第五章 感知机与人工神经网络

李潇

2025年11月

本章目录

01 神经元

02 感知机

03 前馈神经网络

04 BP神经网络

05 玻尔兹曼机

本章目录

01 神经元

02 感知机

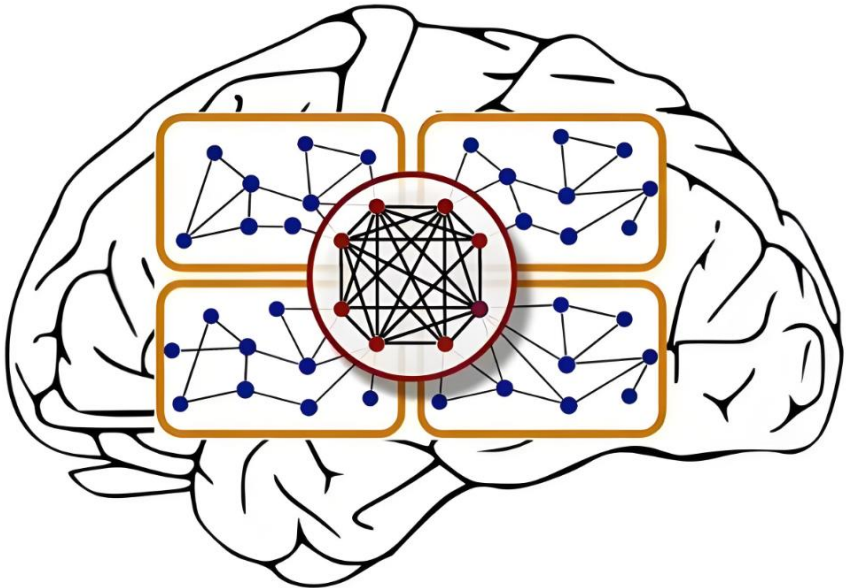
03 前馈神经网络

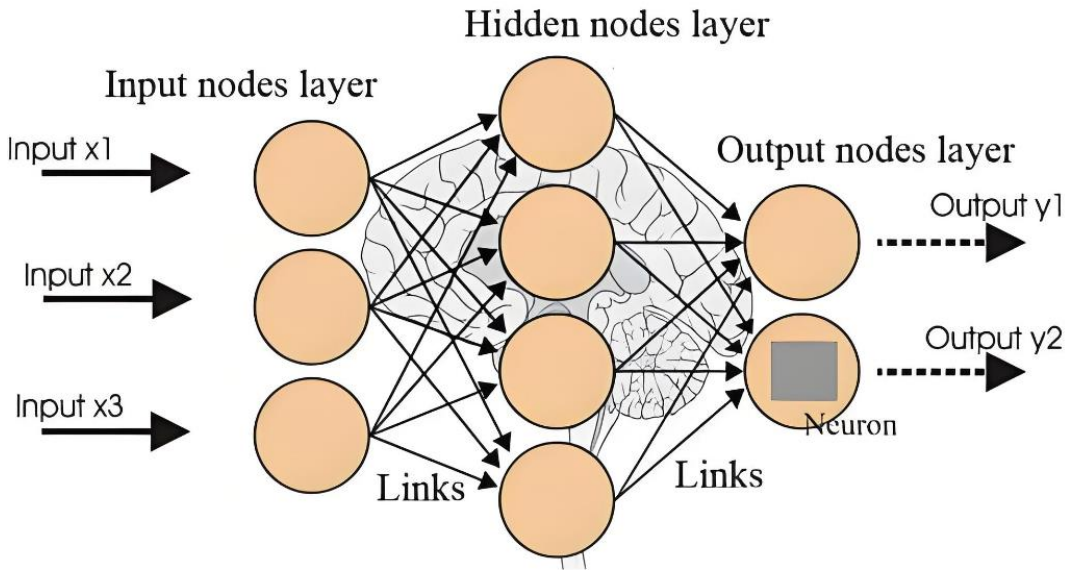
04 BP神经网络

05 玻尔兹曼机

引言

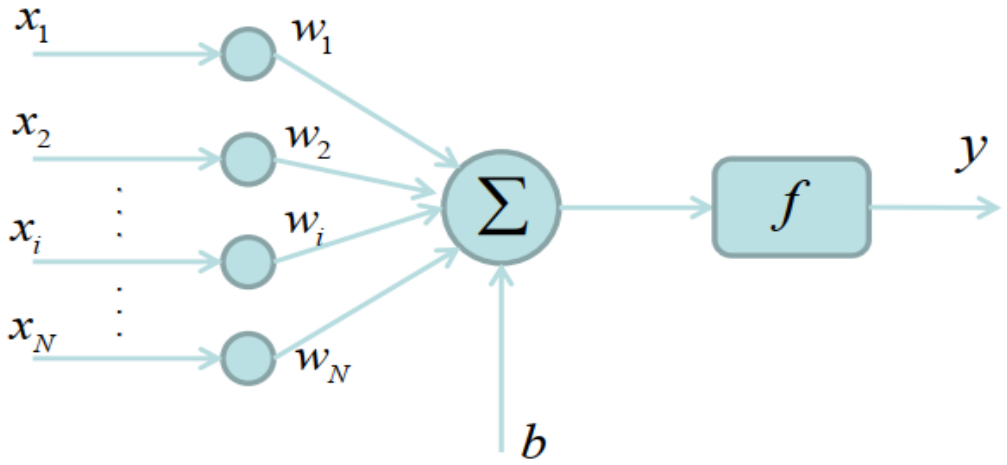
- 神经网络，受生物神经网络的启发，通过数学和计算机科学的方法模拟和实现人工智能系统。
- 局限性：
  - 对于大型神经网络，需要大量的训练数据和计算资源来训练和优化网络。
  - 神经网络的解释性相对较差，难以解释网络内部的决策过程。

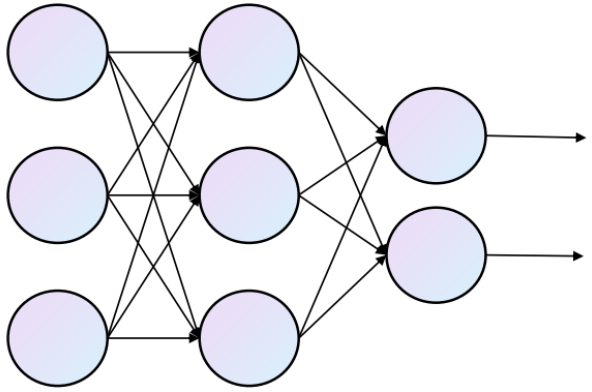




引言

- **人工神经网络的发展历程：**
  - 1890年，首次阐明了人脑结构功能和相关的学习联想记忆规则。
  - 1949年，神经元学习的一般规则，即Hebbin法则。
  - 1953年，神经元的基本特性，提出了神经元的数学模型，即M-P模型。
  - 1958年，历史上第一个具有学习能力的神经网络模型Perceptron。
  - 1960年代初期，自适应线性元结构和Windrow-Hoff算法的提出为神经网络的自适应系统建立了基础。
  - 1969年，单层神经网络的功能局限性，发展走入低谷。

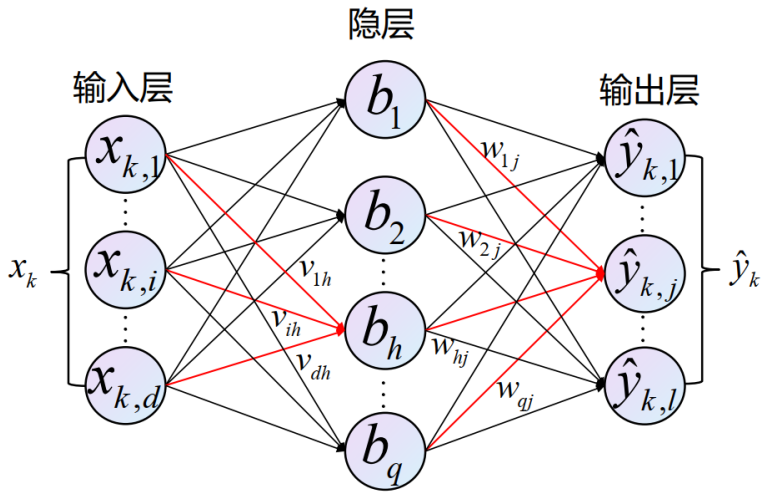


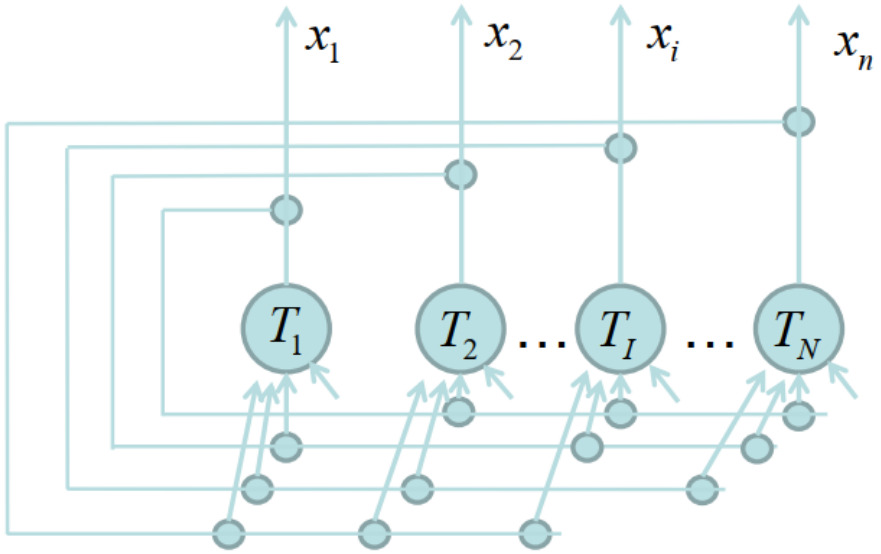


神经元数学模型

引言

- **人工神经网络的发展历程：**
  - 20世纪80年代，成功求解NP完全的旅行商问题，研制了Boltzmann机。
  - 1990年，神经网络集成方法。
  - 1997年，AdaBoost算法。
  - 2012年6月，吴恩达和谷歌科学家合作，用1.6万台计算机搭建并模拟了一个人脑神经网络。
  - 2015年5月，深度学习开始被学术界接受。
  - 2016年3月，AlphaGo打败韩国围棋选手李世石。





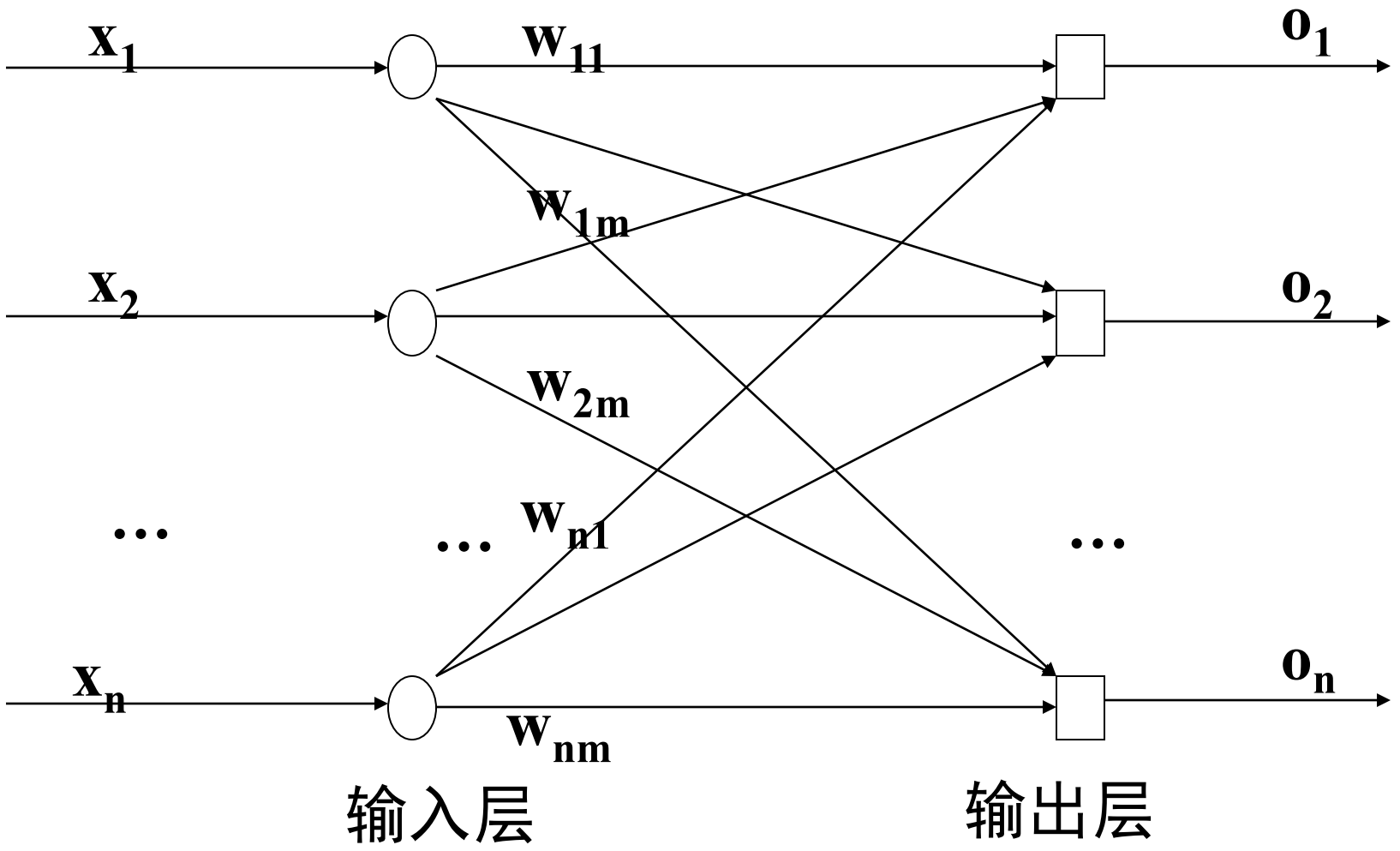
引言

- **人工神经网络的特点：**
  - 分布式存储信息
  - 并行协同处理信息
  - 信息处理与存储合二为一
  - 对信息的处理具有自组织、自学习的特点，便于联想、综合和推广

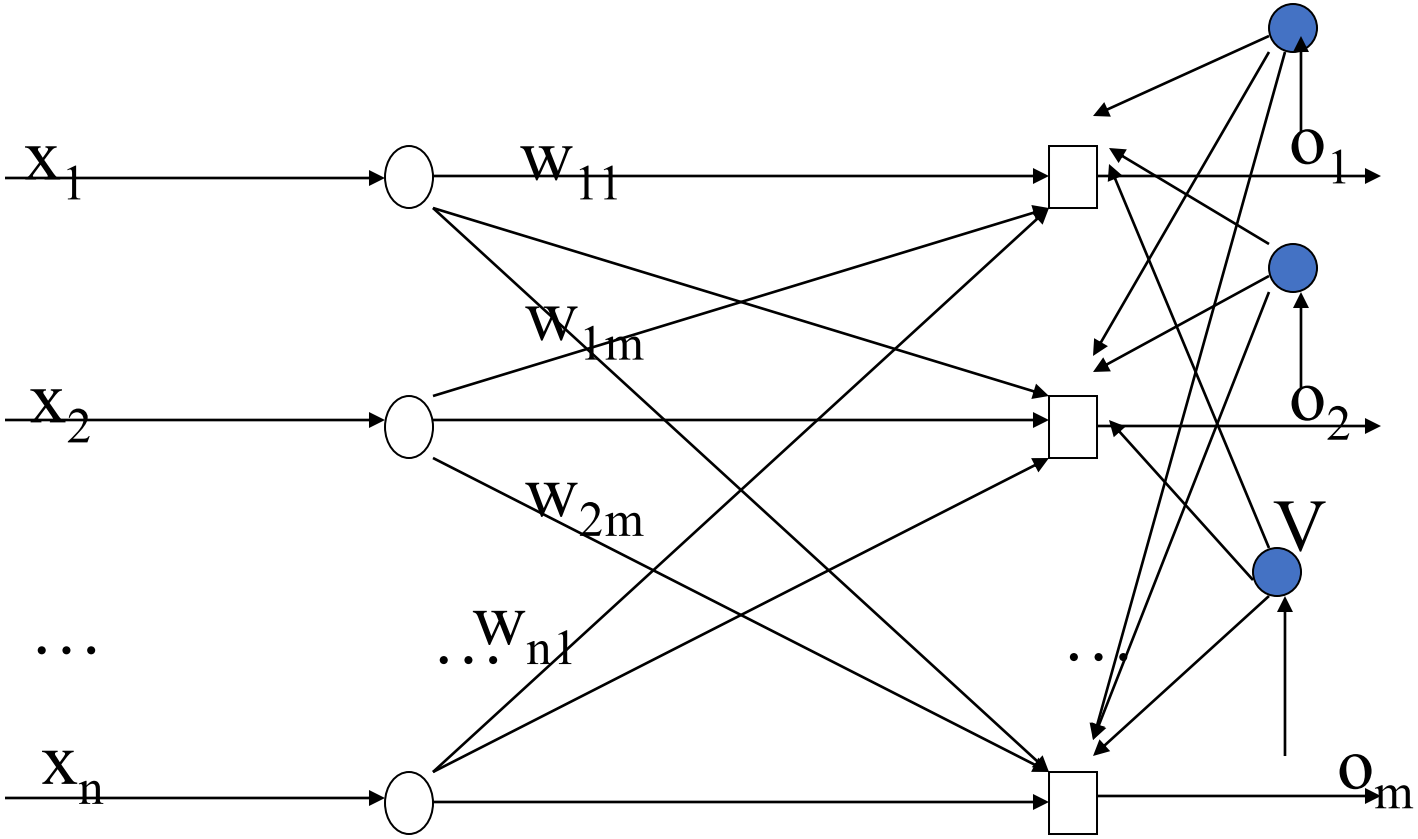
引言

- **人工神经网络的分类：**
  - 前馈神经网络，将神经元分层排列，分别是输入层、隐藏层和输出层。按照层数不同，划分为单层前馈神经网络和多层前馈神经网络。感知机、MLP、BP网络、CNN、RNN。
  - 反馈神经网络，每个神经元同时将自身的输出信号作为输入信号反馈给其他神经元，Hopfield神经网络，是反馈网络中最简单且应用广泛的模型。
  - 自组织神经网络，是无导师学习网络，通常采用竞争原则进行网络学习。

