人工神经网络

Artificial Neural Network

人工神经网络

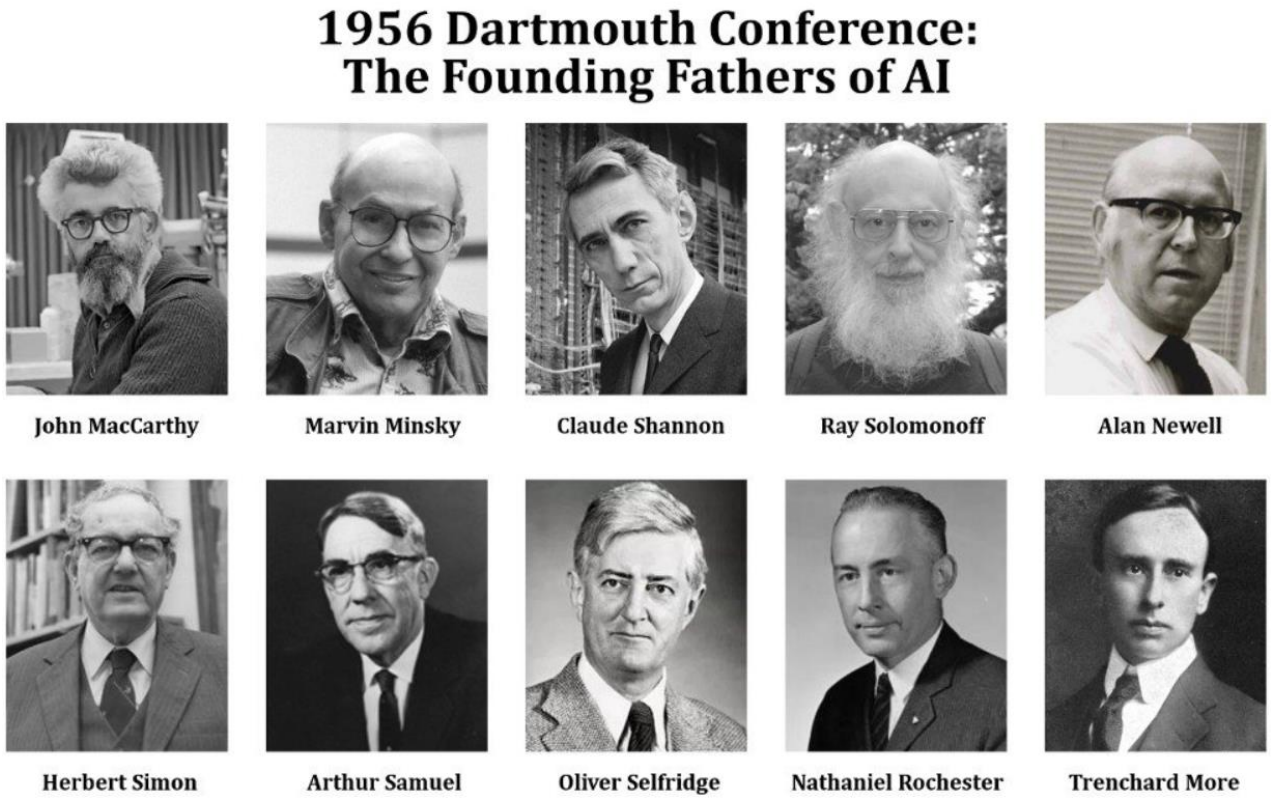
- 概述
- 神经元
- 多层感知机网络
  - 网络结构
  - BP算法

神经网络

- 概述
- 神经元
- 多层感知机网络
  - 网络结构
  - BP算法

概述

- 起源：1956年夏天，在美国东部的达特茅斯召开了一次觉有传奇色彩的学术会议，会上正式出现了**“人工智能”（Artificial Intelligence）**这个术语。在那里，人们首次决定将像人类那样思考的机器称为“人工智能”



概述

1955 年达特茅斯人工智能暑期研讨会建议书中所提的 7 个研究问题

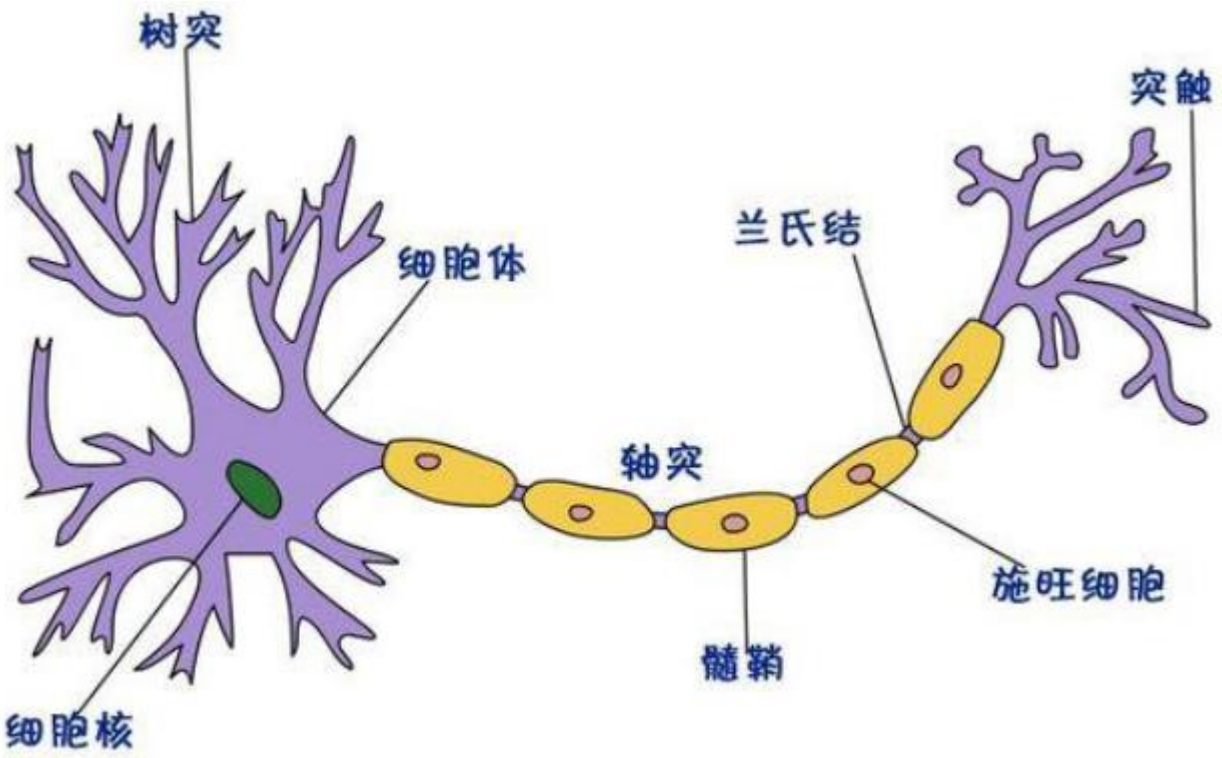
研究问题	当前的拓展性解读
自动计算机	通过编程让机器自动完成特定工作，如语音识别或图像分类
使用语言对计算机进行编程	通过 AI 编程语言来实现某些功能
神经网络	设计神经网络让计算机从感知数据中抽象出概念
计算复杂度	寻找一种方法来度量计算机完成特定任务时所耗费的运行时间和储存空间等开销
算法自我改进能力	AI 算法从过往经验中学习，不断提高其性能
算法抽象能力	从海量数据中归纳出数据中隐藏知识
算法随机性和创造力	模拟人类思维中所特有的创造能力

概述

- AI在其70年发展过程中形成了符号主义(symbolistic)、连接主义(connectionist)和行为主义(actionism)三大学派。
- 符号主义AI认为只要将人类所有知识符号化，把包罗万象、囊括万物的符号化知识组织起来，构建“知识水晶球”，形成如笛卡儿所言的“人类思想字母表”，就可以对所有未知问题进行推理，形成答案，应用于不同领域。
- 在行为主义AI中，学习信号以奖励形式出现，AI算法在与环境交互中去取得最大化收益，这种学习方式既不是从已有数据出发，也不依赖于已有知识，而是从“授之以鱼”迈向“授之以渔”，其代表实现路线是强化学习。

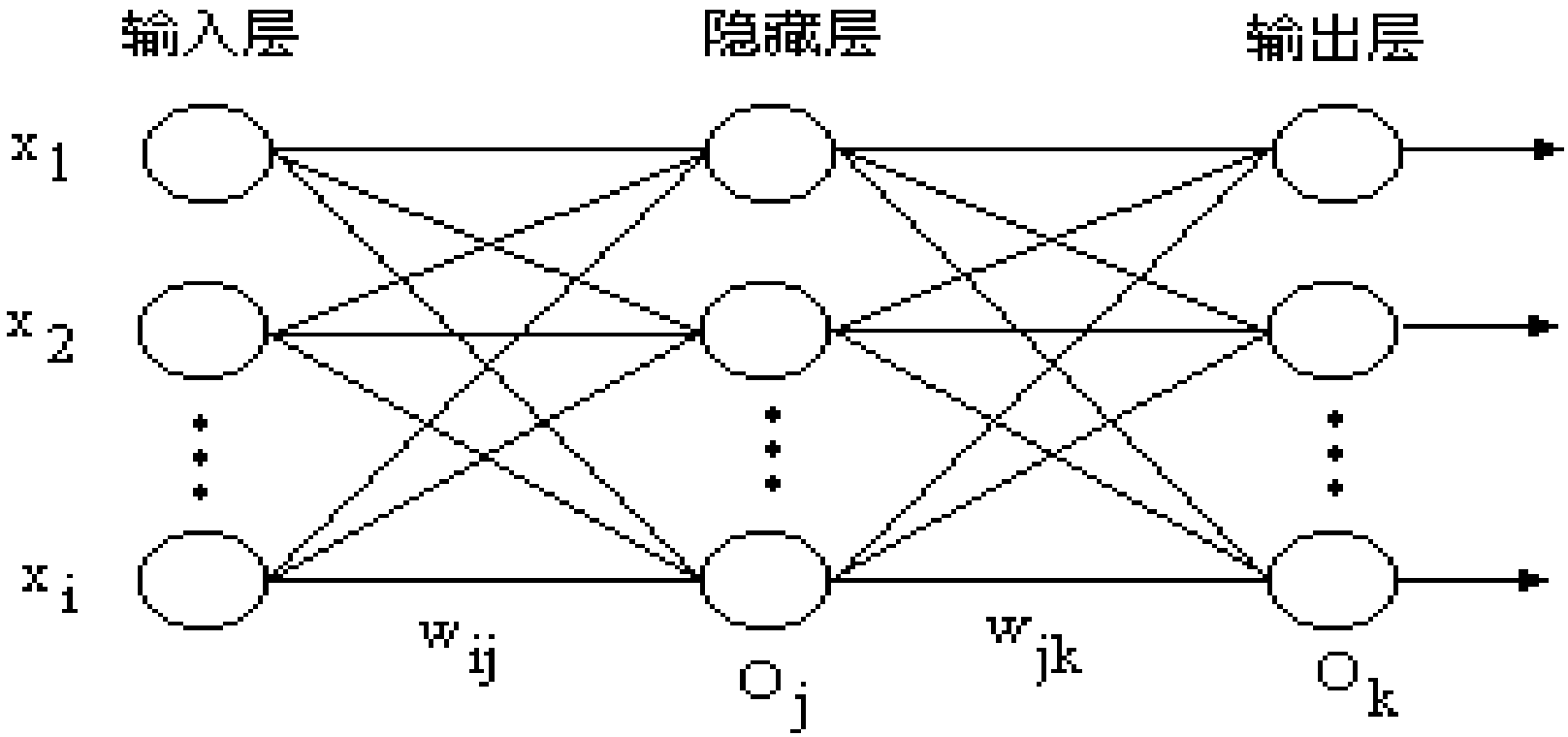
概述

- 所谓连接主义AI就是通过一种刻画人类大脑中神经元之间相互连接的机制，来模拟人类行为。这一方法的基本思想就是对海量数据进行归纳式学习，从数据中挖掘概念模式(即“用数据学”)，从而“化繁为简”



