第8章 人工神经网络及其应用

- 人工神经网络已在模式分类、机器视觉、机器听觉、智能计算、机器人控制、信号处理、组合优化问题求解、联想记忆、编码理论、医学诊断、金融决策和数据挖掘等领域获得了卓有成效的应用。

第8章 人工神经网络及其应用

- 神经网络（Neural Networks，NN）

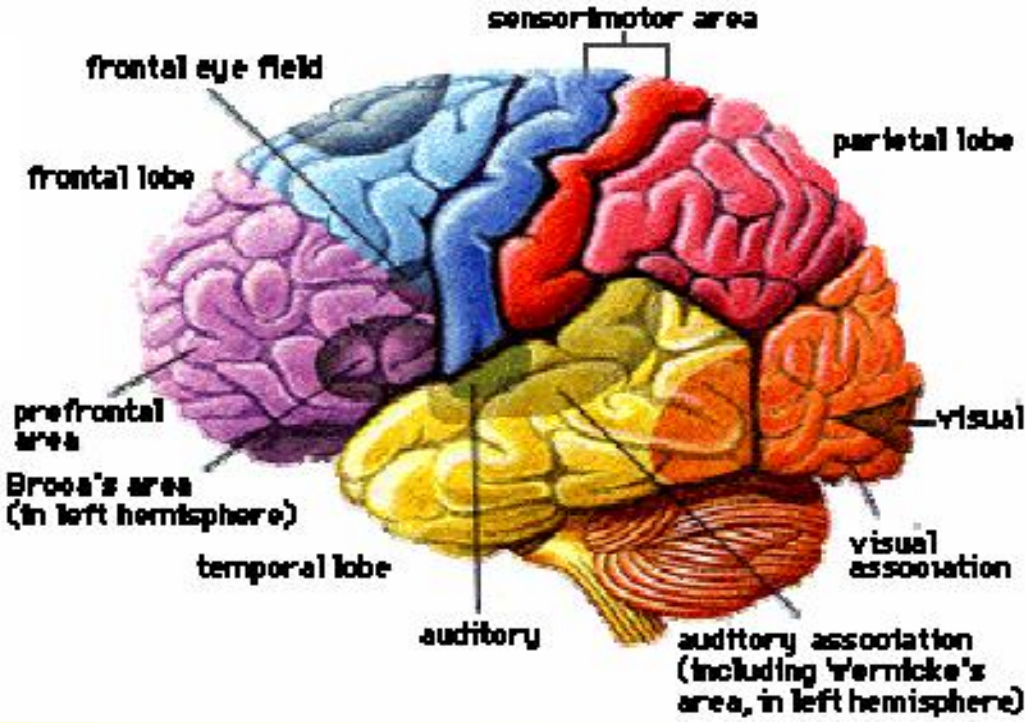
- 生物神经网络(Natural Neural Network, NNN): 由中枢神经系统（脑和脊髓）及周围神经系统（感觉神经、运动神经等）所构成的错综复杂的神经网络，其中最重要的是脑神经系统。

- 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。

大 脑 模型

- 人脑构造：

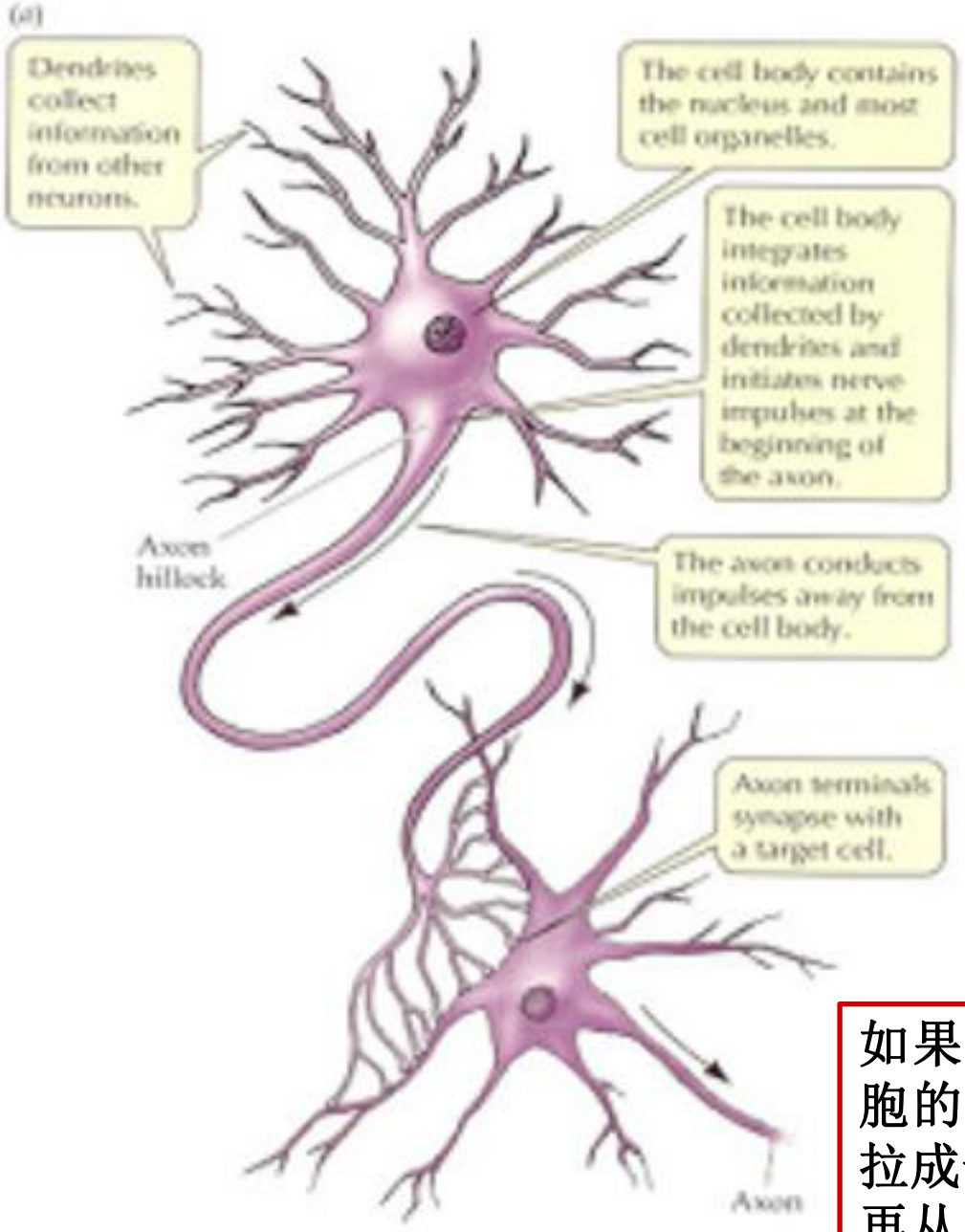
  - 皮层（cortex）
  - 中脑（midbrain）
  - 脑干（brainstem）
  - 小脑（cerebellum）



- 人脑由10^{11}～10^{14}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。

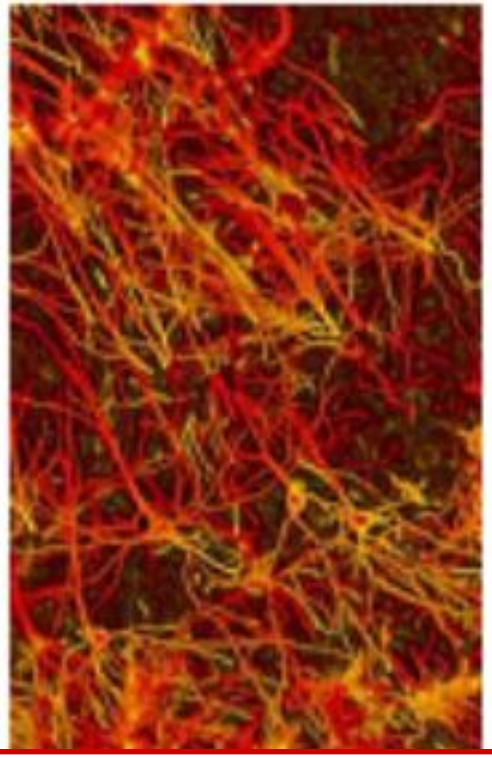
连接总计可能有100,000,000,000,000个

- 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，形成错综复杂而又灵活多变的神经网络。



A generalized diagram of a neuron (p. 906).

-- Purves et. al., 1998



如果将一个人的大脑中所有神经细胞的轴突和树突依次连接起来，并拉成一根直线，可从地球连到月亮，再从月亮返回地球。

第8章 人工神经网络及其应用

- 神经网络（Neural Networks，NN）

▪ 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。

● 硬件实现：电子、光电元件——神经计算机、TPU、NPU等;
● 软件实现：仿真软件，NN模拟软件——TensorFlow，PyTorch，Caffe,PaddlePaddle等
➢神经网络方法：隐式的知识表示方法

大脑模型

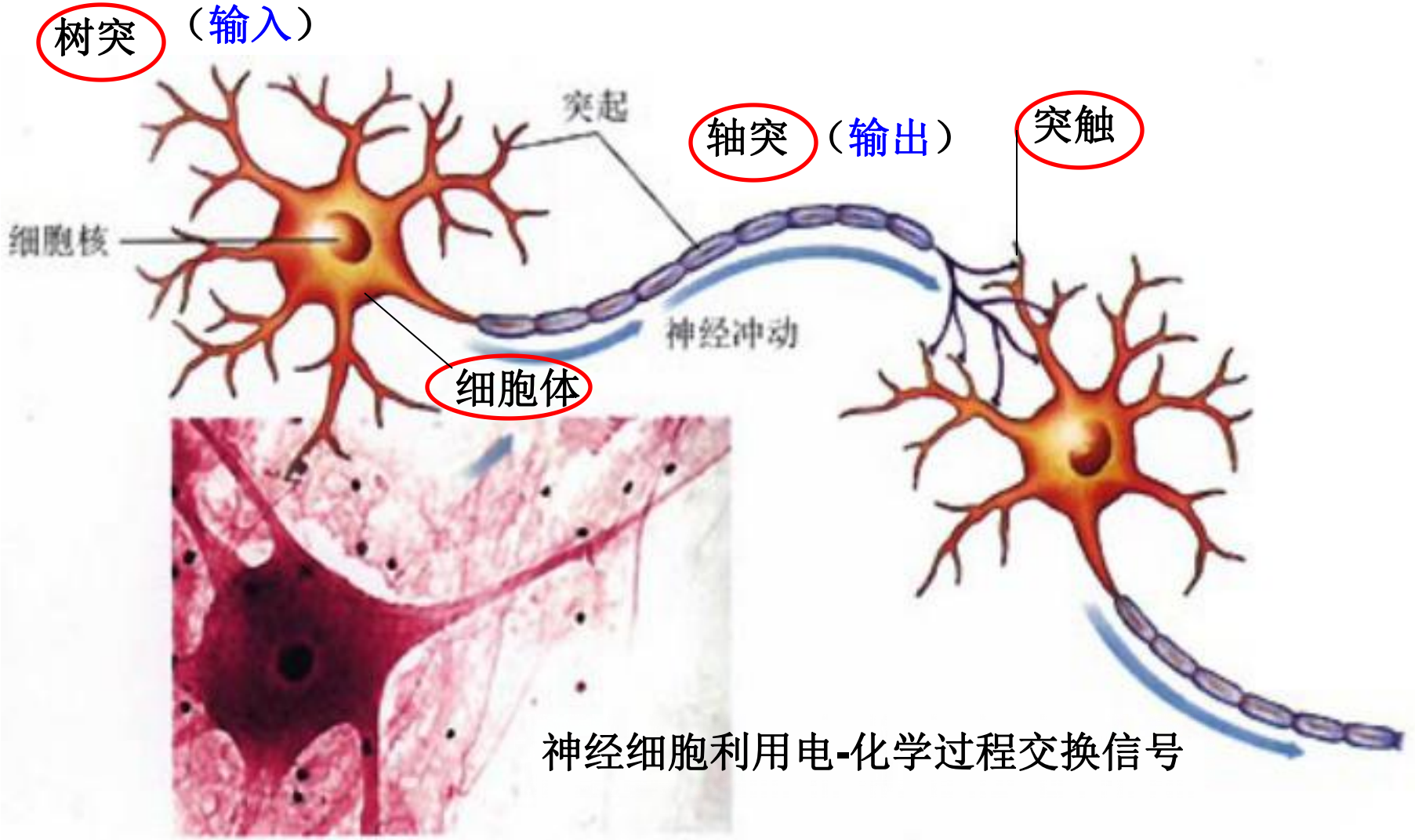
第8章 人工神经网络及其应用

- 8.1 神经元与神经网络
- 8.2 BP神经网络及其学习算法
- 8.3 卷积神经网络及其应用
- 8.4 Hopfield神经网络及其应用
- 其他神经网络

- 重点：BP神经网络、卷积神经网络等典型神经网络模型的结构及工作原理
- 难点：不同激活函数的特点、梯度下降法和反向传播、卷积操作等点

8.1.1 神经元模型

1. 生物神经元结构

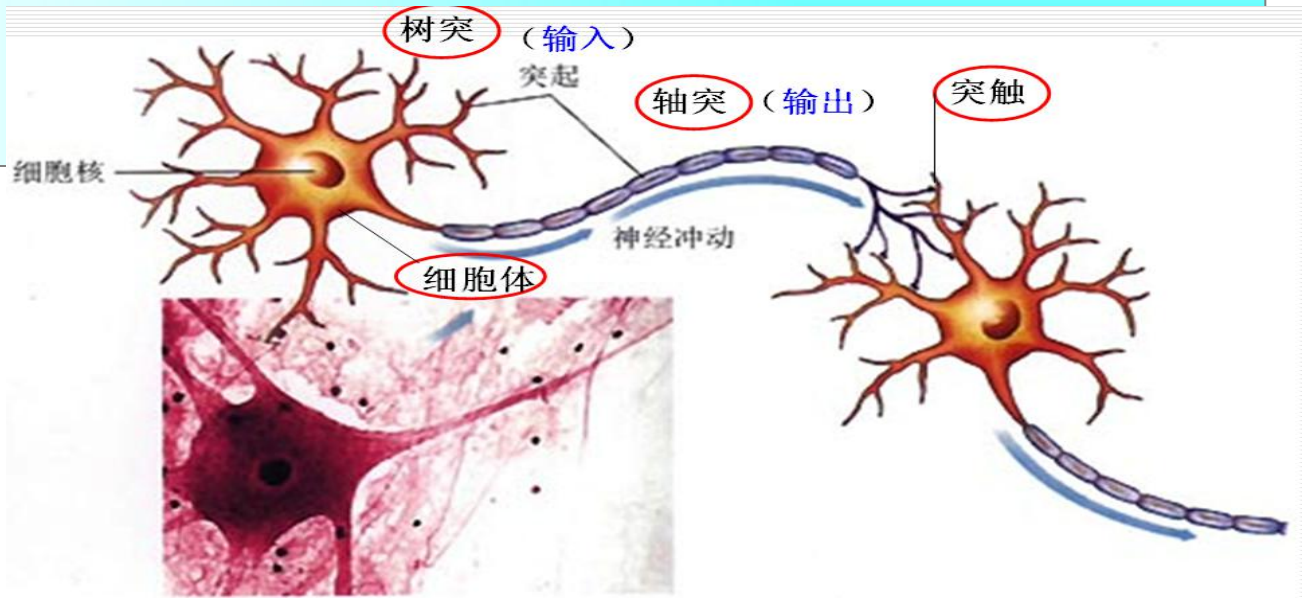


神经细胞利用电-化学过程交换信号

8.1.1 神经元模型

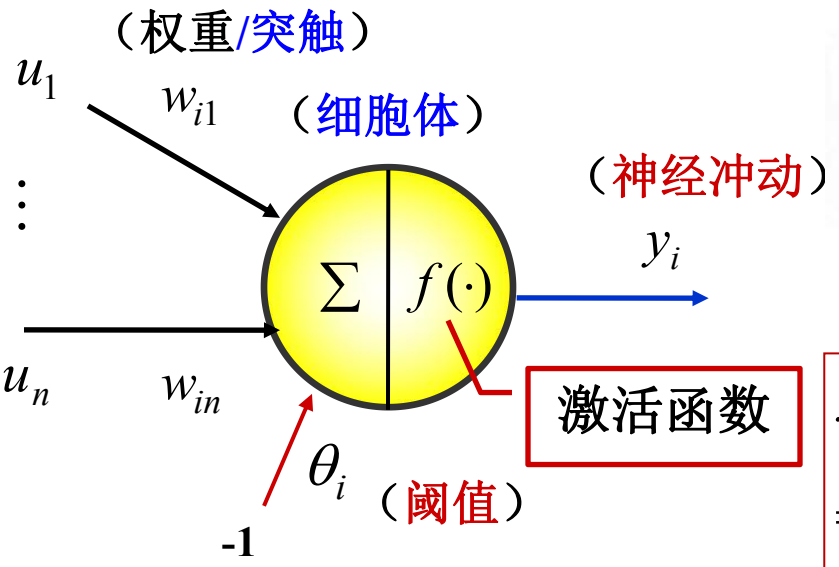
1. 生物神经元结构

- 工作状态：
- 兴奋状态：细胞膜电位 > 动作电位的阈值 → 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 。

