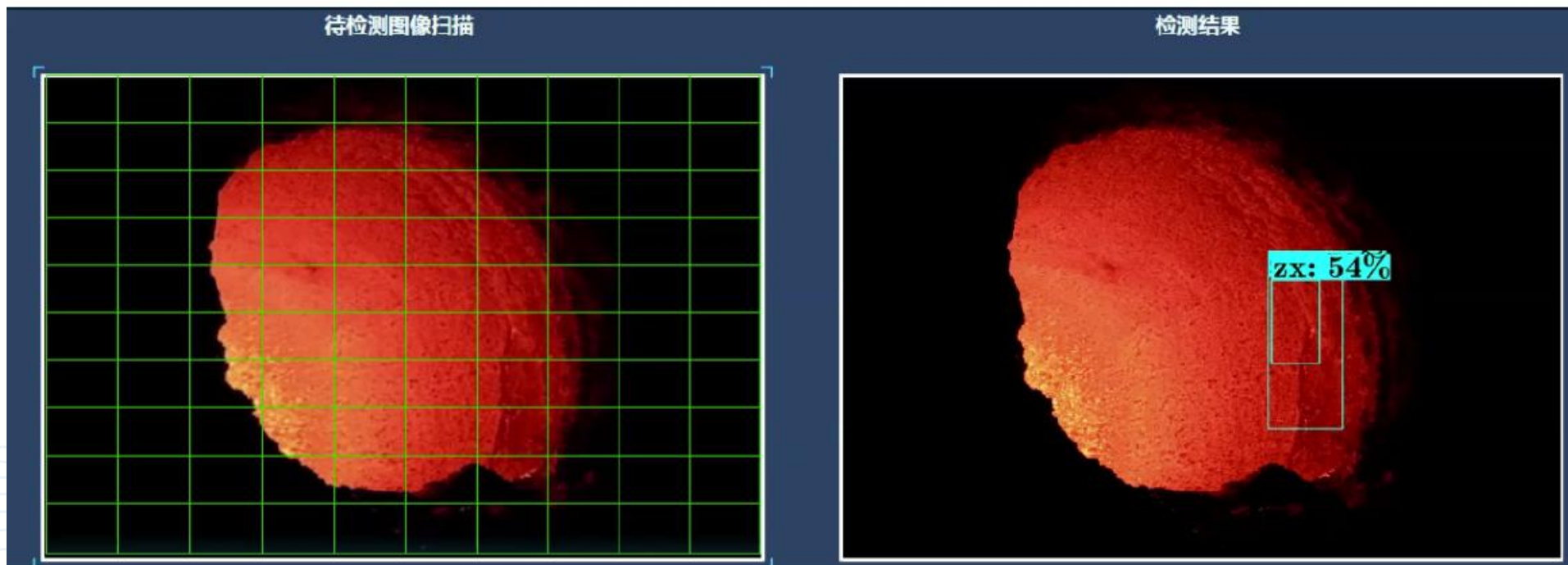


作为生产制造过程中必不可少的一步，物体表面缺陷检测广泛应用于工业领域，如：3C、半导体、电子、汽车、化工、医药、轻工、军工等行业。而医用药瓶作为医药行业基础的包装材料，在出厂之前的检测更加严格。

AI+质检

基于全景钢包内衬图片进行内衬无死角熔损评估

基于人工智能钢包内衬识别模型识别标准客观、
准确率持续提升

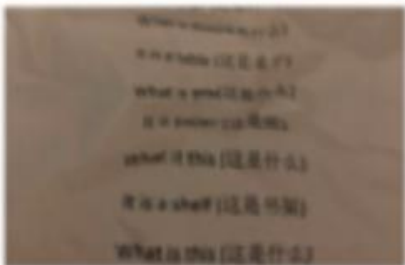


智慧金融——研报识别与关键字段抽取

表格



褶皱



手写



盖章



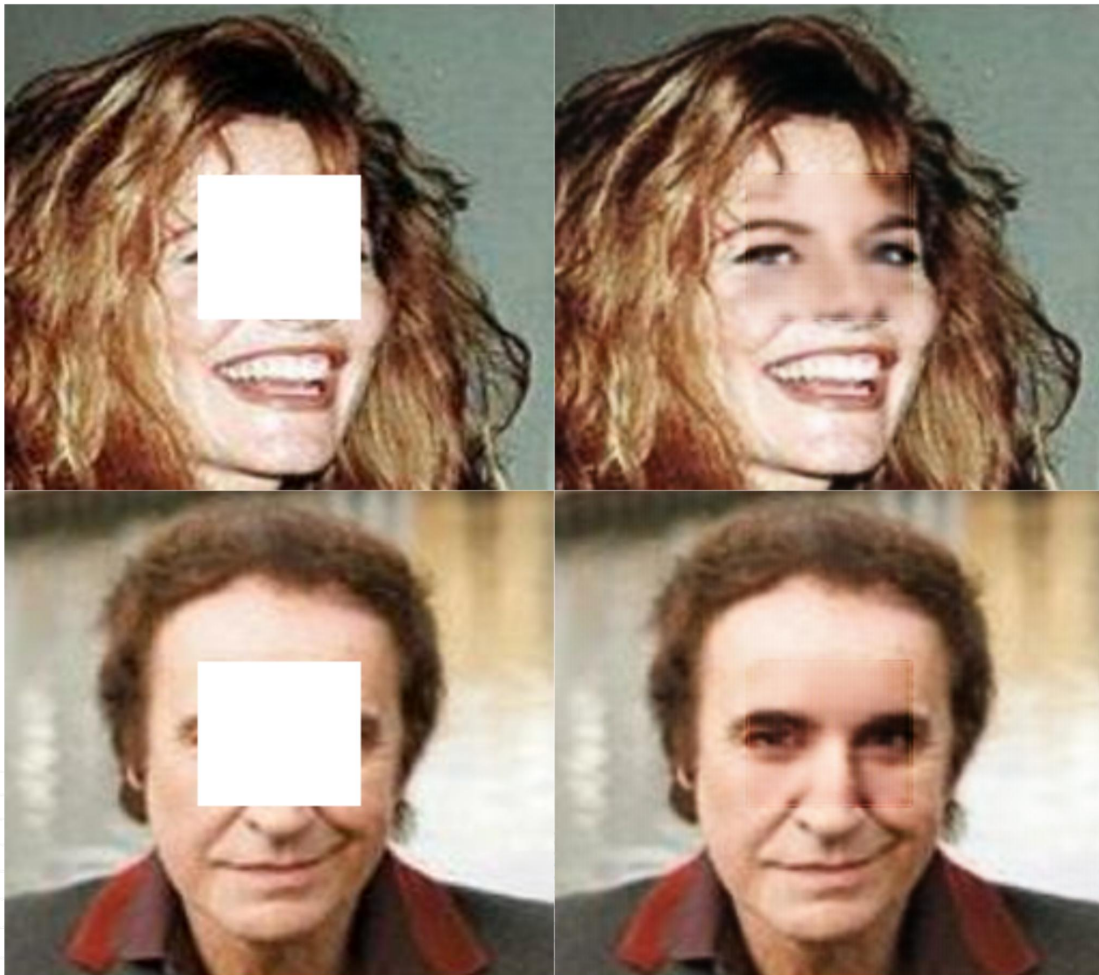
当前，诸多投资机构都通过研报的形式给出对于股票、基金以及行业的判断，让大众了解热点方向、龙头公司等各类信息。然而，分析和学习研报往往花费大量时间，研报数目的与日俱增也使得自动对研报分析和信息抽取的诉求不断提高。

泛交通——人流量统计



在地铁站、火车站、机场、展馆、景区等公共场所，需要实时检测人流数量，人流密度过高时及时预警，并实施导流、限流等措施，防止安全隐患。

图像修复



图像风格迁移



新闻播报机器人

- 2018年11月7-9日，在乌镇第五届世界互联网大会上，新华社对外宣布：中国首个“人工智能主持人”正式上岗。





“从北京冬奥会开始，我将全年无休用AI智慧为听障用户提供手语服务，让他们快捷的获取比赛资讯”，央视新闻联合百度智能云倾心打造的总台首个AI手语主播正式亮相，让“数字人”进入了大众的视野。