概述

- 人工神经网络最早由心理学家、神经学家、数学家共同提出，旨在寻求用于开发和测试神经元（人脑的基本工作单位）的数学模型。



概述

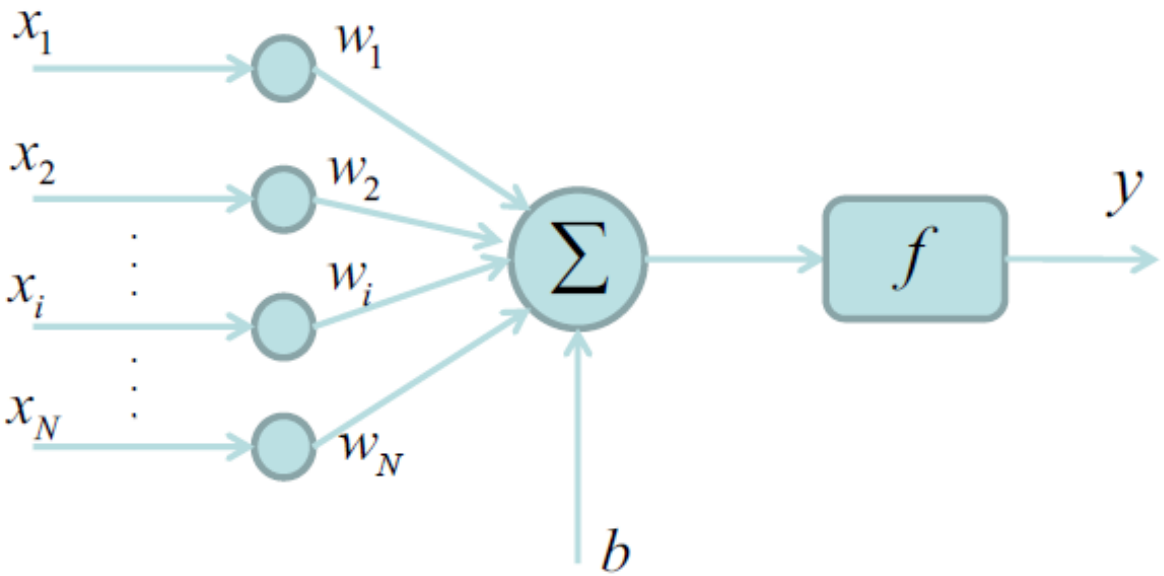
- 人工神经网络神经网络是一个很大的领域，本课程仅讨论最简单的神经网络形式，以及它与机器学习的交集
- 人工神经网络是一个具有适应性的简单单元组成的广泛并行互联的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应。
- 神经网络学得的知识蕴含在连接权重与阈值中

概述

- 发展历程（主要经历三个发展阶段）
  - **第一阶段（启蒙阶段）**
    - 1943年，神经生物学家MeCulloch(麦考克劳斯)和数学家Pitts(皮特斯)合作，提出了第一个神经元模型，从而开创了神经网络的研究。
    - 1949年，Hebb(哈伯)提出了神经元学习准则，为研究神经网络的学习算法奠定了基础。
    - 1958年，Rosenblatt(罗森布莱特)提出了感知器(Perception)模型，并首先将神经网络的研究付诸工程实践，从而激发了大批学者对神经网络的兴趣。

概述

- 感知机



$$y = f\left(\sum_{i=1}^{N} w_i x_i + b\right)$$

单层感知机的数学模型

概述

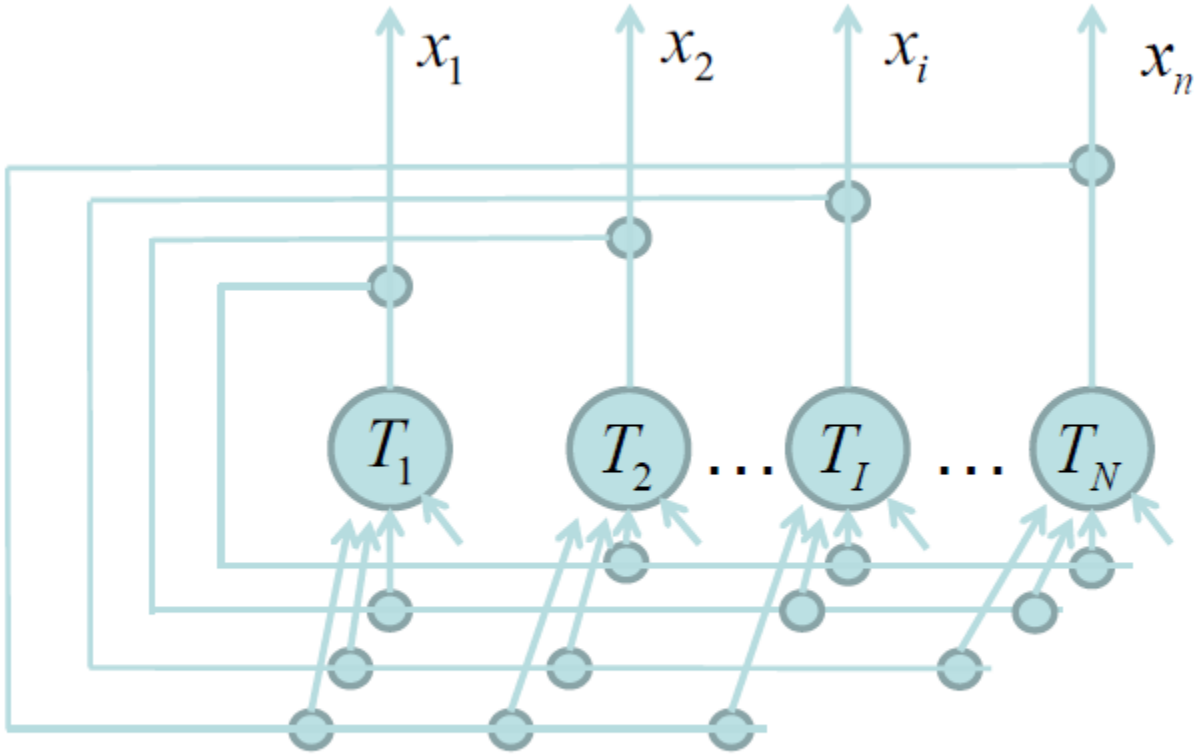
- 发展历程（主要经历三个发展阶段）
  - **第二阶段（低潮期）**
    - 1969年，著名的人工智能专家(也是人工智能的创始人之一)Minsky(明斯基)和Papert(帕伯特)对以感知器为代表的神经元模型进行了深入研究，并指出简单的线性感知器的功能是有限的，它无法解决线性不可分问题。
    - 这一论断给当时神经网络的研究带来了沉重打击，由此出现了神经网络发展史上长达10年的低潮期。

概述

- 发展历程（主要经历三个发展阶段）
  - **第三阶段（复兴时期）**
    - 1982年，Hopfield提出了著名的“Hopfield神经网络模型”，这个模型不仅对神经网络的信息存储和提取功能进行了非线性数学概括，还提出了相应的学习方法，使得神经网络的构造和学习有了理论指导。
    - 在Hopfield神经网络模型的影响下，大批学者又激发起研究神经网络的热情。
    - 此后，神经网络研究步入了一个新的发展时期；一方面，已有理论在不断深化和发展，另一方面，新的理论和方法也不断出现。

概述

- Hopfield神经网络模型

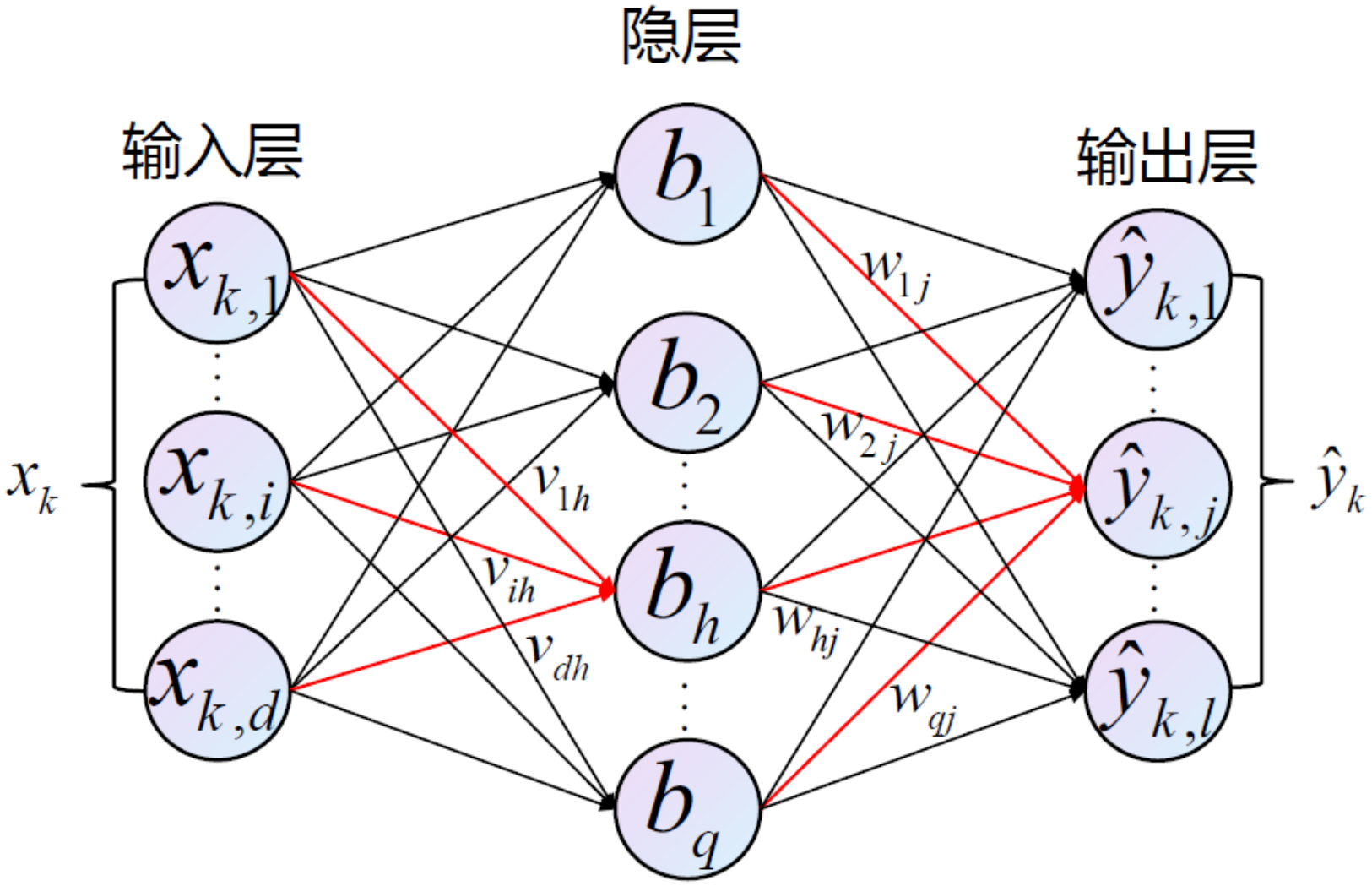


离散Hopfield神经网络模型

概述

- 发展历程（主要经历三个发展阶段）
  - **第三阶段（复兴时期）**
    - Paul J. Werbos 1974 年的哈佛大学博士论文中首次提出通过反向传播算法来训练人工神经网络而闻名，被称为“反向传播之父”
    - 1986 年，Hinton 和 Rumelhart、Williams 在nature发表了论文，详细介绍了“反向传播”技术。一直以来，反向传播被认为是深度学习的根基，也是第三次人工智能浪潮的重要推动因素
    - 1986年，Rumelhart和 McClelland为首的科学家提出了BP（Back Propagation）神经网络的概念，是一种按照误差逆向传播算法训练的多层前馈神经网络，**目前是应用最广泛的神经网络**。