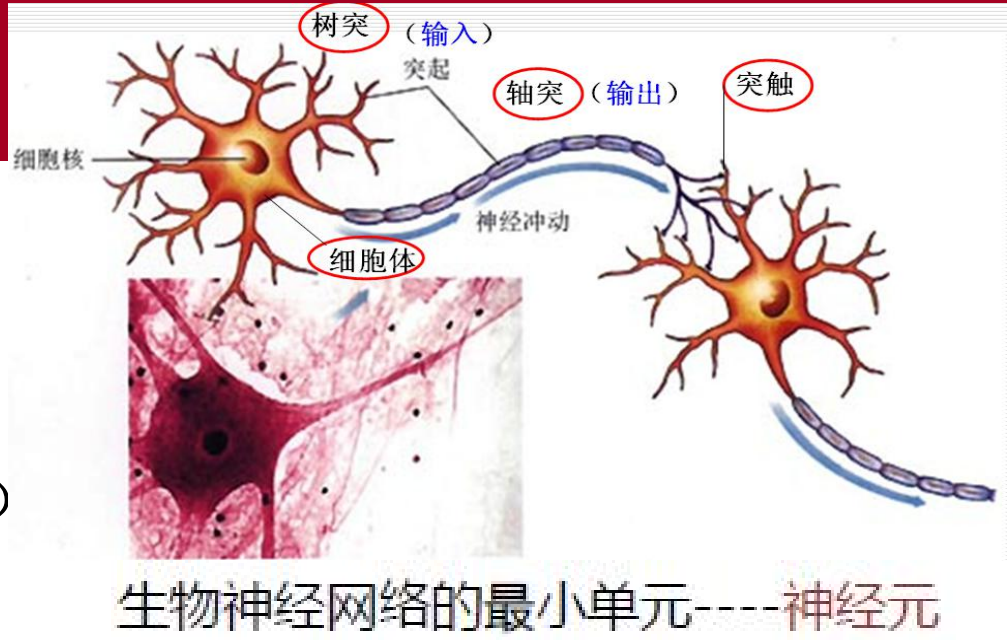


8.1.1 神经元模型

2. 人工神经元模型



人工神经元 *i* 的模型图

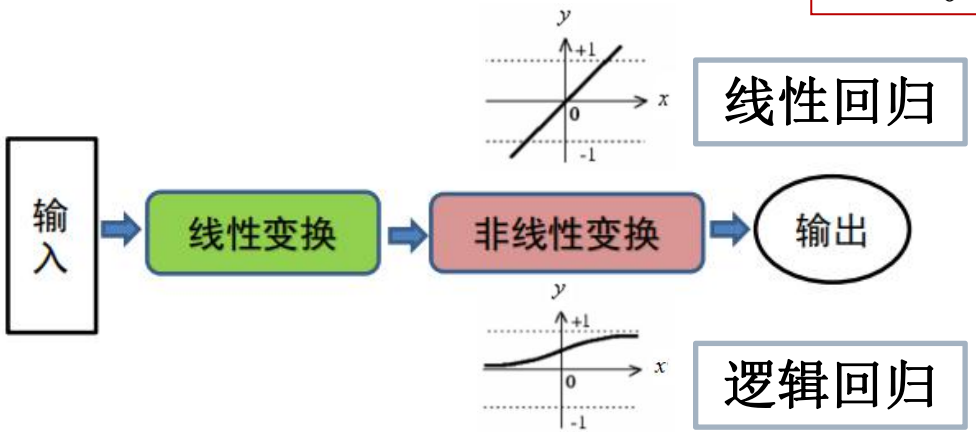


生物神经网络的最小单元----神经元

$$y_i = f(w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{in}u_n - \theta_i)$$

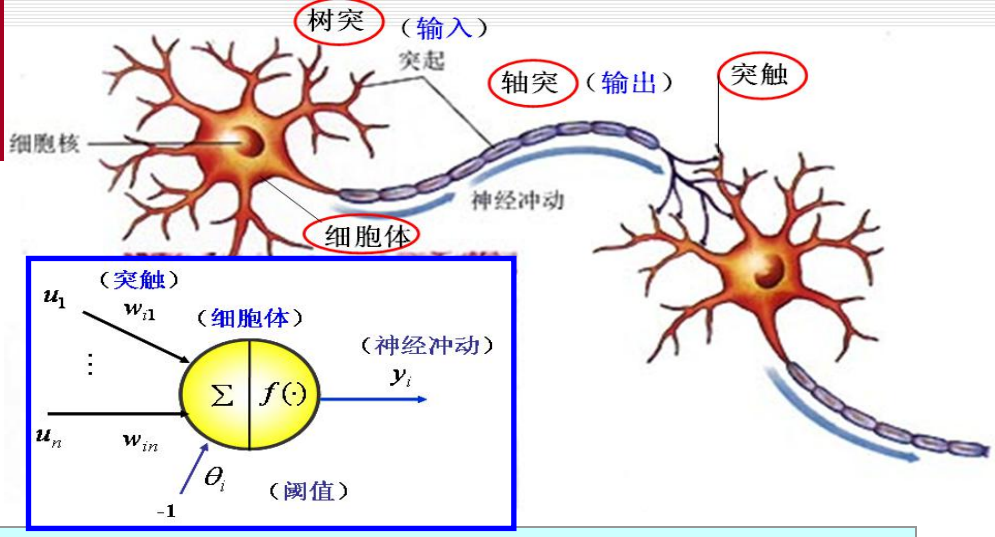
$$= f(\sum_{j=1}^{n} w_{ij}u_j - \theta_i) = f(\sum_{j=0}^{n} w_{ij}u_j) = f(\mathrm{w}_i^{\mathrm{T}}\mathrm{u})$$

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



- 输入：一个向量
- 输出：一个标量
- 运算：
  - 线性变换（加权求和）
  - 非线性变换（非线性函数）

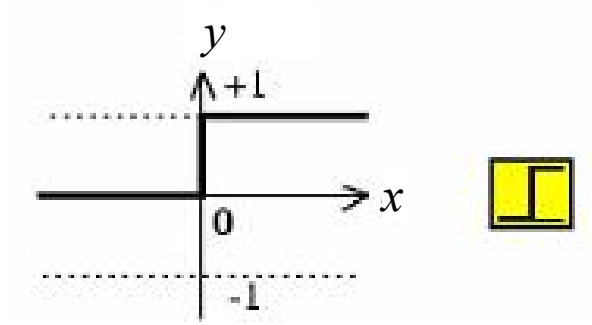
8.1.1 神经元模型



2. 人工神经元模型

- 激活函数(Activation)

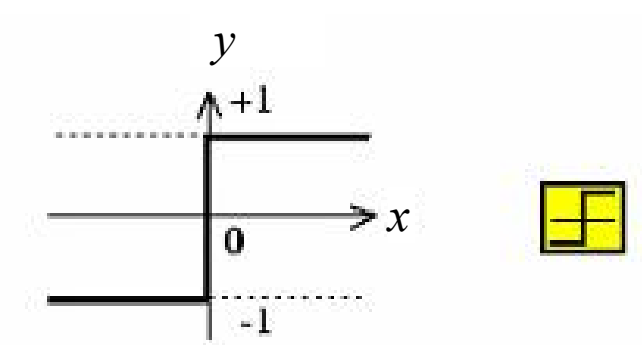
（传输函数、输出变换函数）



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

Hard-Limit Transfer Function

（阶跃函数或硬极限函数）

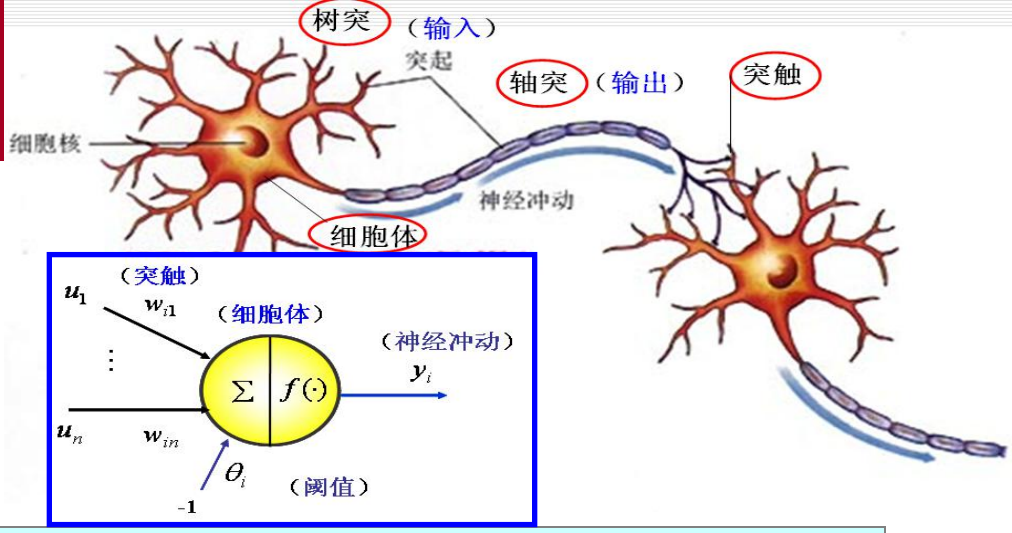


$$y=hardlims(x)=\begin{cases}1 & x\geq 0\\ -1 & x<0\end{cases}$$

Symmetric Hard-Limit Trans. Funct.

（对称硬极限函数）

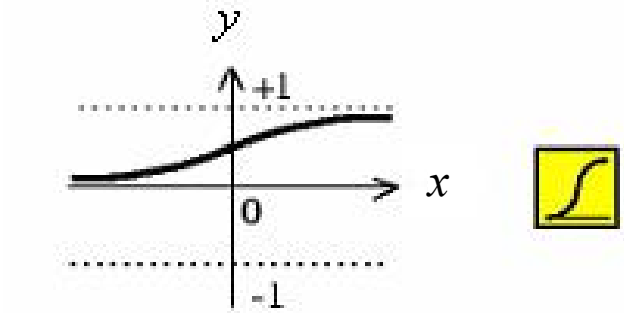
8.1.1 神经元模型



2. 人工神经元模型

- 激活函数(Activation)

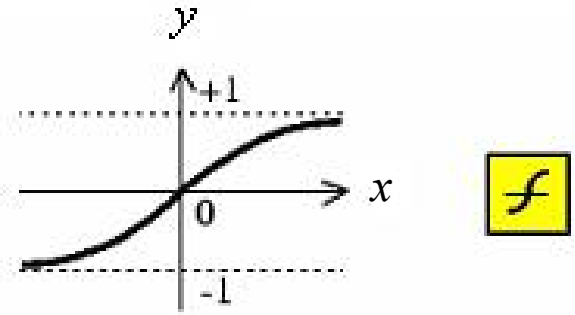
（传输函数、输出变换函数）



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

$$\alpha = 1$$

Log-Sigmoid Transfer Function



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

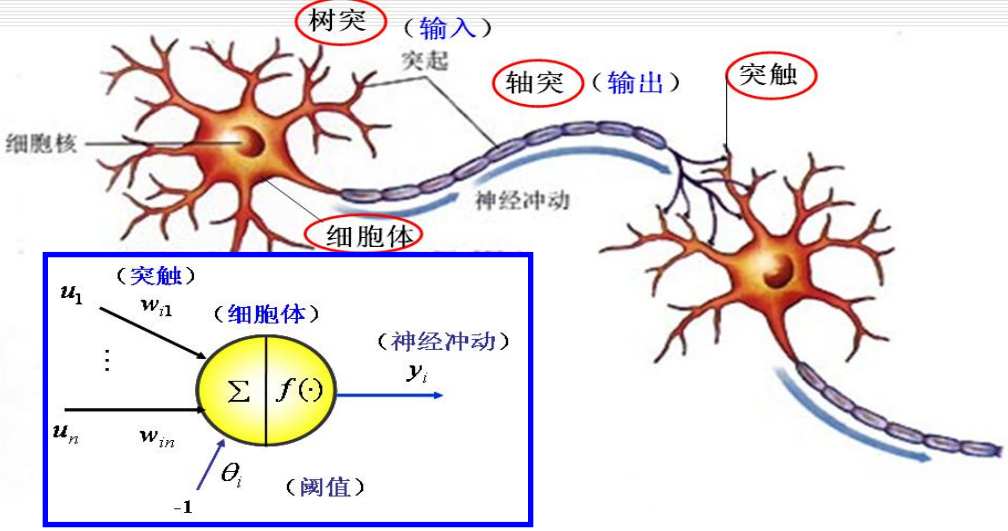
$$\alpha = 1$$

Tan-Sigmoid Transfer Function

（ Sigmoid函数或对数 S 形函数）

（tanh函数或双曲正切S形函数）

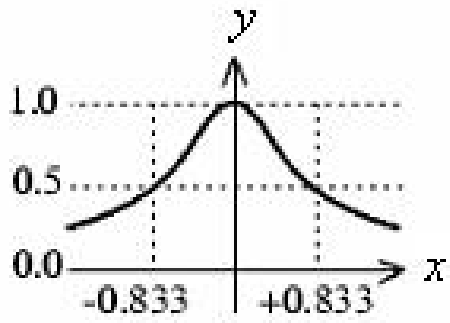
8.1.1 神经元模型



2. 人工神经元模型

- 激活函数(Activation)