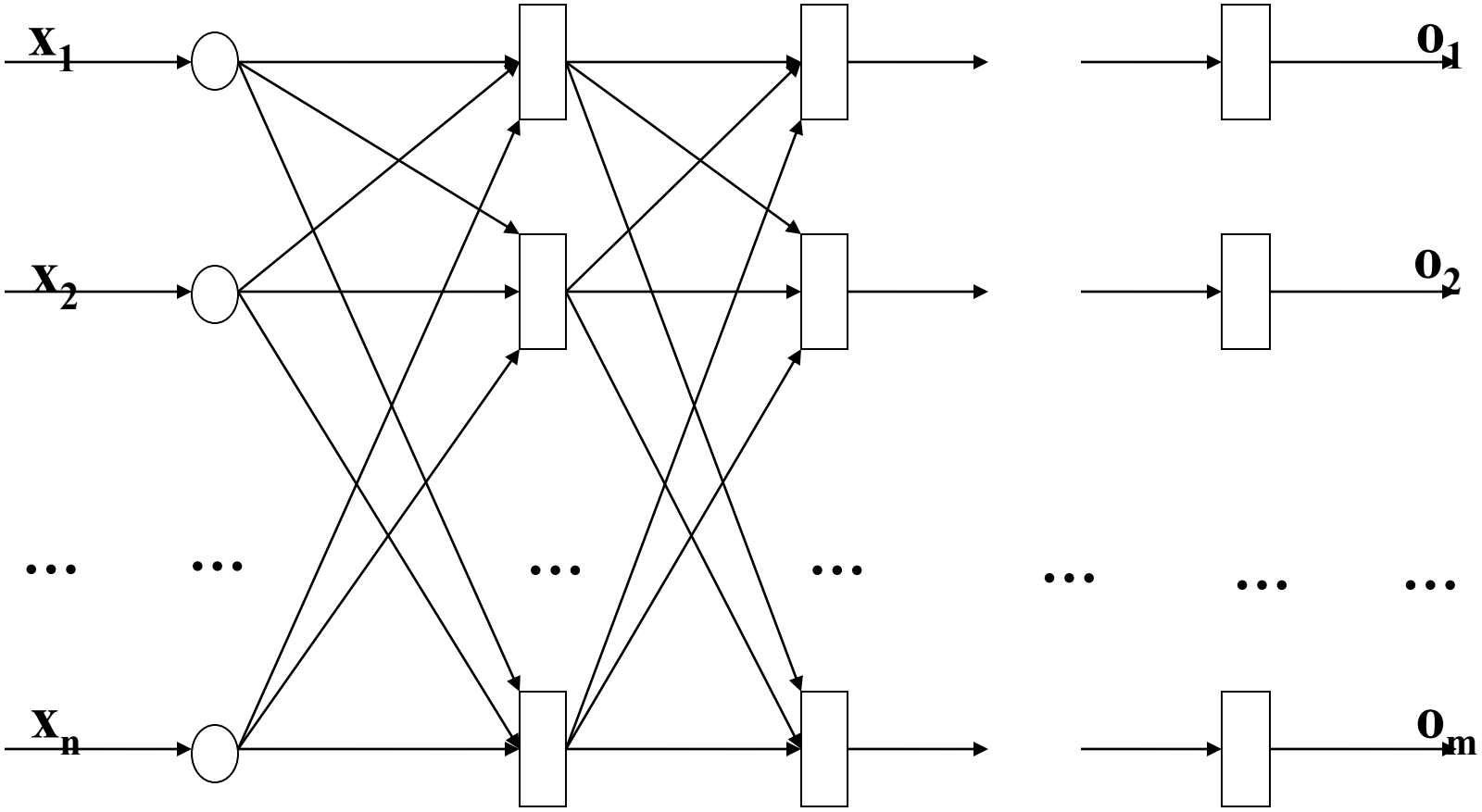
典型网络结构：简单单级网



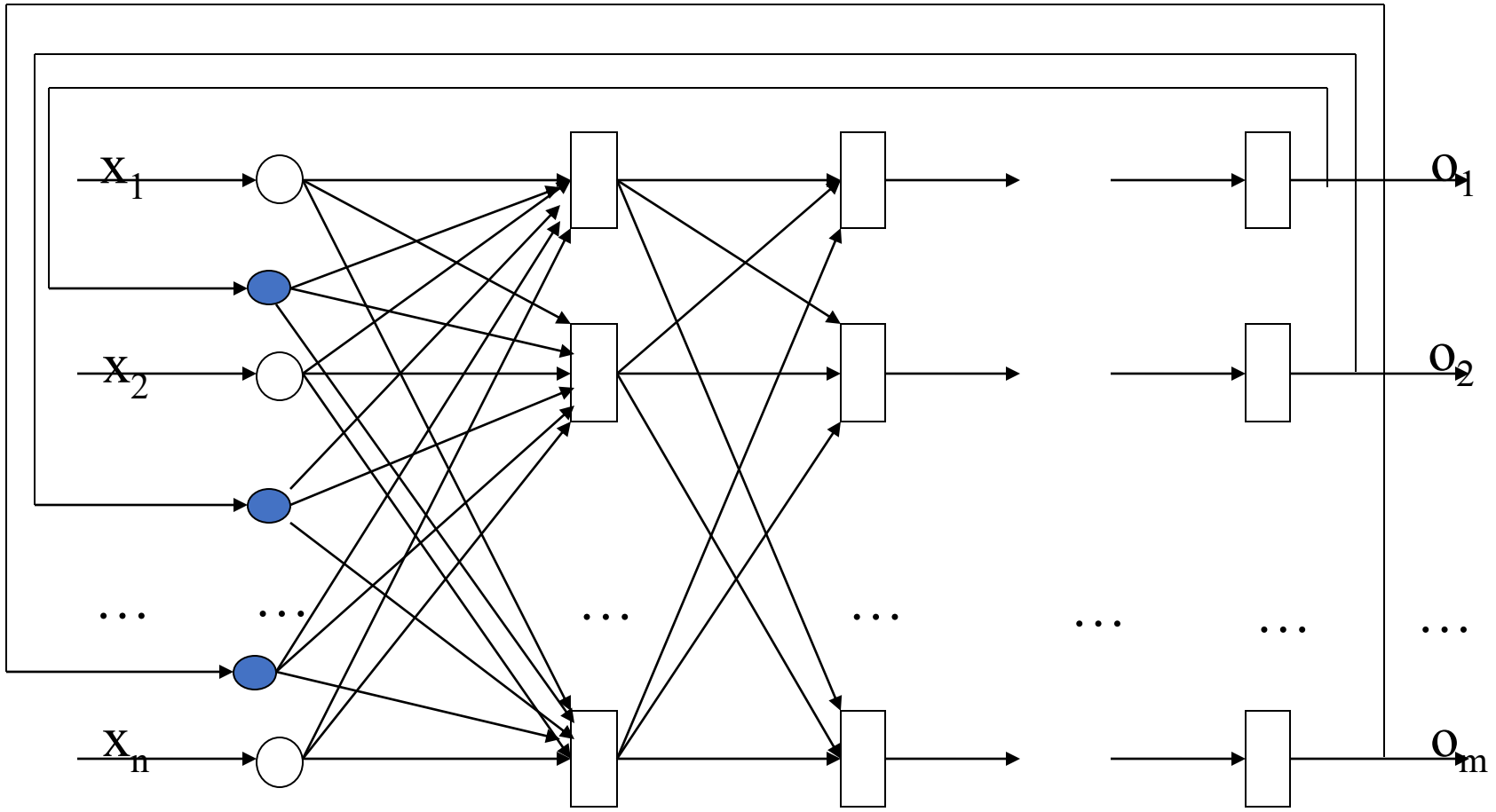
典型网络结构：单级横向反馈网



典型网络结构：多级网

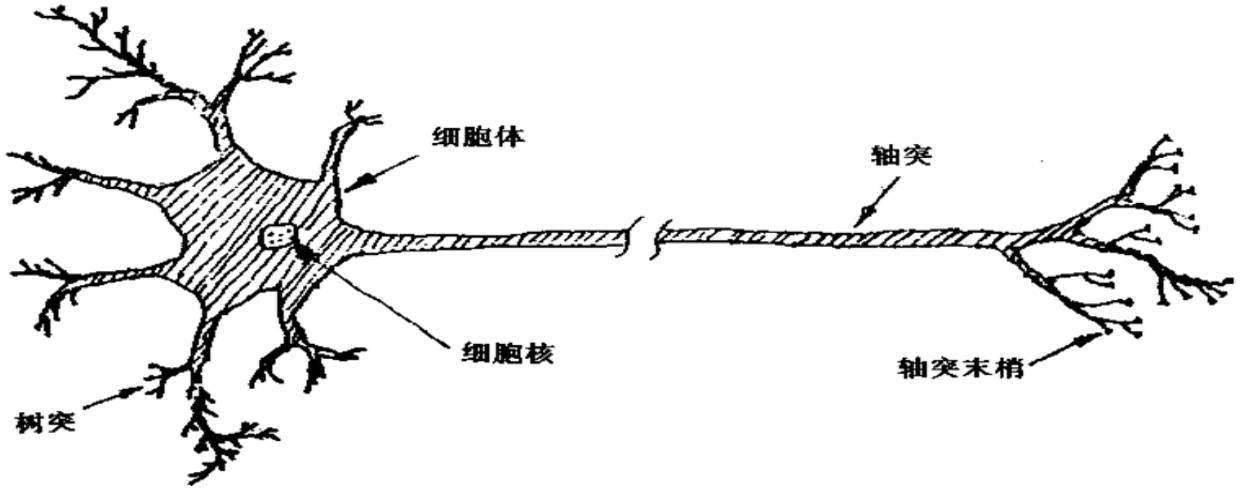


典型网络结构：循环网



神经元模型

- 神经元，是神经网络的基本信息处理单位，是（人工）神经网络的设计基础。
- 神经元起源于早期的生物学研究，一个（生物）神经元主要由细胞体、树突、轴突和突触等组成。
- 人工神经网络的神经元模型，是基于生物神经元的概念进行抽象和简化的。

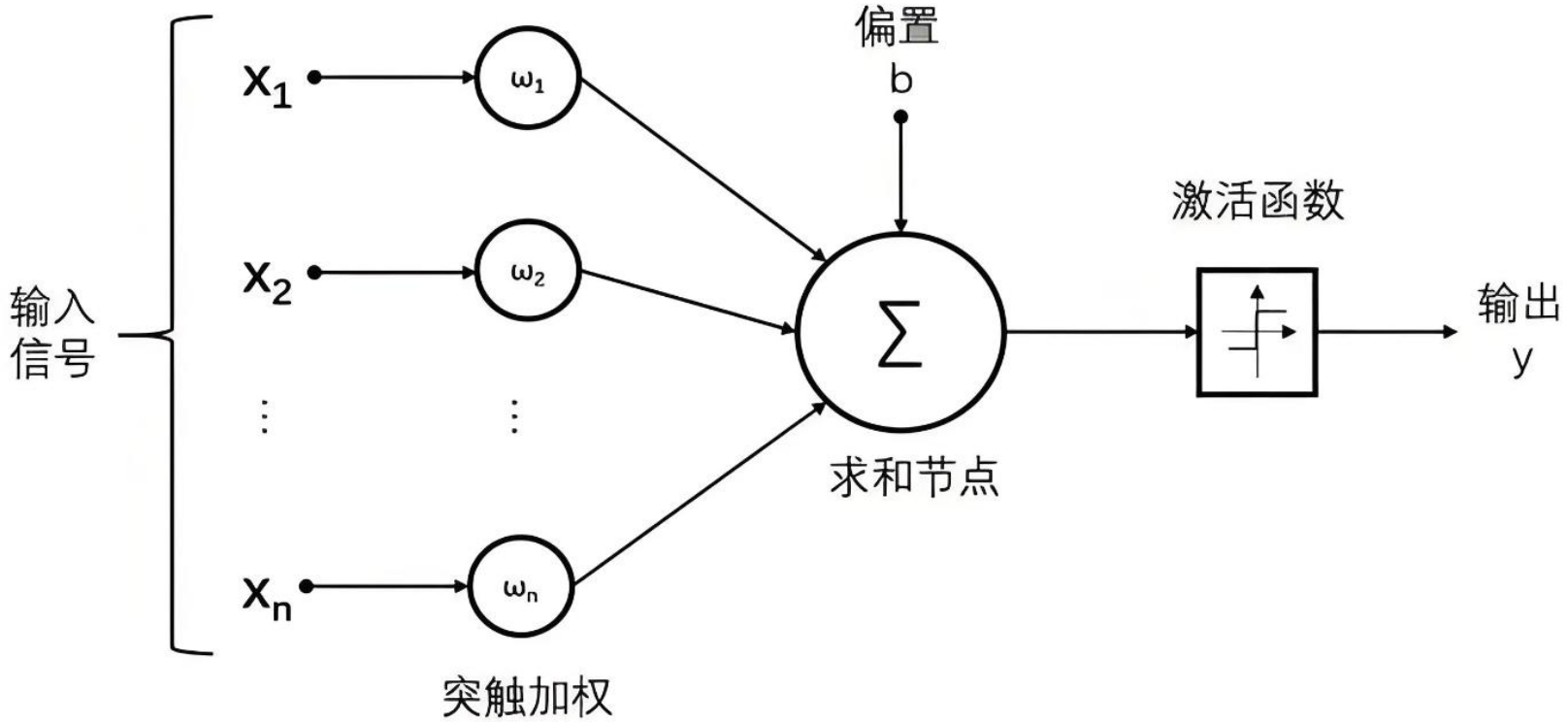


树突是传入纤维，

轴突是传出纤维。

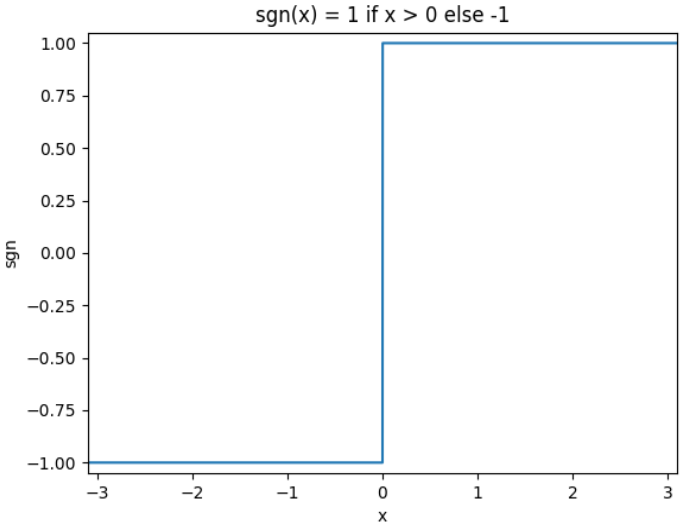
神经元模型

- 1943年，提出模拟生物神经元的计算模型，称为McCulloch-Pitts模型（M-P模型），提出神经元的线性加权求和阈值模型。

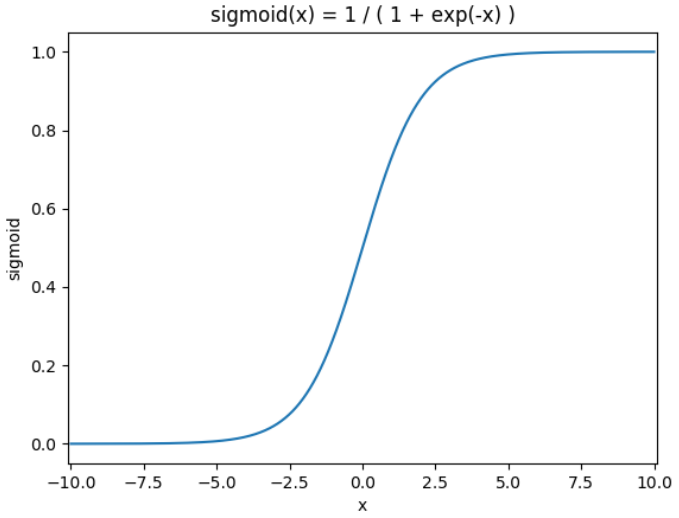


神经元模型

- **神经元阈值函数（激活函数）：**
  - **符号函数Sign(x)**，输出是离散的，只有两个可能的取值。
  - **Logistic函数σ(x)**，输出是连续的，表示输入在某个阈值上下的概率。