2019.2.19新华社发布站立式合成主播

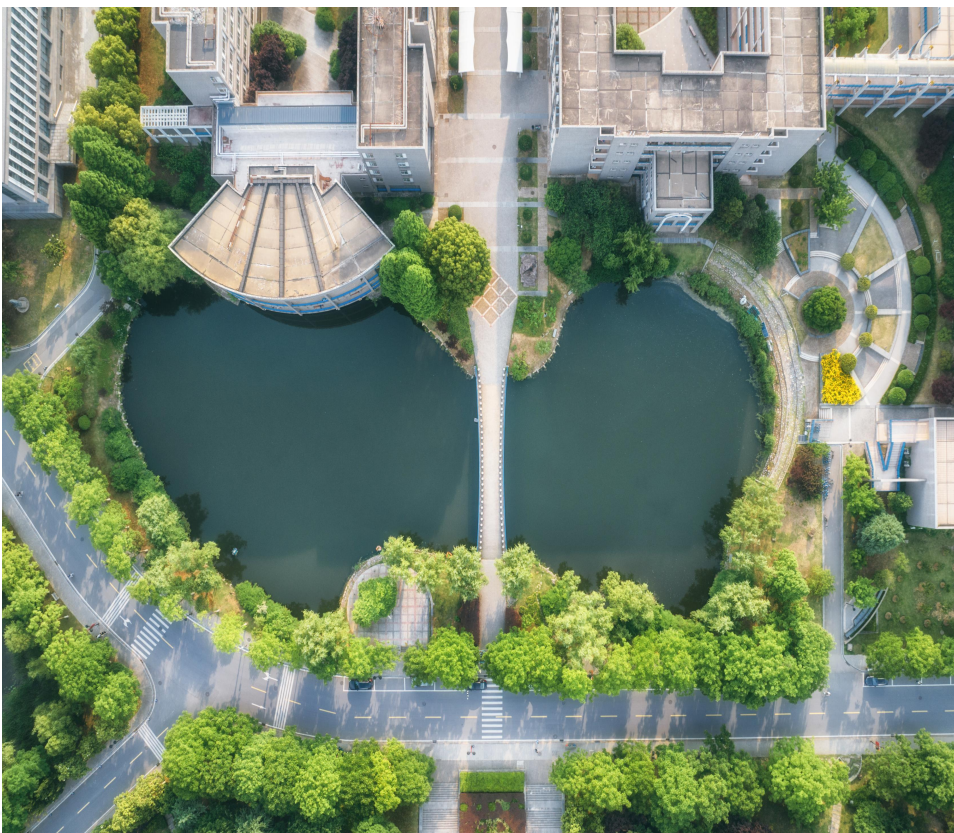


邱浩——新小浩

屈萌——新小萌

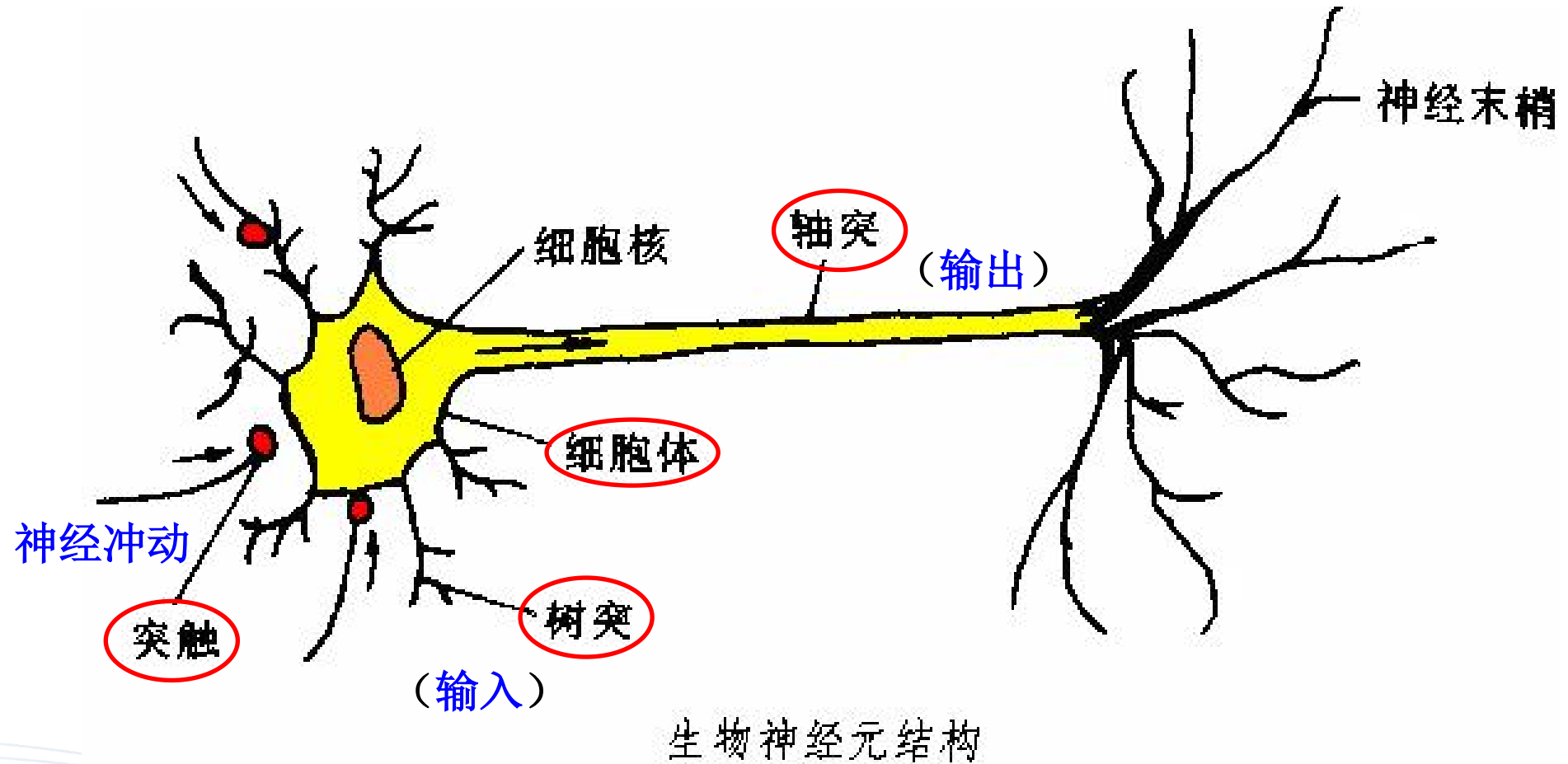
02 / PART ONE

神经元数学模型与激活函数

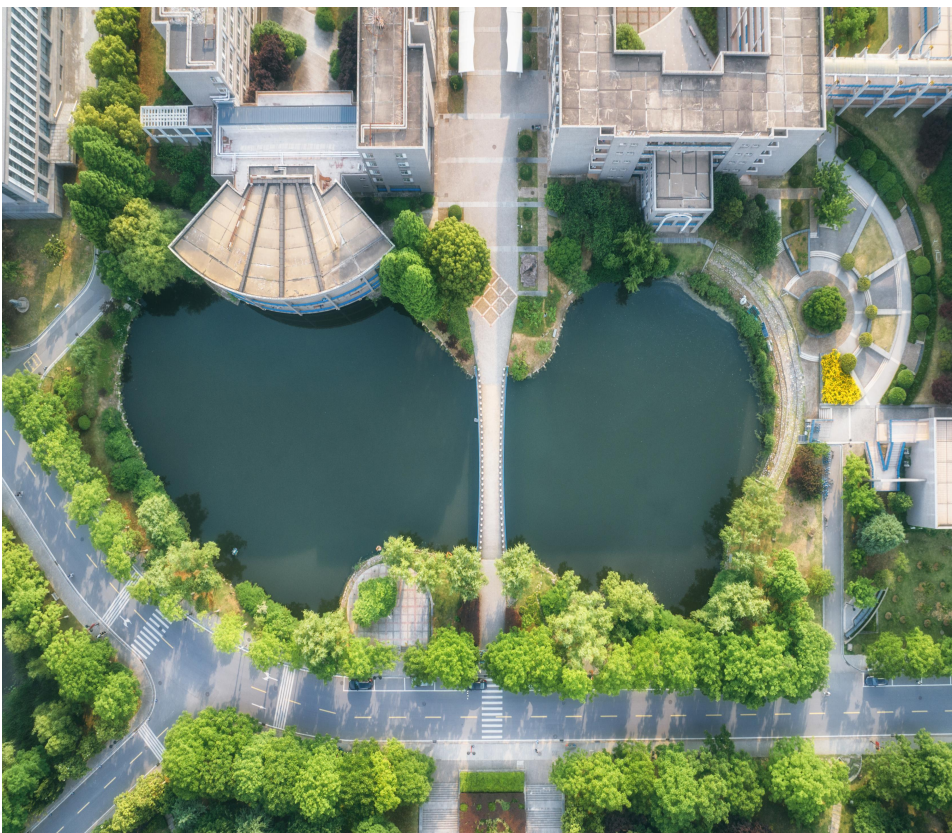


- 人脑由一千多亿（ 10^{11} 亿— 10^{14} 亿）个神经细胞（神经元）交织在一起的网状结构组成，其中大脑皮层约140亿个神经元，小脑皮层约1000亿个神经元。
- 神经元约有1000种类型，每个神经元大约与 10^3 — 10^4 个其他神经元相连接，形成极为错综复杂而又灵活多变的神经网络。
- 人的智能行为就是由如此高度复杂的组织产生的。浩瀚的宇宙中，也许只有包含数千亿颗星球的银河系的复杂性能与大脑相比。





神经细胞利用电-化学过程交换信号



- 工作状态：
 - 兴奋状态：细胞膜电位 $>$ 动作电位的阈值 $\rightarrow$ 神经冲动
 - 抑制状态：细胞膜电位 $<$ 动作电位的阈值
- 学习与遗忘：由于神经元结构的可塑性，突触的传递作用可增强和减弱。





MP神经元原始模型

人工神经元是对生物神经元的抽象与模拟。所谓抽象是从数学角度而言的，所谓模拟是从其结构和功能角度而言的。

MP模型是美国心理学家麦克洛奇 (W. McCulloch) 和数理逻辑学家皮茨 (W. Pitts) 根据生物神经元的功能和结构，于1943年提出的一种将神经元看作二进制阈值元件的简单模型。即MP神经元模型 **实际上是对单个神经元的一种建模**。

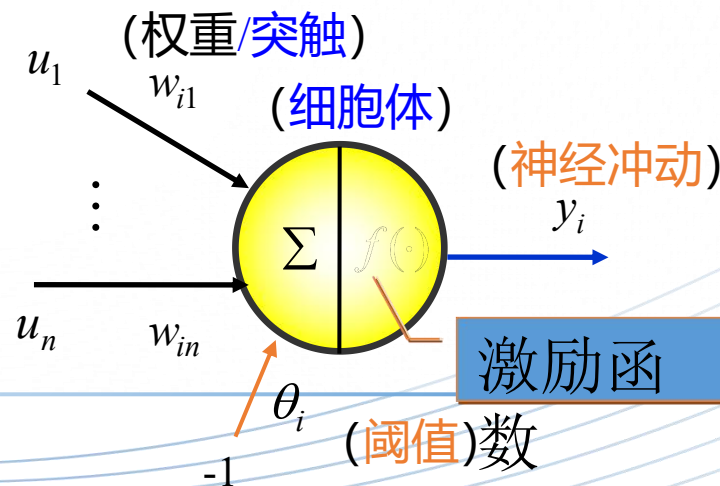
图中的 $x_1, x_2, \dots, x_n$ 表示某一神经元的 n 个输入； w_i 表示第 i 个输入的连接强度，称为连接权值； θ 为神经元的阈值； y 为神经元的输出。