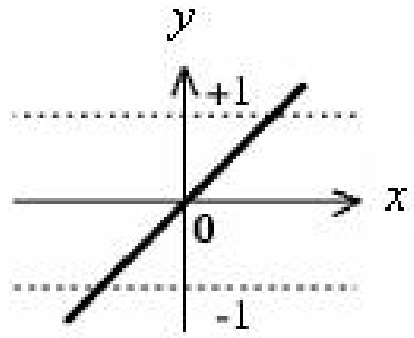
（传输函数、输出变换函数）



$$y = radbas(x) = e^{-\left(\frac{x}{\sigma}\right)^2}$$

Radial Basis Function

（高斯或径向基函数）



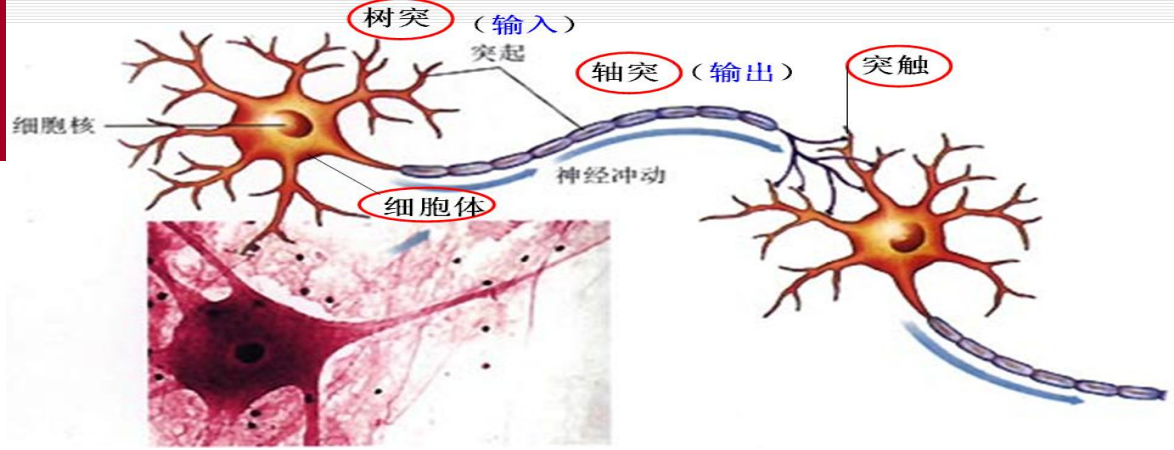
$$y = purelin(x) = x$$

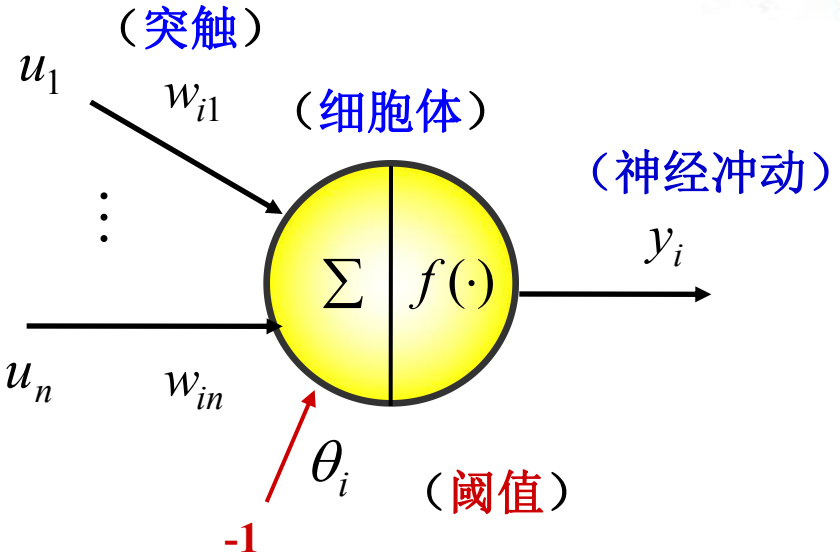
Linear Transfer Function

（Linear/线性函数）

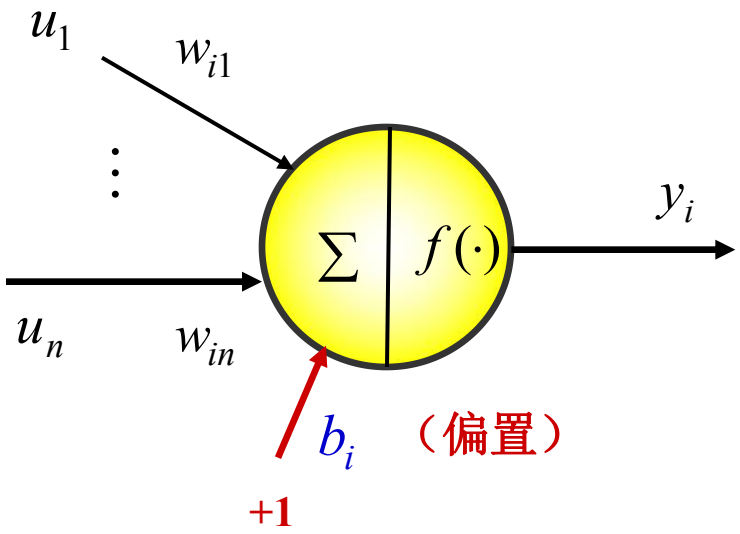
8.1.1 神经元模型

2. 人工神经元模型





人工神经元 *i* 的模型图



人工神经元 *i* 的模型图

$$y_i = f\left(\sum_{j=1}^{n} w_{ij}u_j - \theta_i\right) = f\left(\sum_{j=0}^{n} w_{ij}u_j\right)$$

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

$$y_i = f\left(\sum_{j=1}^{n} w_{ij}u_j + b_i\right) = f\left(\sum_{j=0}^{n} w_{ij}u_j\right)$$

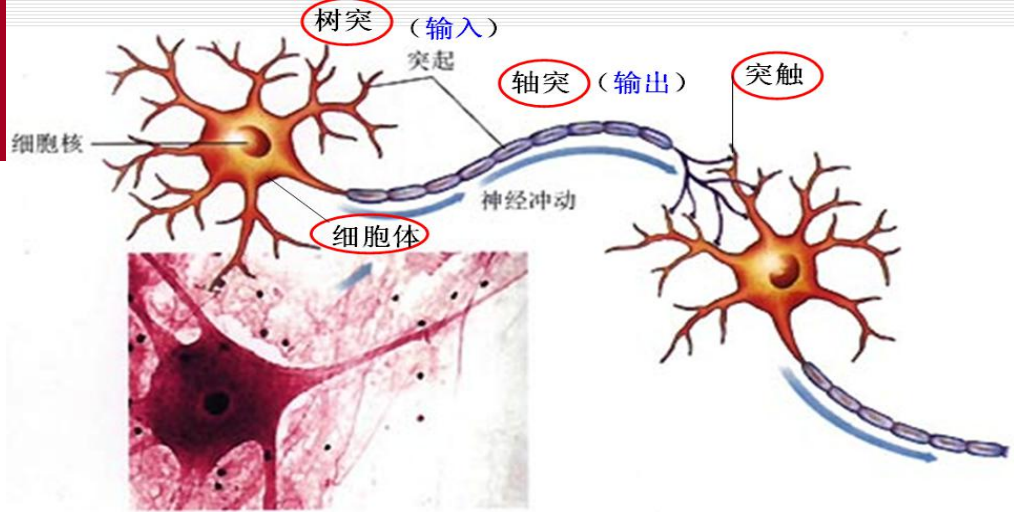
$$(u_0 = 1, w_{i0} = b_i)$$

8.1.1 神经元模型

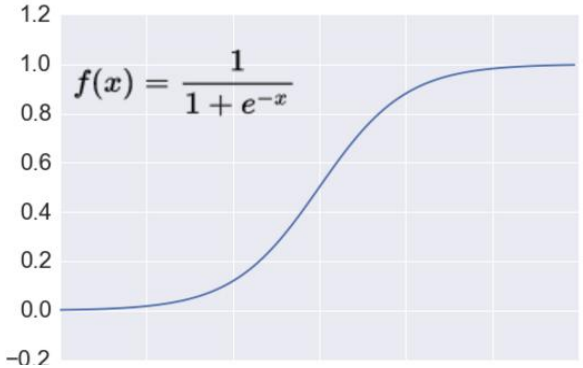
2. 人工神经元模型

- 激活函数(Activation)

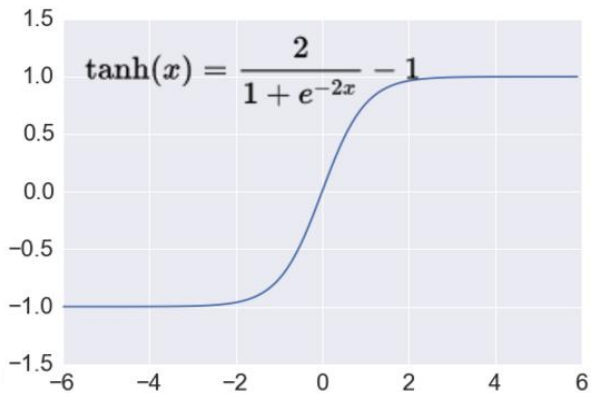
（传输函数、输出变换函数）

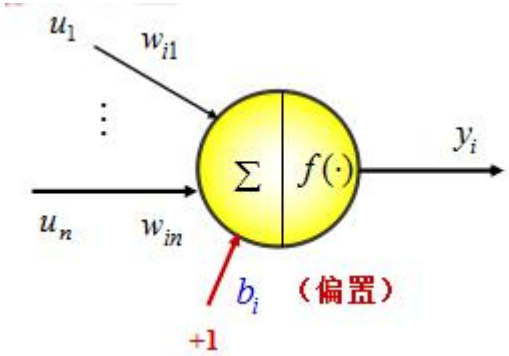


Sigmoid



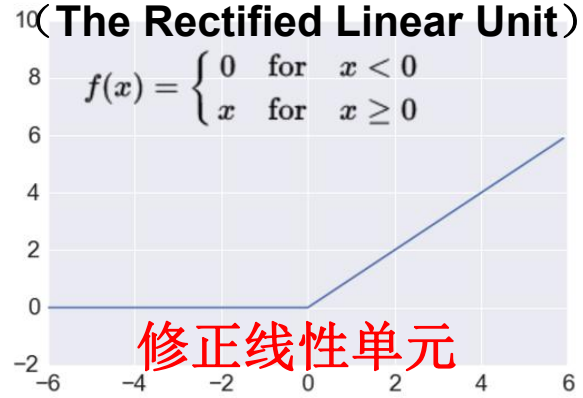
TanH





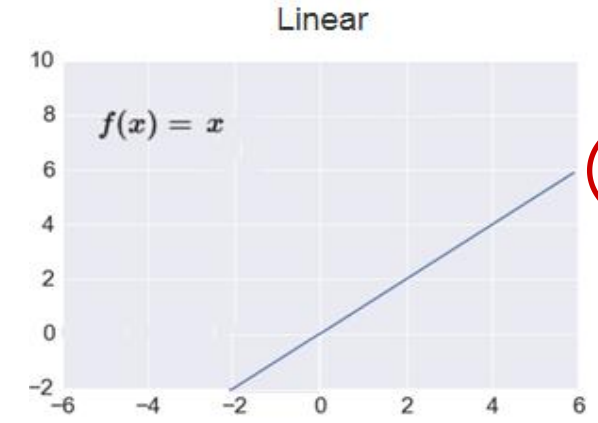
ReLU

（The Rectified Linear Unit）



修正线性单元

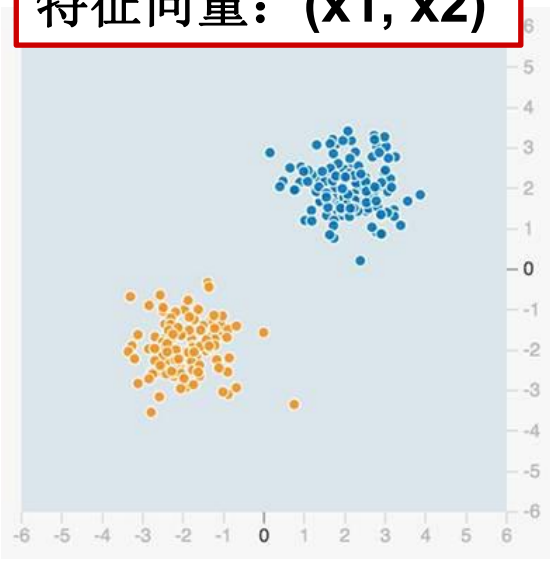
Linear

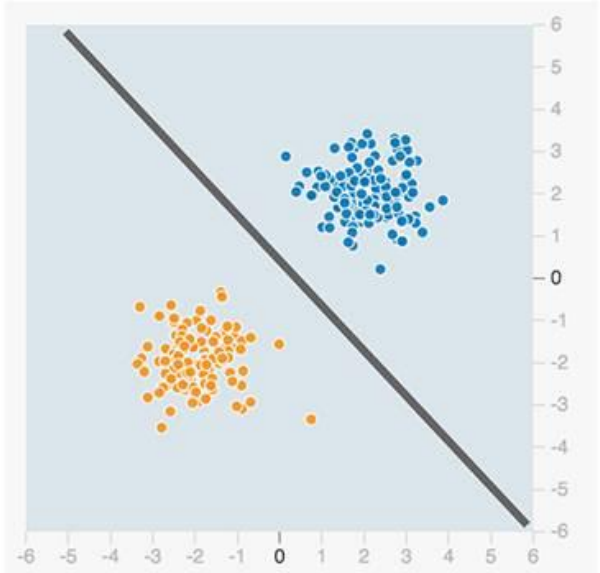


- 一个神经元有什么用？
- 不同的激励函数区别是什么？

简单的线性分类问题

特征向量：(x1, x2)