- **M-P神经元特点：**
  - 多输入单输出
  - 空间整合特性和阈值特性
  - 输入通过权值表征耦合程度
  - 突触接头有时间延迟，网络的活动过程离散化



小节

- 神经元，是神经网络的基本信息处理单位。
- 按照神经网络层数不同，划分为单层神经网络和多层神经网络。
- 激活函数，通过对神经元的输出进行非线性变换，引入了非线性，增强了神经网络的表达能力和复杂问题的建模能力。
- 在神经网络中，每个神经元的输出可以表示为$y=f(wx+b)$，其中 f 是激活函数，w 是权重，x是输入，b 是偏置。

本章目录

01 神经元

02 感知机

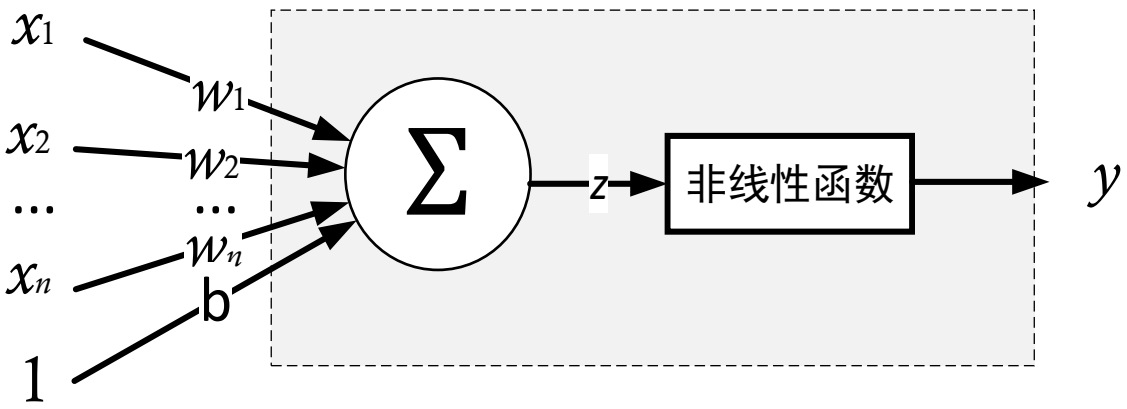
03 前向神经网络分类器

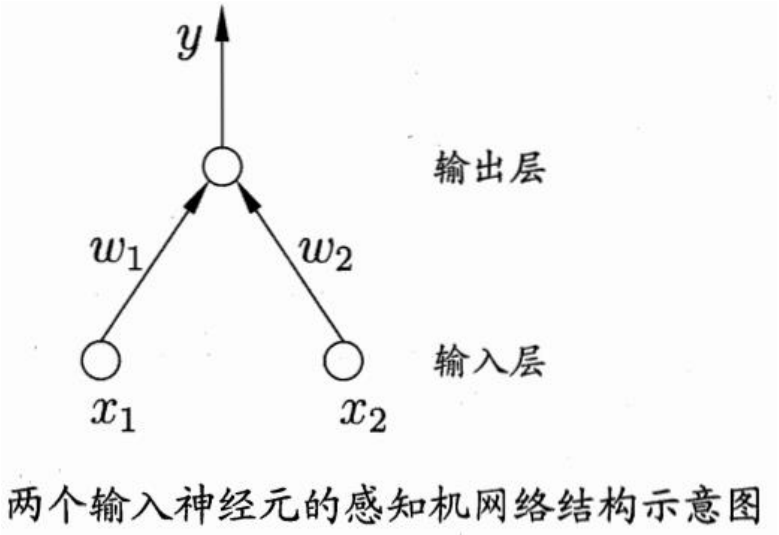
04 BP神经网络

05 玻尔兹曼机

感知机

- 感知机，是简单的二元线性分类器，由美国计算机科学家Frank Rosenblatt于1957年发明。
- 单层感知机，是具有一层神经元的前向神经网络，采用阈值激活函数。
- 单层感知机，适用于线性可分问题，通过一条直线或超平面将不同类别的样本正确分开。



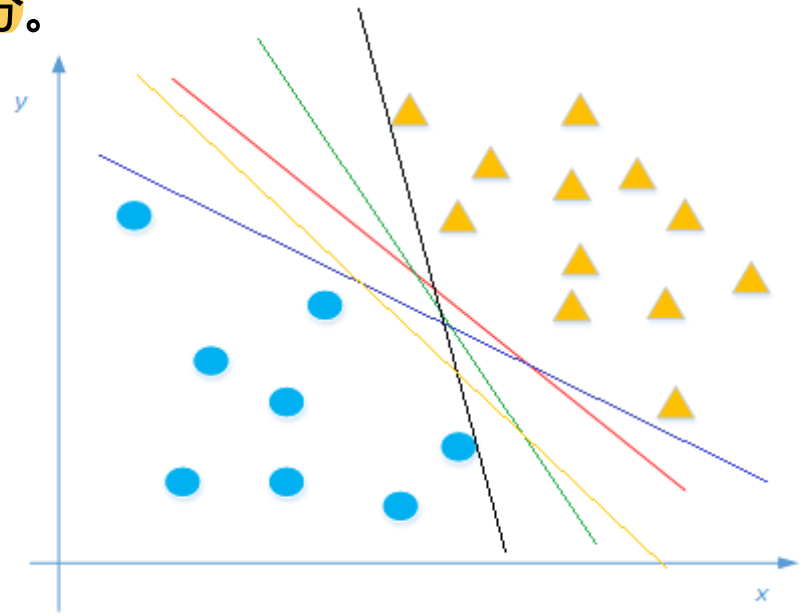


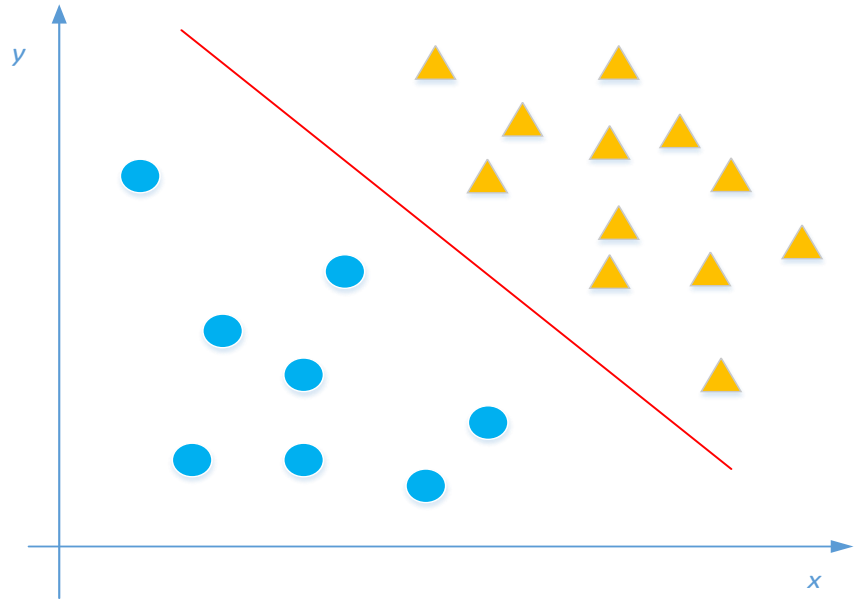
两个输入神经元的感知机网络结构示意图

线性分类

给定一个特征空间上的训练数据集 $T=\{(\boldsymbol{x}_i,y_i)|i=1,2,\cdots,N\}$，其中，$\boldsymbol{x}_i=[x_{i1},x_{i2},\cdots,x_{in}]^{\mathrm{T}}\in X\subset\mathbb{R}^n$，$y_i\in Y=\{-1,+1\},i=1,2,\cdots,N$。$\boldsymbol{x}_i$ 是训练数据集 T 的第 i 个实例，y_i 是 $\boldsymbol{x}_i$ 的类别标签。如果 $y_i=+1$，则称 $\boldsymbol{x}_i$ 为正例；反之如果 $y_i=-1$，则称 $\boldsymbol{x}_i$ 为负例。

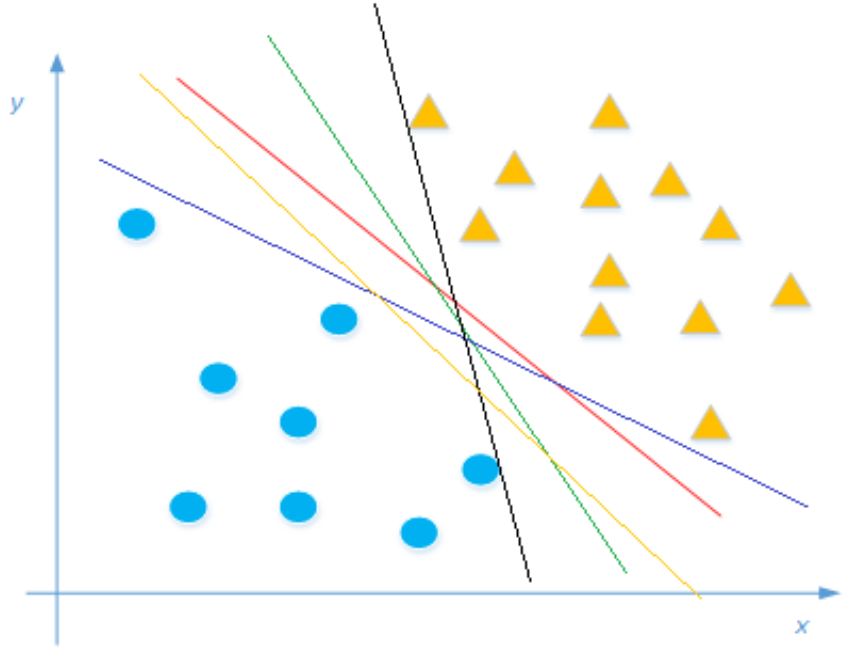
线性分类器的学习目标：是要在 n 维的数据空间中找到一个超平面，将所有的正例划分到超平面的一侧，将所有的负例划分到超平面的另一侧。此时，数据集T为线性可分数据集，否则，数据集T线性不可分。

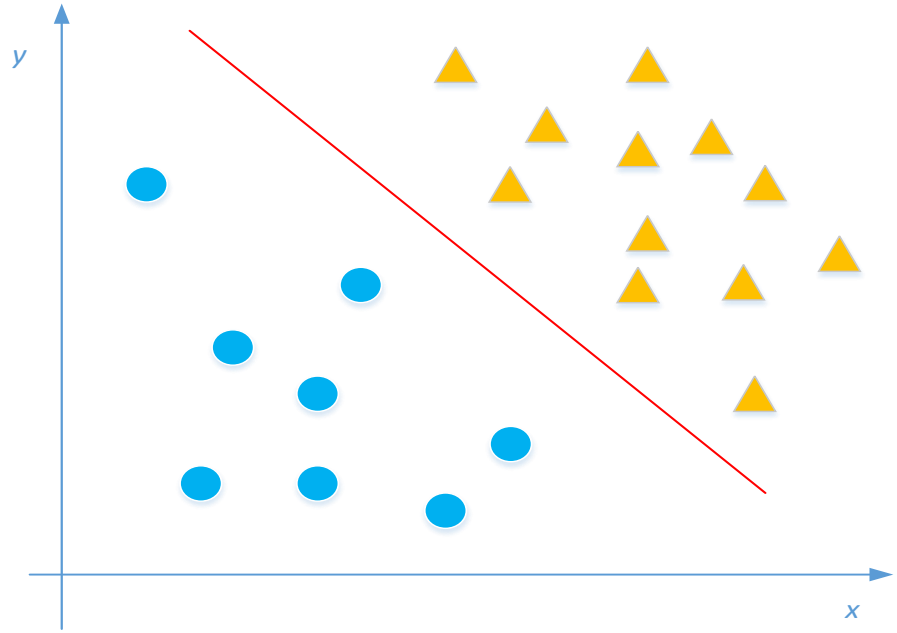




线性分类

假定超平面的方程是$\boldsymbol{w}\cdot\boldsymbol{x}+b=0$，其中 w 为超平面的法向量，b 为截距（偏置项）。显然，当训练数据集线性可分时，会有无穷多个分离超平面，它们都能将上述两类数据正确地区分开来。线性可分支持向量机提出了间隔最大化条件，以此来求得距离两个可分类的最佳超平面。

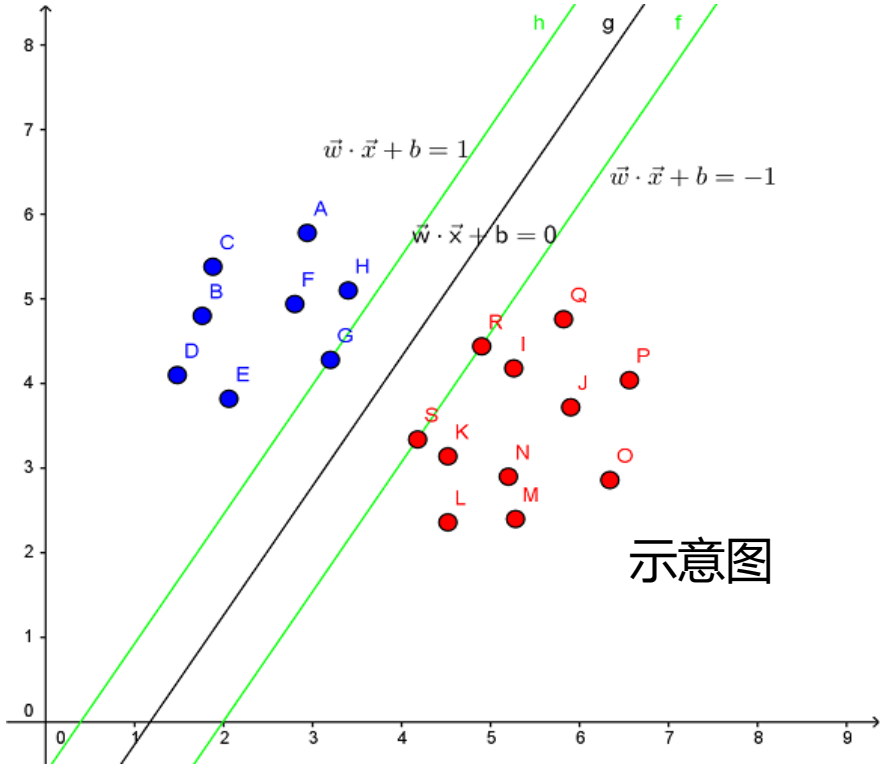




线性分类

- **定义分类函数：**

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right) = \begin{cases} +1, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b > 0 \\ 0, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0 \\ -1, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b < 0 \end{cases}$$



示意图

- **对所有样本$\forall \boldsymbol{x} \in \boldsymbol{T}$，有：**
  - 如果$f(\boldsymbol{x}) > 0$，则该样本的分类（预测）结果为$y = +1$；
  - 如果$f(\boldsymbol{x}) < 0$，则该样本的分类（预测）结果为$y = -1$；
  - 如果$f(\boldsymbol{x}) = 0$，即对于分类超平面上的所有样本点，分类（预测）结果为$y = 0$。

线性分类

- $f(\boldsymbol{x}) = \text{sign}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right) = +1$，样本$\boldsymbol{x}$位于超平面的“上方”，将样本$\boldsymbol{x}$划分为正类。
  - 如果标签$y = +1$，则分类正确，此时$yf(\boldsymbol{x}) = +1$
  - 如果标签$y = -1$，则分类错误，此时$yf(\boldsymbol{x}) = -1$
- $f(\boldsymbol{x}) = \text{sign}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right) = -1$，样本$\boldsymbol{x}$位于超平面的“下方”，将样本$\boldsymbol{x}$划分为负类。
  - 如果标签$y = -1$，则分类正确，此时$yf(\boldsymbol{x}) = +1$
  - 如果标签$y = +1$，则分类错误，此时$yf(\boldsymbol{x}) = -1$

- 显然，可以使用$yf(x)$表示分类的正确性。

感知机模型

定义 2.1（感知机） 假设输入空间（特征空间）是 $\mathcal{X} \subseteq \mathbf{R}^n$，输出空间是 $\mathcal{Y} = \{+1, -1\}$。输入 $x \in \mathcal{X}$ 表示实例的特征向量，对应于输入空间（特征空间）的点；输出 $y \in \mathcal{Y}$ 表示实例的类别。由输入空间到输出空间的如下函数：

$$f(x) = \text{sign}(w \cdot x + b) \tag{2.1}$$

称为感知机。其中，w 和 b 为感知机模型参数，$w \in \mathbf{R}^n$ 叫作权值（weight）或权值向量（weight vector），$b \in \mathbf{R}$ 叫作偏置（bias），$w \cdot x$ 表示 w 和 x 的内积。sign 是符号函数，即

$$\text{sign}(x) = \begin{cases} +1, & x \geqslant 0 \\ -1, & x < 0 \end{cases} \tag{2.2}$$