概述


隐层
输入层
输出层
x_k
$x_{k,1}$
$x_{k,i}$
$x_{k,d}$
b_1
b_2
b_h
b_q
v_{1h}
v_{ih}
v_{dh}
w_{1j}
w_{2j}
w_{hj}
w_{qj}
$\hat{y}_{k,1}$
$\hat{y}_{k,j}$
$\hat{y}_{k,l}$
$\hat{y}_k$

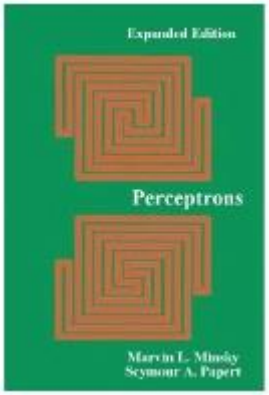
BP神经网络模型

神经网络发展回顾

1940年代 -萌芽期： M-P模型 (1943), Hebb 学习规则 (1945)

1958左右 -1969左右 ~繁荣期 ： 感知机 (1958), Adaline (1960), …

1969年： Minsky & Papert “Perceptrons”



冰 河期

1985左右 -1995左右 ~繁荣期 ： Hopfield (1983), BP (1986), …

1995年左右：SVM 及 统计学习 兴起

沉 寂期

2010左右 -至今 ~繁荣期 ： 深度学习

交替模式：
热十（年）
冷十五（年）

启示

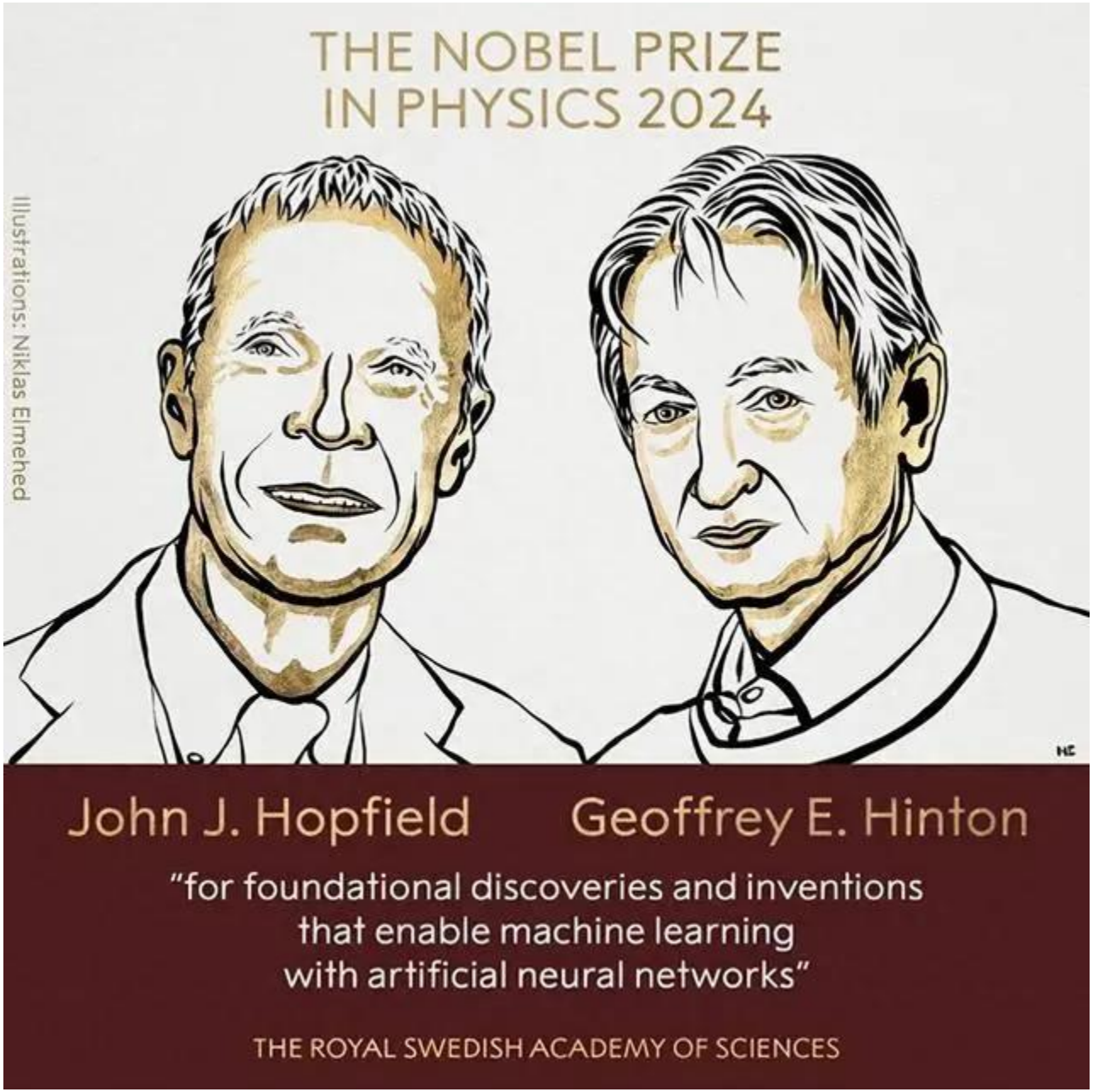
科学的发展总是“螺旋式上升”

三十年河东、三十年河西

坚持才能有结果！

追热门、赶潮流 —— 三思而后行

概述

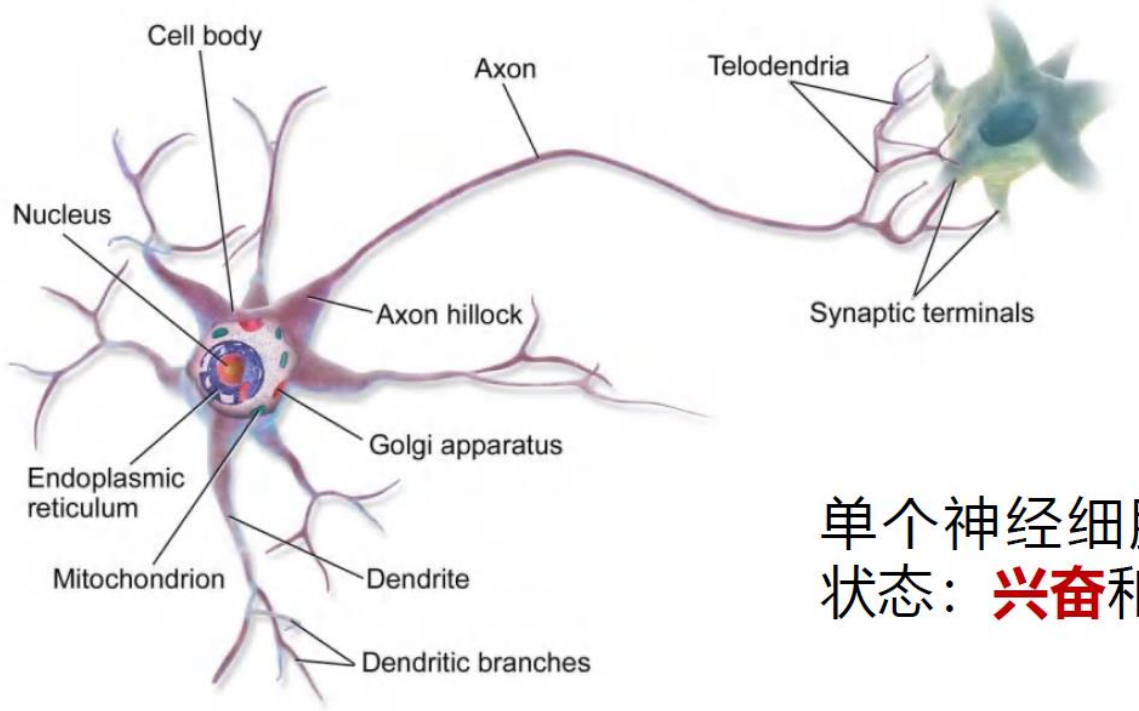

THE NOBEL PRIZE
IN PHYSICS 2024
Illustrations: Niklas Elmehed
John J. Hopfield
Geoffrey E. Hinton
"for foundational discoveries and inventions
that enable machine learning
with artificial neural networks"
THE ROYAL SWEDISH ACADEMY OF SCIENCES

概述

- 依据人工神经网络的结构和学习算法的不同，神经网络可大概分为4类：
  - 前馈神经网络：网络中各个神经元接受前一级的输入，并输出到下一级，网络中没有反馈，可以用一个有向无环路图表示。如多层感知机(Multi-layer Perception, MLP)。
  - 反馈神经网络：至少包含一个反馈回路—Hopfield神经网络、RNN。
  - 随机神经网络：引入随机机制，认为神经元是按照概率原理进行工作的，即：每个神经元的“兴奋”或“抑制”是随机的—玻尔兹曼神经网络。
  - 自组织竞争神经网络：其输出神经元之间相互竞争，竞争的胜利者用于输出—Hamming神经网络。