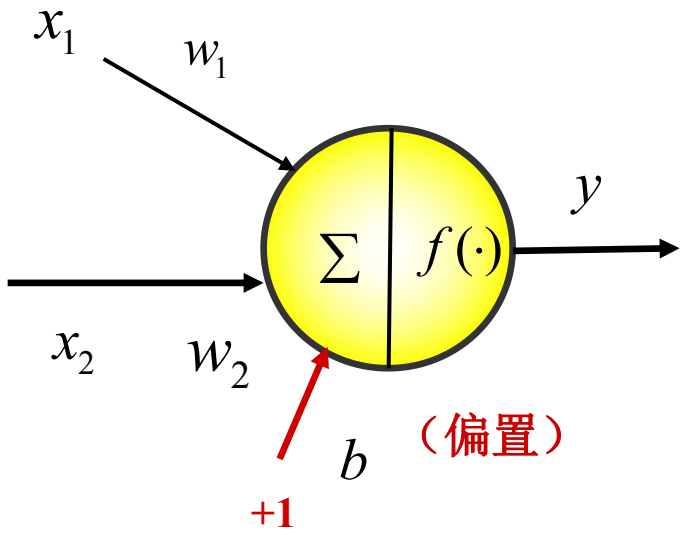
每一个数据点(x_1, x_2)，如何区分一个数据点是橙色的还是蓝色的？

分界线：$w_1x_1 + w_2x_2 + b = 0$

蓝色点：$w_1x_1 + w_2x_2 + b > 0$

黄色点：$w_1x_1 + w_2x_2 + b < 0$

不同的激励函数应用时有什么区别？



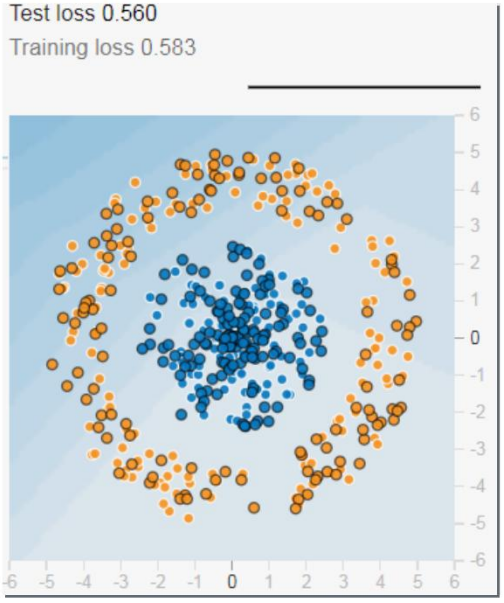
人工神经元的模型图

$$y = f(w_1x_1 + w_2x_2 + b)$$

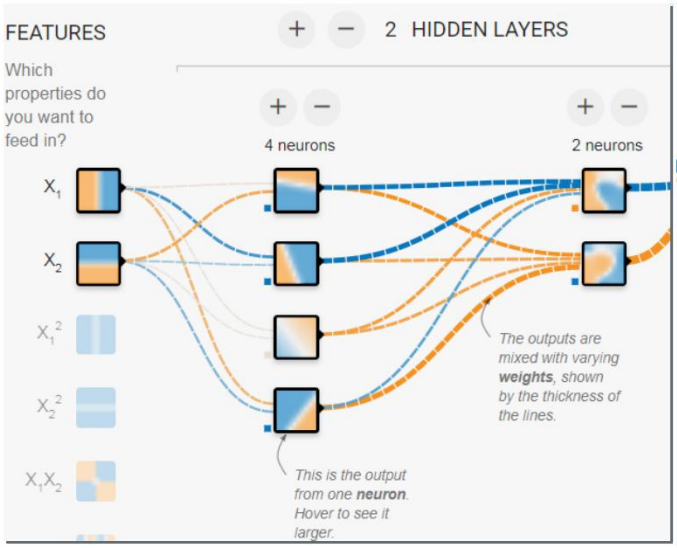
$f(\cdot)$ 可以采用哪些激活函数？

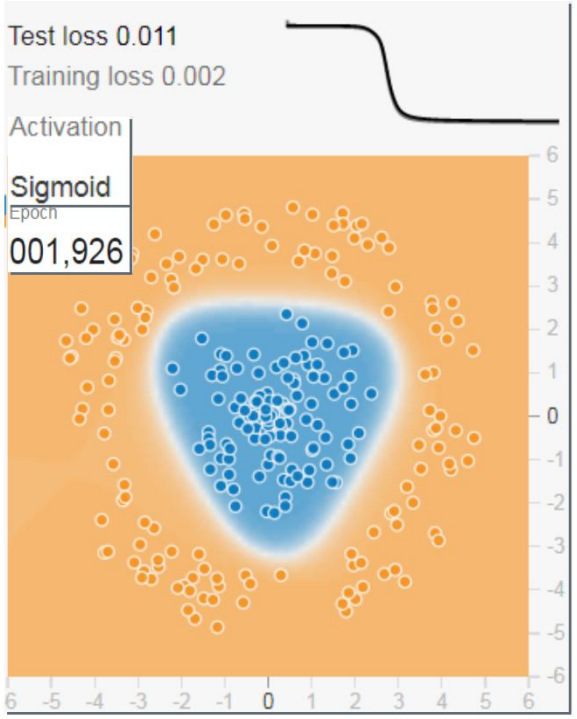
A. Linear　　B. Sigmoid

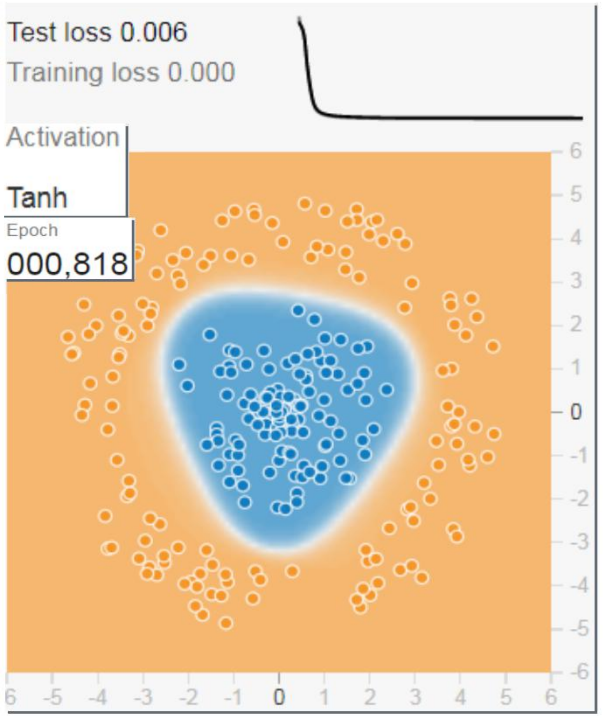
C. Tanh　　D. ReLU函数

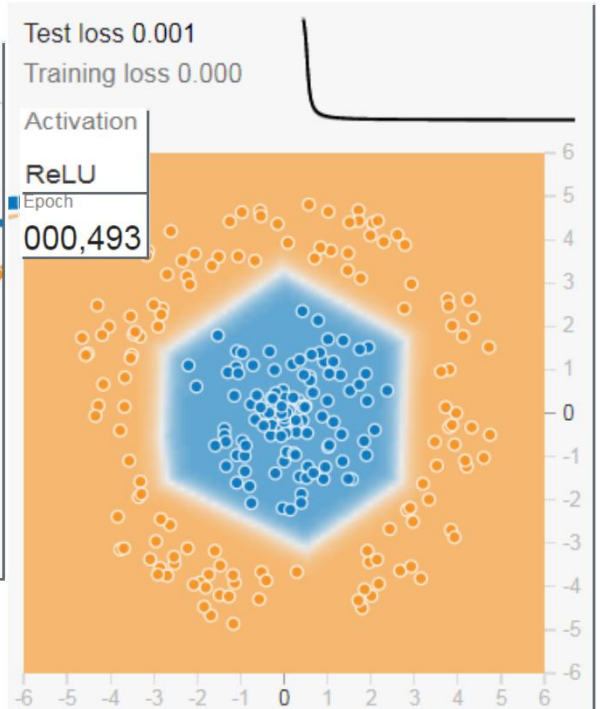

Test loss 0.560
Training loss 0.583

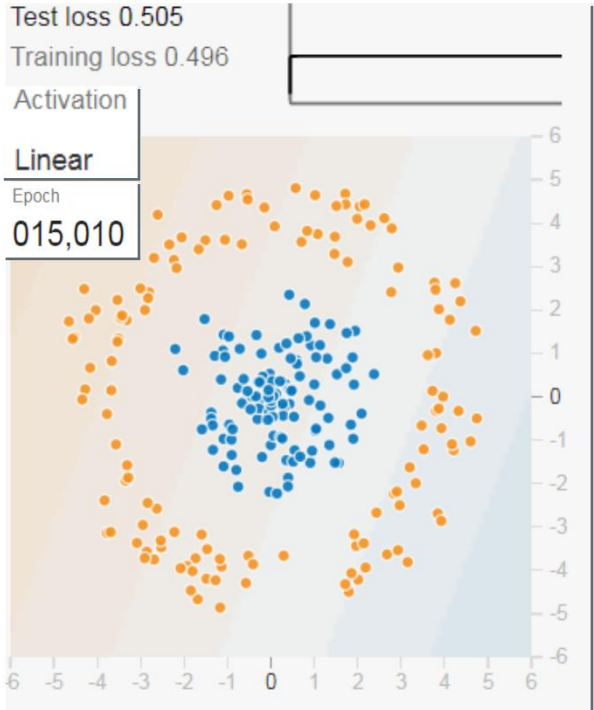
非线性二分类问题


FEATURES
2 HIDDEN LAYERS
Which properties do you want to feed in?
4 neurons
2 neurons
X1
X2
X1²
X2²
X1X2
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.


Test loss 0.011
Training loss 0.002
Activation
Sigmoid
Epoch
001,926


Test loss 0.006
Training loss 0.000
Activation
Tanh
Epoch
000,818

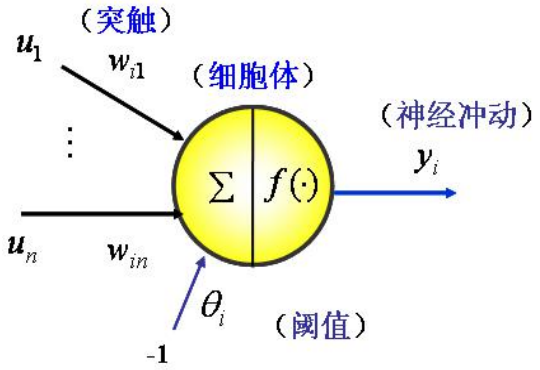

Test loss 0.001
Training loss 0.000
Activation
ReLU
Epoch
000,493

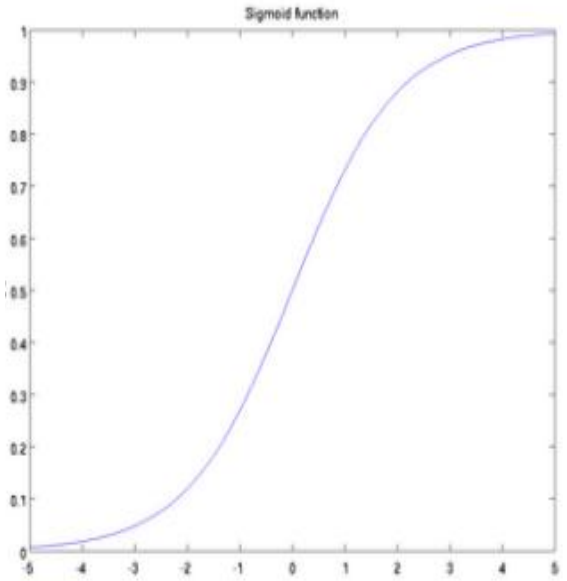

Test loss 0.505
Training loss 0.496
Activation
Linear
Epoch
015,010

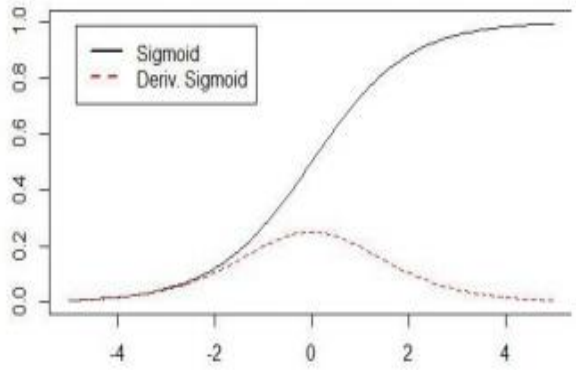
Sigmoid函数

- sigmoid函数也叫 Logistic 函数
- 特点:
  - 它可以将一个实数映射到(0,1)的区间
  - 在特征相差不是特别大时效果比较好
- 用法:
  - 通常用来做二分类
- 缺点:
  - 激活函数计算量大
  - 容易出现梯度消失
    - 当数据分布在曲线平滑位置时很容易就会出现梯度消失

$$f(z) = \frac{1}{1+\exp(-z)}.$$







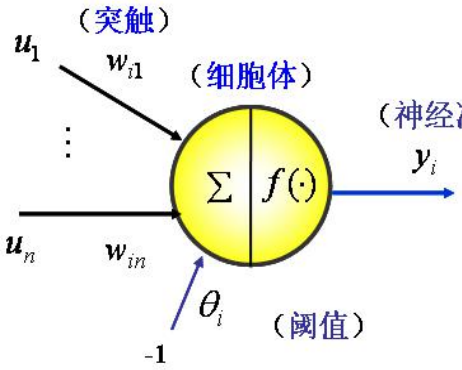
早期最流行的激活函数，激活函数的代名词

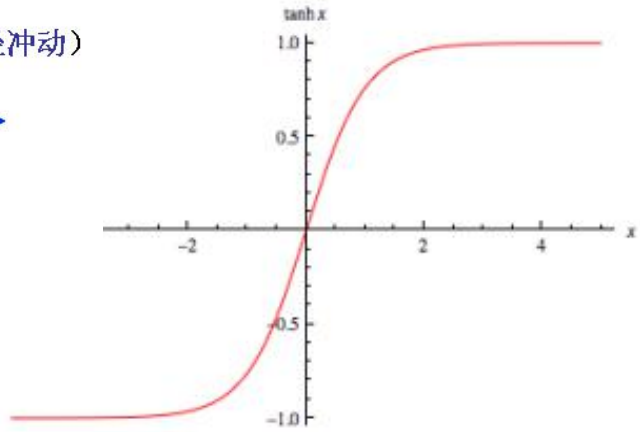
Tanh函数

- Tanh函数 也称为双切正切函数
- 特点：
  - 取值范围为[-1,1]
  - 输出是以0为中心
  - 可以认为是一个放大的sigmoid函数
- 用法：
  - 实际中tanh会比sigmoid更常用
  - 循环神经网络中常用
  - 靠近输出值位置
  - 二分类问题
- 缺点：
  - 梯度消失
  - 在曲线几乎水平的区域学习非常的慢

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

$$tanh(x) = 2sigmoid(2x) - 1$$





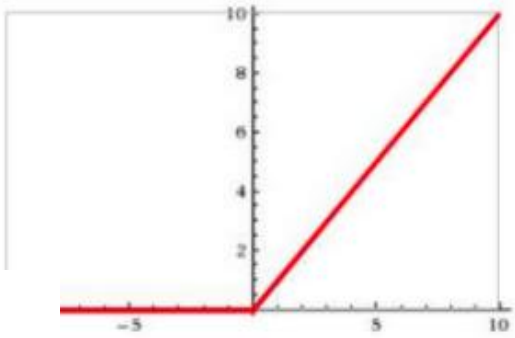
Sigmoid函数的改进版

1、解决了原点对称问题；
2、比sigmoid更快

ReLU

- ReLU函数（矫正的线性单元）
  - Rectified Linear Unit(ReLU)

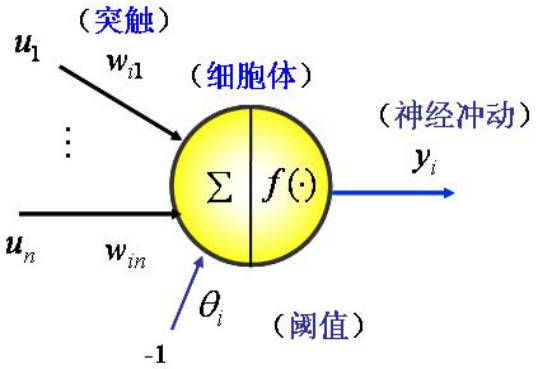
$$\phi(x) = \max(0, x)$$



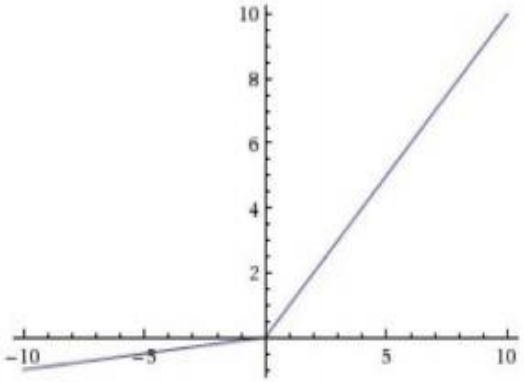
ReLU

y=max(0,x)

- 特点：
  - 大于0的部分输出为数据本身
  - 小于0的部分输出为0
  - ReLU 对于梯度收敛有巨大加速作用
  - 只需要一个阈值就可以得到激活值节省计算量
- 用法：
  - 深层网络中隐藏层常用
- 缺点：
  - 过于生猛，一言不合就判4刑，直接使数据变为0，从此节点后相关信息全部丢失



1、解决了部分梯度消失问题
2、收敛速度更快



y=max(αx,x)

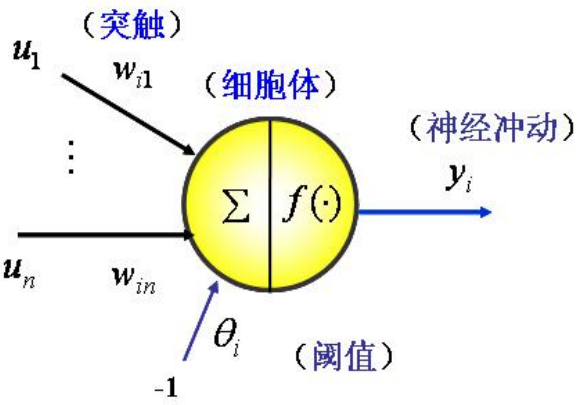
α = small const. (e.g. 0.1)

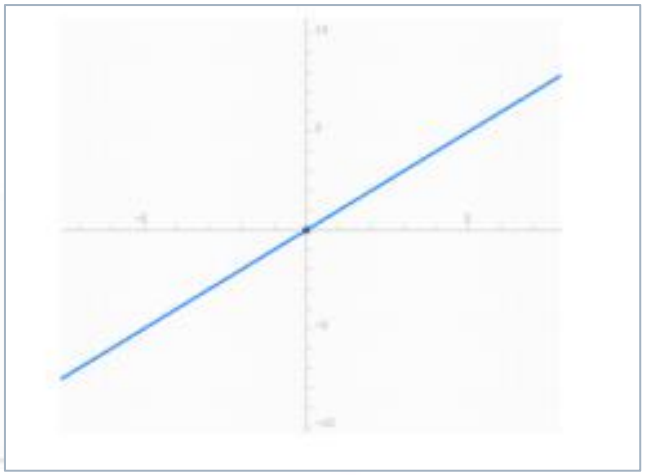
Leaky ReLU

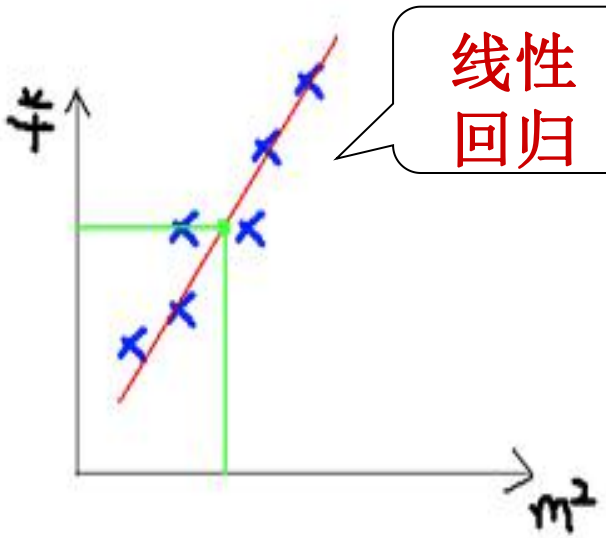
激活函数中的重要发明

线性激活函数

- linear函数 $y = x$
- 特点
  - 输入是什么，输出就是什么
  - 没有做任何 非线性变换
- 用法
  - 仅仅用于线性回归
- 缺点
  - 没有非线性变换
    - 无法拟合非线性函数
  - 无法建立非线性模型





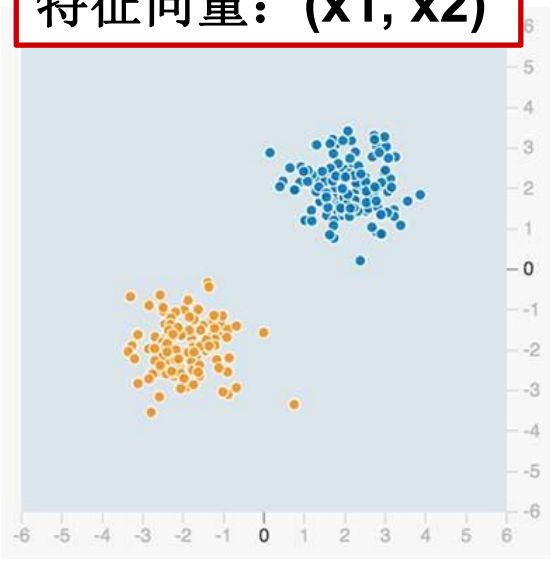


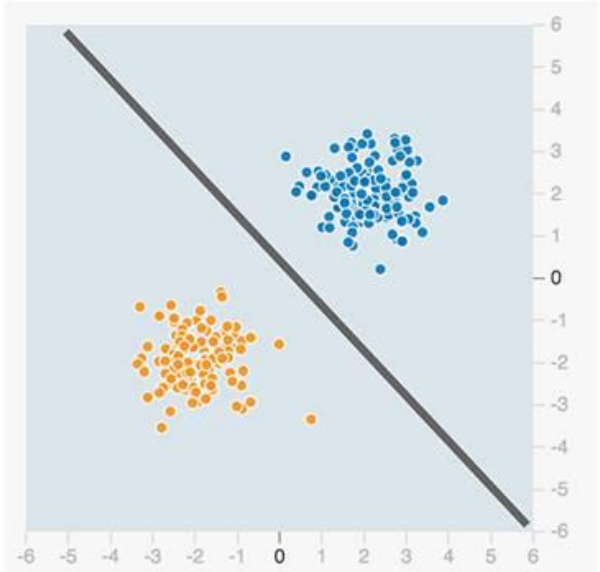
房价 = 面积 * a + b

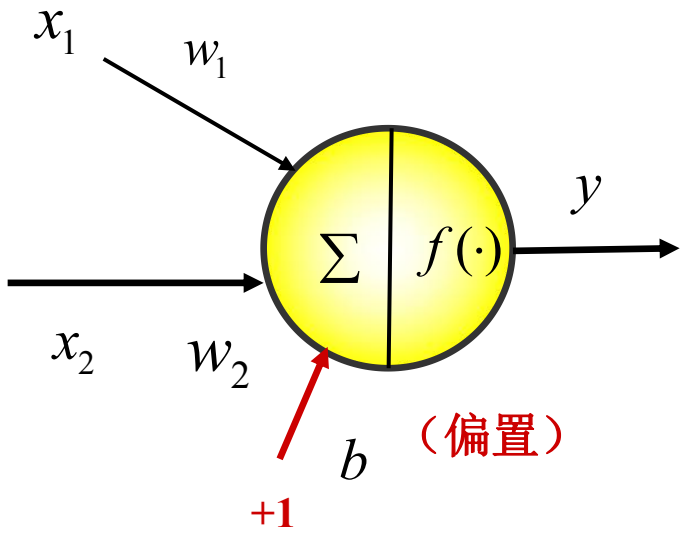
没有非线性变换的“激活函数”