人工神经元是一个多输入、单输出的非线性器件。其输入为：

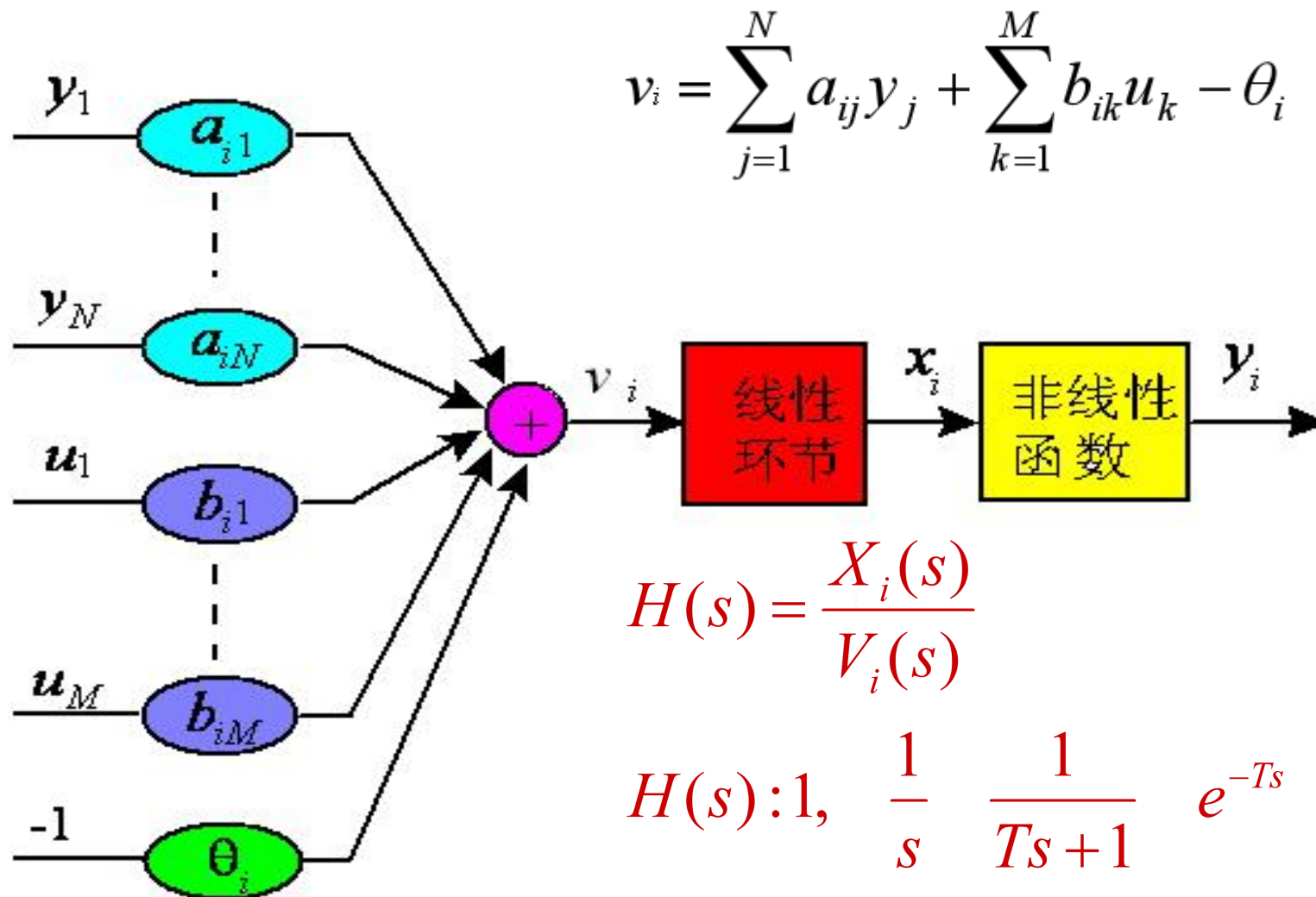
$$\sum_{i=1}^n w_i x_i$$

其输出为：

$$y = f(\sigma) = f\left(\sum_{i=1}^n w_i x_i - \theta\right)$$



一般模型



$y_i(t)$: 第 i 个神经元的输出。

θ_i : 第 i 个神经元的阈值。

$u_k(t) (k = 1, 2, \dots, M)$: 外部输入。

a_{ij}, b_{ik} : 权值。

■ 加权求和:

$$v_i(t) = \sum_{j=1}^N a_{ij} y_j(t) + \sum_{k=1}^M b_{ik} u_k(t) - \theta_i$$

其矩阵形式: $V(t) = AY(t) + BU(t) - \theta$

$$A = \{a_{ij}\}_{N \times N}$$

$$B = \{b_{ik}\}_{N \times M}$$

$$V = [v_1 \quad \cdots \quad v_N]^T$$

$$U = [u_1 \quad \cdots \quad u_M]^T$$

$$\theta = [\theta_1 \quad \cdots \quad \theta_N]^T$$

$$Y = [y_1 \quad \cdots \quad y_N]^T$$

■ 工作过程:

- 从各输入端接收输入信号 u_j ($j = 1, 2, \dots, n$)

- 根据连接权值求出所有输入的加权和

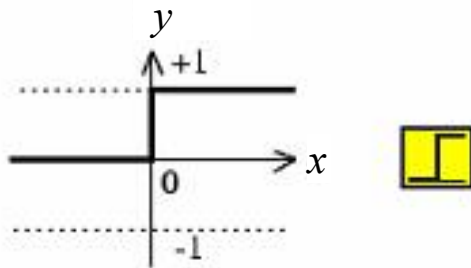
$$x_i = \sum_{j=1}^n w_{ij} u_j - \theta_i = \sum_{j=1}^n w_{ij} u_j + b_i = \sum_{j=0}^n w_{ij} u_j$$

$$(w_{i0} = -1, u_0 = \theta_i \text{ 或 } w_{i0} = 1, u_0 = b_i)$$

- 用非线性激励函数进行转换，得到输出

$$y_i = f(x_i) = f\left(\sum_{j=0}^n w_{ij} u_j\right)$$

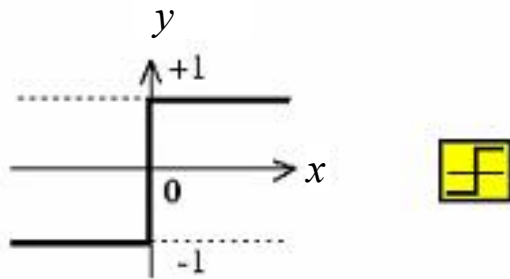
■ 非线性激励函数（传输函数、输出变换函数）



$$y = \text{hardlim}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Hard-Limit Transfer Function

（硬极限函数或阶跃函数）



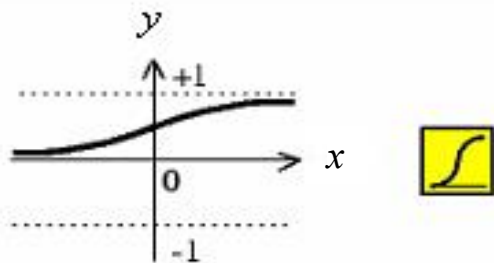
$$y = \text{hardlims}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

Symmetric Hard-Limit Trans. Funct.

（对称硬极限函数）

- 可以将神经元输入值与阈值的差值映射为输出值1或0。若差值大于等于零则输出1，对应兴奋；若差值小于零则输出0，对应抑制。
- 存在问题：
 - 不光滑、不连续，无法“连续可导”

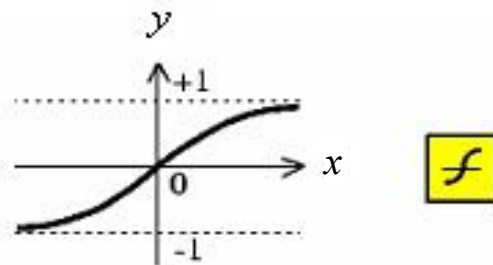
■ 非线性激励函数（传输函数、输出变换函数）



$$y = \text{logsig}(x) = \frac{1}{1 + e^{-\alpha x}}$$

$\alpha = 1$

Log-Sigmoid Transfer Function



$$y = \text{tansig}(x) = \frac{e^{\alpha x} - e^{-\alpha x}}{e^{\alpha x} + e^{-\alpha x}}$$

$\alpha = 1$

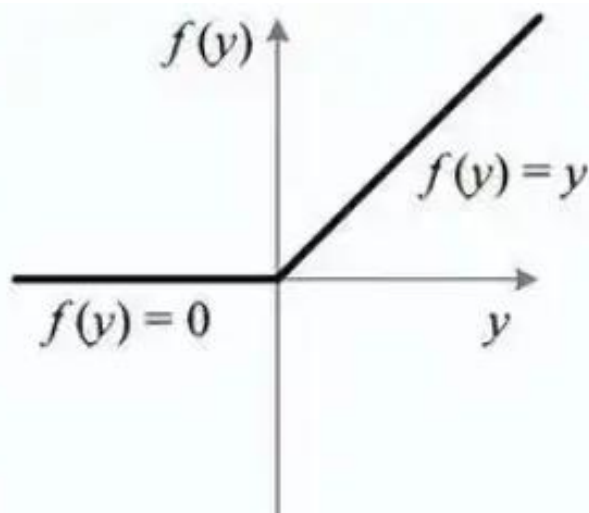
Tan-Sigmoid Transfer Function

（对数-S 形函数或S型函数）

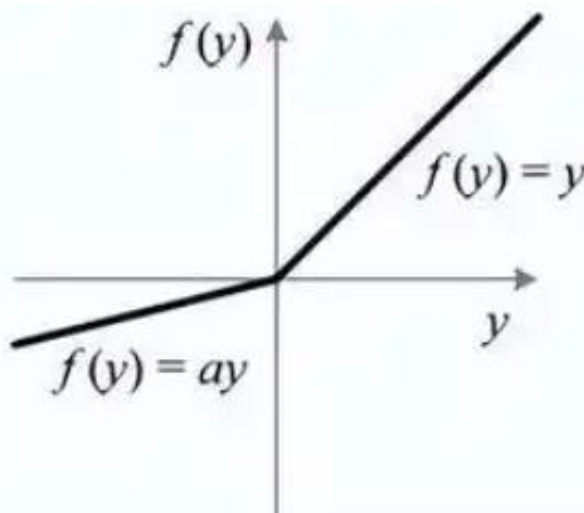
（双曲正切S形函数）

- 无论输入值的范围有多大，这个函数都可以将输出挤压在范围（0,1）之内
- 存在问题：
 - 在趋近0/1附近易梯度消失
 - 计算成本高昂（相比于其他激活函数）