概述

- 本课程主要讨论“前馈神经网络—多层感知机(Multi-layer Perception, MLP)网络”的构造及其学习算法。



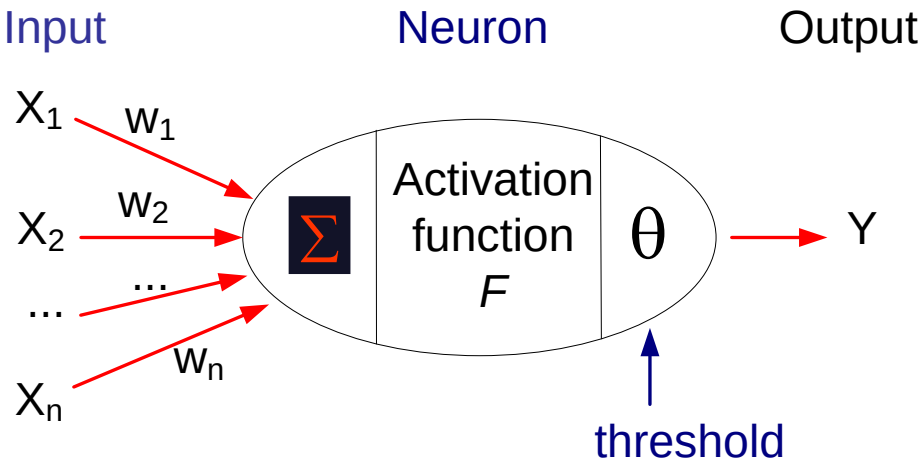
单个神经细胞只有两种状态：**兴奋**和**抑制**

神经网络

- 概述
- 神经元
- 多层感知机网络
  - 网络结构
  - BP算法

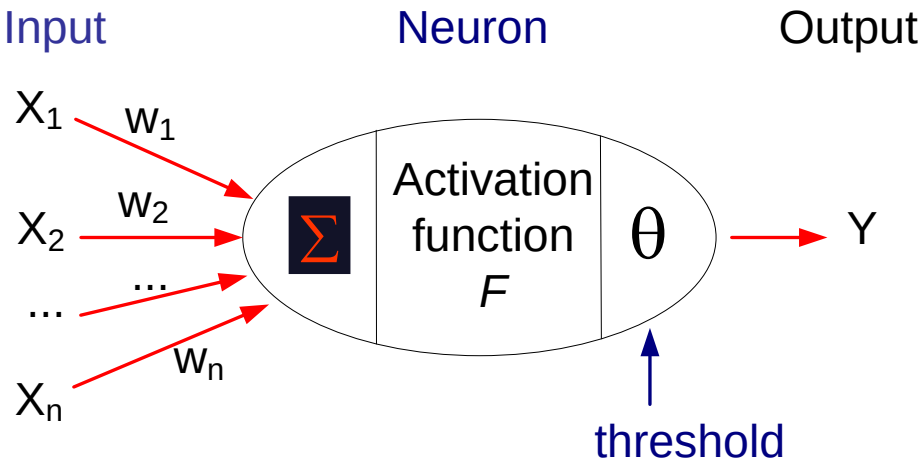
神经元

- 神经元是神经网络的基本计算单元，一个神经元是由多个输入、一个输出、一个内部反馈和阈值组成的非线性单元。
- 拓扑结构：



神经元

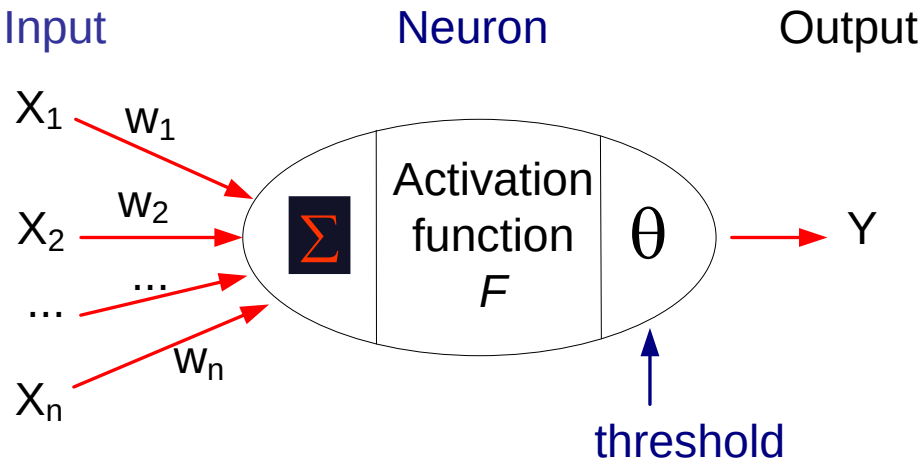
- 神经元是神经网络的基本计算单元，一个神经元是由多个输入、一个输出、一个内部反馈和阈值组成的非线性单元。
- 拓扑结构：



- 其中：$X_i(i=1,2,...,n)$是输入，$w_i(i=1,2,...,n)$是权值。

神经元

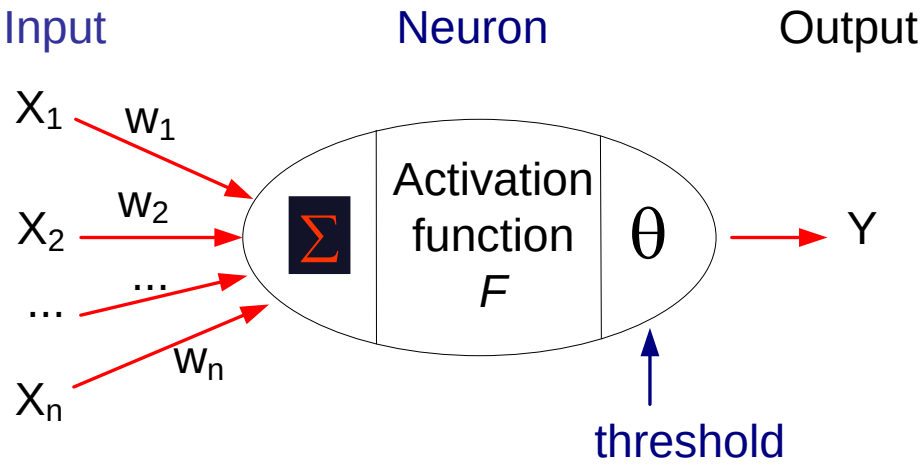
- 神经元是神经网络的基本计算单元，一个神经元是由多个输入、一个输出、一个内部反馈和阈值组成的非线性单元。
- 拓扑结构：



  - 其中： Σ为n个输入的加权和，即： $\Sigma w_i * X_i$。

神经元

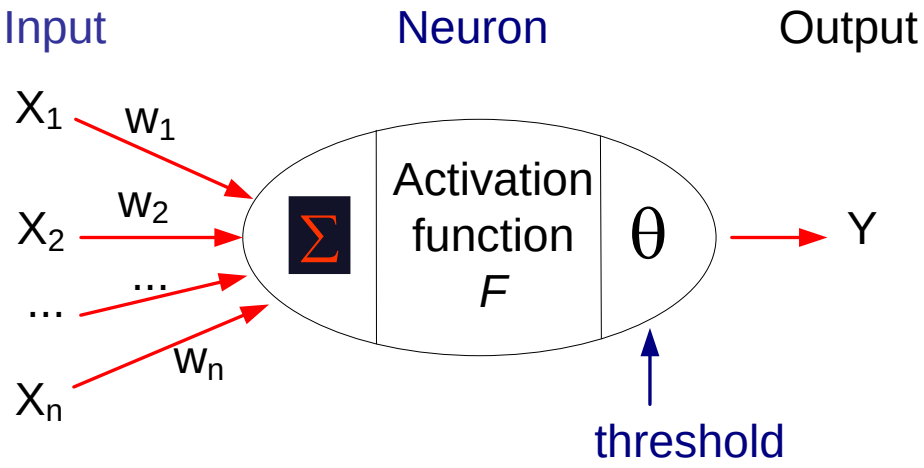
- 神经元是神经网络的基本计算单元，一个神经元是由多个输入、一个输出、一个内部反馈和阈值组成的非线性单元。
- 拓扑结构：



  - 其中： *F*表示神经元的激活函数，一般为S型。

神经元

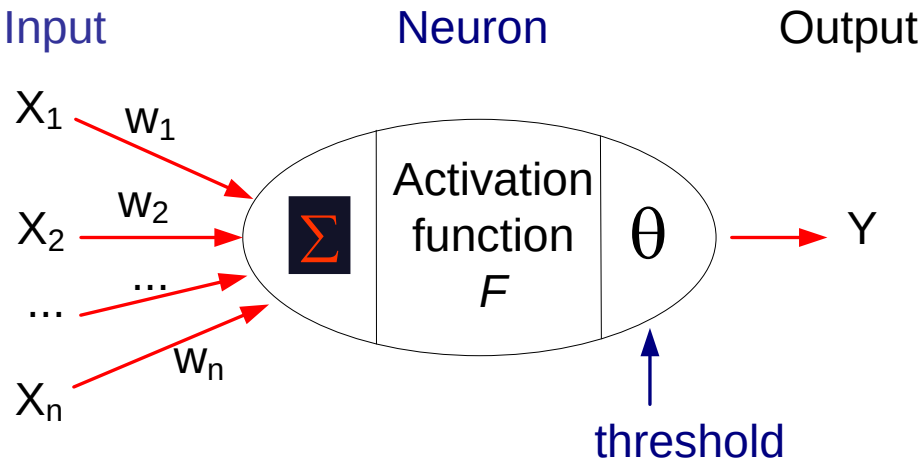
- 神经元是神经网络的基本计算单元，一个神经元是由多个输入、一个输出、一个内部反馈和阈值组成的非线性单元。
- 拓扑结构：



  - 其中：θ是阈值，用于构造神经元的输出。

神经元

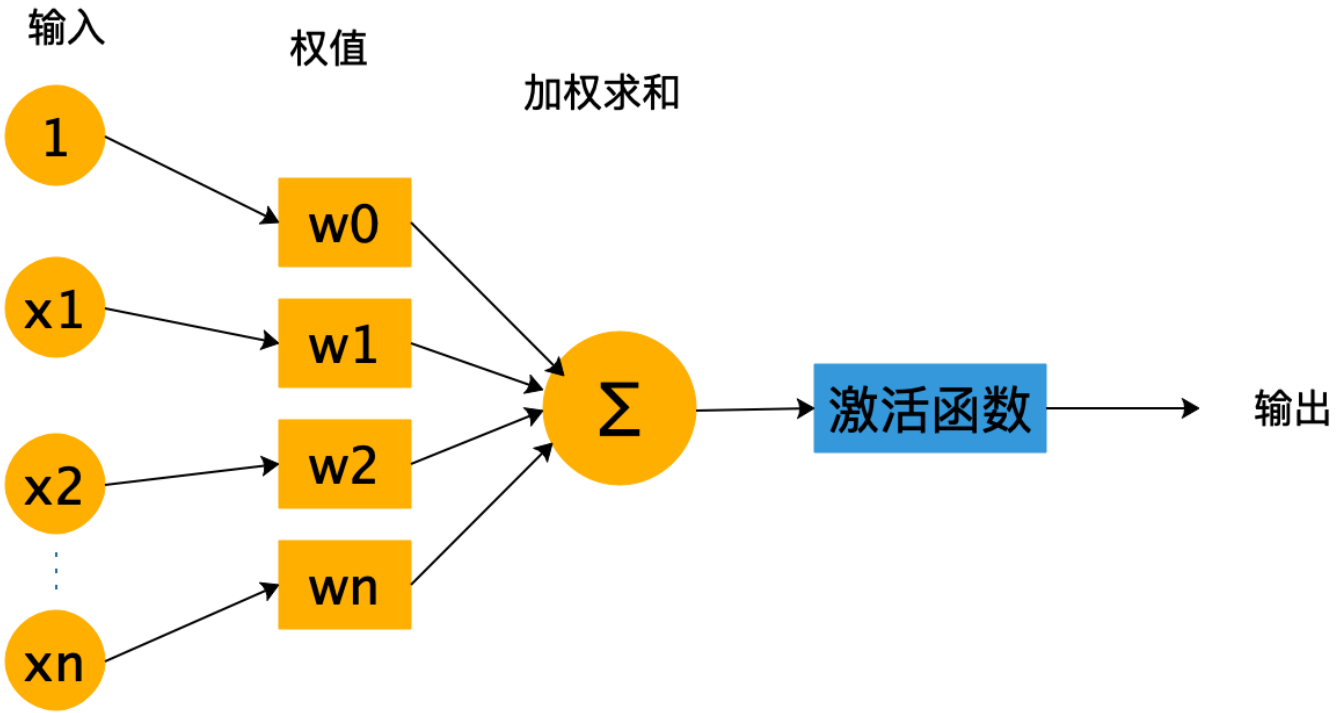
- 神经元是神经网络的基本计算单元，一个神经元是由多个输入、一个输出、一个内部反馈和阈值组成的非线性单元。
- 拓扑结构：