简单的线性分类问题

特征向量：(x1, x2)







人工神经元的模型图

每一个数据点(x_1, x_2)，如何区分一个数据点是橙色的还是蓝色的？

分界线：$w_1x_1 + w_2x_2 + b = 0$

蓝色点：$w_1x_1 + w_2x_2 + b > 0$

黄色点：$w_1x_1 + w_2x_2 + b < 0$

$$y = f(w_1x_1 + w_2x_2 + b)$$

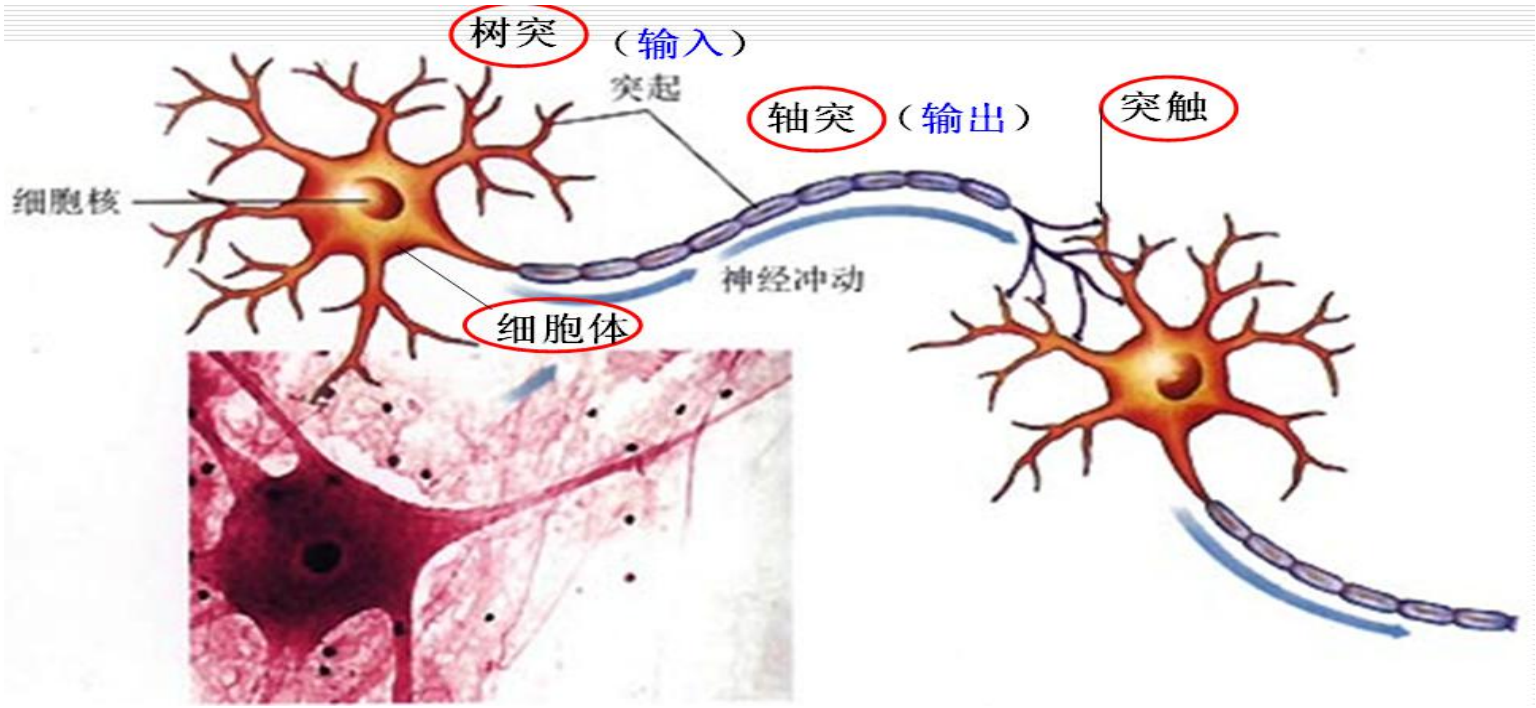
$f(\cdot)$ 可以采用**Linear, Sigmoid, Tanh, ReLU函数**

问题的关键：必须为 w_1、w_2 和 b(可作为w_0)找到合适的值——即所谓的参数值，然后指示计算机如何分类这些数据点

不同的激励函数应用时有什么区别？

8.1 神经元与神经网络

- 8.1.1 神经元模型
- 8.1.2 单神经元学习规则
- 8.1.3 人工神经网络



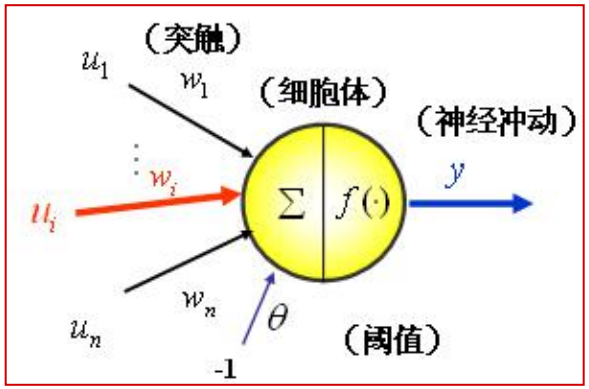
8.1.2 单神经元学习规则

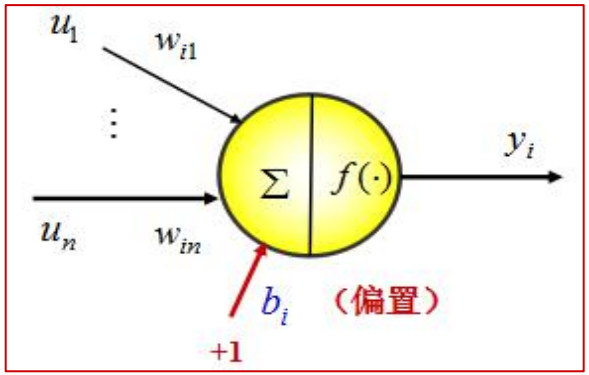
- 单神经元的学习：调整单神经元的连接权，使输入输出具有需要的特性。

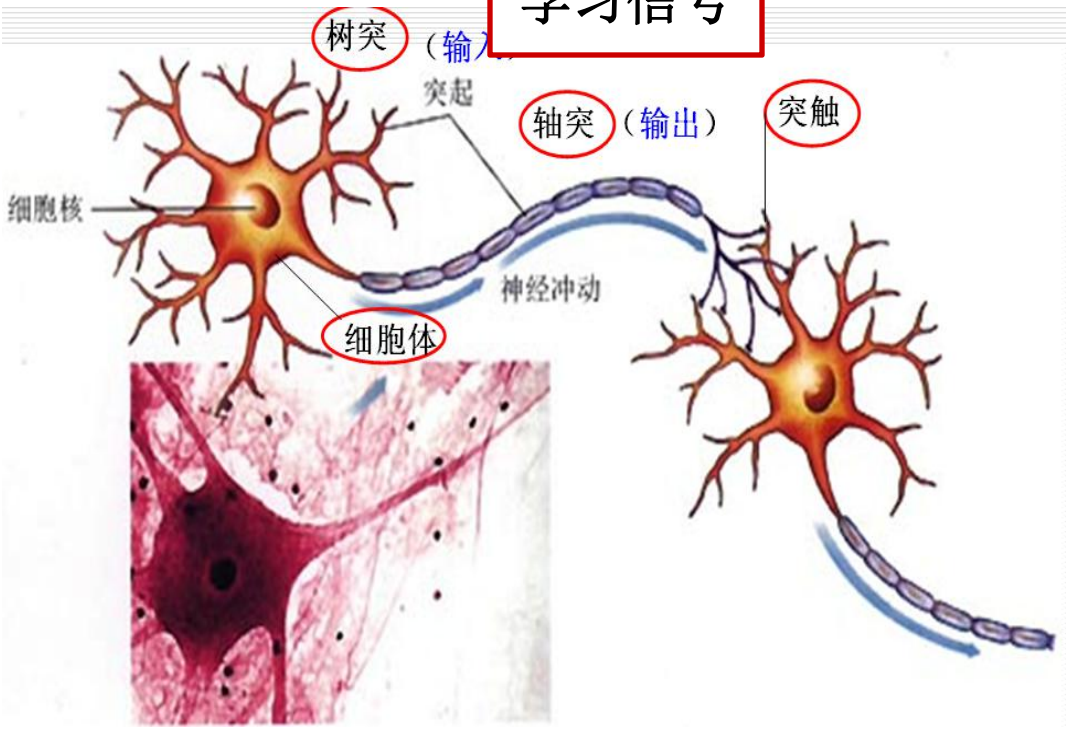
学习率

- 单神经元的连接权修正公式：$w_i(t+1) = w_i(t) + \eta z_i(t)$

学习信号







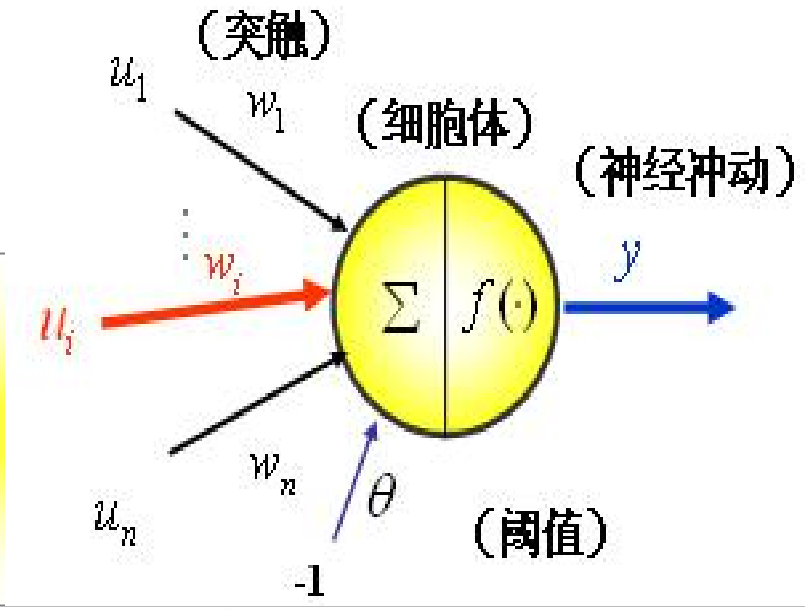
8.1.2 单神经元学习规则

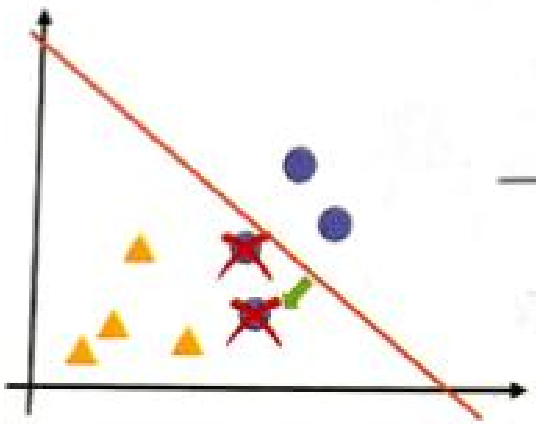
1. 误差纠正学习规则（delta学习规则）

$$\Delta w_i(t) = \eta u_i(t) e(t), \quad \eta > 0$$

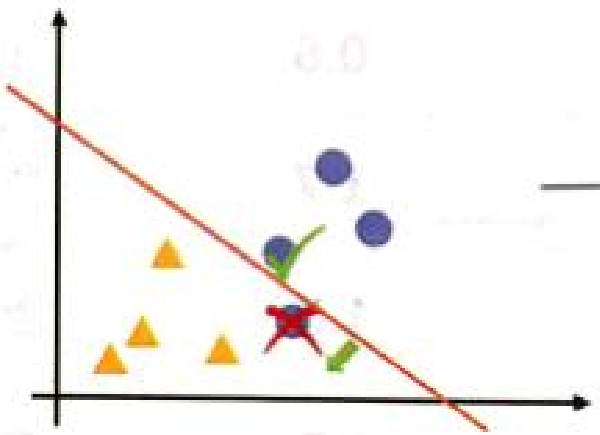
$$e(t) = d - y(t)$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