■ 非线性激励函数（传输函数、输出变换函数）



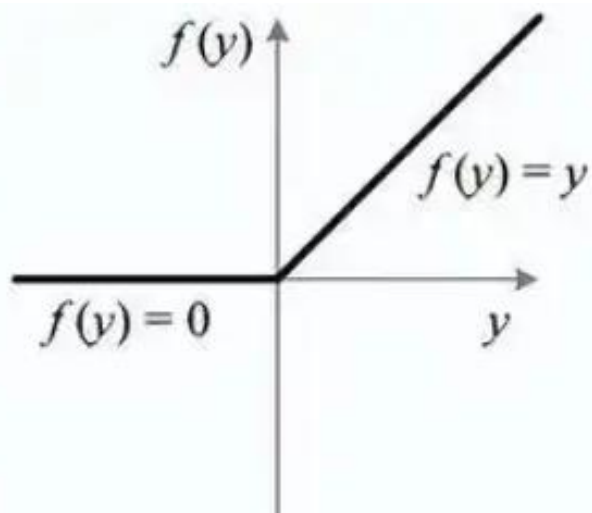
（ReLU函数）



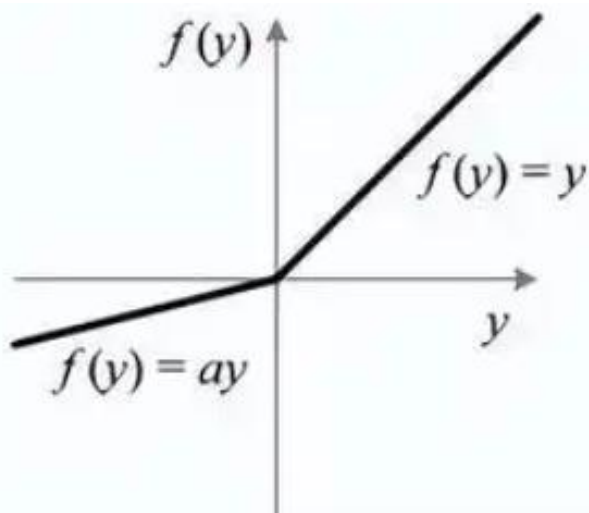
（LeakyReLU函数）

- ReLU是目前较为流行的激活函数之一。它的图像为一条斜率为1的直线和x轴的组合。改善了梯度消失问题，且具有计算速度快的优点。
- 存在问题：
 - 当输入为负时，ReLU函数会完全失效，导致反向传播时梯度为零，称为"DeadReLU"问题。

■ 非线性激励函数（传输函数、输出变换函数）



（ReLU函数）



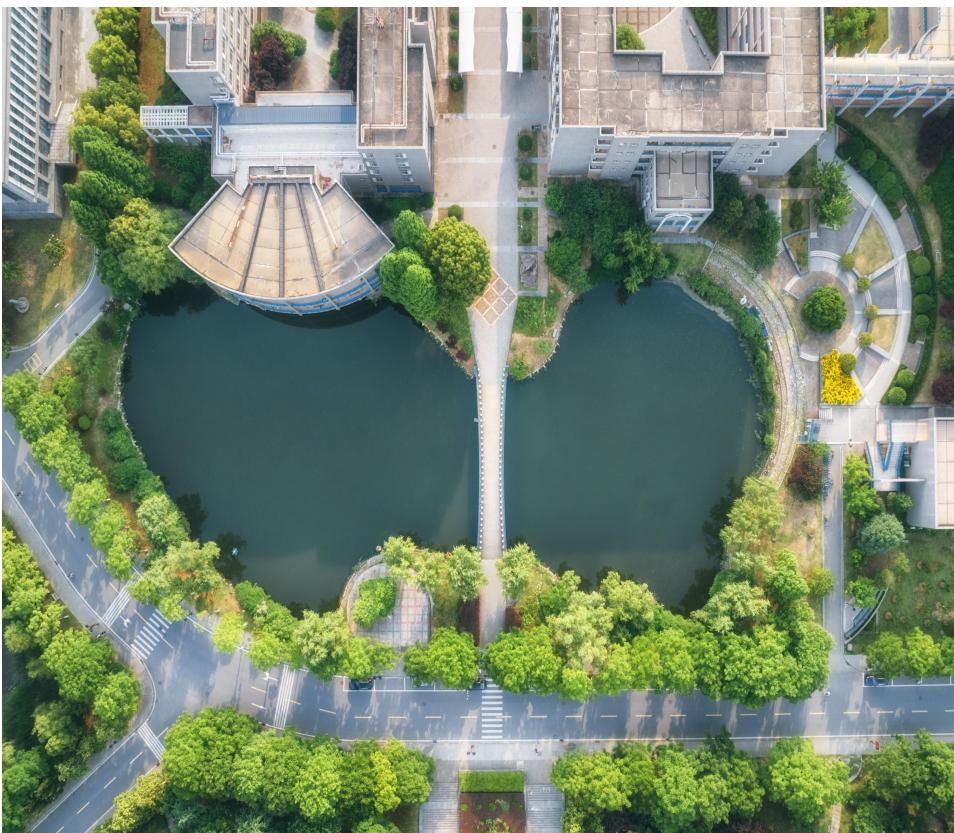
（LeakyReLU函数）

- LeakyReLU是对ReLU函数的改进版本。在负值域加入了一个较小的线性分量，使得负输入的梯度不再为零。在负值区域斜率较小，这也可以避免DeadReLU问题。
- 存在问题：
 - 实践中尚未完全证明LeakyReLU总是比ReLU更好。



03 / PART ONE

前 馈 神 经 网 络



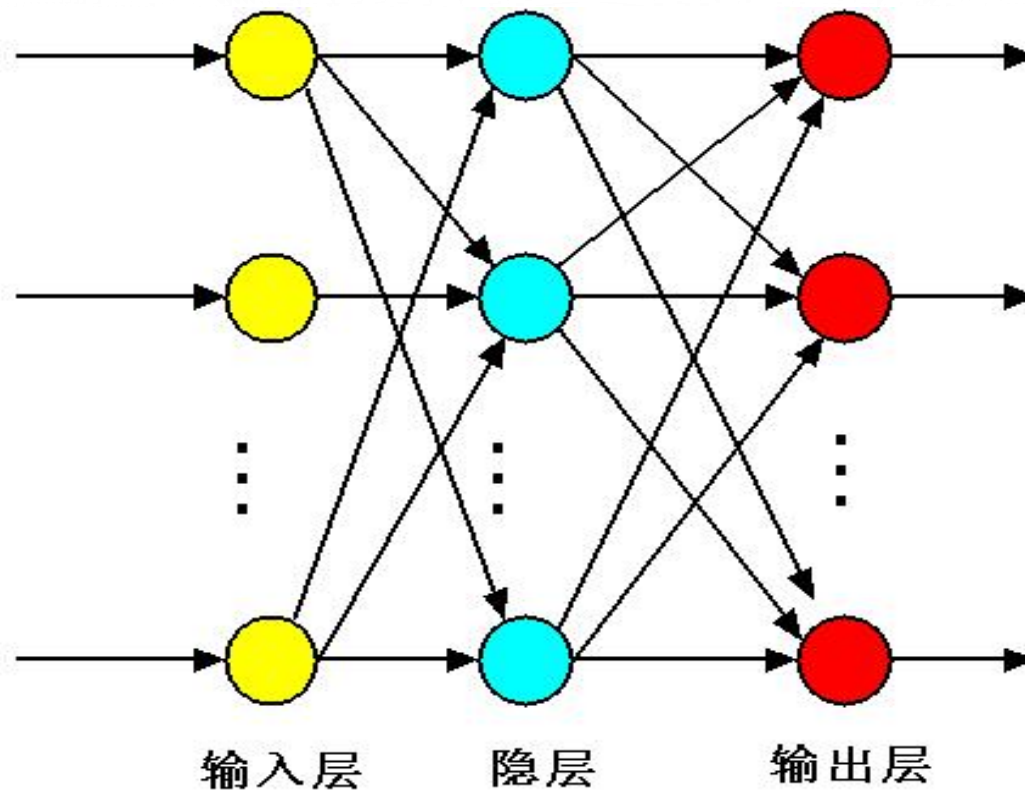
■ 决定人工神经网络性能的三大要素：

- 神经元的特性。
- 神经元之间相互连接的形式——拓扑结构。
- 为适应环境而改善性能的学习规则。



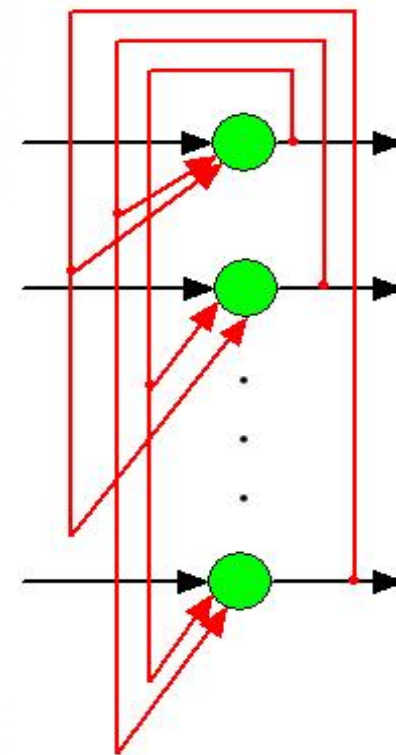
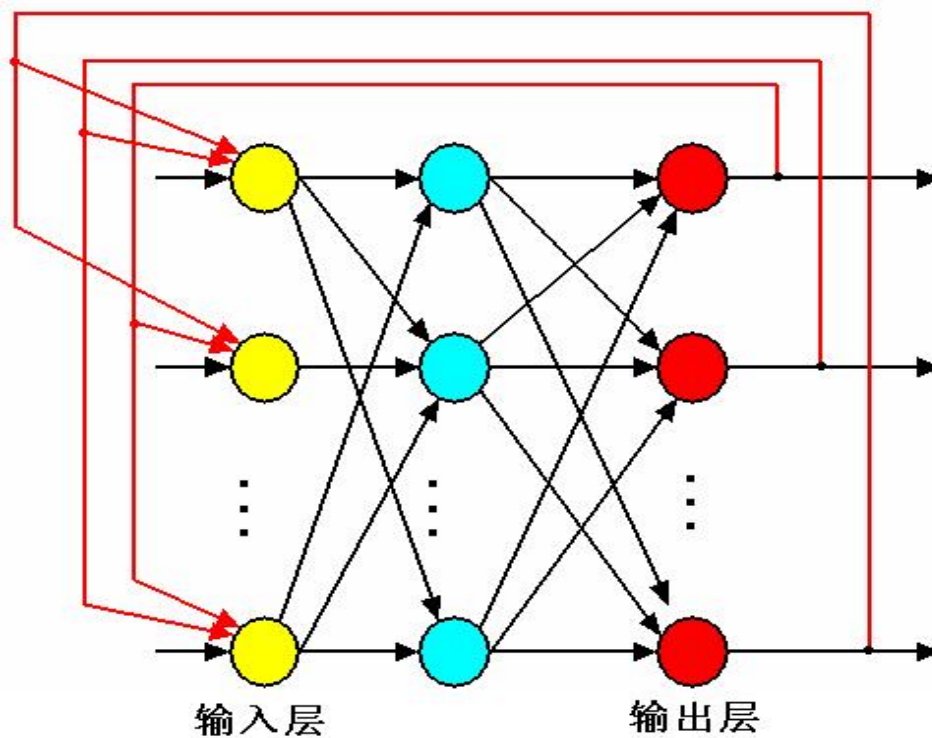