感知机模型

感知机有如下几何解释：线性方程

$$w \cdot x + b = 0 \tag{2.3}$$

对应于特征空间 $\mathbf{R}^n$ 中的一个超平面 S，其中 w 是超平面的法向量，b 是超平面的截距。这个超平面将特征空间划分为两个部分。位于两部分的点（特征向量）分别被分为正、负两类。因此，超平面 S称为分离超平面（separating hyperplane），如图 2.1 所示。

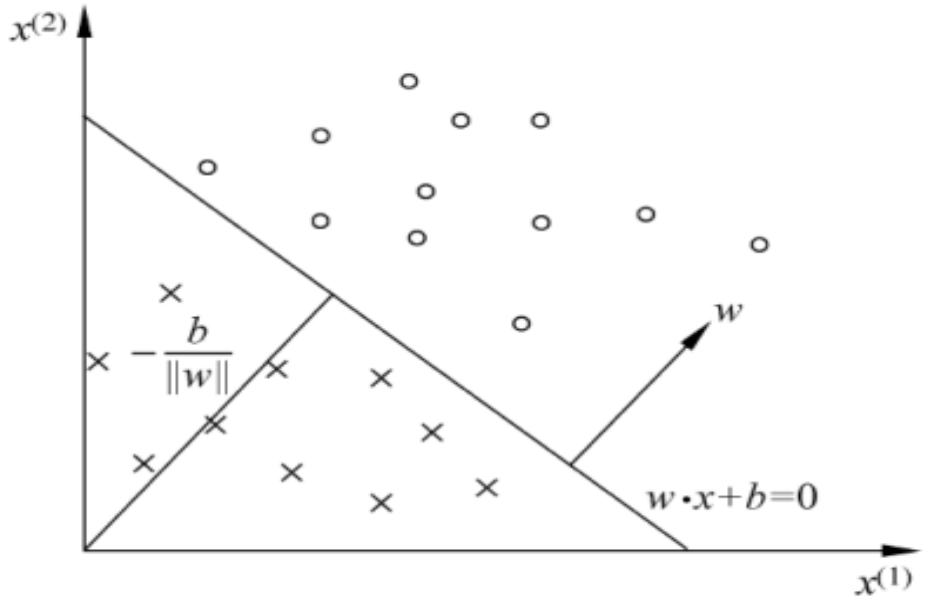


图 2.1　感知机模型

激活函数概述

- **激活函数（activation function）概述：**
  - sign( )函数，被视为一种激活函数。
  - 激活函数，是在神经网络的神经元上运行的函数，负责将神经元的输入映射到输出。
  - 激活函数的作用，引入非线性性质，使得神经网络可以更好地逼近非线性函数和模式，增强网络的表达能力。

由输入空间到输出空间的如下函数:

$$f(x) = \text{sign}(w \cdot x + b)$$

$$\text{sign}(x) = \begin{cases} +1, & x \geqslant 0 \\ -1, & x < 0 \end{cases}$$

- **激活函数类型：**
  - Sigmoid函数
  - Tanh函数
  - ReLU函数
  - Leaky ReLU函数

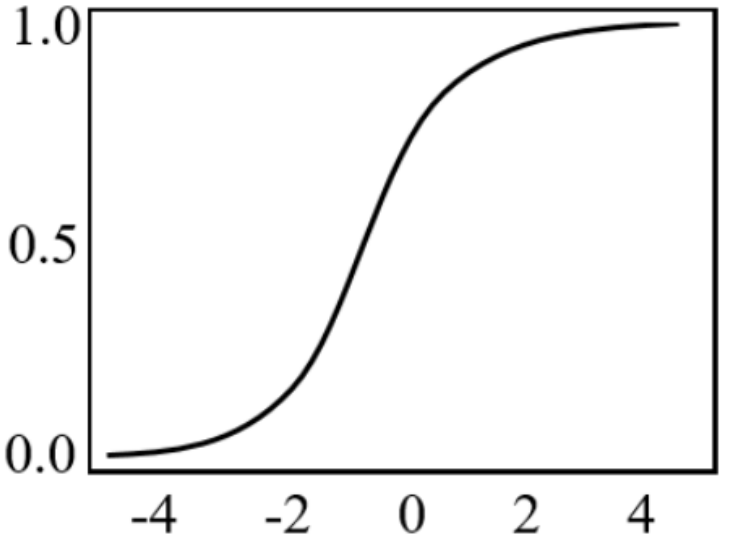
激活函数性质

- **激活函数的性质：**
  - **连续且可导：**激活函数应该在定义域上是连续且可导的，这样的性质使得梯度下降等优化算法可以在激活函数上进行有效的反向传播。
  - **简单性：**激活函数及导函数应该具备简单的表达式，通常意味着计算量较小，可以更快地进行前向传播和反向传播。
  - **适当的导函数值域：**激活函数的导函数值域应该在一个合适的区间内，合适的导函数值域可以促进训练的效率和稳定性。

激活函数类型

- **Sigmoid函数：**

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



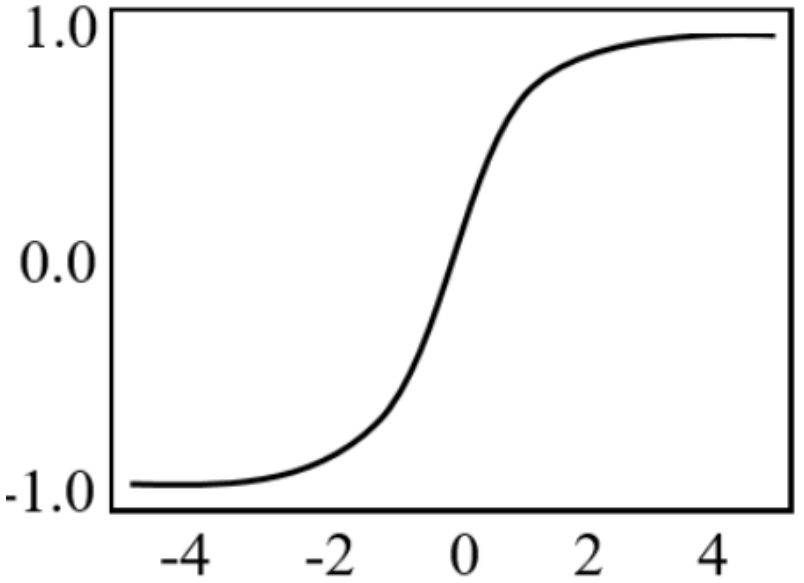
（a）Sigmoid

- Sigmoid函数将输出映射到（0,1）区间，对于任意的x 均连续可微分，提高了模型训练的稳定性。
- 然而，当 x 取极大值或极小值时，Sigmoid 函数的梯度变化极小，甚至梯度消失，导致网络更新缓慢甚至不更新。

激活函数类型

- **Tanh函数：**

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



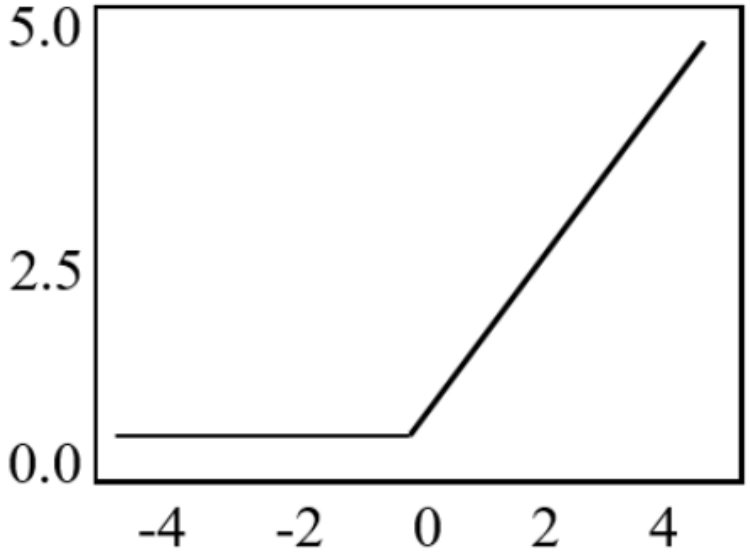
（b）Tanh

- Tanh函数将输出值映射到(-1,1)区间。
- 相比于Sigmoid，Tanh函数具有更好的容错性，并且降低了饱和区。
- 然而，在反向传播过程中梯度消失仍然未得到有效解决。

激活函数类型

- **ReLU函数：**

$$f(x) = max(0, x) = \begin{cases} 0, x \le 0 \\ x, x > 0 \end{cases}$$



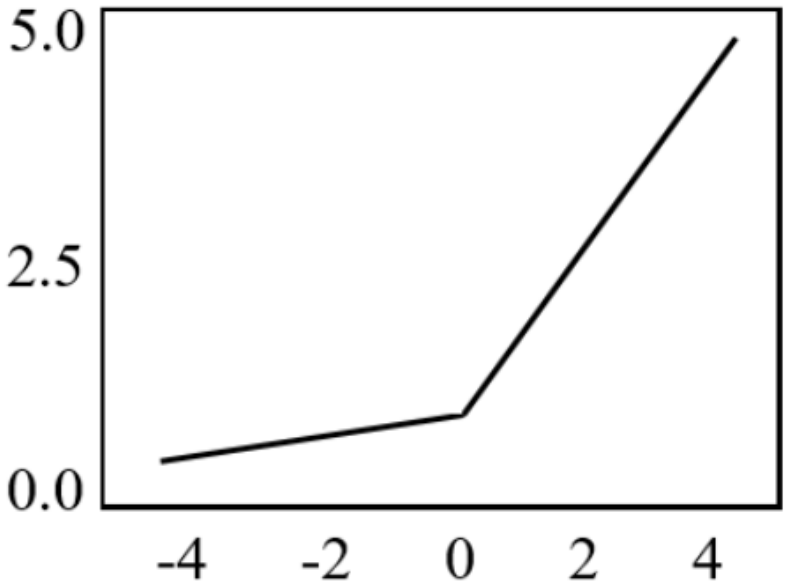
（c）Relu

- ReLU函数将输出映射到 [0,+∞} 区间。
- 当输入x>0时，ReLU函数的输出值为x，克服了梯度消失问题。
- 当输入x<0时，ReLU函数的输出和偏导数都恒为0，权重系数无法更新，出现“神经元死亡”现象。

激活函数类型

- **Leaky ReLU函数：**

$$f(x) = max(\alpha x, x)$$



（d）Leaky Relu

- Leaky ReLU函数，是ReLU函数的一种改进形式。
- Leaky ReLU函数，为负半轴赋予非0斜率，使得负输入区间上的输出和偏导数不再恒为0。
- 有效缓解了“神经元死亡”现象。

感知机学习策略

- 首先，写出输入空间R^n中任一点x_0 到超平面 S 的距离，有**公式1**：

  ||w|| 是 w 的L_2范数。

  向量的L1范数：直接求和
  L2：平方求和开根号

$$\frac{1}{\|w\|}|w \cdot x_0 + b|$$

- 其次，对于误分类的数据(x_i, y_i)来说，有**公式2**：

$$-y_i(w \cdot x_i + b) > 0$$

- 因此，误分类点x_i到超平面S的距离，有**公式3**：

$$-\frac{1}{\|w\|}y_i(w \cdot x_i + b)$$

感知机学习策略

- 假设超平面 S 的误分类点集合为 M，那么所有误分类点到超平面 S 的总距离，有**公式4**：

$$-\frac{1}{\|w\|}\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

不考虑1/||w||，就得到感知机学习的损失函数。

感知机学习策略

- 给定训练数据集：

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

$$x_i \in \mathcal{X} = \mathbf{R}^n,\ y_i \in \mathcal{Y} = \{+1, -1\},\ i = 1, 2, \cdots, N$$

- 感知机sign(wx+b)学习的损失函数定义为**公式5**：

$$L(w, b) = -\sum_{x_i \in M} y_i (w \cdot x_i + b)$$

其中 M 为误分类点的集合。这个损失函数就是感知机学习的经验风险函数。

感知机学习策略

- **感知机学习的经验风险函数：**

$$L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

- 损失函数 L(w,b) 是非负的；
- 如果没有误分类点，损失函数值是 0；
- 误分类点越少，误分类点离超平面越近，损失函数值就越小；
- 给定训练数据集T，损失函数 L(w,b) 是 w, b的连续可导函数；
- 感知机学习策略，是在假设空间中选取使损失函数最小的模型参数 w、b。

为什么可以不写 1/‖w‖?
感知机的核心目标是“把误分类变少并推回正确一侧”，它并不像 SVM 那样强制用几何间隔的规范化来最大化间隔。

从优化角度，保留 1/‖w‖ 会让目标函数变复杂（而且
w 的缩放会引入额外麻烦）；感知机通常用更直接的形式来做迭代更新。

感知机学习算法的原始形式

算法 2.1（感知机学习算法的原始形式）

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathcal{X}=\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{-1,+1\}$，$i=1,2,\cdots,N$；学习率 $\eta(0<\eta\leqslant 1)$；

输出：w,b；感知机模型 $f(x)=\mathrm{sign}(w\cdot x+b)$。

（1）选取初值 w_0,b_0；

（2）在训练集中选取数据 (x_i,y_i)；

（3）如果 $y_i(w\cdot x_i+b)\leqslant 0$，

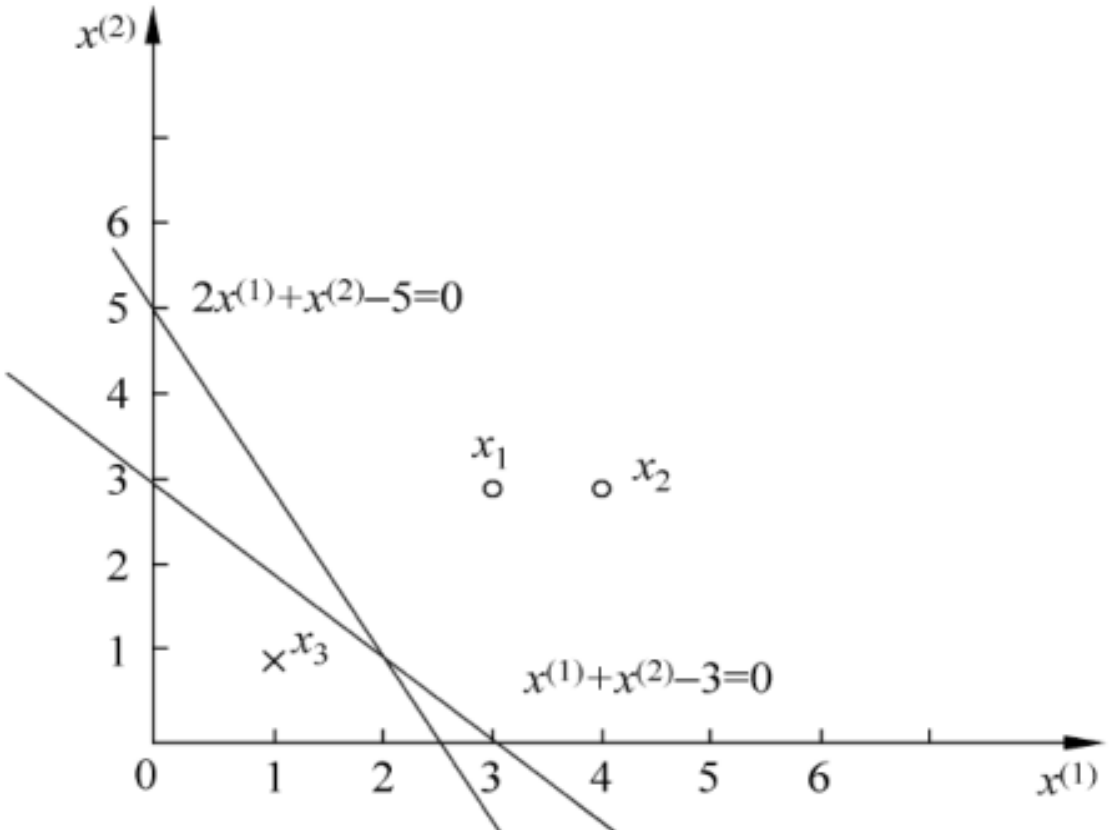
$$w\leftarrow w+\eta y_i x_i$$

$$b\leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点。

例题1

例 2.1 如图 2.2 所示的训练数据集，其正实例点是 $x_1 = (3,3)^{\mathrm{T}}$，$x_2 = (4,3)^{\mathrm{T}}$，负实例点是 $x_3 = (1,1)^{\mathrm{T}}$，试用感知机学习算法的原始形式求感知机模型 $f(x) = \mathrm{sign}(w \cdot x + b)$。这里，$w = (w^{(1)}, w^{(2)})^{\mathrm{T}}$，$x = (x^{(1)}, x^{(2)})^{\mathrm{T}}$。



例题1

解 构建最优化问题：

$$\min_{w,b} L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

按照算法 2.1 求解 w, b。$\eta = 1$。

（1）取初值 $w_0 = 0$，$b_0 = 0$

（2）对 $x_1 = (3,3)^{\mathrm{T}}$，$y_1(w_0 \cdot x_1 + b_0) = 0$，未能被正确分类，更新 w，b

$$w_1 = w_0 + y_1 x_1 = (3,3)^{\mathrm{T}}, \quad b_1 = b_0 + y_1 = 1$$

得到线性模型

$$w_1 \cdot x + b_1 = 3x^{(1)} + 3x^{(2)} + 1$$

例题1

(3) 对 x_1, x_2，显然，$y_i(w_1 \cdot x_i + b_1) > 0$，被正确分类，不修改 w, b;