- 其中：Y是神经元的输出，且$Y=F(\Sigma w_i * X_i - \theta)$。

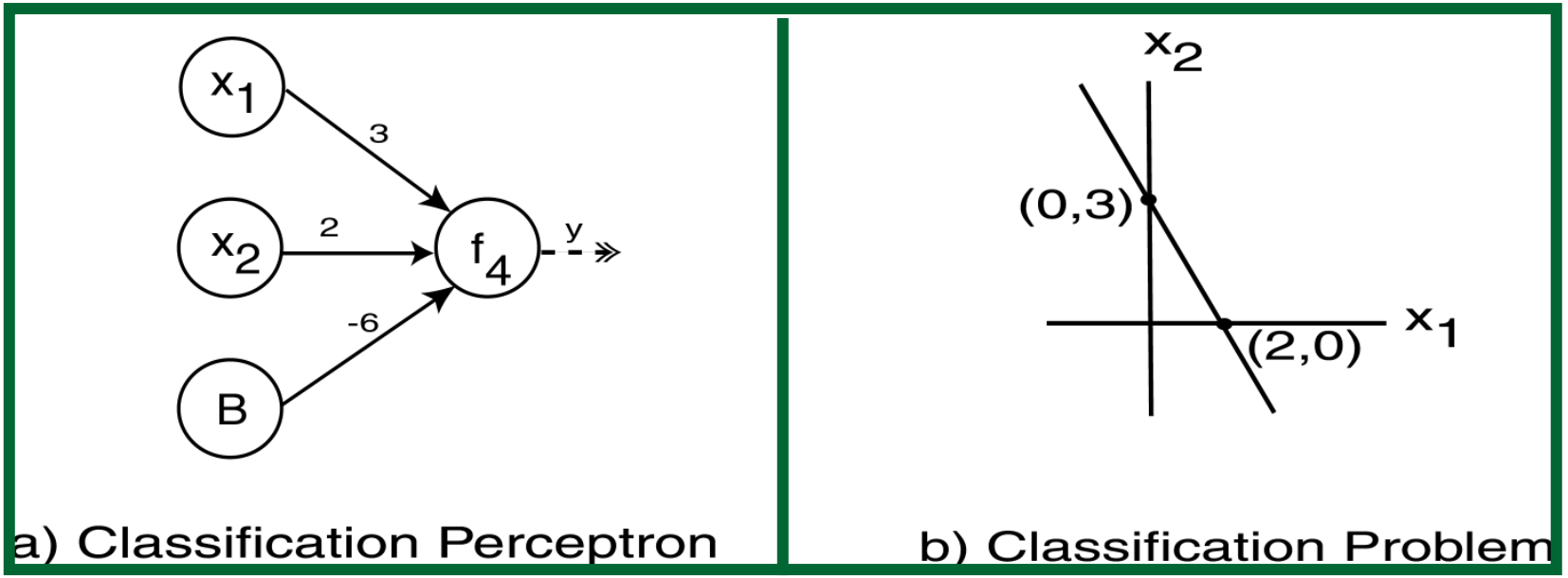
神经元

- 神经元是神经网络的基本计算单元，一个神经元是由多个输入、一个输出、一个内部反馈和阈值组成的非线性单元。
- 拓扑结构：



神经元：例子

- 假设：
  - 有图(a)所示的神经元结构—两类分类问题；
  - 激活函数：$F(x_1,x_2)=3x_1+2x_2-6$
  - 神经元的输出：*if* $F(x_1,x_2)>0$ *then* 输出1 *else* 输出0



神经元：激活函数

- **为什么需要激活函数？**
  - 如果没有激活函数，无论神经网络有多少层，神经网络输入到输出的映射都是输入的线性组合，则神经网络的拟合能力很弱！
  - 引入非线性函数作为激活函数，这样深层神经网络表达能力就更加强大，亦即，输出不再是输入的线性组合，几乎可以学习并拟合任意函数。

神经元：激活函数

- 连续并可导（允许少数点上不可导）的非线性函数。
  - 可导的激活函数可以直接利用数值优化的方法来学习参数
  - 非线性激活函数能够从输入输出之间生成非线性映射
- 激活函数及其导函数要尽可能的简单
  - 有利于提高网络计算效率
- 激活函数的导函数的值域要在一个合适的区间内
  - 不能太大也不能太小，否则会影响训练的效率和稳定性
- 激活函数通常要求有分界点，且在分界点两侧的函数值互不相同，即：严格单调。
- 理想激活函数是阶跃（符号）函数，0表示抑制神经元而1表示激活神经元

神经元：激活函数

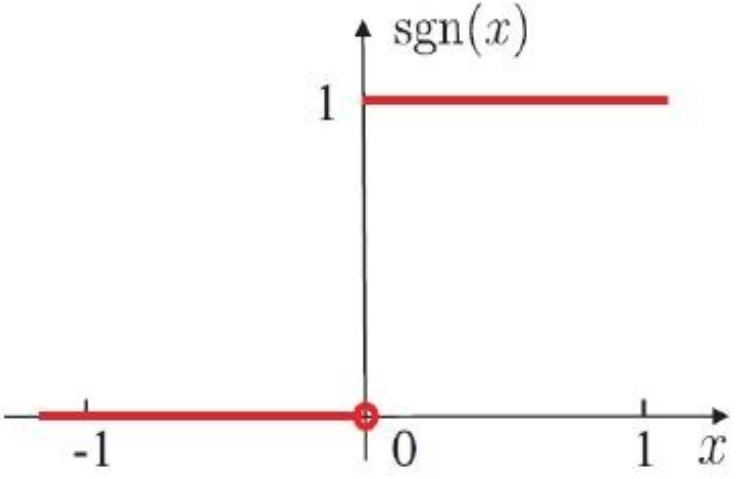
- 常用的激活函数：
  - 阶跃函数：不可微（0是分界点）

$$F(\sum w_i X_i - \theta) = sign(\sum w_i X_i - \theta) = \begin{cases} +1 & 当 \sum w_i X_i > \theta \\ -1 & 当 \sum w_i X_i < \theta \end{cases}$$

  - 阶跃函数具有不连续、不光滑等不好的性质，常用的是 Sigmoid 函数
  - Sigmoid函数：连续、可微（0.5是分界点）

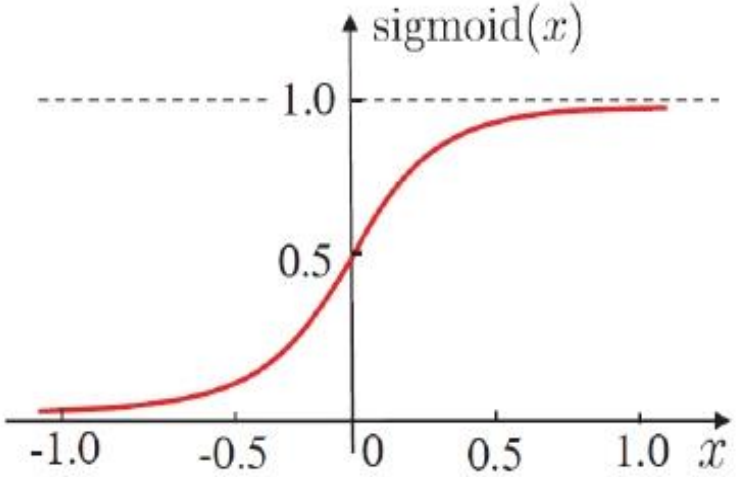
$$F(\sum w_i X_i - \theta) = (1 - e^{-(\sum w_i X_i - \theta)})^{-1} = \begin{cases} 1 & 函数值 \geq 0.5 \\ 0 & 函数值 < 0.5 \end{cases}$$

神经元：激活函数



$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

(a) 阶跃函数



$$\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

(b) Sigmoid 函数

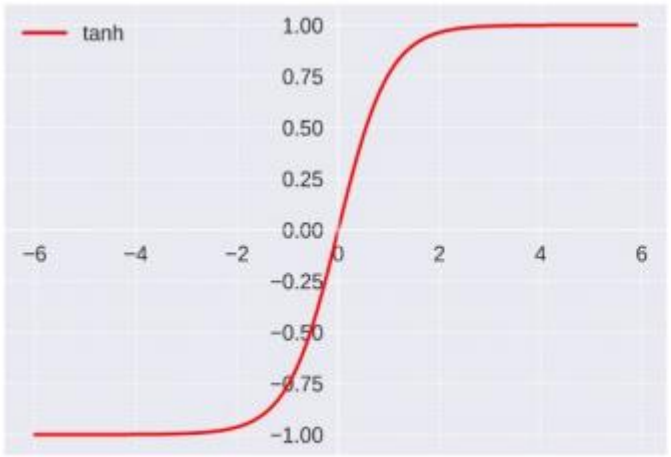
图 5.2 典型的神经元激活函数

神经元：激活函数

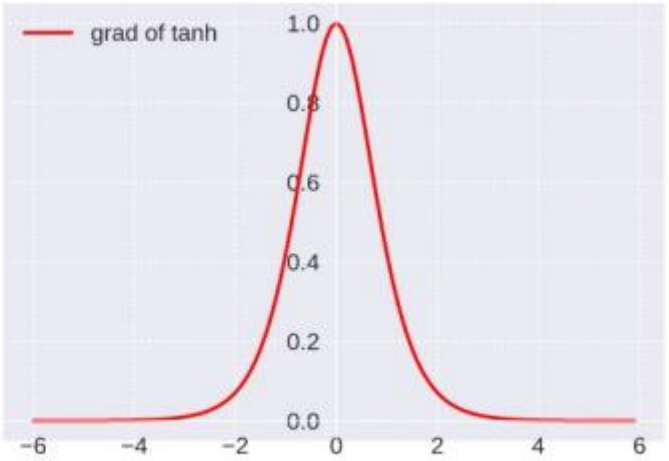
- 常用的激活函数：
  - Tanh 函数：也是一种 Sigmoid 型函数

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

  - 可以看作放大并平移的 Logistic 函数，其值域是 (−1, 1)



Tanh函数



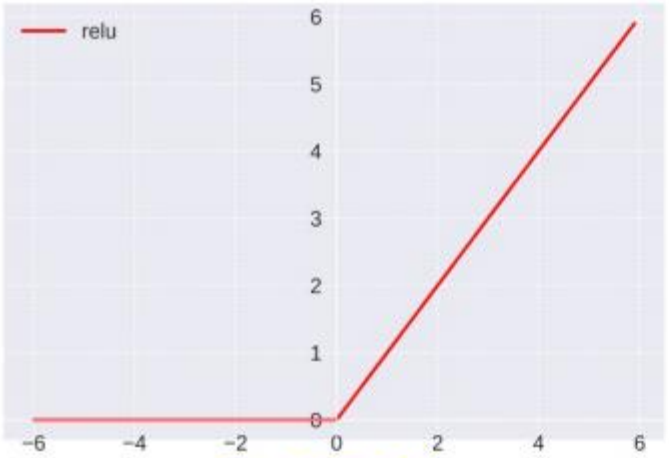
Tanh函数的导数

神经元：激活函数

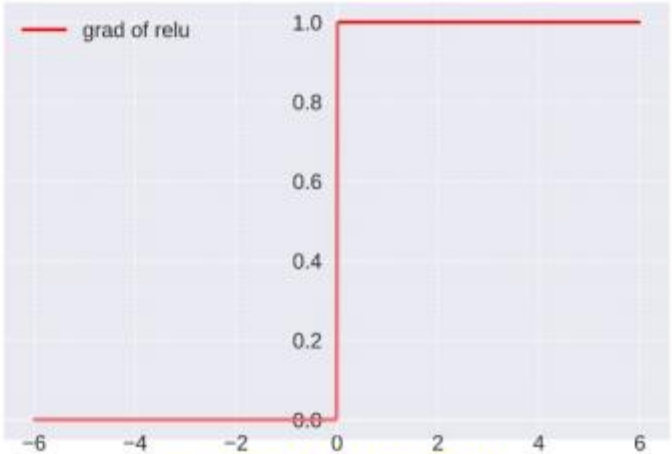
- 常用的激活函数：
  - ReLU（Rectified Linear Unit，修正线性单元）：