前馈型（前向型）

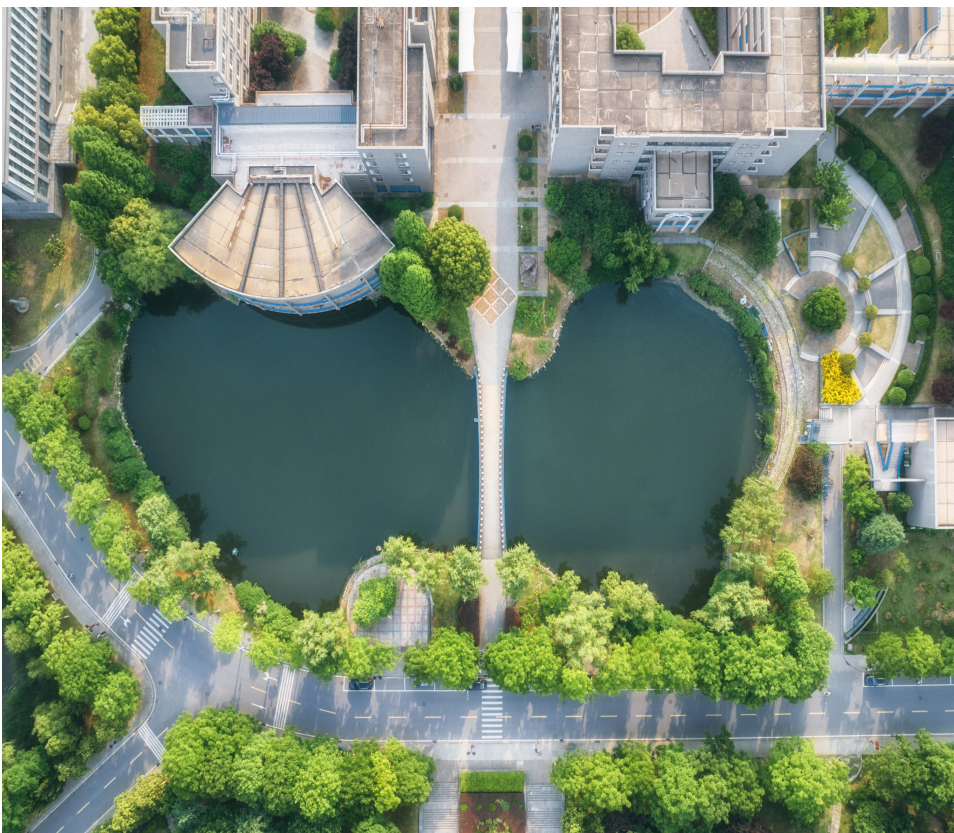




反馈型



(Hopfield神经网络)



- **同步**（并行）方式：任一时刻神经网络中所有神经元同时调整状态。
- **异步**（串行）方式：任一时刻只有一个神经元调整状态，而其它神经元的状态保持不变。



神经网络的学习

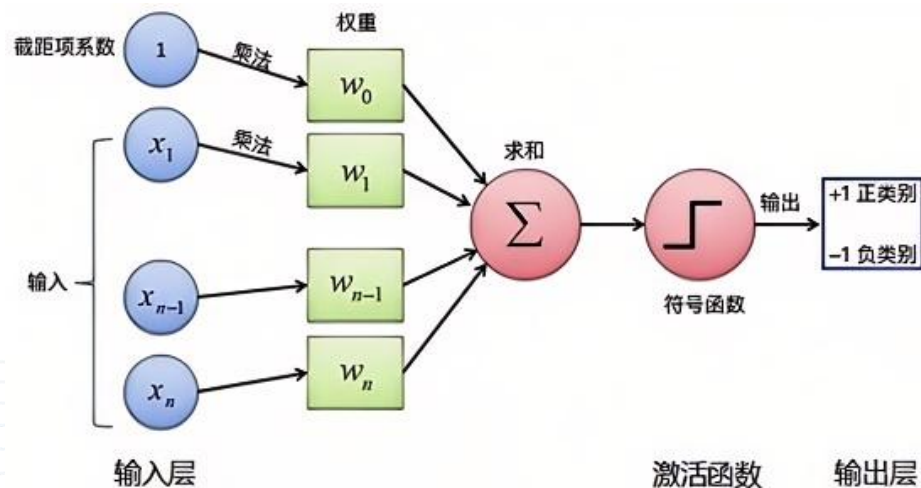
**神经网络方法是一种知识表示方法和推理方法。
神经网络知识表示是一种隐式的表示方法。**

**1944年赫布（Hebb）提出了改变神经元连接强度的
Hebb学习规则。**

Hebb学习规则：当某一突触两端的神经元同时处于兴奋状态，那么该连接的权值应该增强。

1. 单层感知机

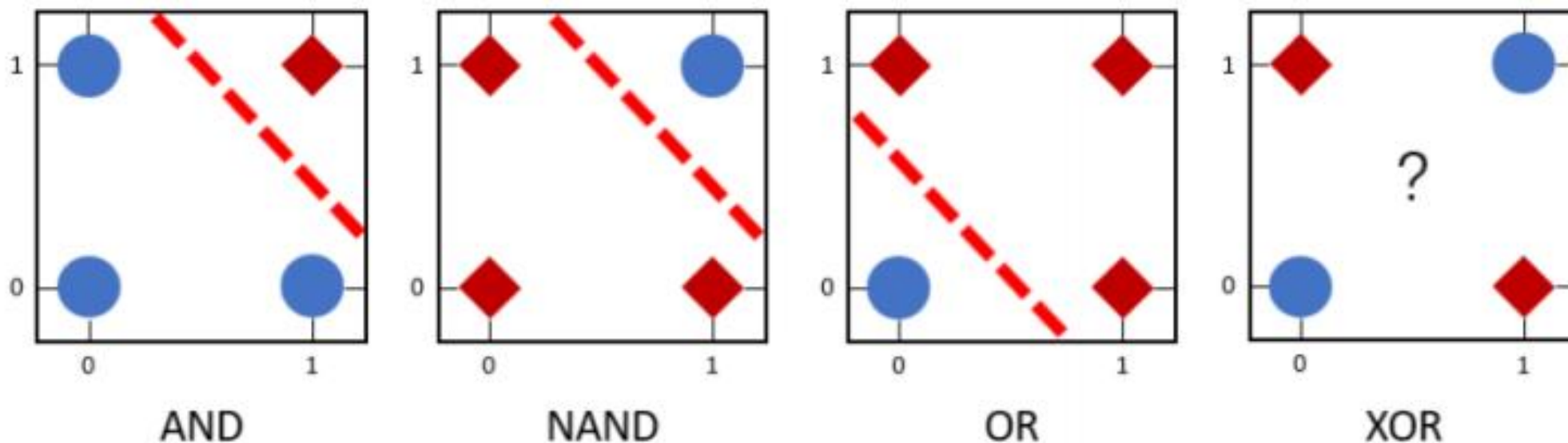
- 感知机是一种判别模型，其目标是求得一个能够将数据集中的正实例点和负实例点完全分开的分离超平面。
- 早期的感知机结构由一个输入层和一个输出层构成，因此也被称为“单层感知机”。



弗兰克·罗森布拉特博士在“感知机”上工作

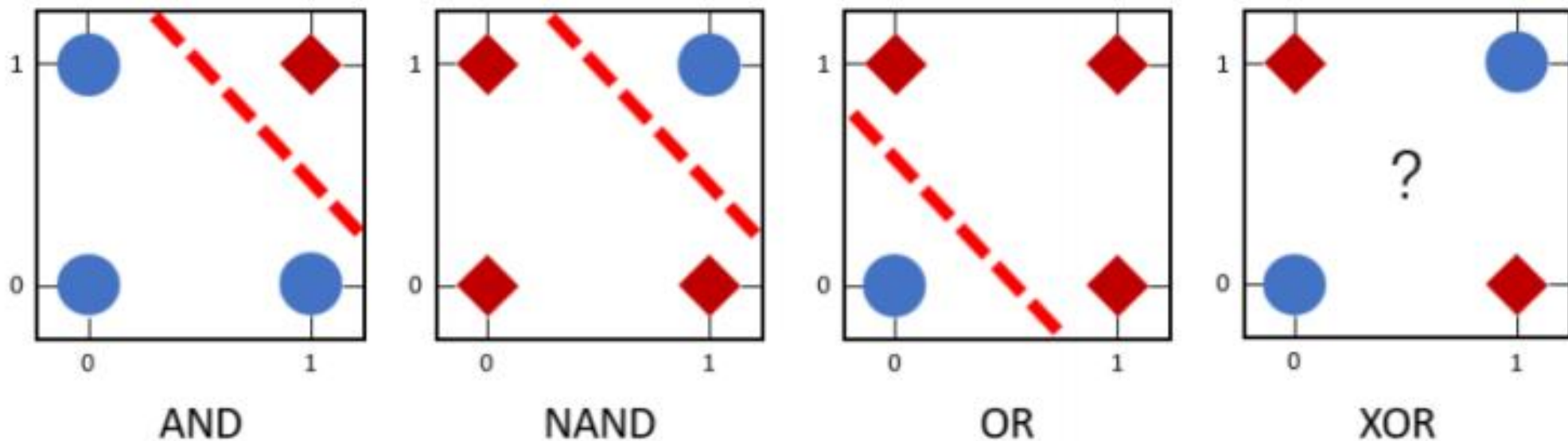
1. 单层感知机

- 感知机是一种判别模型，其目标是求得一个能够将数据集中的正实例点和负实例点完全分开的分离超平面。
- 早期的感知机结构由一个输入层和一个输出层构成，因此也被称为“单层感知机”。
- 单层感知机可被用来区分线性可分数据如与、与非、或等逻辑函数。但是无法模拟非线性可分的逻辑异或函数。



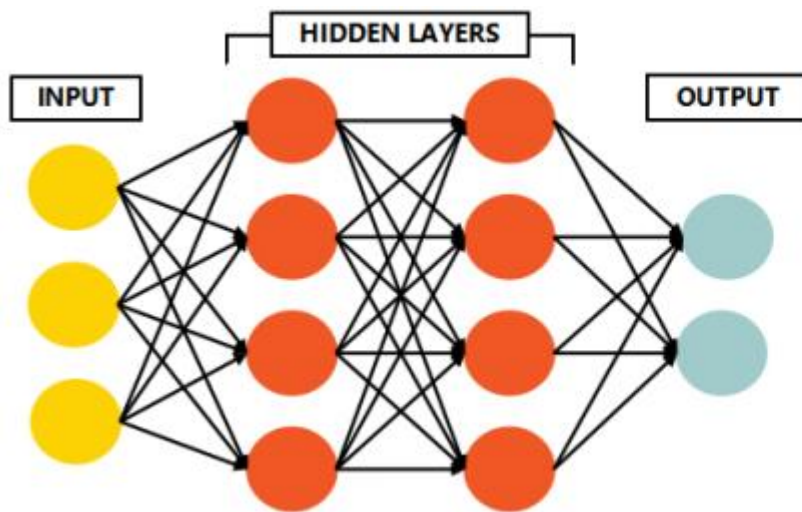
1. 单层感知机

- 感知机是一种判别模型，其目标是求得一个能够将数据集中的正实例点和负实例点完全分开的分离超平面。
- 早期的感知机结构由一个输入层和一个输出层构成，因此也被称为“单层感知机”。
- 单层感知机可被用来区分线性可分数据如与、与非、或等逻辑函数。但是无法模拟非线性可分的逻辑异或函数。