对 $x_3 = (1,1)^{\mathrm{T}}$，$y_3(w_1 \cdot x_3 + b_1) < 0$，被误分类，更新 w, b。

$$w_2 = w_1 + y_3 x_3 = (2,2)^{\mathrm{T}},\ b_2 = b_1 + y_3 = 0$$

得到线性模型

$$w_2 \cdot x + b_2 = 2x^{(1)} + 2x^{(2)}$$

如此继续下去，直到

$$w_7 = (1,1)^{\mathrm{T}},\ b_7 = -3$$

$$w_7 \cdot x + b_7 = x^{(1)} + x^{(2)} - 3$$

对所有数据点 $y_i(w_7 \cdot x_i + b_7) > 0$，没有误分类点，损失函数达到极小。

分离超平面为：$x^{(1)} + x^{(2)} - 3 = 0$

感知机模型为：$f(x) = \mathrm{sign}(x^{(1)} + x^{(2)} - 3)$

例题1

表 2.1　例 2.1 求解的迭代过程

迭代次数	误分类点	w	b	$w \cdot x + b$
0		0	0	0
1	x_1	$(3,3)^{\mathrm{T}}$	1	$3x^{(1)} + 3x^{(2)} + 1$
2	x_3	$(2,2)^{\mathrm{T}}$	0	$2x^{(1)} + 2x^{(2)}$
3	x_3	$(1,1)^{\mathrm{T}}$	-1	$x^{(1)} + x^{(2)} - 1$
4	x_3	$(0,0)^{\mathrm{T}}$	-2	-2
5	x_1	$(3,3)^{\mathrm{T}}$	-1	$3x^{(1)} + 3x^{(2)} - 1$
6	x_3	$(2,2)^{\mathrm{T}}$	-2	$2x^{(1)} + 2x^{(2)} - 2$
7	x_3	$(1,1)^{\mathrm{T}}$	-3	$x^{(1)} + x^{(2)} - 3$
8	0	$(1,1)^{\mathrm{T}}$	-3	$x^{(1)} + x^{(2)} - 3$

1、误分类点，依次取：

x1,x3,x3,x3,x1,x3,x3

得到分离超平面：

$$x^{(1)} + x^{(2)} - 3 = 0$$

2、误分类点，依次取：

x1,x3,x3,x3,x2,x3,x3,x3,x1,x3,x3

得到分离超平面：

$$2x^{(1)} + x^{(2)} - 5 = 0$$

感知机学习算法的对偶形式

算法 2.2（感知机学习算法的对偶形式）

输入：线性可分的数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathbf{R}^n$，$y_i\in\{-1,+1\}$，$i=1,2,\cdots,N$；学习率 $\eta\ (0<\eta\leqslant 1)$；

输出：α,b；感知机模型 $f(x)=\operatorname{sign}\left(\sum_{j=1}^{N}\alpha_j y_j x_j\cdot x+b\right)$，其中 $\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_N)^{\mathrm{T}}$。

（1）$\alpha\leftarrow 0$，$b\leftarrow 0$；

（2）在训练集中选取数据 (x_i,y_i)；

（3）如果 $y_i\left(\sum_{j=1}^{N}\alpha_j y_j x_j\cdot x_i+b\right)\leqslant 0$，

$$\alpha_i\leftarrow\alpha_i+\eta$$

$$b\leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点。

将w替换为j=1∑3α_jy_jx_j

只要f(x_i)*y_i 小于等于0
a_i 一定加1 b_i 加y_i(正负一)

感知机学习算法的对偶形式

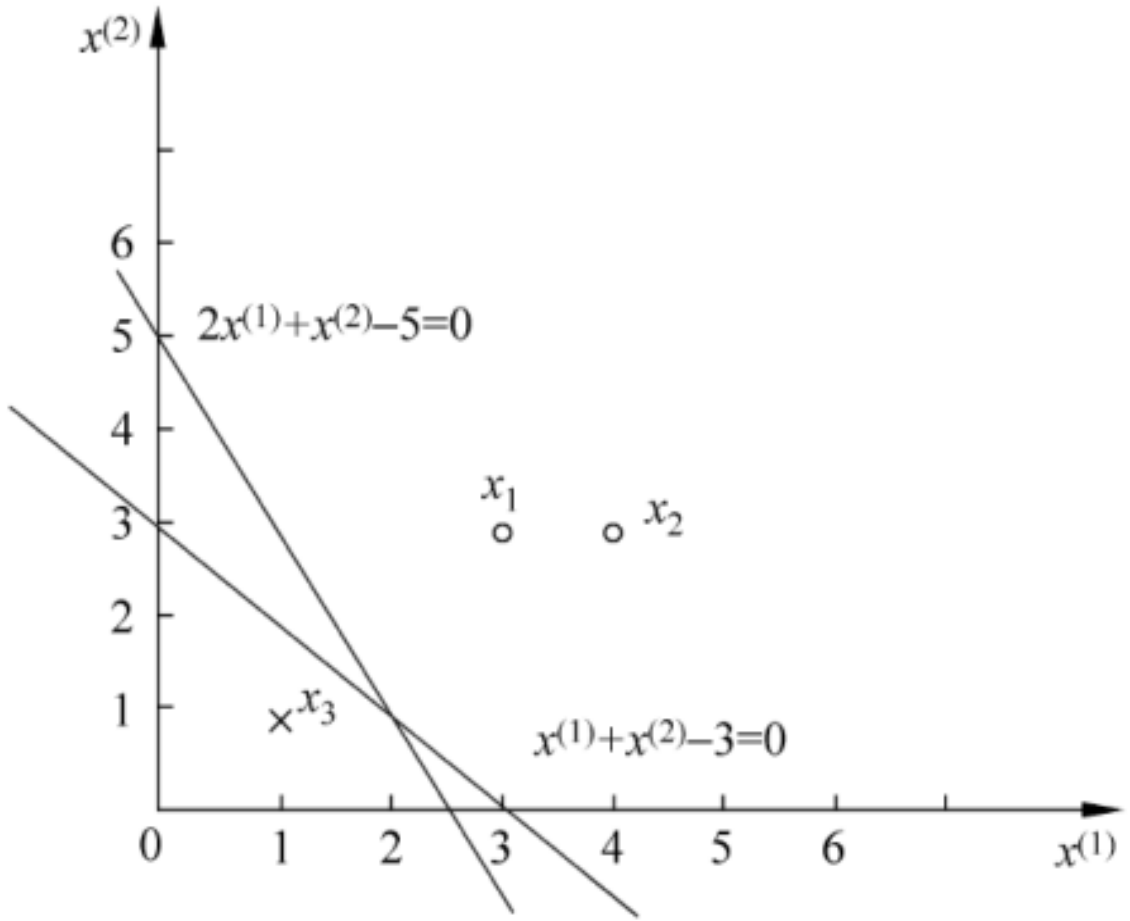
Gram 矩阵：预先将训练集中实例间的内积计算出来，并以矩阵的形式存储。

$$G = [x_i \cdot x_j]_{N \times N}$$

内积，是指两个向量之间对应元素相乘后的累加和。

例题2

例 2.2 数据同例 2.1，正样本点是 $x_1 = (3,3)^{\mathrm{T}}$，$x_2 = (4,3)^{\mathrm{T}}$，负样本点是 $x_3 = (1,1)^{\mathrm{T}}$，试用感知机学习算法对偶形式求感知机模型。



例题2

例 2.2 数据同例 2.1，正样本点是 $x_1 = (3,3)^{\mathrm{T}}$，$x_2 = (4,3)^{\mathrm{T}}$，负样本点是 $x_3 = (1,1)^{\mathrm{T}}$，试用感知机学习算法对偶形式求感知机模型。

解 按照算法 2.2，

（1）取 $\alpha_i = 0$，$i = 1,2,3$，$b = 0$，$\eta = 1$；

（2）计算 Gram 矩阵

$$\boldsymbol{G} = \begin{bmatrix} 18 & 21 & 6 \\ 21 & 25 & 7 \\ 6 & 7 & 2 \end{bmatrix}$$

对于点 x_1，有

$$y_1\left(\sum_{j=1}^{N} \alpha_j^{<0>} y_j x_j \cdot x_1 + b^{<0>}\right) = 0$$

（3）误分条件

$$y_i\left(\sum_{j=1}^{N} \alpha_j y_j x_j \bullet x_i + b\right) \leqslant 0$$

参数更新

$$\alpha_i \leftarrow \alpha_i + 1,\ b \leftarrow b + y_i$$

例题2

（4）迭代。过程从略，结果列于表 2.2；

（5）

$$w = 2x_1 + 0x_2 - 5x_3 = (1, 1)^{\mathrm{T}}$$

$$b = -3$$

分离超平面

$$x^{(1)} + x^{(2)} - 3 = 0$$

感知机模型

$$f(x) = \mathrm{sign}(x^{(1)} + x^{(2)} - 3)$$

例题2

表 2.2　例 2.2 求解的迭代过程

k	0	1	2	3	4	5	6	7
		x_1	x_3	x_3	x_3	x_1	x_3	x_3
α_1	0	1	1	1	1	2	2	2
α_2	0	0	0	0	0	0	0	0
α_3	0	0	1	2	3	3	4	5
b	0	1	0	-1	-2	-1	-2	-3

原始形式vs对偶形式

- **感知机学习算法的原始形式**
  - 直接基于权重向量和偏置项的表示，通过迭代对每个训练样本进行更新。
  - 直观易懂，适用于线性分类问题。
  - 对于大规模数据集，原始形式算法可能需要进行大量的权重更新，计算复杂度较高。

- **感知机学习算法的对偶形式**
  - 将权重向量和偏置项表示为训练样本的线性组合，通过 Gram 矩阵来存储样本之间的内积。
  - 避免了重复计算样本之间的内积，适用于处理大规模数据集和非线性分类问题。
  - 使用 Gram 矩阵存储样本之间的内积，通过Gram矩阵操作更新权重。相比于原始形式，对偶形式的计算复杂度更低。

感知机的局限性

- **感知机神经网络的局限性**
  - **二分类输出：**感知机的传输函数一般采用阈值函数，导致其输出只能是两种离散值；
  - **线性可分性限制：**单层感知机网络只能解决线性可分的分类问题；
  - **单层结构：**感知机学习算法只适用于单层感知机网络，即只有一个神经元层的结构。

前情回顾--感知机学习算法的原始形式

算法 2.1（感知机学习算法的原始形式）

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathcal{X}=\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{-1,+1\}$，$i=1,2,\cdots,N$；学习率 $\eta(0<\eta\leqslant 1)$；

输出：w,b；感知机模型 $f(x)=\mathrm{sign}(w\cdot x+b)$。

（1）选取初值 w_0,b_0；

（2）在训练集中选取数据 (x_i,y_i)；

（3）如果 $y_i(w\cdot x_i+b)\leqslant 0$，

$$w\leftarrow w+\eta y_i x_i$$

$$b\leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点。

前情回顾--感知机学习算法的对偶形式

算法 2.2（感知机学习算法的对偶形式）

输入：线性可分的数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathbf{R}^n$，$y_i\in\{-1,+1\}$，$i=1,2,\cdots,N$；学习率 $\eta\ (0<\eta\leqslant 1)$；

输出：α,b；感知机模型 $f(x)=\operatorname{sign}\left(\sum_{j=1}^{N}\alpha_j y_j x_j\cdot x+b\right)$，其中 $\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_N)^{\mathrm{T}}$。

（1）$\alpha\leftarrow 0$，$b\leftarrow 0$；

（2）在训练集中选取数据 (x_i,y_i)；

（3）如果 $y_i\left(\sum_{j=1}^{N}\alpha_j y_j x_j\cdot x_i+b\right)\leqslant 0$，

$$\alpha_i\leftarrow\alpha_i+\eta$$

$$b\leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点。

前情回顾--感知机学习算法的对偶形式

Gram 矩阵：预先将训练集中实例间的内积计算出来，并以矩阵的形式存储。

$$G = [x_i \cdot x_j]_{N \times N}$$

内积，是指两个向量之间对应元素相乘后的累加和。

本章目录

01 神经元

02 感知机

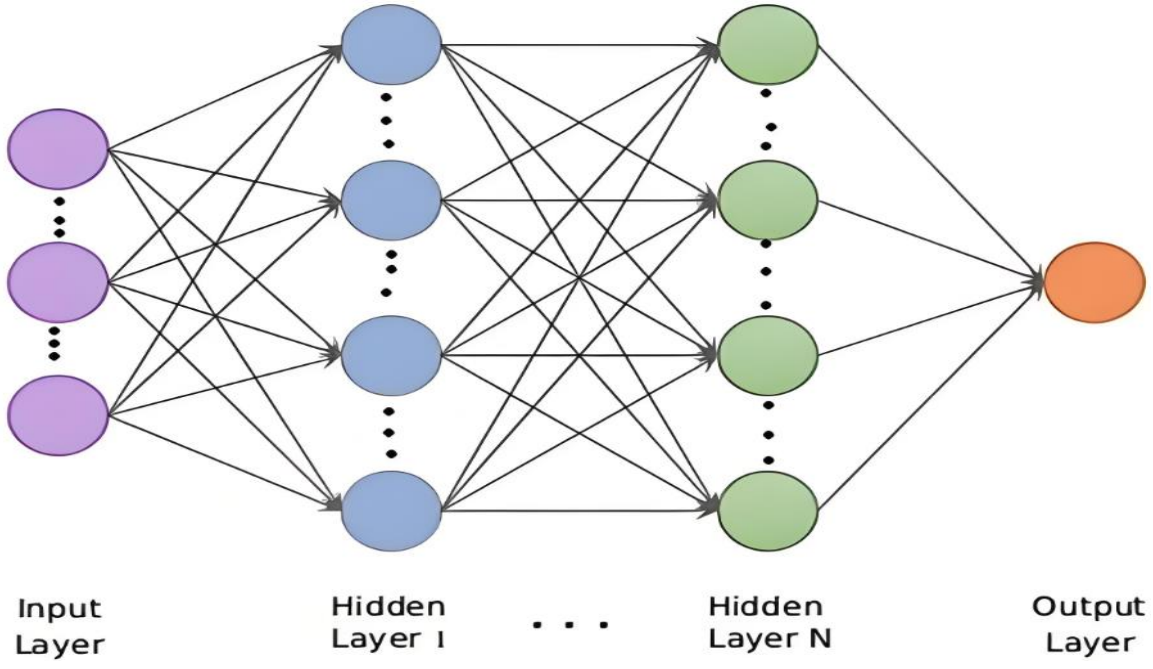
03 前馈神经网络

04 BP神经网络

05 玻尔兹曼机

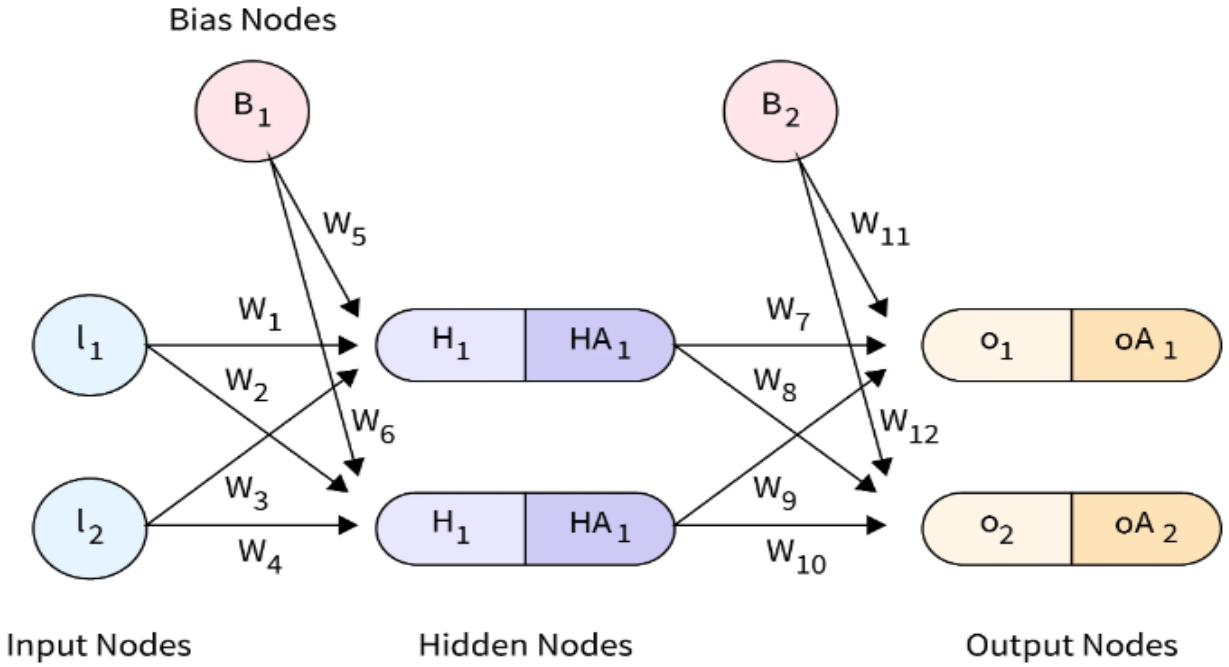
前馈神经网络

- 前馈神经网络，由相互连接的“神经元”组成，输入通过第一层，输出由最后一层产生，输入层和输出层之间可能有任意数量的隐藏层。
- 每个神经元都有相关的权重和偏差，这些权重和偏差在训练过程中会进行调整。
- 经过训练后，网络可以处理新的输入，捕捉输入数据中的模式和相关性，并产生输出。