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

  - 深度神经网络中经常使用的激活函数． ReLU 实际上是一个斜坡（ramp）函数



ReLU函数



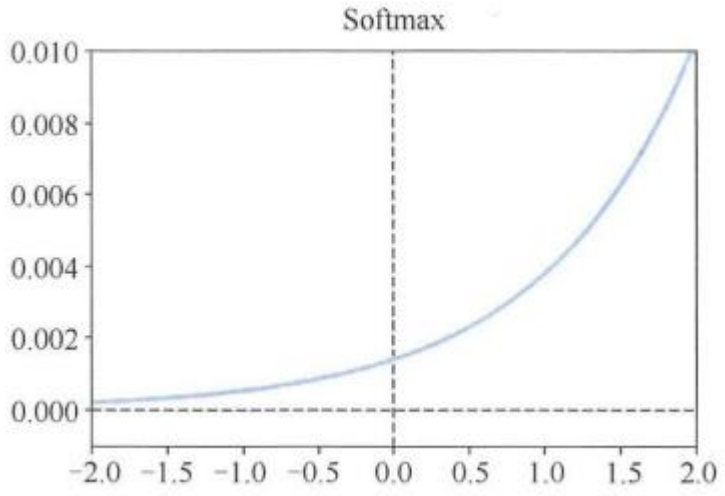
ReLU函数的导数

神经元：激活函数

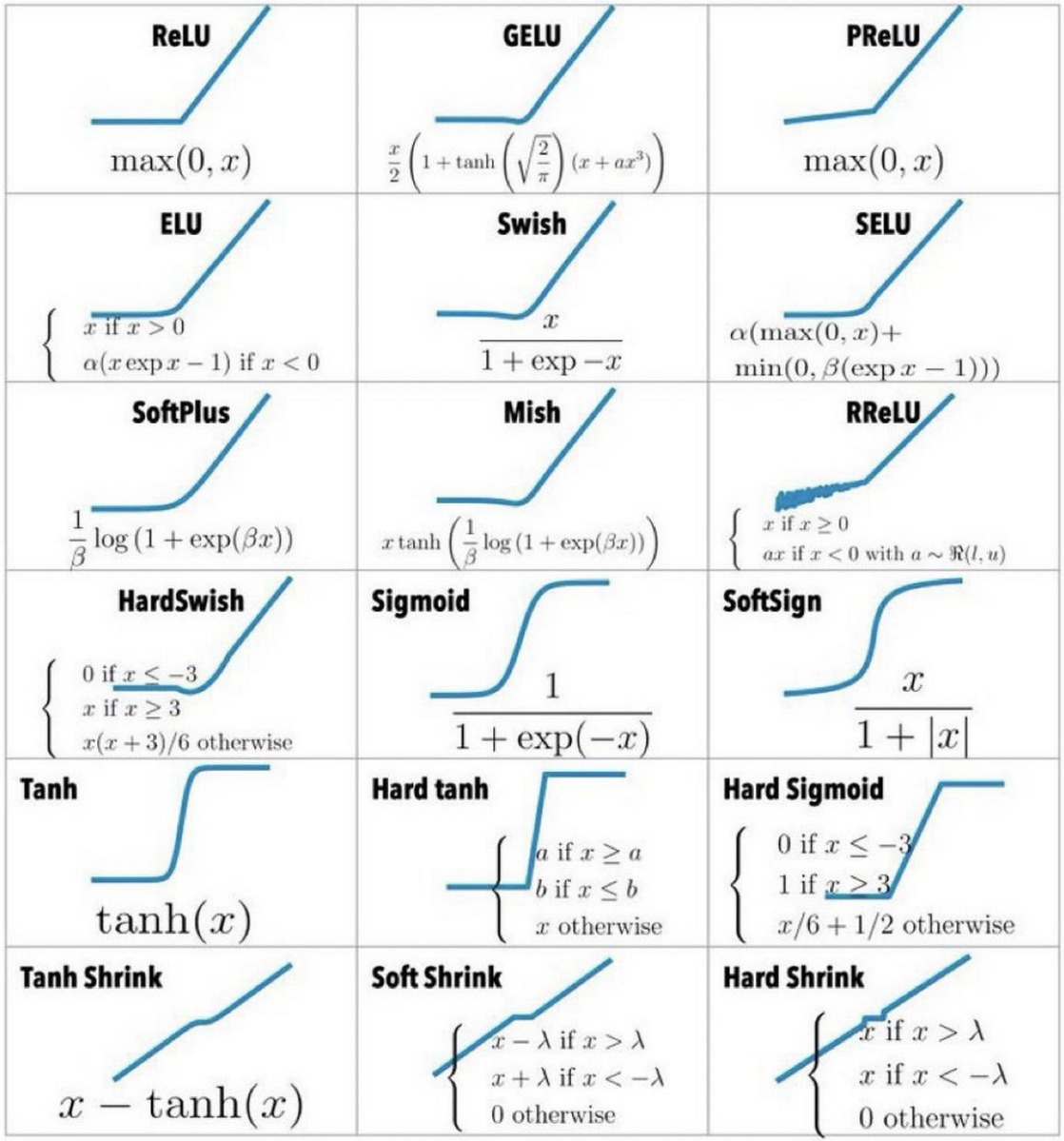
- 常用的激活函数：
  - Softmax函数：

$$\mathrm{softmax}(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

  - 本质是将一个 K 维的任意 实数向量, 压缩（映射）成另一个 K 维的实数向量, 其中向量中的每个元素取值都 介于 (0,1) 范围内
  - Softmax 用于处理多分类问题。在数学上, Softmax 函数会返回输出类的互斥概率分布，因此通常把Softmax 作为输出层的激活函数



神经元：激活函数


ReLU
$\max(0, x)$
GELU
$\frac{x}{2}\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}\left(x+ax^3\right)\right)\right)$
PReLU
$\max(0, x)$
ELU
x if $x > 0$
$\alpha(x \exp x - 1)$ if $x < 0$
Swish
$\frac{x}{1+\exp -x}$
SELU
$\alpha(\max(0, x)+$
$\min(0, \beta(\exp x - 1)))$
SoftPlus
$\frac{1}{\beta}\log\left(1+\exp(\beta x)\right)$
Mish
$x\tanh\left(\frac{1}{\beta}\log\left(1+\exp(\beta x)\right)\right)$
RReLU
x if $x \geq 0$
ax if $x < 0$ with $a \sim \Re(l, u)$
HardSwish
0 if $x \leq -3$
x if $x \geq 3$
$x(x+3)/6$ otherwise
Sigmoid
$\frac{1}{1+\exp(-x)}$
SoftSign
$\frac{x}{1+|x|}$
Tanh
$\tanh(x)$
Hard tanh
a if $x \geq a$
b if $x \leq b$
x otherwise
Hard Sigmoid
0 if $x \leq -3$
1 if $x \geq 3$
$x/6 + 1/2$ otherwise
Tanh Shrink
$x - \tanh(x)$
Soft Shrink
$x - \lambda$ if $x > \lambda$
$x + \lambda$ if $x < -\lambda$
0 otherwise
Hard Shrink
x if $x > \lambda$
x if $x < -\lambda$
0 otherwise

神经网络

- 概述
- 神经元
- 多层感知机网络
  - 网络结构
  - BP算法

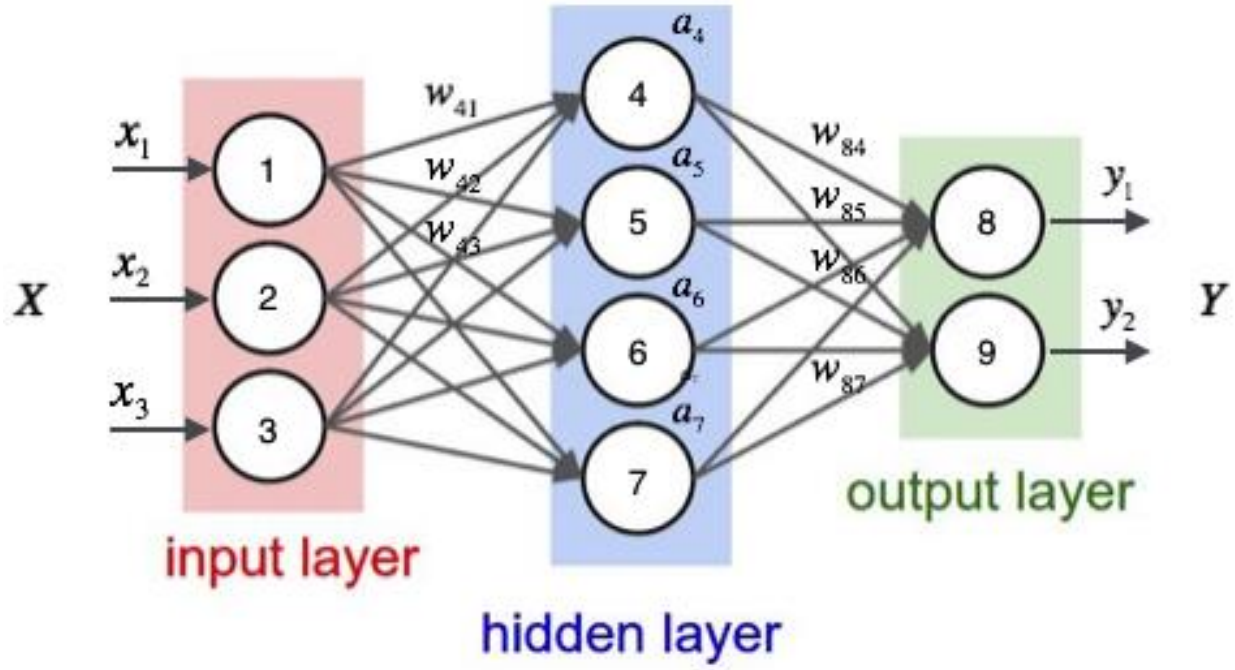
神经网络

- 概述
- 神经元
- 多层感知机网络
  - 网络结构
  - BP算法

人工神经网络

- 神经网络其实就是按照一定规则连接起来的多个神经元，以全连接（ full connected ）神经网络为例，它的规则包括：
  - 神经元按照层来布局。最左边的层叫做输入层，负责接收输入数据；最右边的层叫输出层，我们可以从这层获取神经网络输出数据。输入层和输出层之间的层叫做隐藏层，因为它们对于外部来说是不可见的。
  - 同一层的神经元之间没有连接。
  - 第N层的每个神经元和第N-1层的所有神经元相连(这就是full connected的含义)，第N-1层神经元的输出就是第N层神经元的输入。
  - 每个连接都有一个权值。