2. 多层感知机

- 多层感知机由输入层、输出层和至少一层的隐藏层构成。各个隐藏层中神经元可接收相邻前序隐藏层中所有神经元传递而来的信息，经过加权激活后将信息输出给相邻后续层中的所有神经元。
- 层与层之间通过“全连接”进行链接，即两个相邻层之间的神经元完全成对连接，但层内的神经元不相互连接。整个模型中没有反馈。

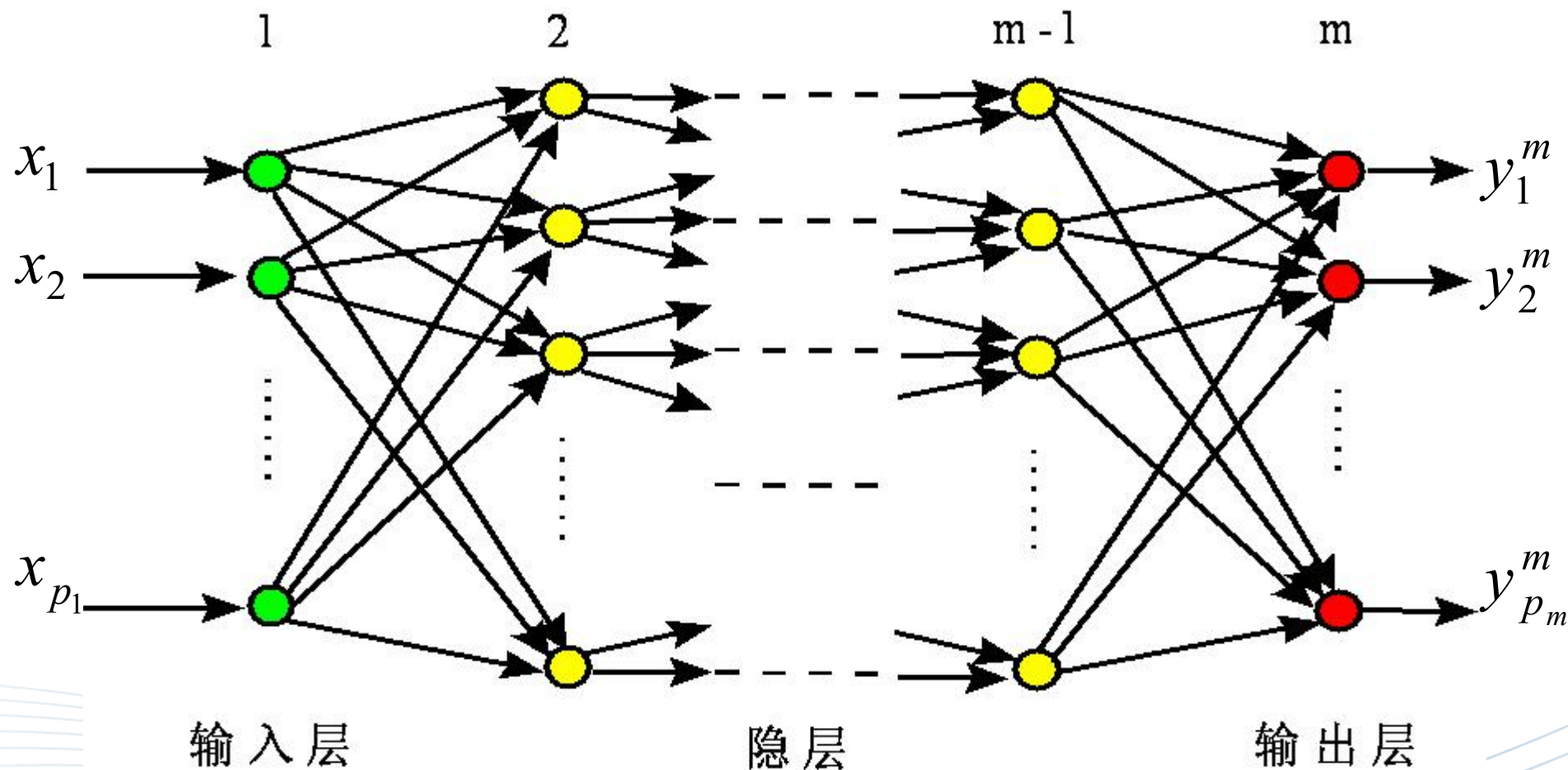


$$\text{Hidden layer1: } a^{(1)} = \sigma(W^{(0)}a^{(0)} + b^{(0)})$$

$$\text{Hidden layer2: } a^{(2)} = \sigma(W^{(1)}a^{(1)} + b^{(1)})$$

$$\text{Output: } a^{(3)} = \text{softmax}(W^{(2)}a^{(2)} + b^{(2)})$$

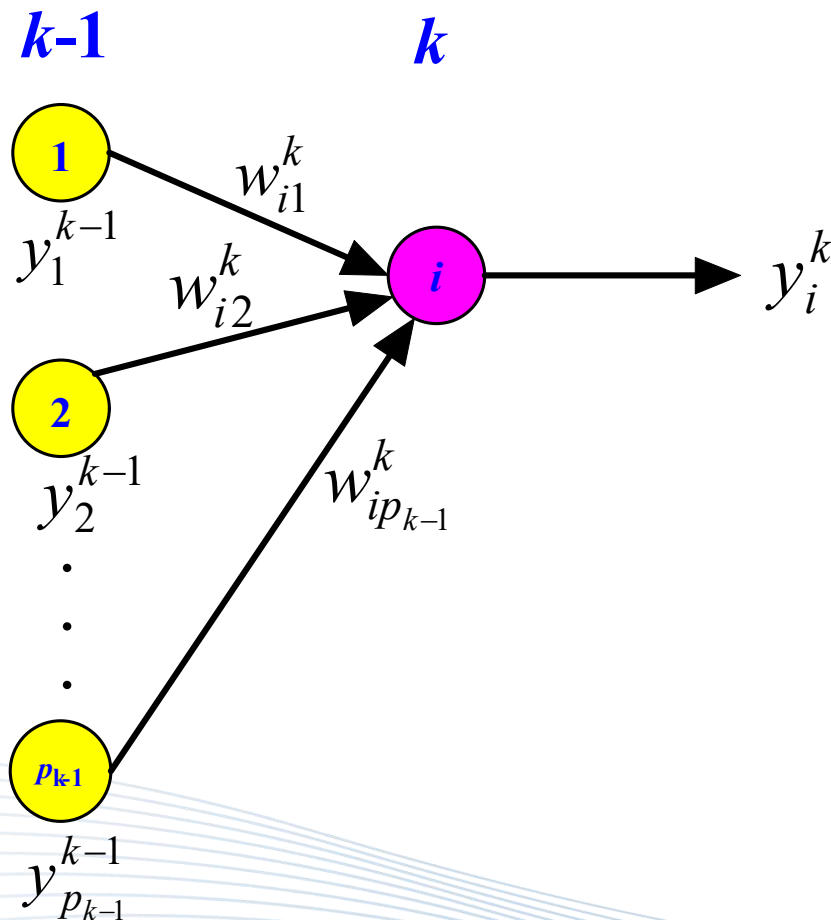
1. BP 网络结构



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

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

2. 输入输出变换关系



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

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

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

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

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

3. 工作过程

- 第一阶段或网络训练阶段：

- N 组输入输出样本： $x_i=[x_{i1}, x_{i2}, \dots, x_{ip1}]^T$

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

$$i=1, 2, \dots, N$$

- 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。

- (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。

■ 1. 基本思想

- 目标函数:
$$J = \frac{1}{2} \sum_{j=1}^{p_m} (y_j^m - d_j)^2$$

- 约束条件:
$$u_i^k = \sum_j w_{ij}^{k-1} y_j^{k-1} \quad i = 1, 2, \dots, p_k$$

$$y_i^k = f_k(u_i^k) \quad k = 1, 2, \dots, m$$

- 连接权值的修正量:

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

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
$$\frac{\partial J}{\partial y_i^k} = \frac{\partial J}{\partial y_i^m} = y_i^m - y_{si}$$