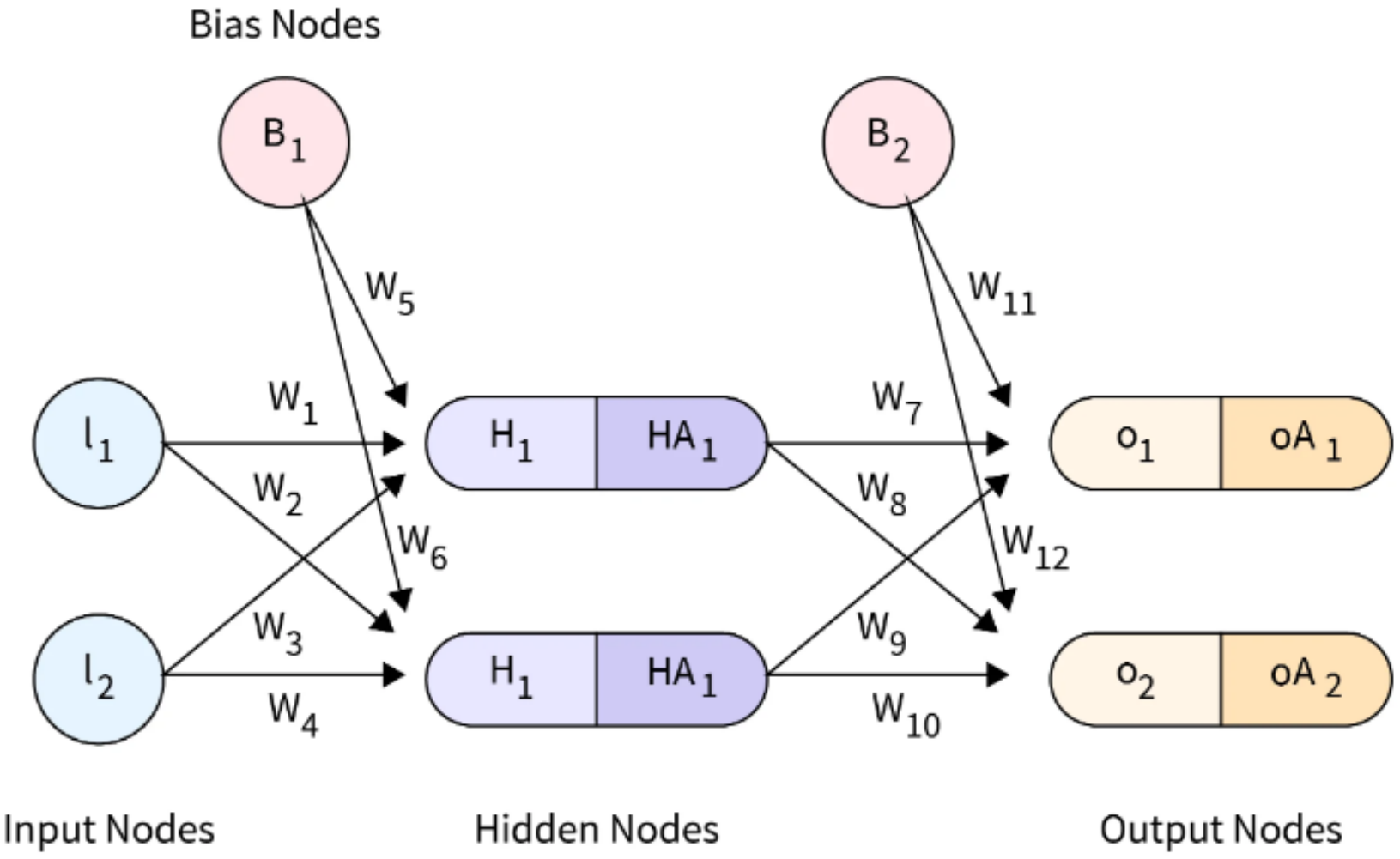


前馈神经网络

- **前馈神经网络**，多层感知器（Multilayer Perceptron，MLP），不会循环回馈。
  - **输入层：**接受原始输入数据，并将其传递到下一层。
  - **隐藏层：**处理和转换输入数据的一个或多个隐藏层，激活函数引入非线性。
  - **输出层：**生成最终输出，输出层的神经元数量取决于问题的类型。



前馈神经网络



L：输入节点

W：连接权重

H：隐藏节点

HA：激活的隐藏节点

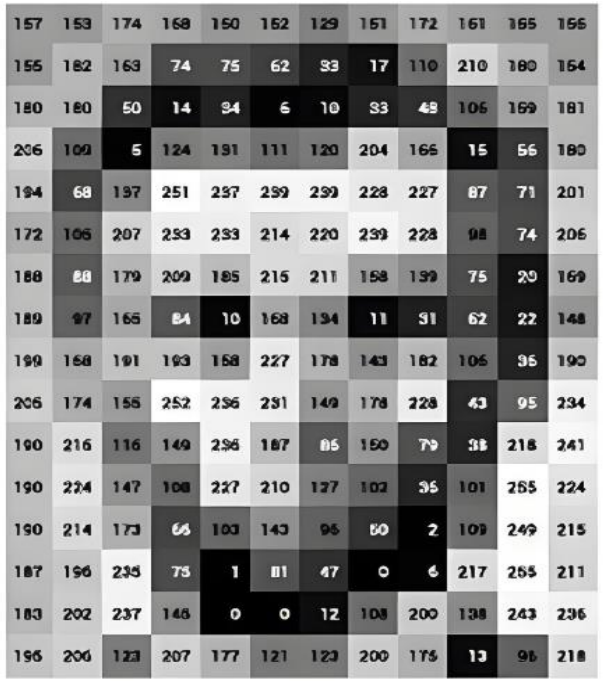
O：输出节点

OA：激活的输出节点

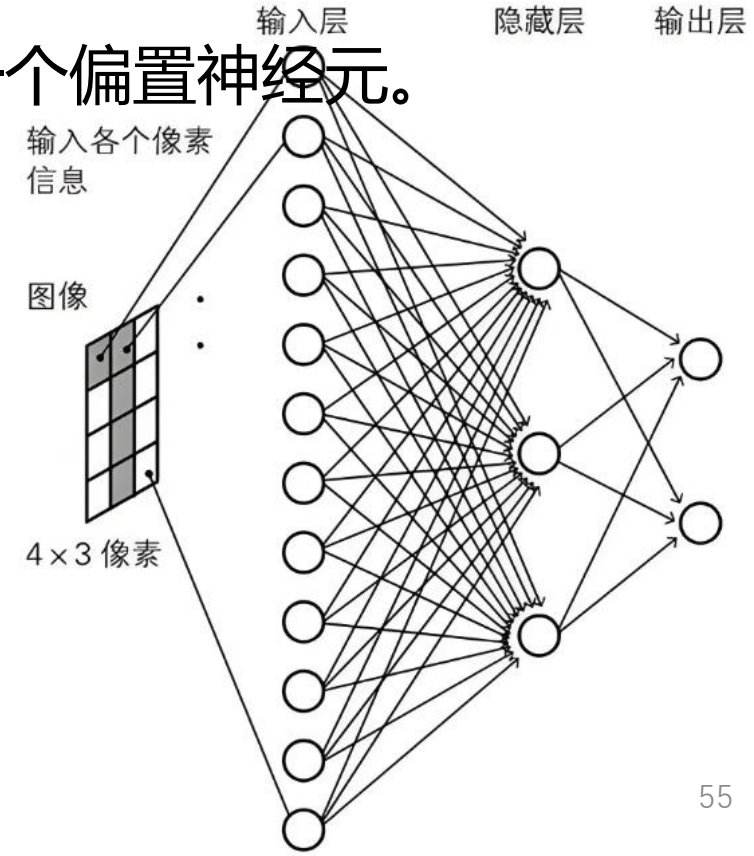
B：偏置节点

前馈神经网络--输入层

- 输入层不会对数据执行任何计算或转换，仅充当输入数据的占位符。
- 输入层具有多个神经元，对应于输入数据中的要素数量。
- 输入层接受输入数据并将其馈送到下一层，没有可学习的参数。
- 输入层向输入数据添加信息，例如，在输入层添加一个偏置神经元。

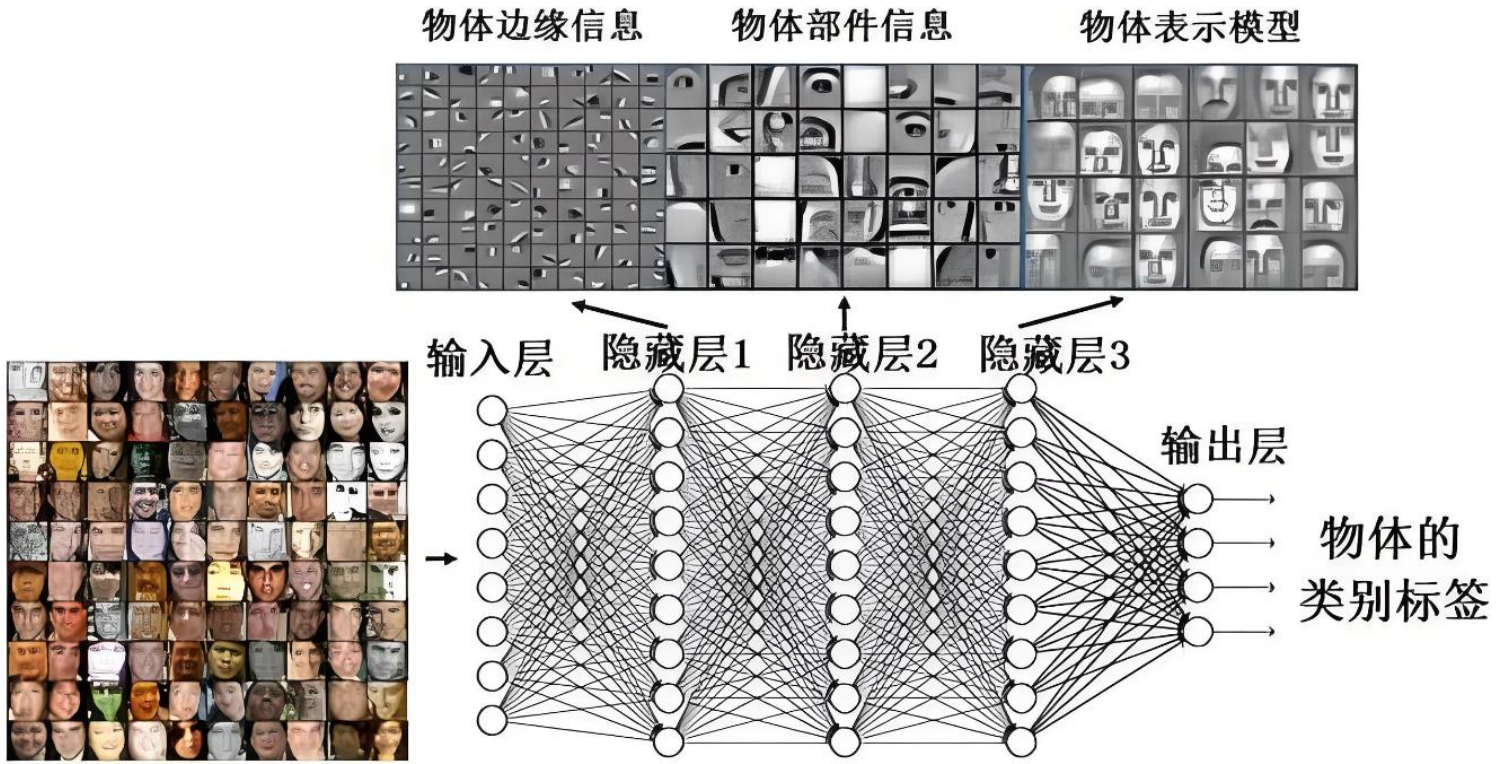


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
206	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218



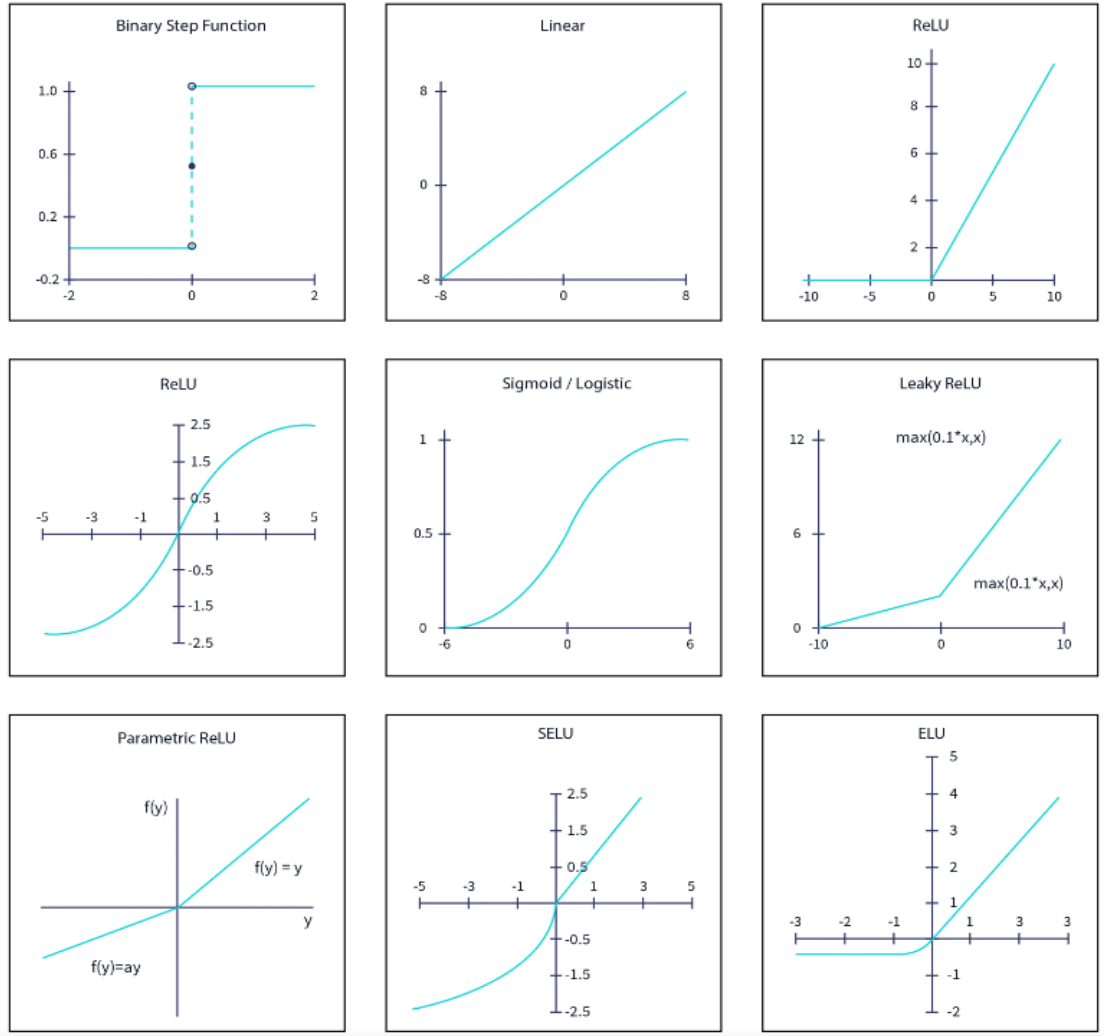
前馈神经网络--隐藏层

- 在前馈神经网络中，隐藏层不直接与外部环境交互。
- 提取输入数据的特征和抽象表示，学习输入和输出之间更复杂和抽象的关系。
- 激活函数用于隐藏层，将非线性引入网络。



前馈神经网络--输出层

- 输出层，根据处理后的输入数据，生成网络的最终输出。
- 输出层的神经元数量，通常与任务的输出维度或类别数量相对应，每个神经元都会生成一个概率值，指示属于每个类的输入数据的概率。
- 输出层，通常具有一组可学习的参数，通过最小化损失函数来优化网络的预测能力。
- 输出层的常见激活函数，用于二元分类的 sigmoid 和用于多类分类的 softmax。



小节

- 前馈神经网络，将神经元分层排列，分别是_____、 _____和_____ 。
- 前馈神经网络是一种没有_____的神经网络，信号只在一个方向上流动，从输入层到输出层。
- 前馈神经网络的训练过程，通常使用_____来更新网络的权重和偏置。
- 在前馈神经网络中，二分类问题常用的激活函数为_____ ； 多分类问题常用的激活函数为_____。