人工神经网络



- 上面这些规则定义了全连接神经网络的结构。事实上还存在很多其它结构的神经网络，比如卷积神经网络(CNN)、循环神经网络(RNN)，他们都具有不同的连接规则。

人工神经网络

- 人工神经网络主要由大量的神经元以及它们之间的有向连接构成因此考虑三方面：
- 神经元的激活规则

• 主要是指神经元输入到输出之间的映射关系，一般为非线性函数(由激活函数决定)。

- 网络的拓扑结构

• 不同神经元之间的连接关系（由模型决定）。

- 学习算法和优化方法

• 通过训练数据来学习神经网络的参数

人工神经网络

- 人工神经网络主要由大量的神经元以及它们之间的有向连接构成因此考虑三方面：
- 神经元的激活规则

• 主要是指神经元输入到输出之间的映射关系，一般为非线性函数(由激活函数决定)。

- 网络的拓扑结构

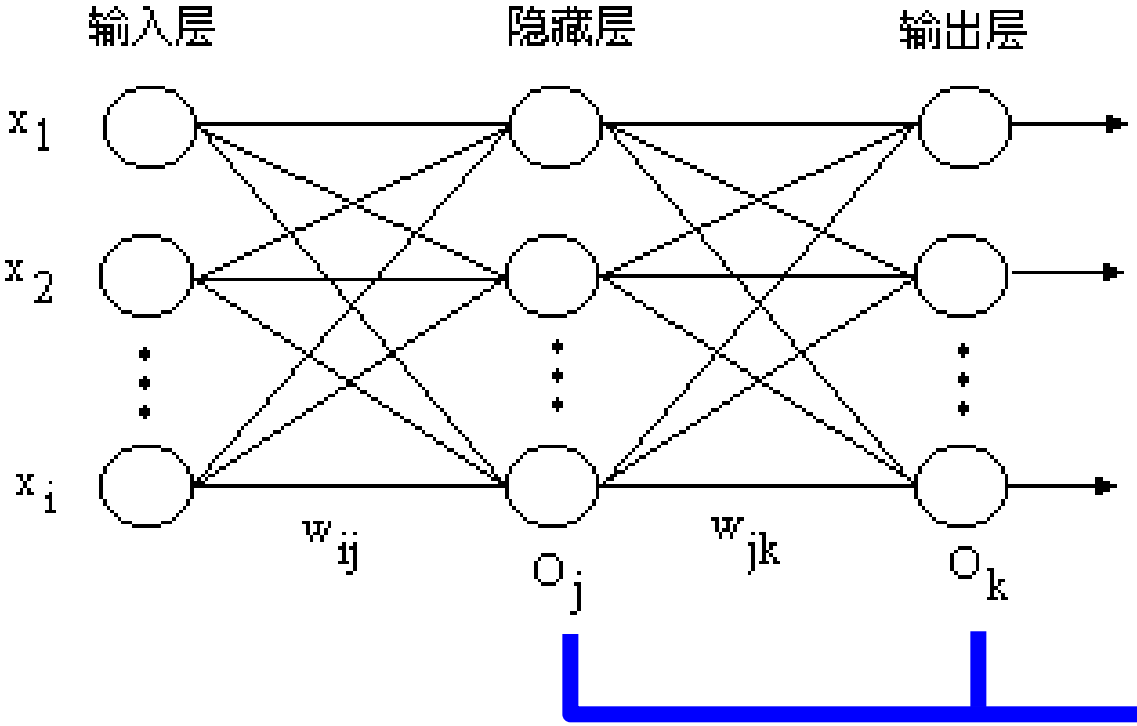
• 不同神经元之间的连接关系（由模型决定）。

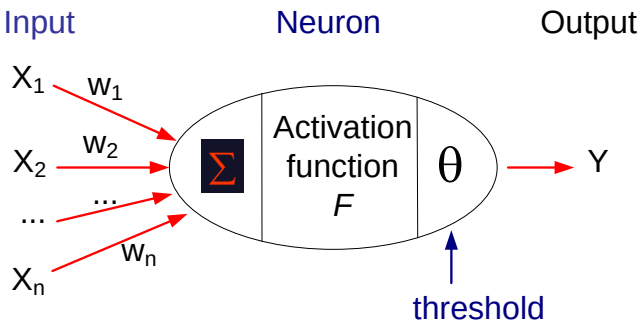
- 学习算法和优化方法

• 通过训练数据来学习神经网络的参数（如反向传播算法）

多层感知机网络：网络结构

- 一个多层感知器网络具有**一个输入层，一个输出层**，以及**一个或者多个隐含层**。
- 拓扑结构：





多层感知机网络：网络结构

- 特点：
  - 全连接网络：上一层神经元的输出，将传递给下一层的每个神经元作为输入。
  - **输入层的神经元个数由样本数据的特征个数决定。**
    - 样本数据为d维特征向量，则输入层神经元为d个。
  - **输出层的神经元个数由待分类问题决定。**
    - 对于M类分类问题，输出层神经元个数为M个。
  - **隐含层的数目以及每个隐含层中神经元的个数尚无理论依据(通过经验或者实验来确定)。**

Where n, m and h are the number of the input layer's neurons, the output layer's neurons and the hidden layer's neurons respectively; a is usually an integer between 1 and 10.

$$h = \sqrt{n+m} + a$$

$$h = 1.5 * n$$

$$h = \sqrt{n * m}$$

神经网络

- 概述
- 神经元
- 多层感知机网络
  - 网络结构
  - BP算法

多层感知机网络：BP算法

- 当根据具体问题设计出一个多层感知机网络之后，如何使用该网络进行分类？
  - 需要先**确定网络中的所有权值及阈值**，然后才能对未知类标号的样本数据进行分类。
- 反向传播算法(Back-propagation Algorithm, BP)
  - 是多层感知机网络的一种“有监督”训练算法，能根据给定的训练样本求出使网络正确分类的权值和阈值。

多层感知机网络：BP算法

- BP算法的基本思想：
  - 对于一个已知类标号的样本，经过权值累积求和、阈值以及激活函数等一系列的计算后，将得到一个输出，可根据该输出与该样本期望的目标输出（已知标号）进行比较。
  - 如果有偏差，则从网络的输出开始反向传递这个偏差，依次对网络中各层的权值和阈值进行调整，使网络的输出逐渐与期望的目标输出一致。

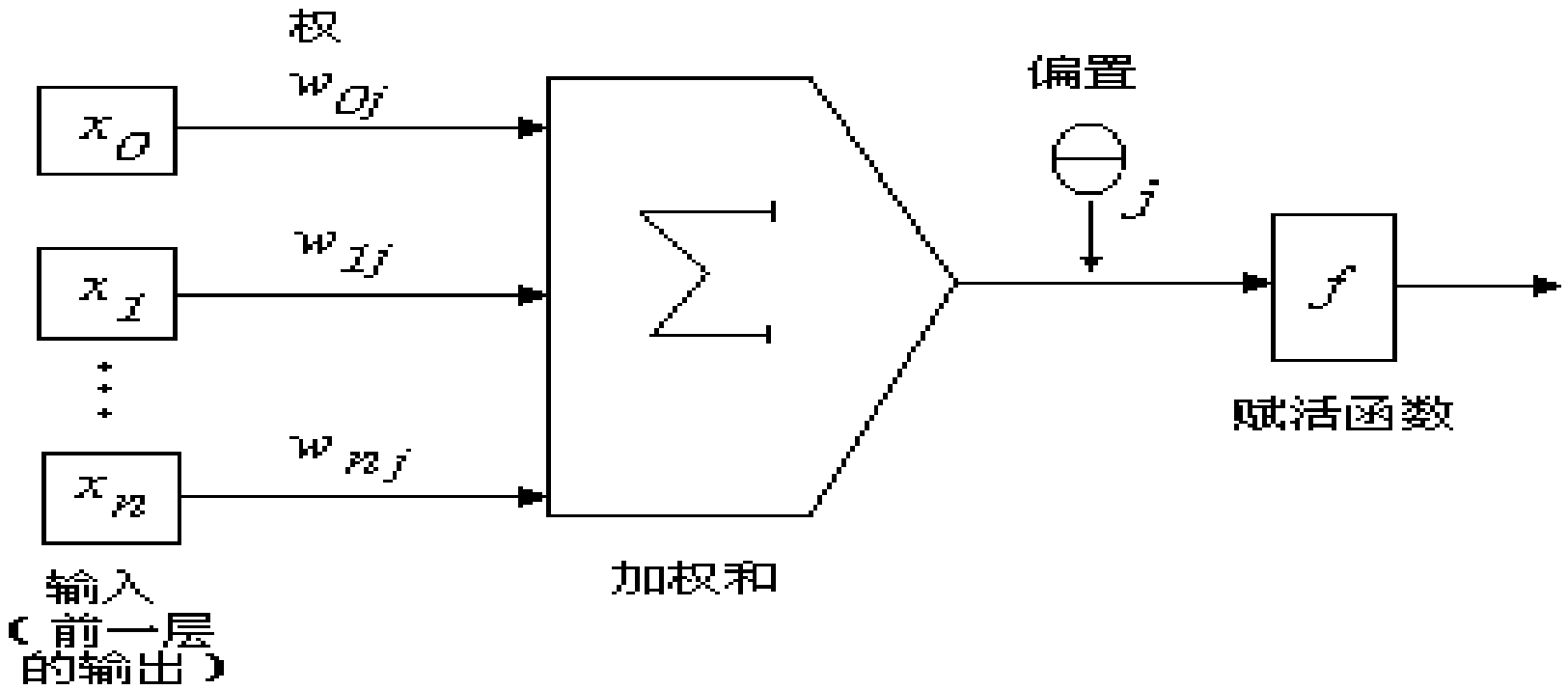
多层感知机网络：BP算法

- BP算法可分为两个阶段：
  - 第一阶段（正向阶段）：样本数据从输入层经过隐含层逐层计算各单元的输出值，最终得到各输出单元的输出值。
  - 第二阶段（反向阶段）：计算各输出单元的误差，并逐层向前计算各隐层单元的误差，并采用此误差来修正前一层的权值。（BP 算法基于梯度下降策略）

多层感知机网络：BP算法

- BP算法的具体步骤：
  - (1) **初始化权值**：网络中的权值都被初始化为很小的随机数（例如：-1.0到1.0的随机数）；每个单元的阈值（偏置）也初始化为很小的随机数。
  - (2) **向前传播输入**：在这一步，每个样本*X*都分别计算网络隐含层和输出层中各单元的净输入和输出。
    - 首先，训练样本提供给网络的输入层。注意，对于输入层的单元*j*，它的输出等于它的输入，即：对于单元*j*，$O_j = I_j$。

多层感知机网络：BP算法



$$I_j = \sum_i w_{ij} O_i + \theta_j$$

- 其中，w_{ij}是由上层的单元i到本层单元j的连接权值；O_i是上层单元i的输出；而θ_j是本层单元j的偏置。

多层感知机网络：BP算法

- BP算法的具体步骤：
  - (1) **初始化权值**：网络中的权值都被初始化为很小的随机数（例如：-1.0到1.0的随机数）；每个单元的阈值（也称偏置）也初始化为很小的随机数。
  - (2) **向前传播输入**：在这一步，每个样本X都分别计算网络隐含层和输出层中各单元的净输入和输出。
    - 最后，将隐含层和输出层中每个单元的净输入I_j送入各自的激活函数（假设为Sigmoid函数），计算其输出O_j：

$$O_j = \frac{1}{1 + e^{-I_j}}$$

多层感知机网络：BP算法

- BP算法的具体步骤：
  - (1) **初始化权值**：网络中的权值都被初始化为很小的随机数（例如：-1.0到1.0的随机数）；每个单元的阈值（也称偏置）也初始化为很小的随机数。
  - (2) **向前传播输入**：在这一步，每个样本X都分别计算网络隐含层和输出层中各单元的净输入和输出。
  - (3) **反向传播误差**：通过更新权值和反映网络预测误差的偏置，向后传播误差。
    - 对于输出层单元j，误差Err_j用下式计算：

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

    - 其中，O_j是单元j的实际输出，而T_j是j的期望输出。

多层感知机网络：BP算法

- BP算法的具体步骤：
  - (1) **初始化权值**：网络中的权值都被初始化为很小的随机数（例如：-1.0到1.0的随机数）；每个单元的阈值（也称偏置）也初始化为很小的随机数。
  - (2) **向前传播输入**：在这一步，每个样本*X*都分别计算网络隐含层和输出层中各单元的净输入和输出。
  - (3) **反向传播误差**：通过更新权值和反映网络预测误差的偏置，向后传播误差。
    - 从后往前，依次计算各隐层单元*j*的误差Err_j：

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

    - 其中，w_{jk}是由下一较高层中单元*k*到单元*j*的连接权，而Err_k是单元*k*的误差。

多层感知机网络：BP算法

- BP算法的具体步骤：
  - (1) **初始化权值**：网络中的权值都被初始化为很小的随机数（例如：-1.0到1.0的随机数）；每个单元的阈值（也称偏置）也初始化为很小的随机数。
  - (2) **向前传播输入**：在这一步，每个样本X都分别计算网络隐含层和输出层中各单元的净输入和输出。
  - (3) **反向传播误差**：通过更新权值和反映网络预测误差的偏置，向后传播误差。
    - 计算权值的修正量，并更新权值：

$$\Delta w_{ij} = (l) Err_j O_i \quad \Rightarrow \quad w_{ij} = w_{ij} + \Delta w_{ij}$$

    - 其中：l是学习率，通常取0~1之间的值。

多层感知机网络：BP算法

- BP算法的具体步骤：
  - (1) **初始化权值**：网络中的权值都被初始化为很小的随机数（例如：-1.0到1.0的随机数）；每个单元的阈值（也称偏置）也初始化为很小的随机数。
  - (2) **向前传播输入**：在这一步，每个样本X都分别计算网络隐含层和输出层中各单元的净输入和输出。
  - (3) **反向传播误差**：通过更新权值和反映网络预测误差的偏置(阈值)，向后传播误差。
    - 计算权值的修正量，并更新权值：

$$\Delta\theta_j = (l)Err_j \quad \Rightarrow \quad \theta_j = \theta_j + \Delta\theta_j$$