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

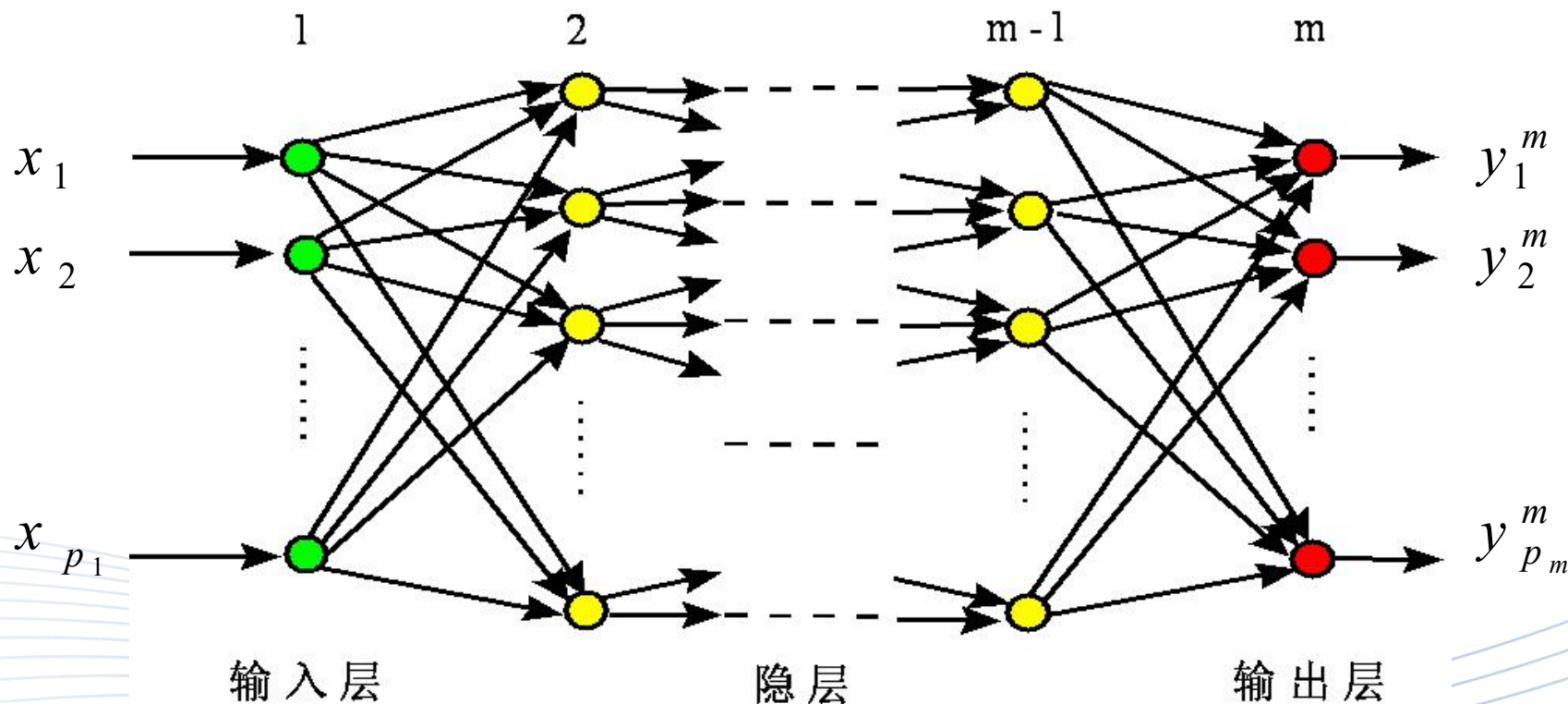
(2) 对隐单元层, 则有
$$\frac{\partial J}{\partial y_i^k} = \sum_l \frac{\partial J}{\partial u_l^{k+1}} \frac{\partial u_l^{k+1}}{\partial y_i^k} = \sum_l d_l^{k+1} w_{li}^k$$

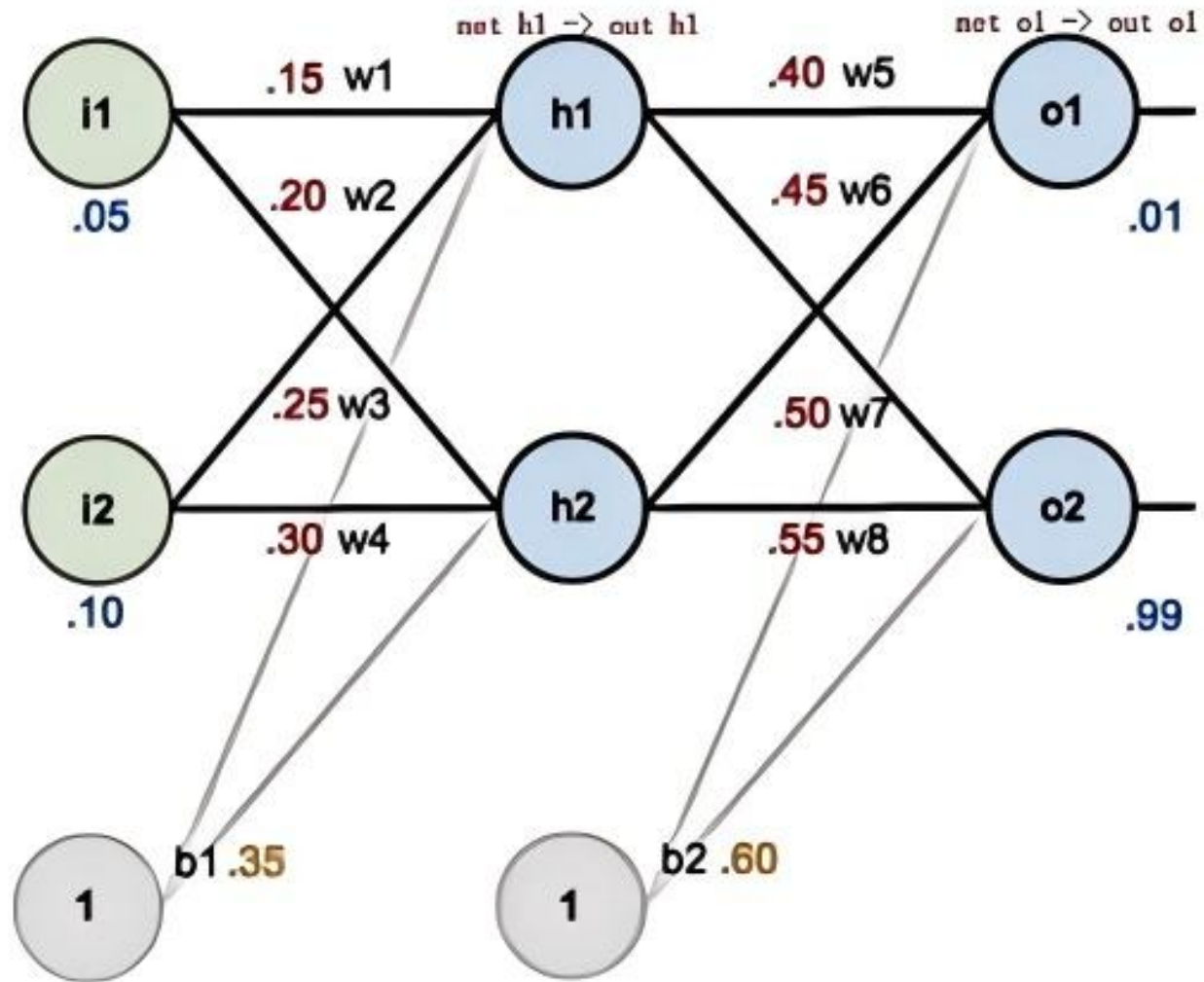
$$d_i^k = f'_k(u_i^k) \sum_l d_l^{k+1} w_{li}^k$$

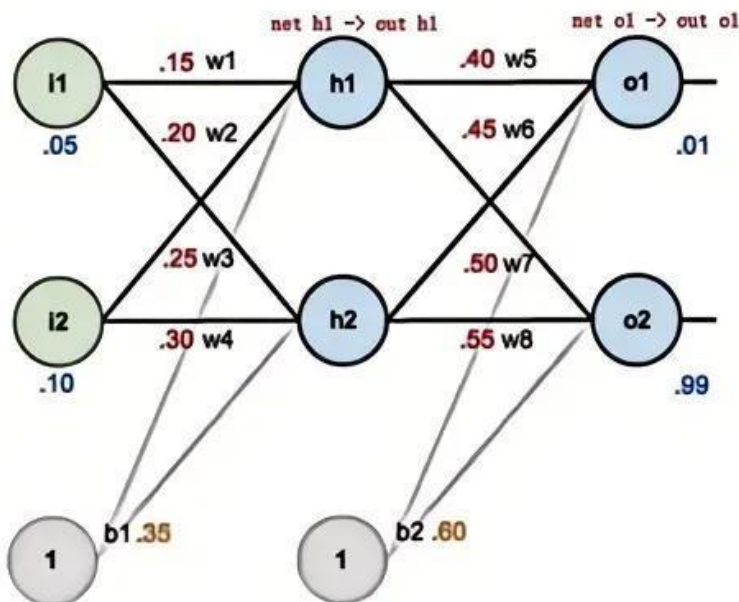
$$\Delta w_{ij}^{k-1} = -\varepsilon d_i^k y_j^{k-1}$$

■ 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。







$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

$$out_{h2} = 0.596884378$$



$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

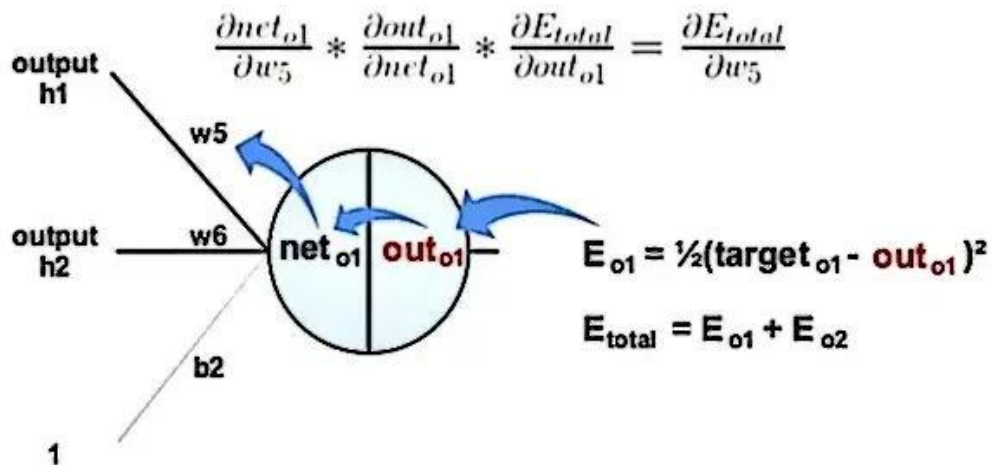
$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

$$out_{o2} = 0.772928465$$



$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$



$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

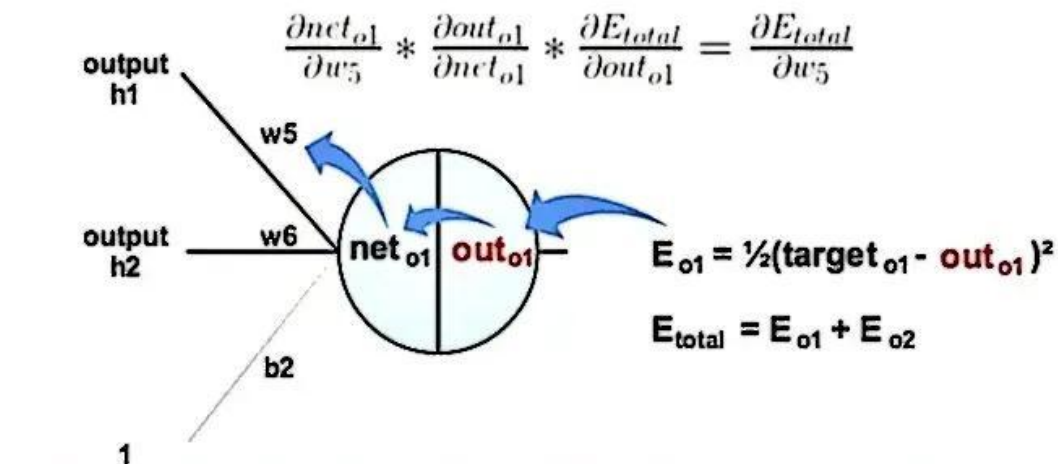
$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992$$