本章目录

01 神经元

02 感知机

03 前馈神经网络

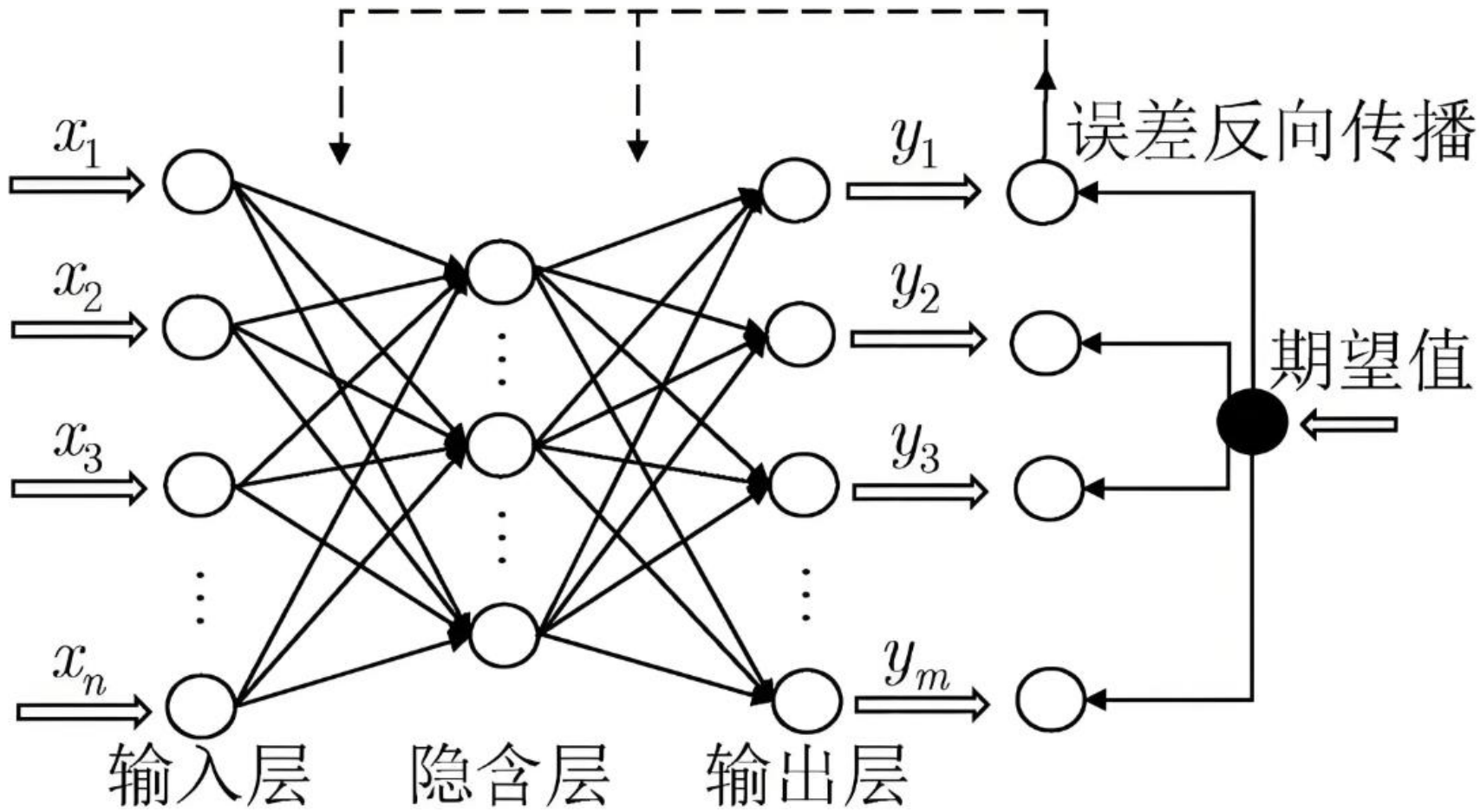
04 BP神经网络

05 玻尔兹曼机

BP神经网络概述

- BP神经网络，反向传播神经网络（Backpropagation Neural Network），多层的前馈神经网络，主要特点是：**信号是前向传播的，误差是反向传播的。**
- BP神经网络的过程主要分为两个阶段：
  - 信号的前向传播，从输入层，经隐含层，最后到输出层；
  - 误差的反向传播，从输出层，到隐含层，最后到输入层，调整权重和偏置。

BP神经网络概述



BP神经网络算法流程

- **前向传播：**将输入样本送入网络，通过网络的每一层进行信号传递，计算出网络的输出结果。
- **误差计算：**将网络的输出结果与期望输出进行比较，计算出网络的误差。
- **反向传播：**从输出层开始，将误差信号反向传播回隐藏层和输入层。通过链式法则，计算每个连接权重对误差的贡献，并将误差信号传递给前一层的神经元。
- **参数更新：**根据计算得到的梯度信息，使用梯度下降优化方法更新网络的连接权重和偏置，以减小误差。（梯度的负方向）
- **重复迭代：**重复执行前向传播、误差计算、反向传播和参数更新的步骤，直到达到停止条件，如达到最大迭代次数或误差降至可接受的范围。

反向传播算法

- **核心思想：**
  - 将输出误差以某种形式反传给所有的单元，各层按照本层误差修正连接权值。

- **两个问题需要考虑：**
  1. 误差如何表示？--损失函数
  2. 权值如何修正？--梯度下降

损失函数

- 神经网络损失函数，也称为目标函数，衡量神经网络的输出与预期值之间的距离，以便控制、调节参数。
  - L1Loss
  - 均方误差损失
  - 平滑L1损失
  - 交叉熵损失
  - 二进制交叉熵损失

损失函数-- L1Loss

- **L1范数损失（也称为曼哈顿距离）**，用于衡量模型预测值与真实值之间的差异。
- **L1范数损失的定义：**

$$\text{L1 范数损失} = \sum |y - \hat{y}|$$

其中，y表示真实值，ŷ表示模型的预测值，∑表示求和运算。

- 通过最小化L1范数损失，鼓励模型产生稀疏的权重或特征选择。

损失函数-- L1Loss

L1 范数损失的特点

- **稀疏性**：L1 范数损失能够产生稀疏的权重，即某些权重会被直接设置为零。这种稀疏性有助于减少模型的复杂度，提高模型的可解释性。
- **鲁棒性**：由于 L1 范数对异常值的敏感度较低，因此在存在异常值的数据集中，L1 范数损失可能比 L2 范数损失表现得更好。
- **特征选择**：L1 范数损失可以作为一种特征选择方法，通过将不重要的特征权重设置为零，自动选择出对模型贡献较大的特征。

损失函数--MSELoss

- **均方误差损失（Mean Squared Error Loss，简称MSE损失）**，用于衡量预测值与真实值差异的平方的平均值。
- **MSE损失函数定义：**

$$\text{MSE 损失} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

其中，y表示真实值，ŷ表示模型的预测值，n表示样本数量，∑表示求和运算。

损失函数--MSELoss

- **MSE损失的优点：**

  对较大的预测误差有较高的惩罚

  在数学性质上比较好

  但是，MSE损失对异常值较为敏感

损失函数--SmoothL1Loss

- **平滑L1损失（也称为Huber损失）**，用于回归任务的损失函数，在L1损失和L2损失之间提供了平滑过渡，减少异常值对损失函数的敏感性。
- **SmoothL1Loss的定义：**

$$\text{SmoothL1}(y, \hat{y}) = \begin{cases} 0.5\,(y-\hat{y})^2, & |y-\hat{y}| < 1 \\ |y-\hat{y}| - 0.5, & \text{otherwise} \end{cases}$$

其中，y表示真实值，ŷ表示模型的预测值。

损失函数--SmoothL1Loss

- **平滑L1损失特点：**

平滑度：相比于L2损失，SmoothL1Loss在接近真实值时具有更好的平滑性，对离群值的响应相对较小。

鲁棒性：由于平滑过渡区域，SmoothL1Loss对于异常值的影响较小，使得模型更具鲁棒性。

渐进性：在预测值与真实值之间的差异较小时，SmoothL1Loss接近平方差损失，有利于模型学习小的差异。

损失函数--CELoss

- **交叉熵损失**（CrossEntropyLoss，简称CE损失），衡量模型的预测概率分布与真实标签之间的差异，交叉熵损失的计算方式依赖于模型的输出形式。
- **交叉熵损失的定义：**

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

其中，y表示真实值，ŷ表示模型的预测值。

损失函数--CELoss

- **CE损失的本质：**

衡量两个概率分布之间的距离，当两个概率分布越接近时，交叉熵损失越小，表示模型预测结果越准确。

- **CE损失特点：**

CE损失的计算方式，依赖于模型的输出形式和真实标签的表示方式。

在多分类任务中，通常采用 softmax 激活函数+交叉熵损失函数的组合。

损失函数--BCELoss

- **二进制交叉熵损失**（Binary Cross-Entropy Loss，简称BCE损失），用于衡量模型的二分类输出与真实标签之间的差异。
- **BCE损失的定义：**

  BCE损失 = -[y * log(ŷ) + (1 - y) * log(1 - ŷ)]

  L_{BCE} = -y * log(ŷ)，当真实标签为1时；

  L_{BCE} = -(1-y) * log(1-ŷ)，当真实标签为0时；

  其中，y表示真实标签（取值为0或1），ŷ表示模型的预测概率（取值在0到1之间）。

损失函数--BCELoss

- **BCE损失函数的优点：**

  在二分类问题中比较直观且易于计算

  对于模型预测接近真实标签的情况有较低的损失

  对于模型预测明显错误的情况有较高的惩罚

梯度下降算法概述

- 在反向传播算法中，每一层都需要依据反传回来的误差调整权重，目标是尽可能让误差最小。
- 沿着梯度的负方向调整，将会最快逼近误差的最小值，即：损失函数 (目标函数) 的极小值。
- **步骤：**
  - 定义损失函数
  - 优化损失函数：通过求损失函数极小值确定参数，损失函数E对参数w的梯度表示为：

$$\nabla \mathrm{E} = \frac{dE}{dw} = \frac{\partial E}{\partial w}$$

- 梯度方向：函数值增大最快的方向
- 梯度负方向：下降最快的方向

梯度下降算法概述

- **概述：**
  - 梯度下降 (Gradient Descent，简称GD)，最小化风险函数或损失函数。

- **先导知识：**
  - 导数：当函数定义域和取值都在实数域中，导数表示函数曲线上的切线斜率。
  - 偏导数：是多元函数的导数的一种形式，表示函数在某个特定变量上的变化率。
  - 方向导数：在多元函数的微积分中，方向导数表示函数在给定方向上的变化率。
  - 梯度：梯度是一个矢量，在其方向上的方向导数最大，也就是函数在该点处沿着梯度的方向变化最快，变化率最大。

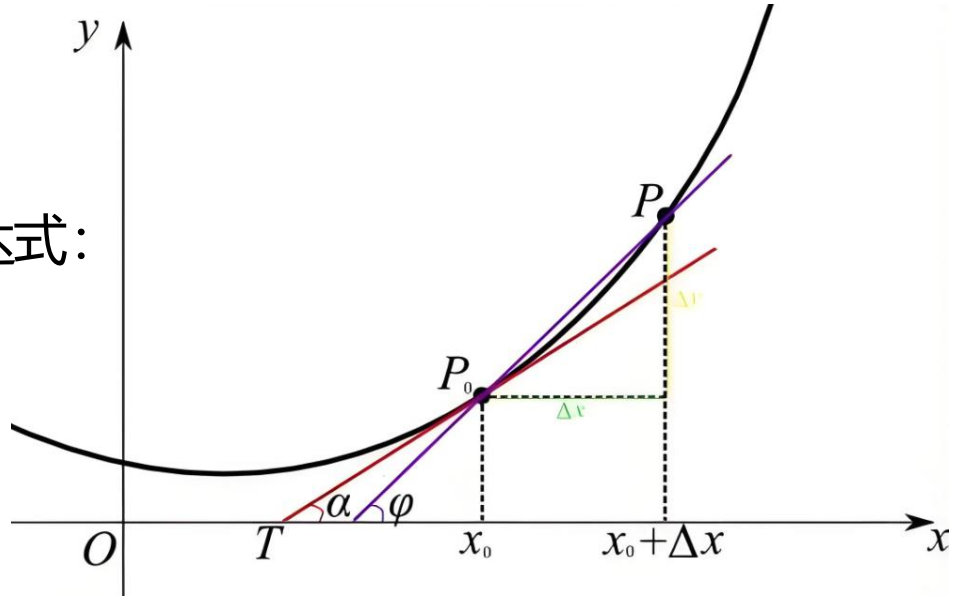
导数

- **导数：**当函数定义域和取值都在实数域中，导数表示函数曲线上的切线斜率。

- **导数的定义：**
  - 对于一个函数 f(x)，它在某一点 x 处的导数，使用极限表达式：

  f'(x) = lim (h→0) [f(x+h)-f(x)] / h

  其中，h 是自变量的增量。

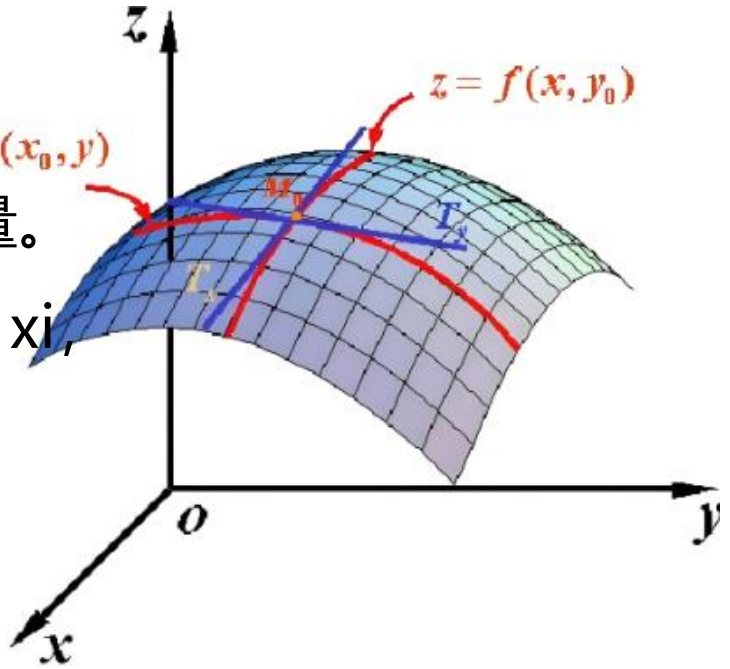


- **导数的意义：**
  - 导数有助于理解函数的变化情况，包括函数在某一点的变化速率和变化趋势。
  - 导数的正负号指示函数的增减性，导数的绝对值表示函数的斜率大小。
  - 导数在机器学习中有广泛应用，例如求解函数的最值、优化问题、判断函数的凸凹性等。

偏导数

- **偏导数：**是多元函数的导数的一种形式，表示函数在某个特定变量上的变化率，同时将其他变量视为常数。
- **偏导数的定义：**
  - 考虑一个多元函数 $f(x_1, x_2, ..., xn)$，其中 $x_1, x_2, ..., xn$ 是自变量。
  - 偏导数表示函数在某个特定变量上的变化率，对于每个自变量 xi，偏导数表示为：

    ∂f/∂xi
- **偏导数的意义：**
  - 偏导数是函数在不同自变量上的变化趋势
  - 只关注某个特定自变量的变化对函数的影响
  - 每个偏导数都表示了函数在相应自变量上的变化率



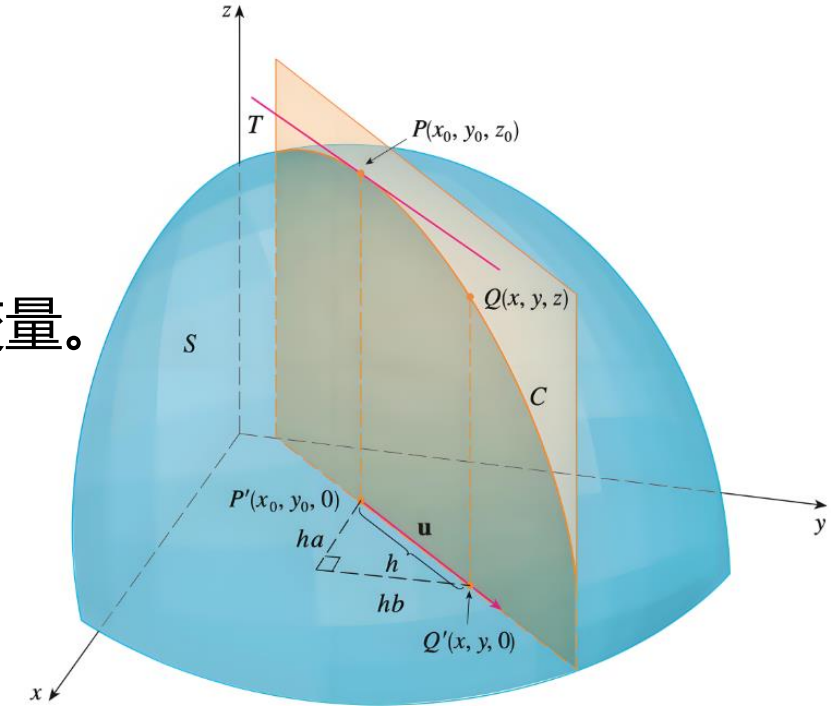
方向导数

- **方向导数：**方向导数表示函数在给定方向上的变化率。

- **方向导数的定义：**
  - 假设多元函数 $f(x_1, x_2, ..., xn)$，其中 $x_1, x_2, ..., xn$是自变量。
  - 对于给定的方向向量 $v = (v_1, v_2, ..., vn)$：