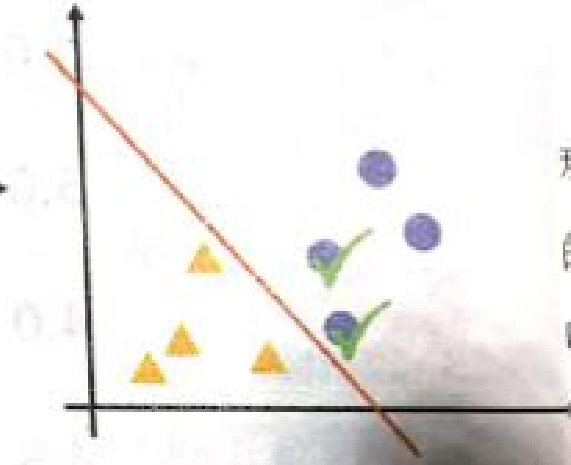




两个样本被分错，直线向误分类样本一侧移动



一个误分类样本被纠正，直线向另一误分类样本侧移动



直到所有训练数据都被正确分类

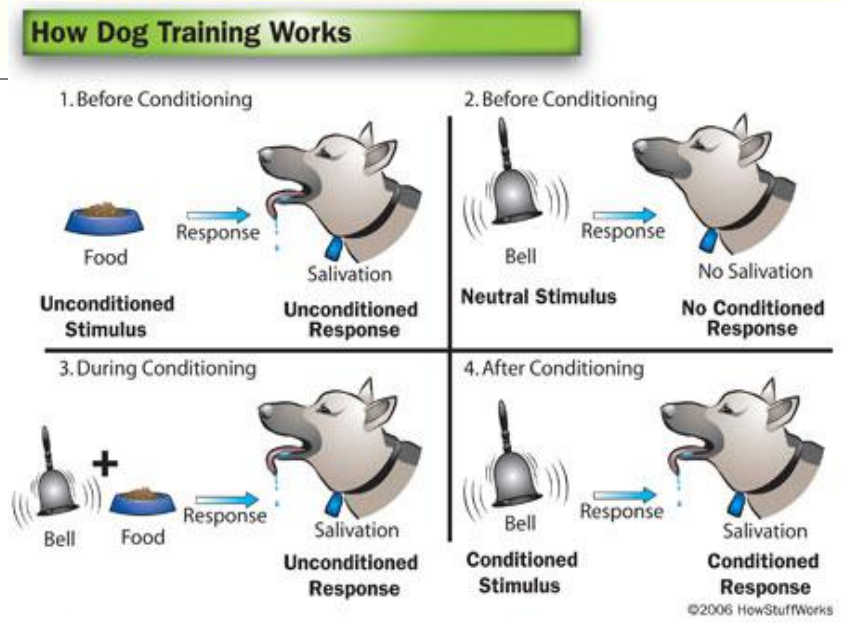
8.1.2 单神经元学习规则

2. Hebb学习规则（1944）

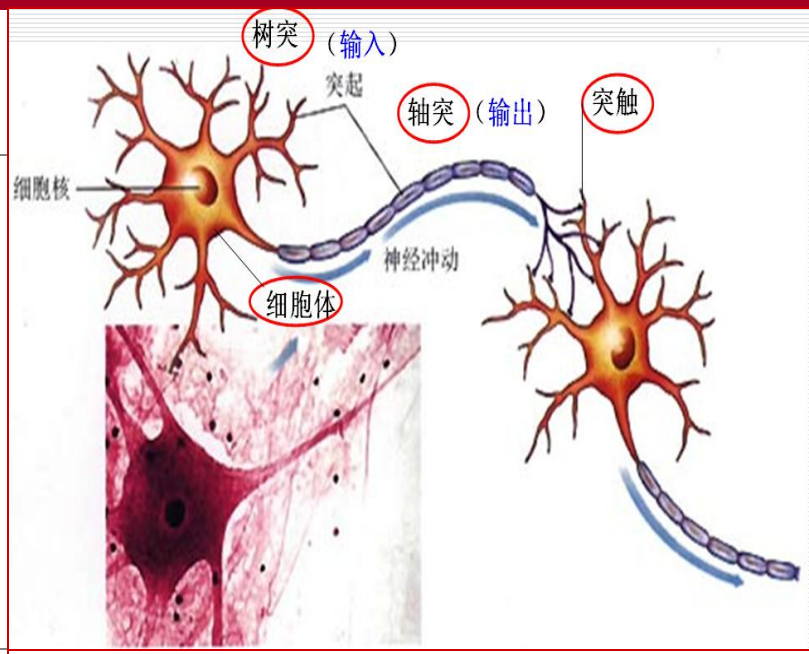
- 当某一突触两端的神经元同步激活时，该连接的强度增强，反之减弱。

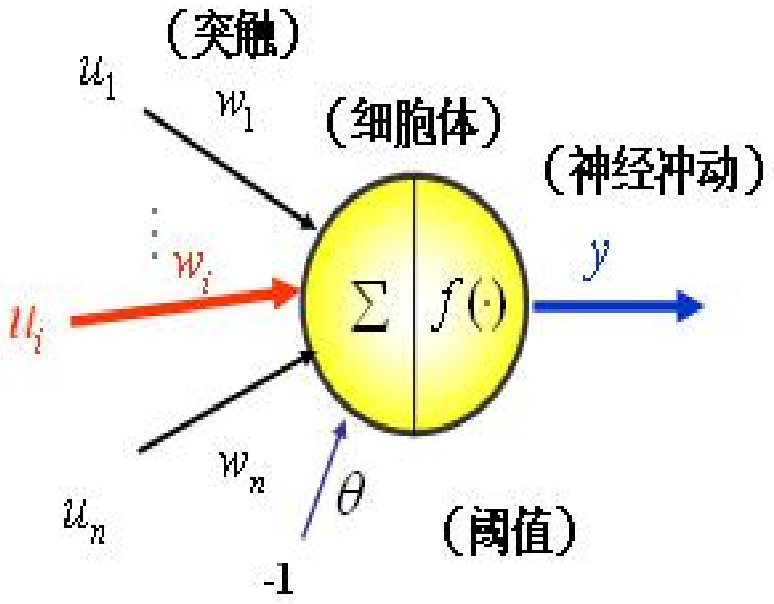
$$\Delta w_i(t) = \eta u_i(t) y(t), \quad \eta > 0$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$



巴甫洛夫的条件反射实验





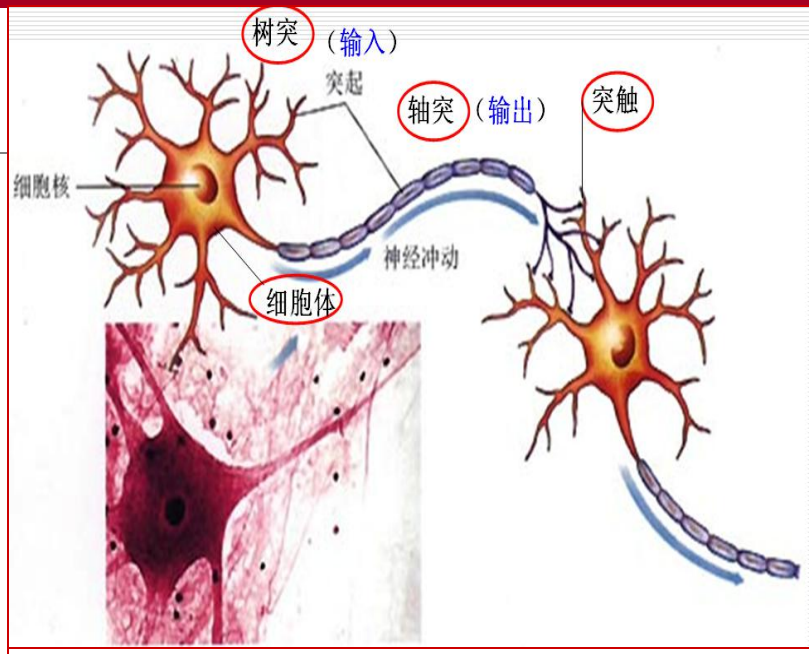
8.1.2 单神经元学习规则

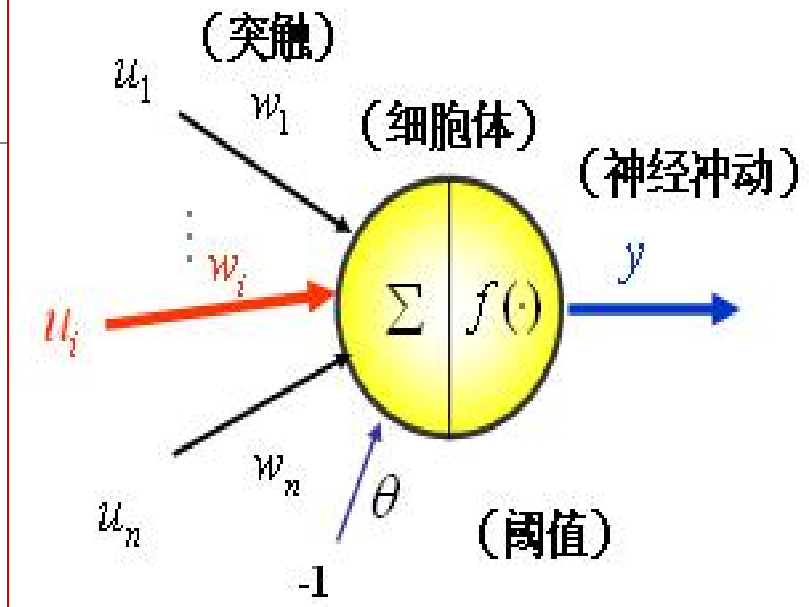
3. 竞争学习

- 以某种内部规则（与外部环境无关）确定竞争层获胜神经元，对获胜神经元与输入间的连接权值进行调整，其余不变。

$$\Delta w_i(t) = \eta[u_i(t) - w_i(t)], \quad \eta > 0$$

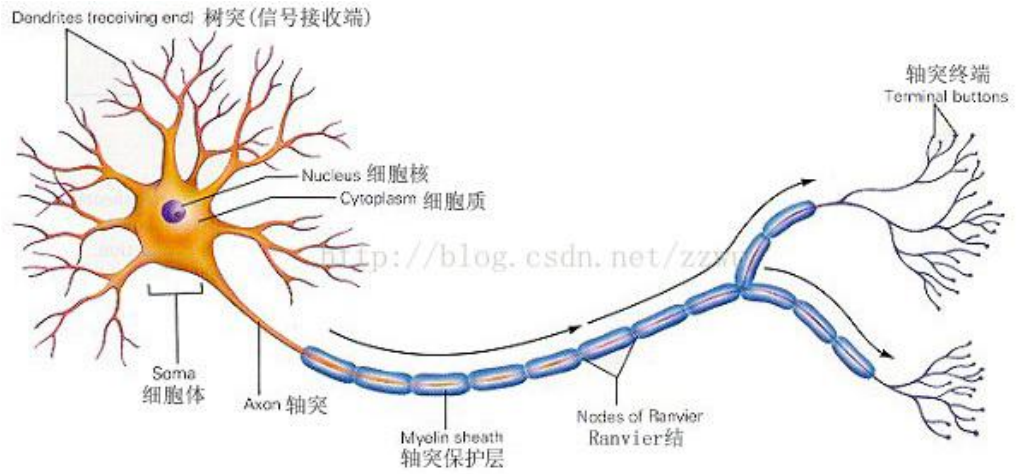
$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

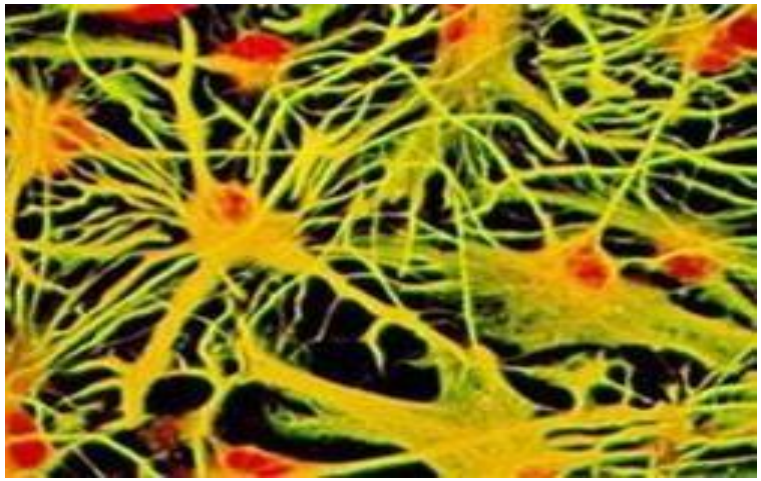




8.1 神经元与神经网络

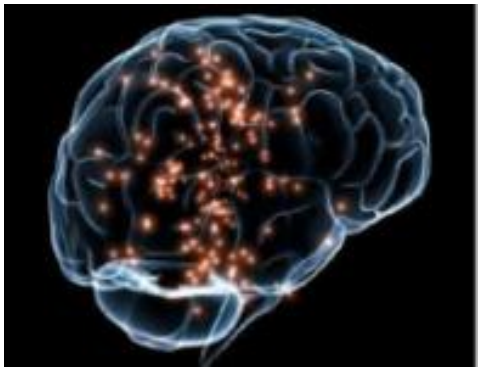
- 8.1.1 神经元模型
- 8.1.2 单神经元学习规则
- 8.1.3 人工神经网络

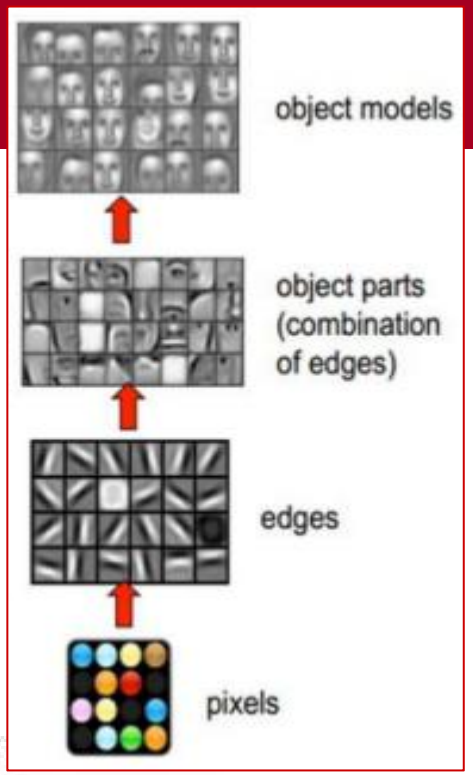




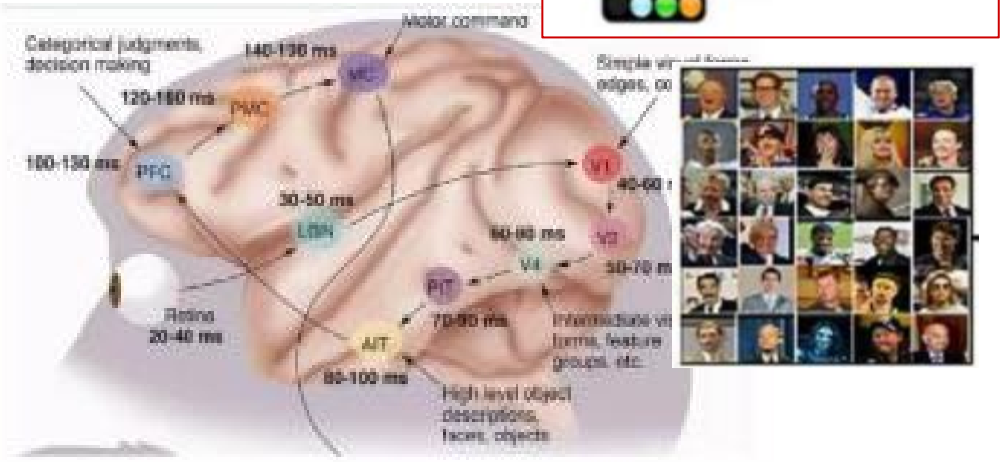
8.1 神经元与神经网络

- 8.1.1 神经元模型
- 8.1.2 单神经元学习规则
- 8.1.3 人工神经网络



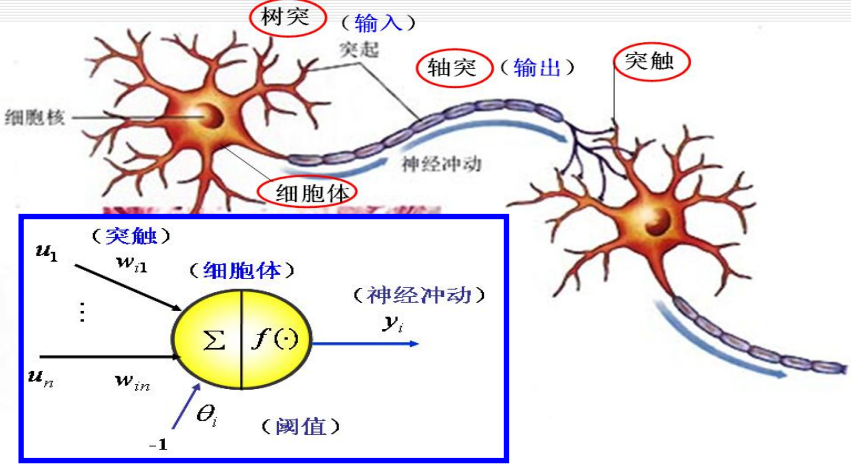


1958年，**约翰霍普金斯大学** David Hubel 和 Torsten Wiesel 发现人的视觉系统的信息处理是分级的，人对物品的识别可能是一个不断迭代不断抽象的过程。



视网膜得到原始信息后，经由区域V1初步处理得到边缘和方向特征信息，其次经由区域V2的进一步抽象得到轮廓和形状特征信息，如此迭代地经由更多更高层的抽象最后得到更为精细的分类。