    - 其中：l是学习率，通常取0~1之间的值。

多层感知机网络：BP算法

- BP算法的具体步骤：
  - (1) **初始化权值**：网络中的权值都被初始化为很小的随机数（例如：-1.0到1.0的随机数）；每个单元的阈值（也称偏置）也初始化为很小的随机数。
  - (2) **向前传播输入**：在这一步，每个样本X都分别计算网络隐含层和输出层中各单元的净输入和输出。
  - (3) **反向传播误差**：通过更新权值和反映网络预测误差的偏置(阈值)，向后传播误差。
  - (4) **终止条件**：
    - 前一周期所有的Δw_{ij}都太小，小于某个阈值；
    - 前一周期未正确分类的样本百分比小于某个阈值；
    - 超过预先指定的周期数（**实际中常用的终止条件**）。

多层感知机网络：BP算法

算法流程回顾：

1.将输入样本提供给输入层神经元

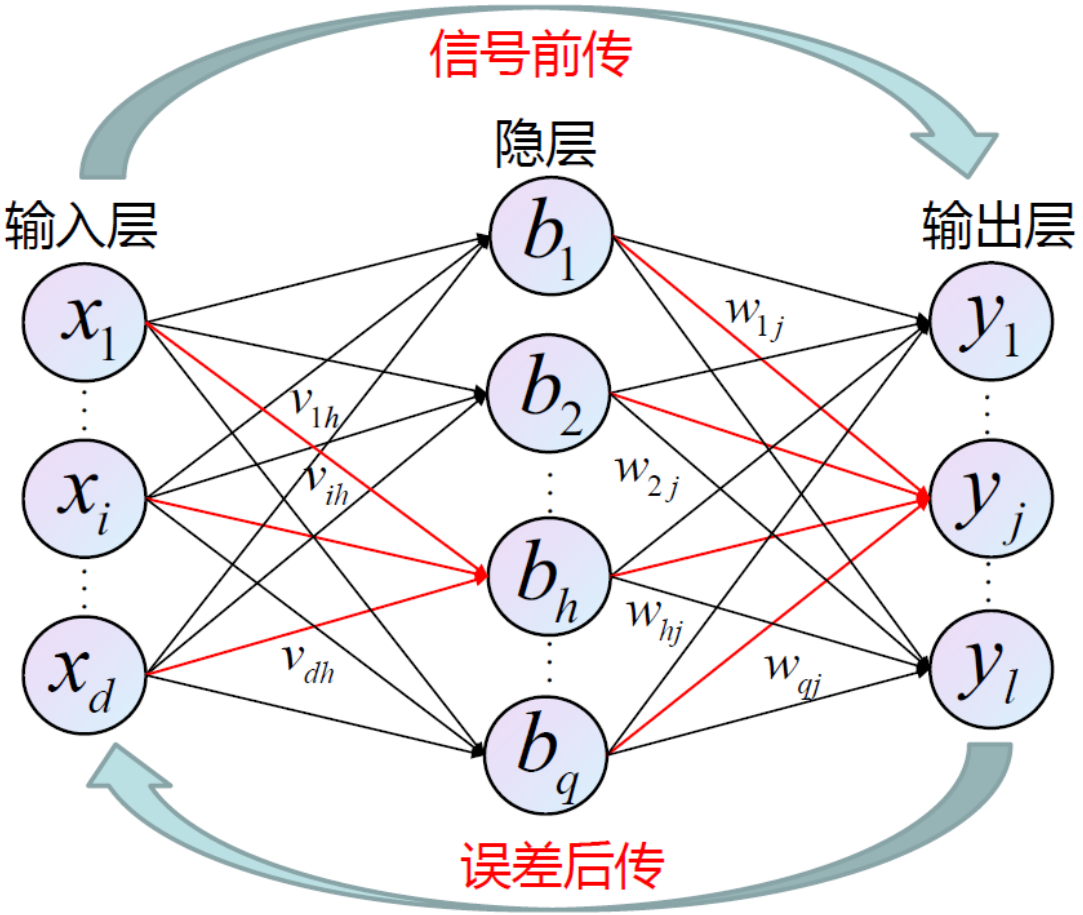
2.逐层将信号前传至隐层、输出层，产生输出层的结果

3.计算输出层误差

4.将误差反向传播至隐藏层神经元

5.根据隐层神经元对连接权重和阈值进行调整

6.上述过程循环进行，直至达到某些停止条件为止



多层感知机网络：BP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

图 5.8　误差逆传播算法

多层感知机网络：BP算法

- 参数对BP算法的影响：
  - (1) 权重的初始值
  - (2) 学习率*l*的取值

多层感知机网络：BP算法

- 参数对BP算法的影响：
  - (1) 权重的初始值
    - 当训练样本足够多时，对于仅包含一个隐含层的多层感知器网络，权重的初始值不会影响BP算法的收敛。
    - 但当训练样本不足或者网
      的初始值将影响BP算法的
    - 实际中，为权重选取初值

***d*是输入层中单元的个数；**
***n*是隐含层中单元的个数。**

输入层到隐含层的权重初值：$-\frac{1}{\sqrt{d}} < w_{ij} < \frac{1}{\sqrt{d}}$

隐含层到输出层的权重初值：$-\frac{1}{\sqrt{n}} < w_{jk} < \frac{1}{\sqrt{n}}$

  - (2) 学习率*l*的取值

多层感知机网络：BP算法

- 参数对BP算法的影响：
  - (1) 权重的初始值
  - (2) 学习率*l*的取值
    - 理论上，只要学习率足够小，都能保证BP算法收敛，即：*l*的大小只影响训练的速度（快慢），而不影响最终权值的大小。
    - 实际上，多层感知机网络很少能充分训练，使误差达到最小值，因此学习率的选值会影响到BP算法的收敛，从而影响最终权值的大小。
    - 学习率的经验值为：0.01~1。

BP算法常常导致过拟合

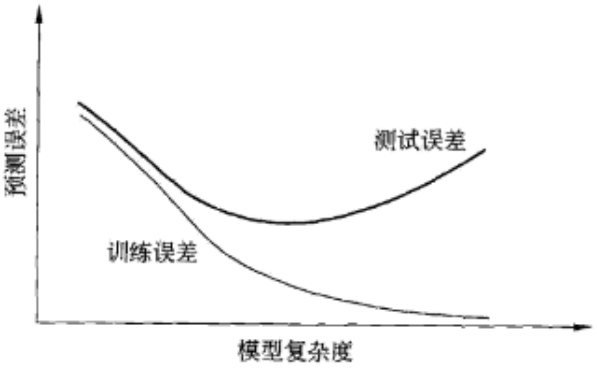


图 1.3 训练误差和测试误差与模型复杂度的关系

主要策略：

- 早停 (early stopping)
  - 若训练误差连续 a 轮的变化小于 b, 则停止训练
  - 使用验证集：若训练误差降低、验证误差升高 , 则停止训练

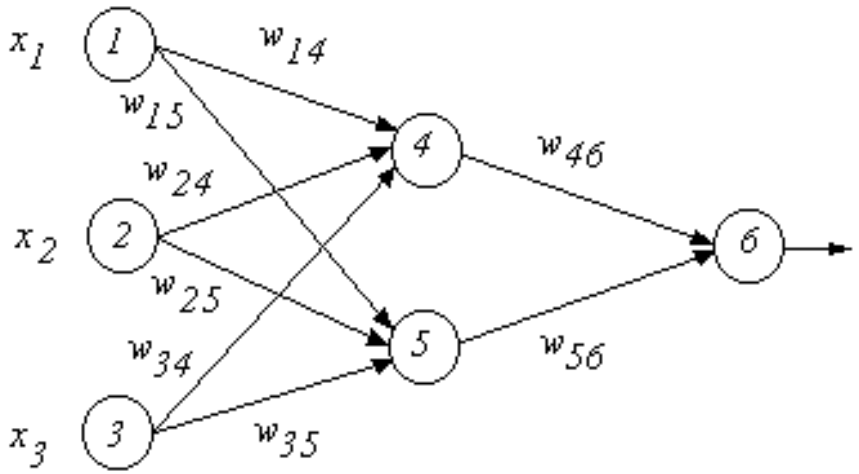
- 正则化 (regularization)
  - 在误差目标函数中增加一项描述网络复杂度

例如 $E = \lambda \frac{1}{m} \sum_{k=1}^{m} E_k + (1-\lambda) \sum_i w_i^2$

偏好比较小的连接权和阈值，
使网络输出更“光滑”

多层感知机网络：BP算法示例

- 例：给定一个多层感知机网络，使用BP算法，计算一个训练样本数据训练该多层感知机网络的过程。$X = (1, 0, 1)$，假定类标号为1。



x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

多层感知机网络：BP算法示例

- 首先，将样本提供给网络，计算每个单元的净输入和输出。

单元j	净输入I_j	输出O_j
4	0.2+0-0.5-0.4 = -0.7	$1+(1+e^{0.7}) = 0.332$
5	-0.3+0+0.2+0.2 = 0.1	$1+(1+e^{-0.1}) = 0.525$
6	(-0.3)(0.332)-(0.2)(0.525)+0.1 = -0.105	$1+(1+e^{-0.105}) = 0.474$

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

$$O_j = \frac{1}{1+e^{-I_j}}$$

多层感知机网络：BP算法示例

- 计算每个单元的误差，并反向传播。

单元j	Err_j
6	(0.474)(1-0.474)(1-0.474) = 0.1311
5	(0.525)(1-0.525)(0.1311)(-0.2) = -0.0065
4	(0.332)(1-0.332)(0.1311)(-0.3) = -0.02087

输出层单元的误差 → $Err_j = O_j(1-O_j)(T_j - O_j)$

隐含层单元的误差 → $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$

多层感知机网络：BP算法示例

- 更新每个单元的权值和偏置。每个单元权值和偏置的更新如下表所示。(学习率取值0.9)

权值和偏置	新值
w_{46}	-0.3 + (0.9)(0.1311)(0.332) = -0.261
w_{56}	-0.2 + (0.9)(0.1311)(0.525) = -0.138
w_{14}	0.2 + (0.9)(-0.0087)(1) = 0.192
w_{15}	-0.3 + (0.9)(0.0065)(1) = -0.306
w_{24}	0.4 + (0.9)(-0.0087)(0) = 0.4
w_{25}	0.1 + (0.9)(-0.0065)(0) = 0.1
w_{34}	-0.5 + (0.9)(-0.0087)(1) = -0.508
w_{35}	0.2 + (0.9)(-0.0065)(1) = 0.194
θ_6	0.1 + (0.9)(0.1311) = 0.218
θ_5	0.2 + (0.9)(-0.0065) = 0.194
θ_4	-0.4 + (0.9)(-0.0087) = -0.408

$$\Delta w_{ij} = (l) Err_j O_i \quad \Rightarrow \quad w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta \theta_j = (l) Err_j \quad \Rightarrow \quad \theta_j = \theta_j + \Delta \theta_j$$

多层感知机网络

- 优点
  - 预测精度总的来说较高；
  - 健壮性好，训练样本中包含错误时也可正常工作；
  - 输出可能是离散值、连续值或者是离散或量化属性的向量值；
  - 对目标进行分类速度较快。
- 缺点
  - 训练（学习）时间通常较长；
  - 隐含层个数及每个隐含层中单元数目尚无理论依据，常用经验选取。

神经网络

- 参考
  - https://www.3blue1brown.com/lessons/backpropagation
  - https://mp.weixin.qq.com/s/vmoIxE9hnKToYO1WiLLbjQ