$$\frac{\partial E_{total}}{\partial w5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

$$w_6^+ = 0.408666186$$

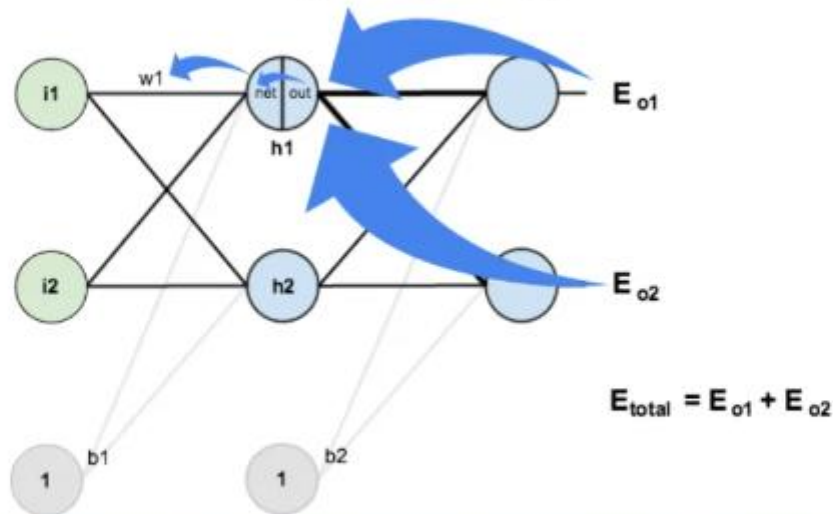
$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\downarrow$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}} \right) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

■ 1. BP算法的设计

- (1) **隐层数及隐层神经元数的确定**：目前尚无理论指导。
- (2) **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。
- (3) **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。
- (4) **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。

■ 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。**

■ 1. 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。

深度神经网络结构复杂、参数众多，很容易造成过拟合（over-fitting）。为了缓解神经网络在训练过程中出现的过拟合现象，需要采取一些正则化技术来提升神经网络的泛化能力（generalization）。

- ◆ **Dropout**: 指在训练神经网络的过程中随机丢掉一部分神经元来降低神经网络的复杂度，从而防止过拟合。Dropout的实现方法很简单：在每次迭代训练中，以一定概率随机屏蔽每一层中的若干神经元，用余下神经元构成的网络继续训练。
- ◆ **批归一化**: 批归一化（batch normalization）就是通过规范化手段，把神经网络每层中任意神经元的输入值分布改变成均值为0、方差为1的标准正态分布，把偏移较大的分布强制映射为标准正态分布。经过批归一化处理，激活函数的输入值被映射到非线性函数梯度较大的区域，使得梯度变大从而克服梯度消失问题，进而加快收敛速度。

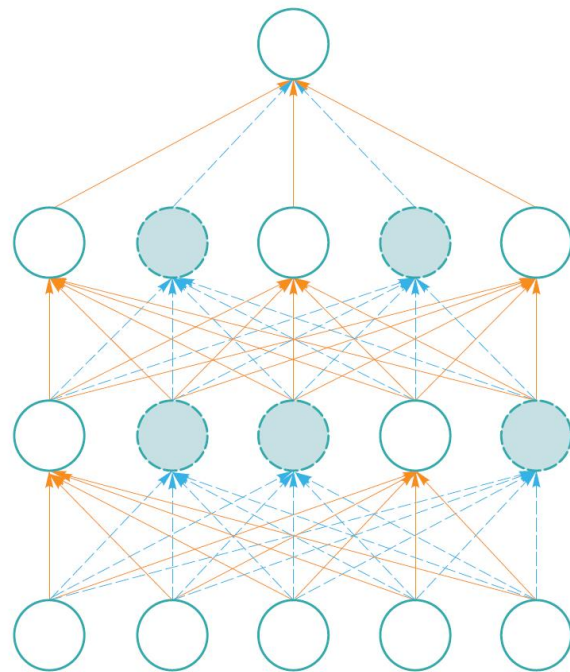


图 5.22 使用 Dropout 的神经网络模型

神经网络正则化

深度神经网络结构复杂、参数众多，很容易造成过拟合（over-fitting）。为了缓解神经网络在训练过程中出现的过拟合现象，需要采取一些正则化技术来提升神经网络的泛化能力（generalization）。

L_1 和 L_2 正则化：对于具有 n 个训练数据 $\{(x_1, y_1), \dots, (x_n, y_n)\}$ 的神经网络，其中 $(x_i, y_i) (1 \leq i \leq n)$ 分别为输入样本及其对应的标签。加入正则化项后，神经网络的损失函数一般可如下表示：

$$\min \frac{1}{n} \sum_{i=1}^n \underbrace{\text{Loss}(y_i, f(\mathbf{W}, x_i))}_{\text{损失函数}} + \underbrace{\lambda}_{\text{正则化权重}} \times \underbrace{\Phi(\mathbf{W})}_{\text{正则化项}}$$

其中， $f(\mathbf{W}, x_i)$ 表示参数为 $\mathbf{W}$ 的神经网络对输入 x_i 的预测、 $\Phi(\mathbf{W})$ 为正则化项（又称惩罚项）、 λ 为正则项权重。

正则化项 $\Phi(\mathbf{W})$ 一般用模型参数 $\mathbf{W}$ 的范数形式来表示。假设神经网络模型中参数数目为 N ，则参数的范数形式主要有：

L_0 范数：数学表示为 $\|\mathbf{W}\|_0 = \sum_{i=1}^N \mathbb{I}[w_i \neq 0]$ ，其中 $\mathbb{I}$ 是指示函数，若括号内表达式为真，函数值为1，反之为0。所以 L_0 范数指模型参数 $\mathbf{W}$ 中非零元素个数。 L_0 可实现模型参数的稀疏化。但是，由于 L_0 范数正则化是一个NP Hard问题，难以求解，因此一般不用 L_0 范数。

L_1 范数：数学表示为 $\|\mathbf{W}\|_1 = \sum_{i=1}^N |w_i|$ ，指模型参数 $\mathbf{W}$ 中各个元素的绝对值之和。 L_1 范数也被称为“稀疏规则算子”（Lasso regularization）。

L_2 范数：数学表示为 $\|\mathbf{W}\|_2 = \sqrt{\sum_{i=1}^N w_i^2}$ ，指模型参数 $\mathbf{W}$ 中各个元素平方和的开方。

03 / PART ONE

B P 神经网络的应用

■ 1. BP神经网络在模式识别中的应用

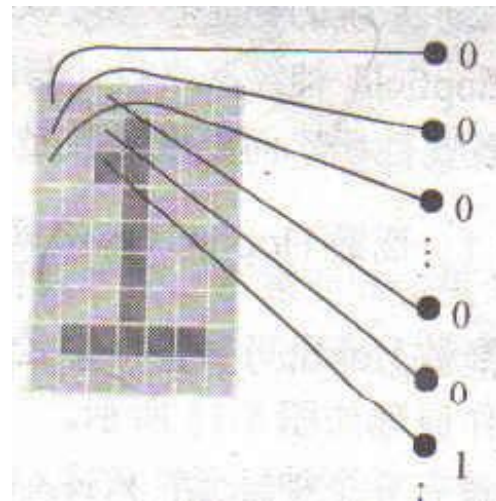
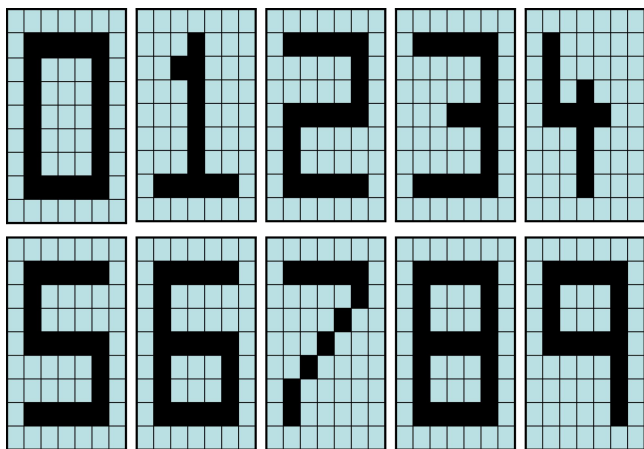
模式识别研究用计算机模拟生物、人的感知，对模式信息，如图像、文字、语音等，进行识别和分类。

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网络的特点：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。

1. BP神经网络在模式识别中的应用-设计一个三层BP网络对数字0至9进行分类。



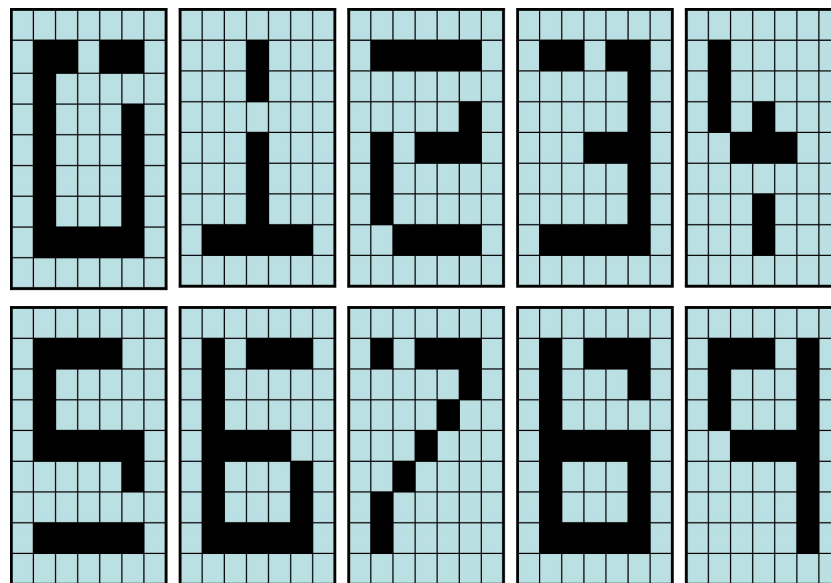
每个数字用 9×7 的网格表示，灰色像素代表0，黑色像素代表1。将每个网格表示为0, 1的长位串。位映射由左上角开始向下直到网格的整个一列，然后重复其他列。

选择BP网络结构为63-6-9。63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。

1. BP神经网络在模式识别中的应用-设计一个三层BP网络对数字0至9进行分类。

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。



河海大学
HOHAI UNIVERSITY

谢 谢!