8.1.3 人工神经网络

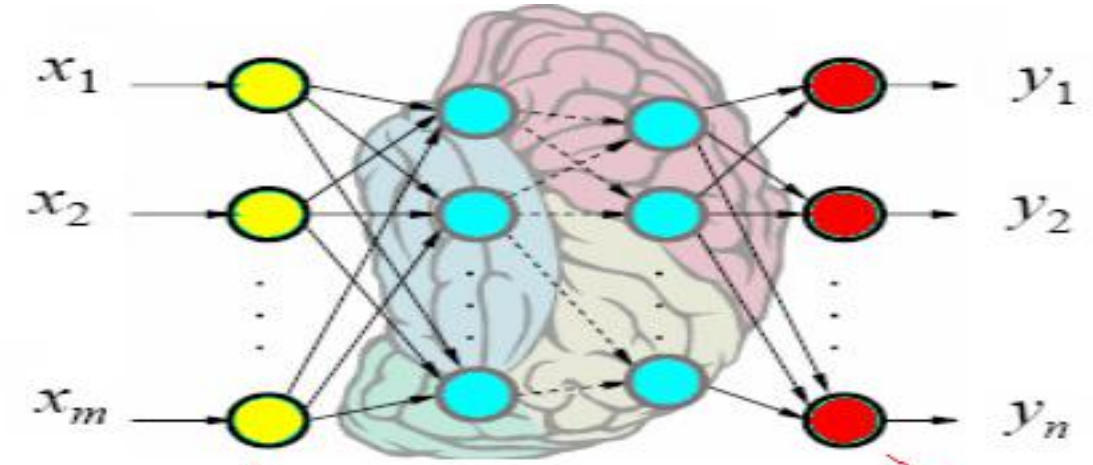


❑ 1. 神经网络的结构

（1）前馈型（ 前向型）

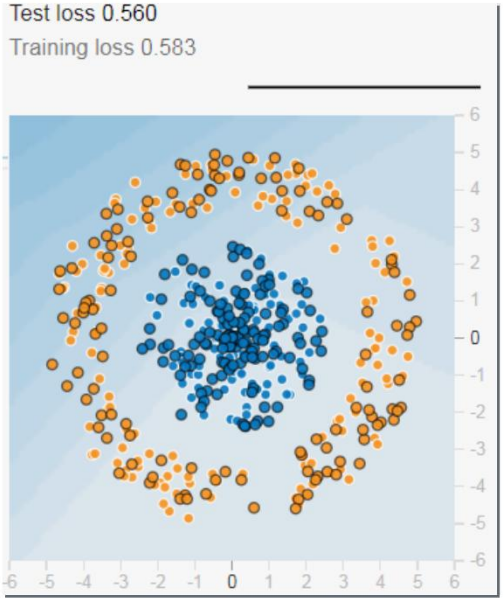
- 全连接：每个神经元都和下一层的所有神经元相连

BP神经网络

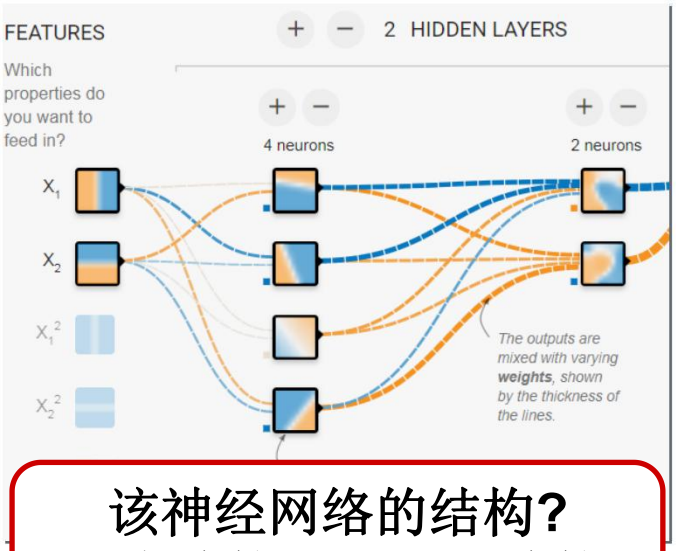


输入层不是神经细胞

卷积神经网络也是前馈型的，但是局部连接的

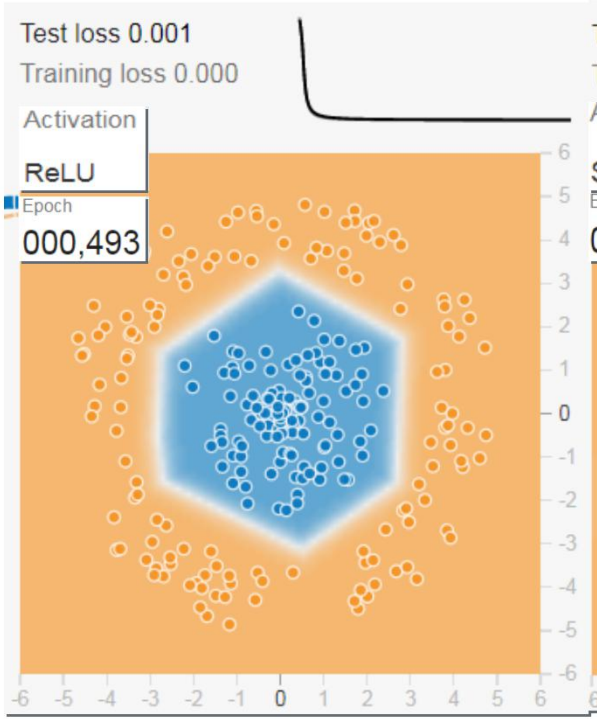


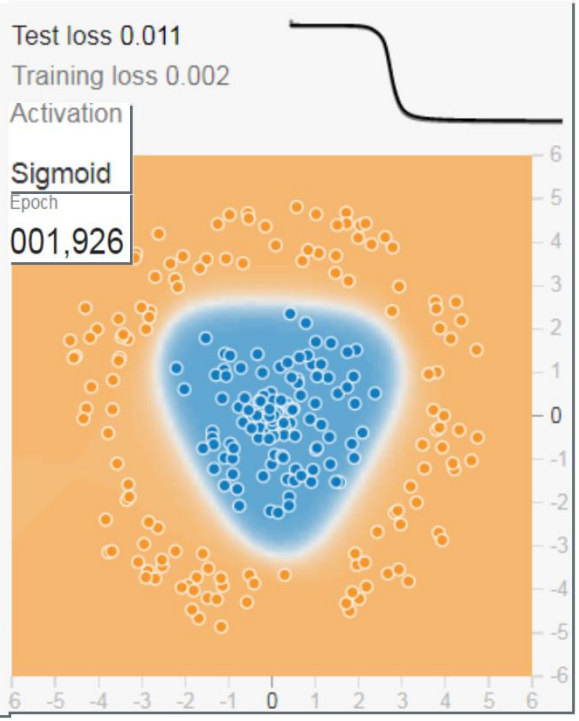
非线性分类问题

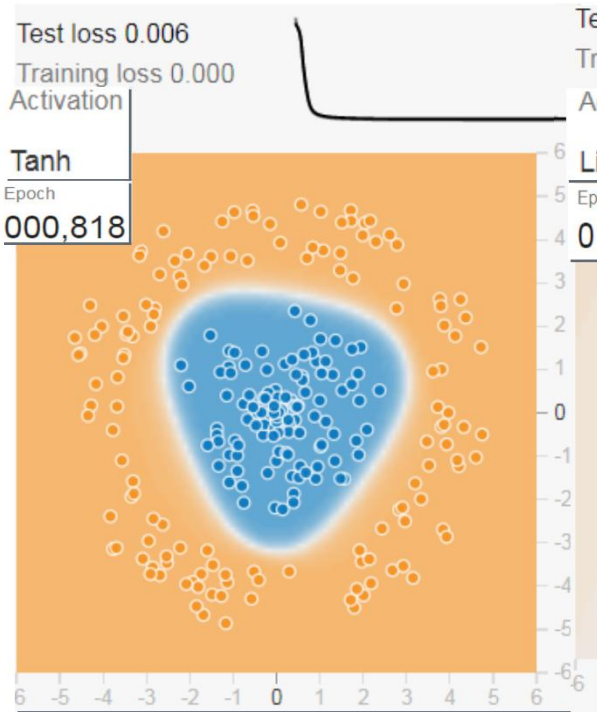


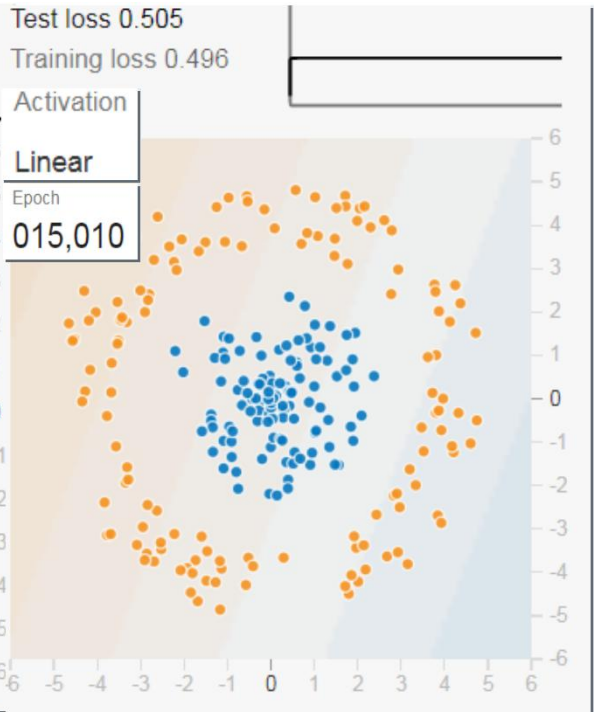
该神经网络的结构?

A.全连接　B.局部连接





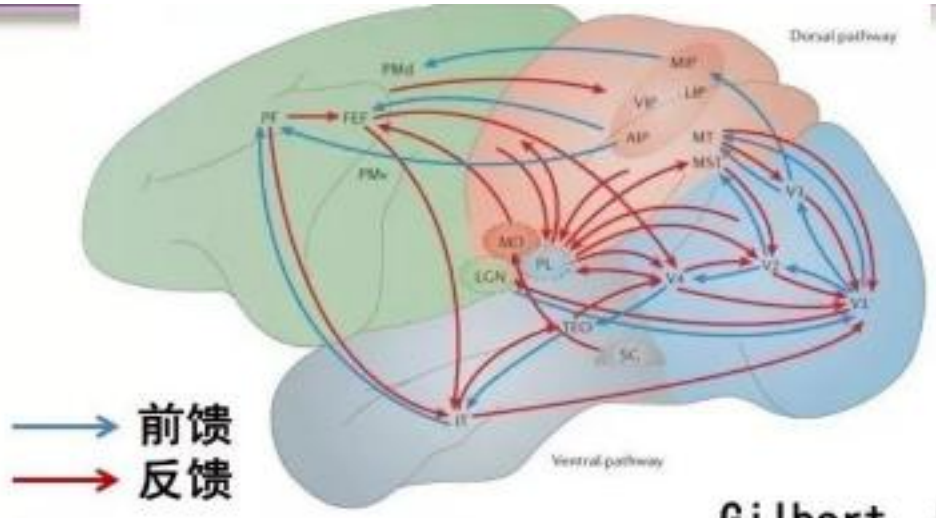




8.1 神经元与神经网络

❑ 1. 神经网络的结构

（2）反馈型



反馈连接在各级视觉皮层中广泛存在。例如初级视皮层（V1）4C 层神经元只有不超过8%的突触输入是来自于 LGN 前馈输入(Peters et al., 1994)

Gilbert, Li, 2013

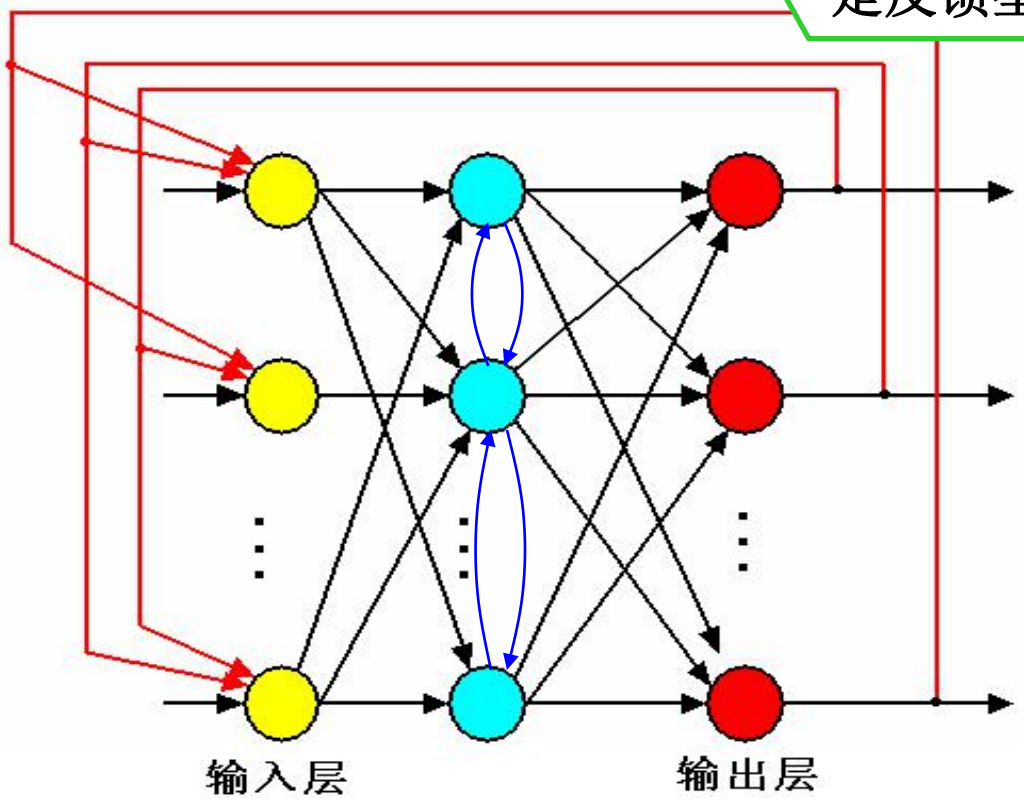
- 高级皮层的反馈对低纸皮层的反应进行调制
- 同级皮层中的反馈也对神经元的反应进行调制
- 现在的用于图像处理的深度学习模型主要前馈模型，只是视觉皮层的一个粗糙近似

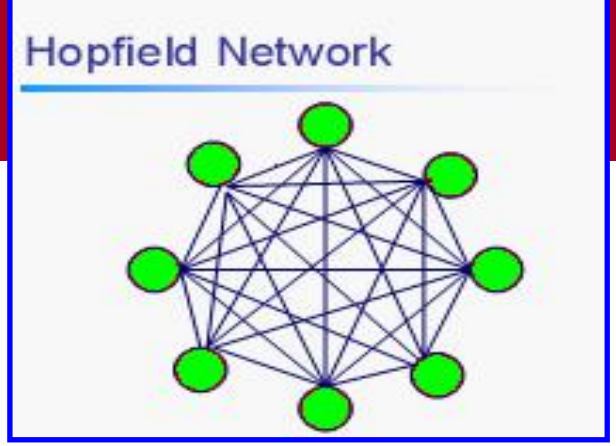
8.1.3 人工神经网络

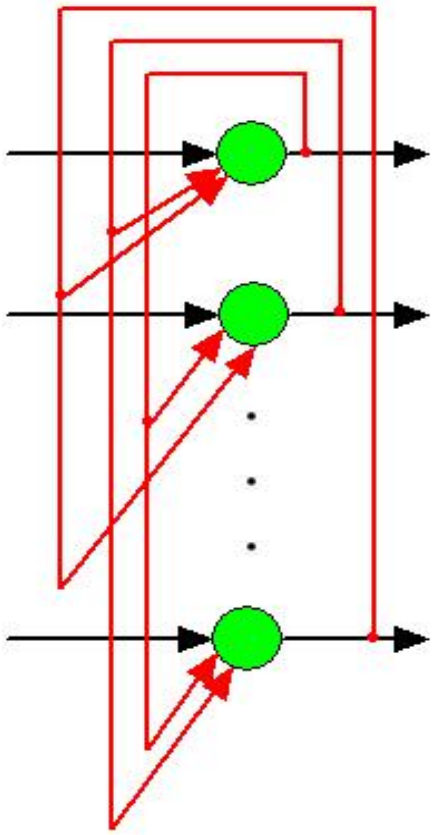
- 1. 神经网络的结构

（2）反馈型

递归神经网络（RNN，LSTM）也是反馈型的





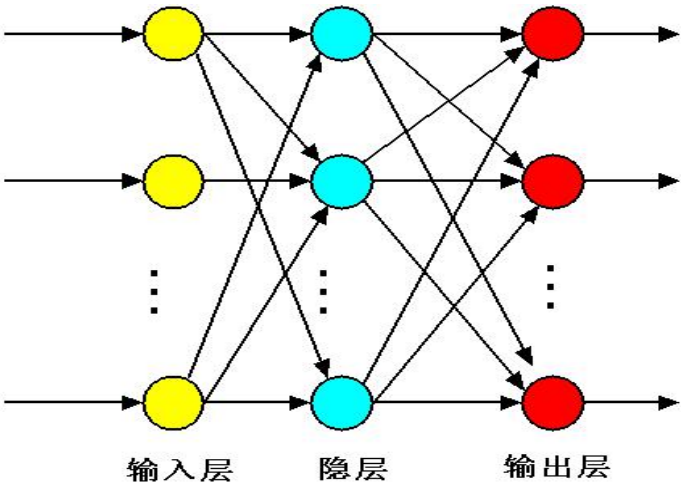


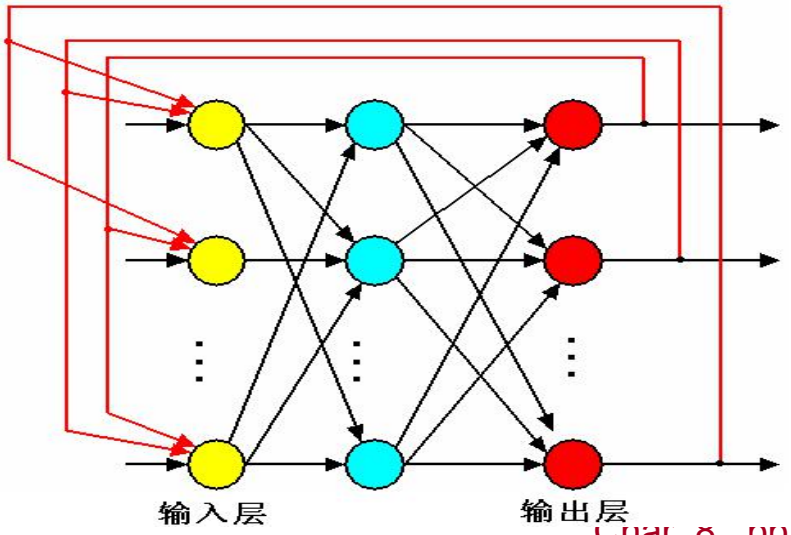
Hopfield神经网络

8.1.3 人工神经网络

❑ 2. 神经网络的工作方式

- 同步（synchronous, 或并行）方式：任一时刻神经网络中所有神经元同时调整状态；
- 异步（asynchronous, 或串行）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。

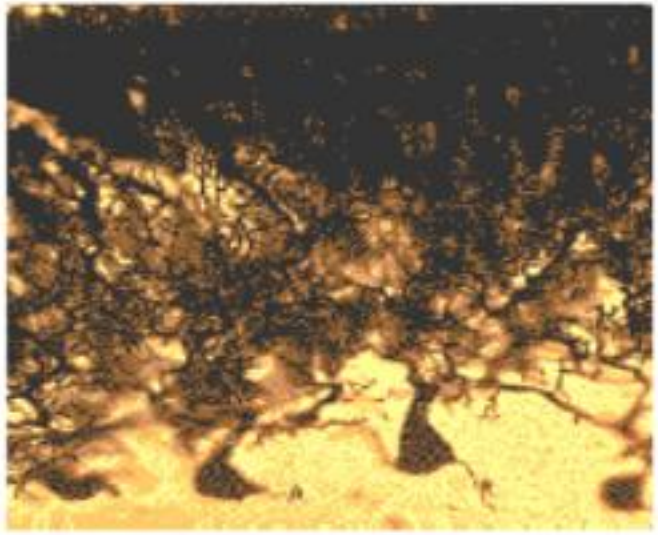


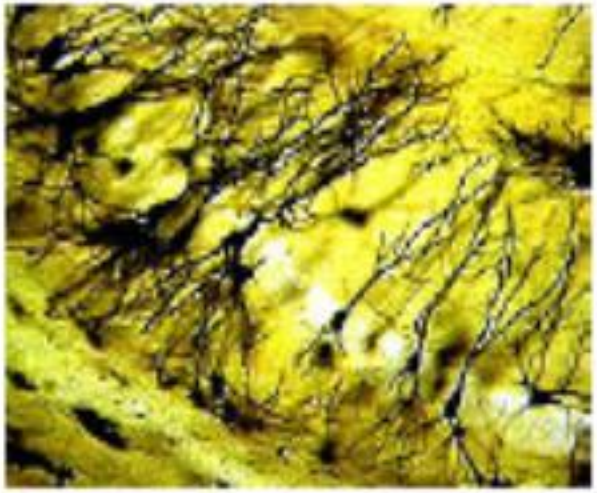


8.1.3 人工神经网络

- 决定人工神经网络性能的3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。

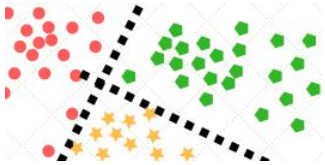


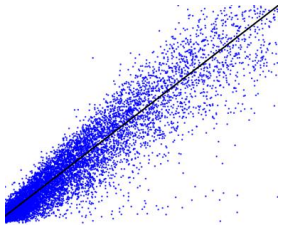


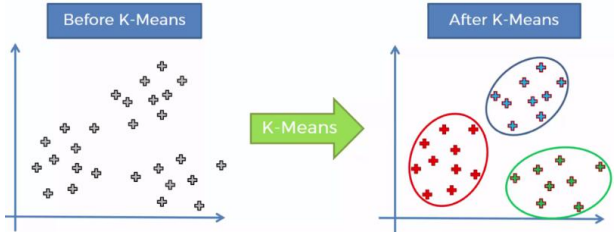
8.1.3 人工神经网络

❑ 3. 神经网络的学习方式

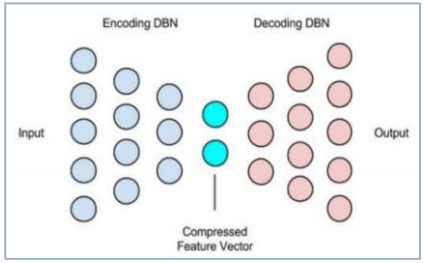
- ◆有监督学习
- ◆无监督学习
- ◆半监督学习

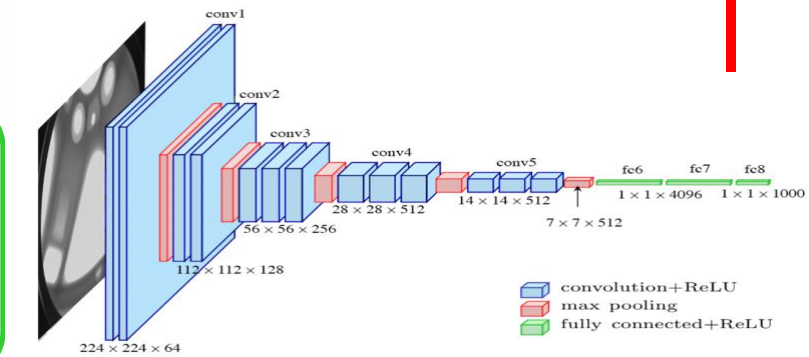




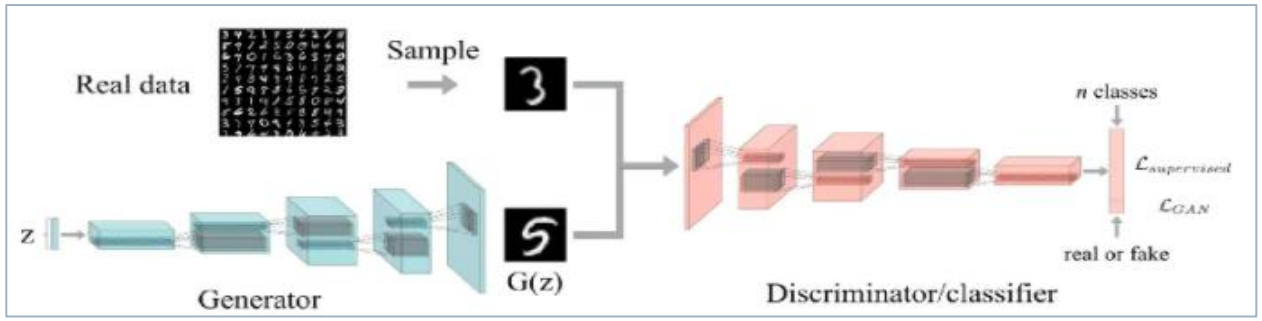


无监督学习：自组织神经网络、自动编码器等





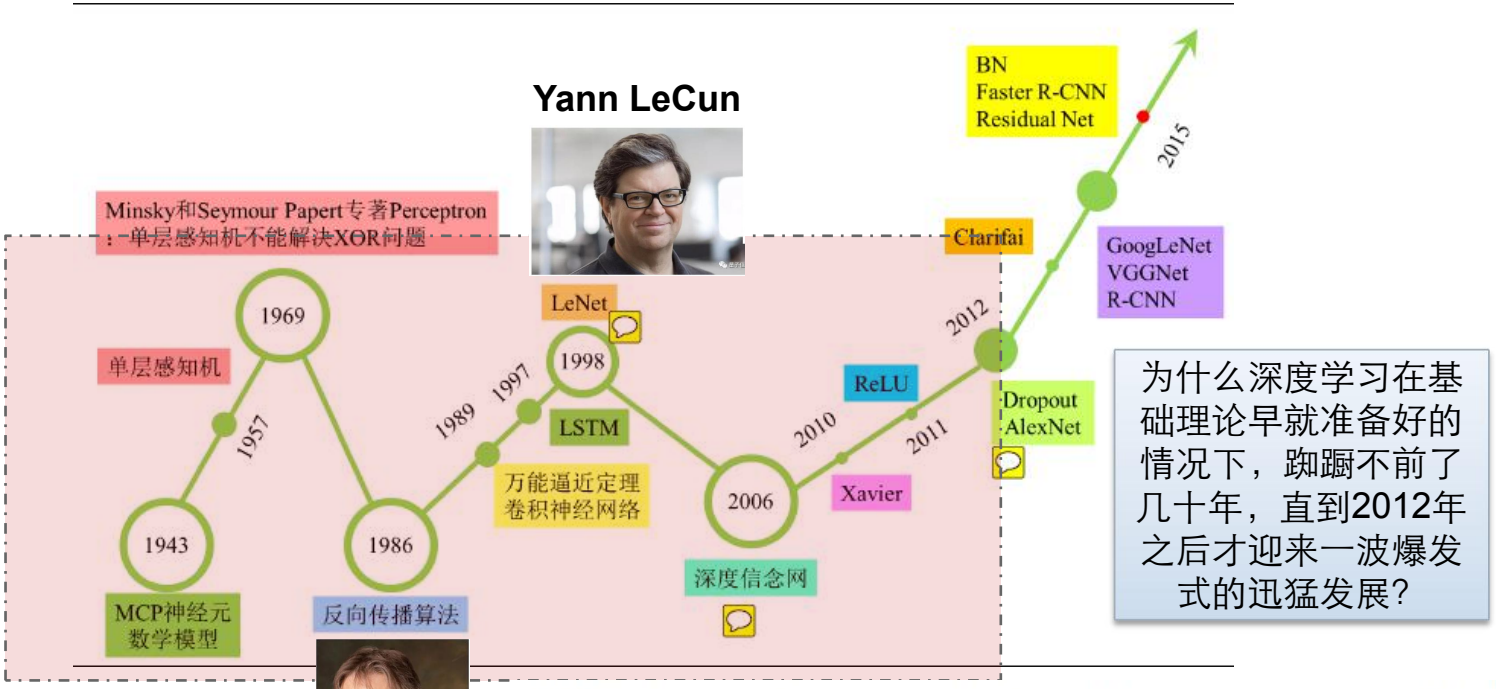
有监督学习：**BP神经网络**、**卷积神经网络**、胶囊网络、图卷积神经网络等



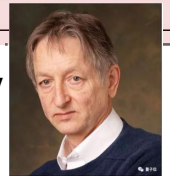
半监督学习：**生成对抗网络**等

8.1.3 人工神经网络

❑ 4. 神经网络的发展概况



本图来自于“深度学习大讲堂”微信公众号



2012年，Hinton课题组为了证明深度学习的潜力，首次参加ImageNet图像识别比赛，其通过构建的CNN网络**AlexNet**一举夺得冠军，且碾压第二名（SVM方法）的分类性能。也正是由于该比赛，CNN吸引到了众多研究者的注意。

Geoffrey Hinton， Yann LeCun，和Yoshua Bengio共同获得了2019年的图灵奖

Geoffrey Hinton
加拿大多伦多大学

Yann LeCun
纽约大学

Yoshua Bengio
加拿大蒙特利尔大学

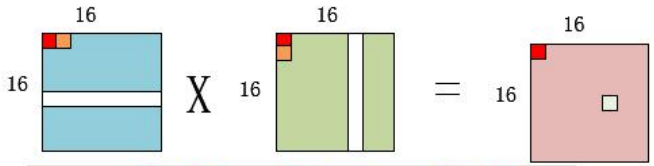
2019年3月27日 ——ACM宣布，深度学习的三位创造者Yoshua Bengio， Yann LeCun， 以及Geoffrey Hinton获得了2019年的图灵奖。

图片出处：https://tech.sina.com.cn/d/i/2019-03-27/doc-ihsxncvh6044779.shtml

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

从一个矩阵乘说起




```
float a[16][16], b[16][16], c[16][16];
```

CPU:

```
for(int i=0; i< 16; i++)
    for (int j=0; j<16; j++)
        for(int k=0; k<16; k++) {
            c[i][j] += a[i][k] * b[k][j
        }
```

Vector:

```
for(int i=0; i<16; i++)
    for ( int j =0; j<16; j++) {
        c[i][j] = a[i][:] *+  b[:][j
    }
```

CUBE:

```
CUBE:  c[:][:] = a[:][:] X b[:][:]
```

算力密度高

灵活

假设三种硬件都是一拍
完成一个基本操作

Cycle=16*16*16*2 = 8192
DataNum per cycle: Rd 2, Wr 1

Cycle=16*16 = 256
DataNum per cycle: Rd 2*16, Wr 1

Cycle=1
DataNum per cycle: Rd 2*16*16
Wr:16*16

华为昇腾**310/910** 芯片（**2018.10.10**），
同等功耗和面积下比**Nvidia V100/TPU**
极致算力都高

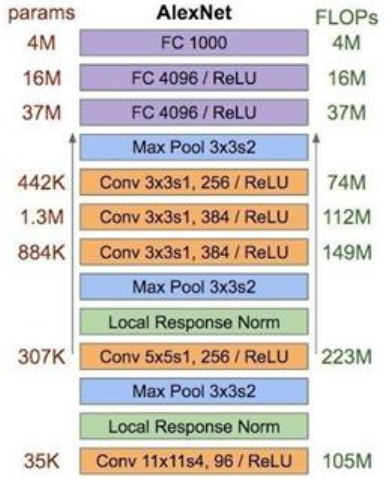


图2 AlexNet模型每层每秒浮点运算次数及参数数量

CNN经典模型的内存，计算量和参数数量对比

	AlexNet	VGG16	Inception-v3
模型内存(MB)	> 200	> 500	90-100
参数（百万）	60	138	23.2
计算量（百万）	720	15300	5000

从计算量角度，**99%**以上计算都是矩阵乘

图片出处：https://www.cnblogs.com/sindy-zhang/p/9012340.html

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

主流机器学习框架

百度PaddlePaddle(2016.9)、
华为MindSpore(2018.10)

库名	发布者	支持语言	支持系统
TensorFlow	Google	Python/C++/ Java/Go	Linux/Mac OS/ Windows/Android/iOS
Caffe	UC Berkeley	Python/C++/ Matlab	Linux/Mac OS/Windows
CNTK	Microsoft	Python/C++/ BrainScript	Linux/Windows
MXNet	DMLC (分布式机器学习社区)	Python/C++/Matlab/ Julia/Go/R/Scala	Linux/Mac OS/ Windows/Android/iOS
Torch/Pytorch	Facebook	C/Lua/Python	Linux/Mac OS/ Windows/Android/iOS
Theano	蒙特利尔大学	Python	Linux/Mac OS/Windows
Neon	Intel	Python	Linux

主流机器学习框架

百度PaddlePaddle(2016.9)、
华为MindSpore(2018.10)

库名	学习材料丰富程度	CNN建模能力	RNN建模能力	易用程度	运行速度	多GPU支持程度
TensorFlow	★★★	★★★	★★	★★★	★★	★★
Caffe	★	★★	★	★	★	★
CNTK	★	★★★	★★★	★	★★	★
MXNet	★★	★★	★	★★	★★	★★★
Torch	★	★★★	★★	★★	★★★	★★
Theano	★★	★★	★★	★	★★	★★
Neon	★	★★	★	★	★★	★★

主流机器学习框架

百度PaddlePaddle(2016.9)、
华为MindSpore(2018.10)

框架	github过去两年增长率		2019年7月		2017年9月		发布时间	公司	开发语言
	star	fork	star	fork	star	fork			
pytorch ↑↑↑	303%	394%	29635	7186	7361	1456	2016	facebook	c++, lua
tensorflow ↑↑	87%	121%	130540	75929	69781	34355	2015	google	c++
paddlepaddle ↑↑	71%	72%	9246	2484	5405	1447	2016	baidu	c++
mxnet ↑	56%	47%	17316	6145	11127	4179	2017	apache	c++
dl4j ↑	53%	31%					2014	eclipse	java
caffe2	50%	73%					2017	facebook	c++
caffe	41%	39%				2371	2014	berkeley vision	c++
cntk	31%	35%	16258	4316	12366	3190	2016	microsoft	c++
theano	28%	9%	8834	2493	6902	2290	2007	MILA	python
keras	-	-	42504	16187	-	-	2015	google	python
chainer	-	-	4887	1294	-	-	2015	chainer	python

用过或了解过这些平台？

A. 有　B.没有

十大深度学习框架GitHub数据变化(caffe，caffe2分开统计)

神经网络的种类

- 神经网络基础：单层感知器、线性神经网络、**BP**神经网络、**Hopfield**神经网络等
- 神经网络进阶：玻尔兹曼机、受限玻尔兹曼机、递归神经网络等
- 深度学习网络：深度置信网络、卷积神经网络、深度残差网络、**LSTM**网络、胶囊网络、生成对抗网络等
- 深度网络应用：应用于传统的数据挖掘与机器学习问题，手写体识别，图像识别，应用于自然语言处理，**AlphaGo**等