    $Dv\ f = \nabla f \cdot v$

    其中，∇f 表示函数 f 的梯度，$\cdot$ 表示向量的点积操作。



- **方向导数的意义：**
  - 描述函数在某个特定方向上的变化率，而不仅仅是某个特定变量上的变化率。
  - 梯度的方向与方向导数最大的方向一致，而梯度的模表示方向导数的最大值。

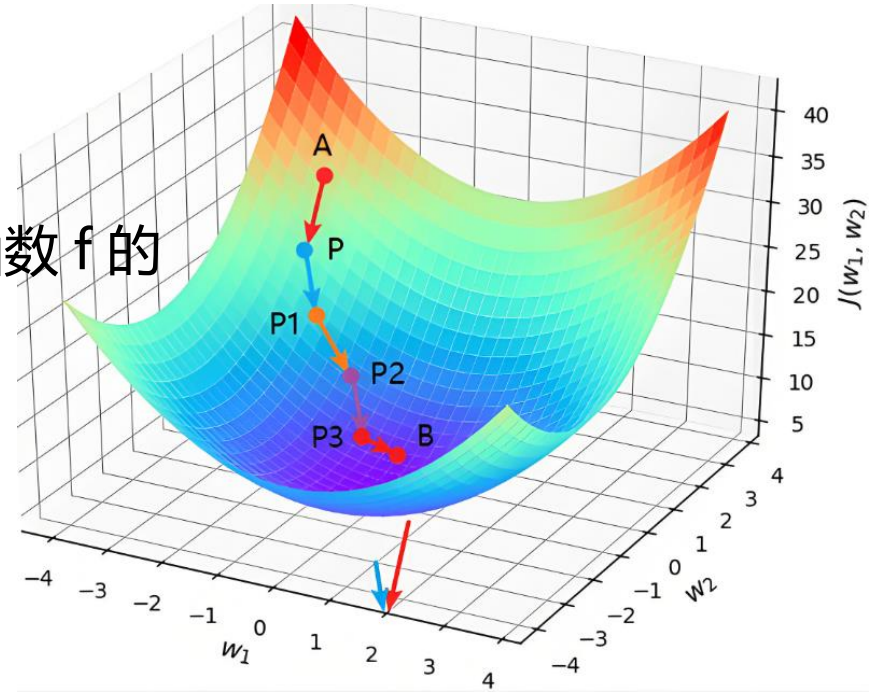
梯度

- **梯度：**矢量，表示多元函数在某一点上的变化率最大的方向。

- **梯度的定义：**
  - 考虑多元函数 $f(x_1, x_2, ..., xn)$，其中 $x_1, x_2, ..., xn$ 是自变量。函数 f 的梯度表示为 ∇f 或 $grad(f)$。
  - 梯度是一个矢量，其分量为函数 f 在每个自变量上的偏导数。

    $\nabla f = (\partial f/\partial x_1, \partial f/\partial x_2, ..., \partial f/\partial xn)$

    其中，$\partial f/\partial xi$ 表示函数 f 关于第 i 个自变量的偏导数。

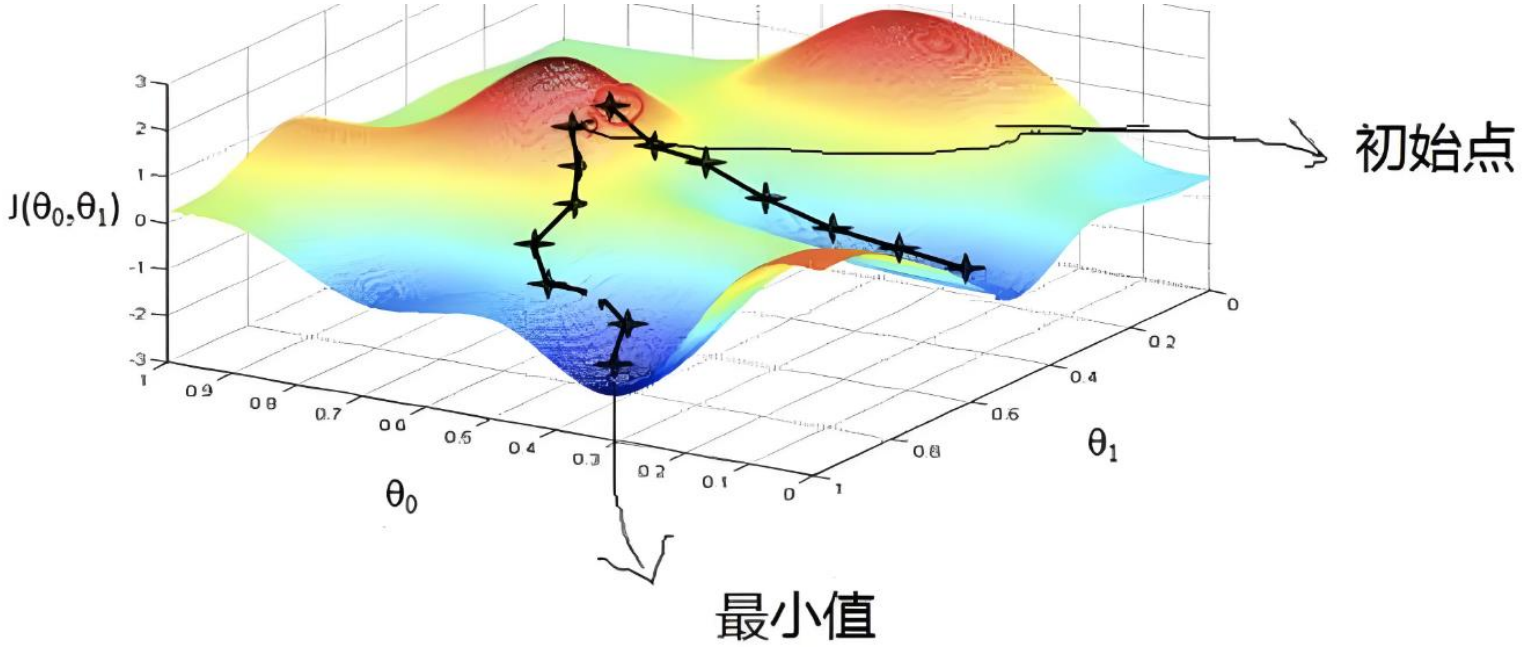


- **梯度的意义：**
  - 梯度的方向是函数在某一点上变化最快的方向，梯度的模表示变化率的最大值。
  - 梯度下降指导搜索方向以寻找函数的最小值点。
  - 梯度只提供了局部变化率的信息，而不是整个函数的变化趋势。

梯度下降

- **举个例子：**

你站在山上某处，想要尽快下山，于是决定走一步算一步，也就是每走到一个位置时，求解当前位置的梯度，沿着梯度的负方向，也就是当前最陡峭的位置向下走，这样一直走下去，很可能走不到山脚，而是某个局部的山峰最低处。



梯度下降

- **一些重要的概念：**

1. **步长（Learning rate）：** 每一步梯度下降时，向目标方向前行的长度。
   - 步长越长，在陡峭区域下降的越快，在平缓区易出现反复抖动而找不到最优点；
   - 步长越短，越不易产生抖动，但是容易陷入局部最优解。
2. **假设函数（hypothesis function）：** 在监督学习中，为了拟合输入样本，而使用的假设函数。
   - 用h( )表示，对于线性回归模型，Y = W0 + W1X1 + W2X2 + ... + WnXn 。
3. **损失函数（loss function）：** 为了评估模型的好坏，通常用损失函数来度量拟合的程度。
   - 损失函数最小化，意味着拟合程度最好，对应的模型参数即为最优参数。

梯度下降

- **梯度下降法步骤：**

1. 确定模型的假设函数和损失函数
2. 初始化模型的参数
3. 根据当前参数值计算损失函数的值
4. 计算损失函数对参数的梯度
5. 根据梯度和学习率（控制每次更新的步长），更新参数值
6. 重复步骤3-5，直到达到停止条件（例如，达到最大迭代次数或损失函数的变化很小）

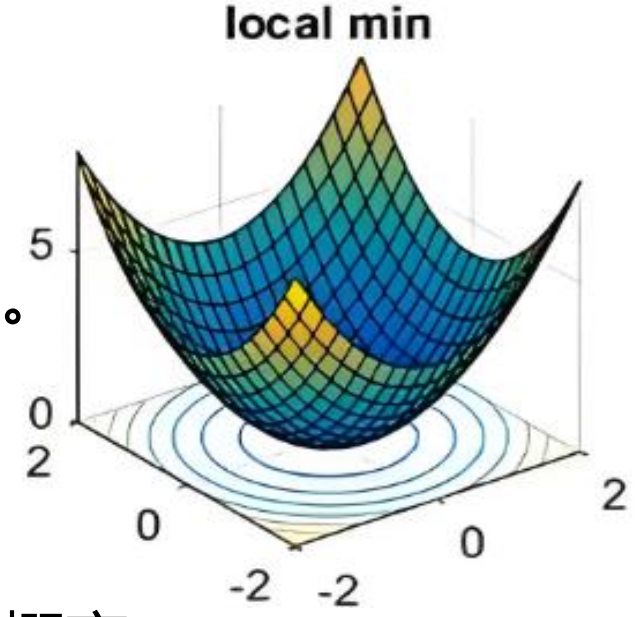
梯度下降

- **梯度下降会遇到所有最优化问题中常见的两个问题：局部最小值和鞍点。**
  - **局部最小值：**

    在复杂的非凸优化问题中，存在多个局部最小值。

    梯度下降算法，可能会收敛到局部最小值而不是全局最小值。

    算法可能会陷入局部最小值，无法达到全局最小值。



  - **解决方法：**

    多次运行：每次使用不同的初始点，增加找到全局最小值的概率。

    改进的优化算法：使用随机梯度下降、牛顿法或拟牛顿法等。

梯度下降

- **梯度下降会遇到所有最优化问题中常见的两个问题：局部最小值和鞍点。**
  - **鞍点：**

    鞍点是指在某一维度上是局部最小值，而在其他维度上是局部最大值的点。

    在高维优化问题中，鞍点比局部最小值更常见。

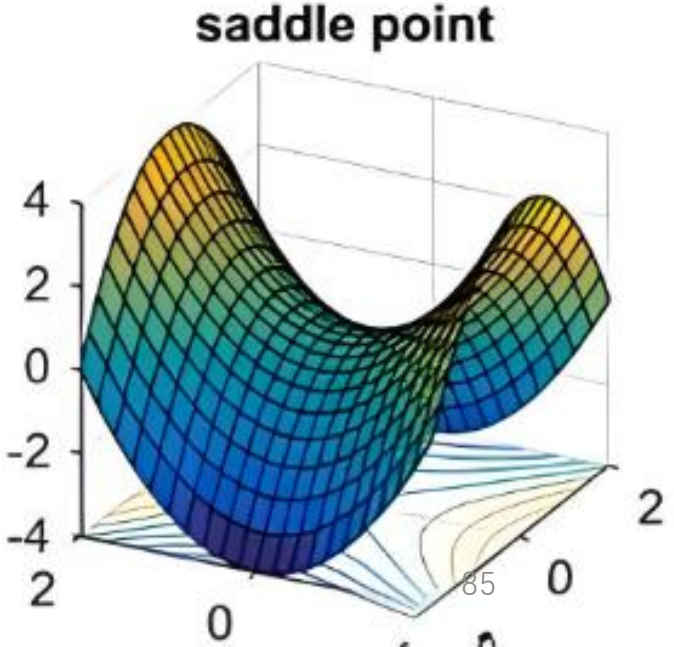
    鞍点的存在使得梯度接近于零，导致梯度下降算法收敛速度变慢或停滞不前。

  - **解决方法：**

    随机性：引入随机性，跳出鞍点。

    加入噪声：引入噪声，跳出鞍点。

    改进的优化算法：牛顿法、拟牛顿法、随机梯度下降的变体等。



梯度下降

- **梯度下降法的不同形式：**
  - 批量梯度下降BGD （Batch GradientDescent）
  - 随机梯度下隆SGD （Stochastic GradientDescent）
  - 小批量梯度下降MBGD （Mini-BatchGradient Descent）

梯度下降

- **批量梯度下降（Batch Gradient Descent，BGD）：**
  - 批量梯度下降是最基本的梯度下降形式。
  - 在每次迭代中，使用训练集的所有样本计算损失函数关于参数的梯度，并更新参数。
  - 由于需要计算所有样本的梯度，BGD的计算成本通常较高，特别是在大型数据集上。
  - 适用于小型数据集和凸优化问题。

梯度下降

- **随机梯度下降（Stochastic Gradient Descent，SGD）：**
  - 在每次迭代中，随机选择单个样本进行梯度计算和参数更新的梯度下降形式。
  - 与BGD相比，SGD的计算成本较低。
  - 单个样本的梯度估计，可能存在较大方差，SGD在更新参数时，可能出现更大的波动。
  - SGD通常在大规模数据集上更高效，但收敛速度可能较慢。

梯度下降

- **小批量梯度下降（Mini-Batch Gradient Descent，MBGD）：**
  - 小批量梯度下降是介于BGD和SGD之间的一种形式。
  - 在每次迭代中，MBGD选择一个较小的批量样本进行梯度计算和参数更新。
  - MBGD综合了BGD的稳定性和SGD的计算效率。
  - 在实际应用中， MBGD是一种平衡计算效率和收敛速度的一种折中选择。

梯度下降

- **梯度下降法的应用：**
  - 线性回归
  - 逻辑回归
  - 神经网络

通过最小化损失函数，梯度下降使模型能够逐步优化参数，以便更好地拟合训练数据，并提高在新数据上的泛化能力。

小节

- **填空题**
- BP神经网络，是多层的前馈神经网络，主要特点是：_____，_____。
- 神经网络的_____，衡量了神经网络输出与预期值之间的距离。
- 沿着_____调整参数，是实现误差最小化的最高效路径。
- 梯度下降法通常会陷入：_____和_____问题。
- **简答题**
- 请简述BP 神经网络的训练流程。
- 请简述梯度下降法。

本章目录

01 神经元

02 感知机

03 前馈神经网络

04 BP神经网络

05 玻尔兹曼机

玻尔兹曼机概述

- 玻尔兹曼机（Boltzmann Machine，简称BM），是一类对称耦合的、随机反馈型的二值单元神经网络。
- BM网络结点分为可见单元和隐藏单元
- 神经元取值1或0两种状态，状态1表示该接通，状态0表示断开。
- 每个神经元有两个状态：$s_i \in \{0,1\}$，网络的能量函数：

$$E = -\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} w_{ij}s_i s_j - \sum_{i=1}^{N}\theta_i s_i$$

其中，w_{ij}是神经元*i*和*j*之间的连接权值，θ_i是神经元s_i的阈值。

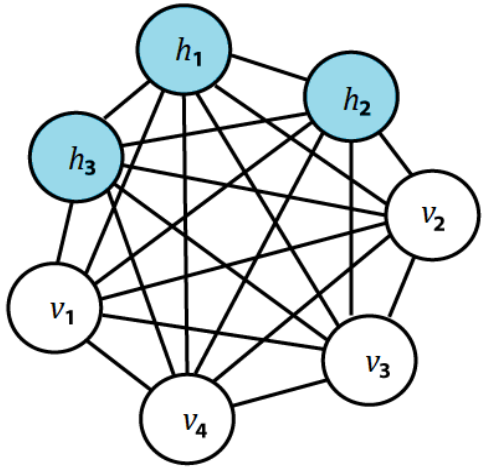
- 网络中任意的2个全局状态α和β，全局状态α和β出现的概率与相应的网络能量之间满足：

$$\frac{p_\alpha}{p_\beta} = e^{-(E_\alpha - E_\beta)/T}$$

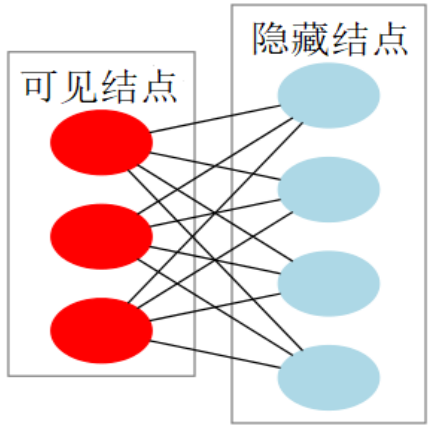
即为著名的Boltzmann分布。

玻尔兹曼机的拓扑结构

- $\boldsymbol{S} = [\boldsymbol{V}^{\mathrm{T}}, \boldsymbol{H}^{\mathrm{T}}]^{\mathrm{T}}$，
  - **可见单元向量**$\boldsymbol{V} = \{0,1\}^{D}$，由输入结点和输出结点组成，是网络和环境之间的接触面，表示可观察的数据；
  - **隐藏单元向量**$\boldsymbol{H} = \{0,1\}^{K}$，由隐藏结点组成，不与外界环境直接接触，用于隐含信息和学习特征。
- **W**是可见层结点到隐藏层结点的对称连接权重矩阵，**L**是可见层结点到可见层结点的对称连接权重矩阵，**R**是隐藏层结点到隐藏层结点的对称连接权重矩阵。

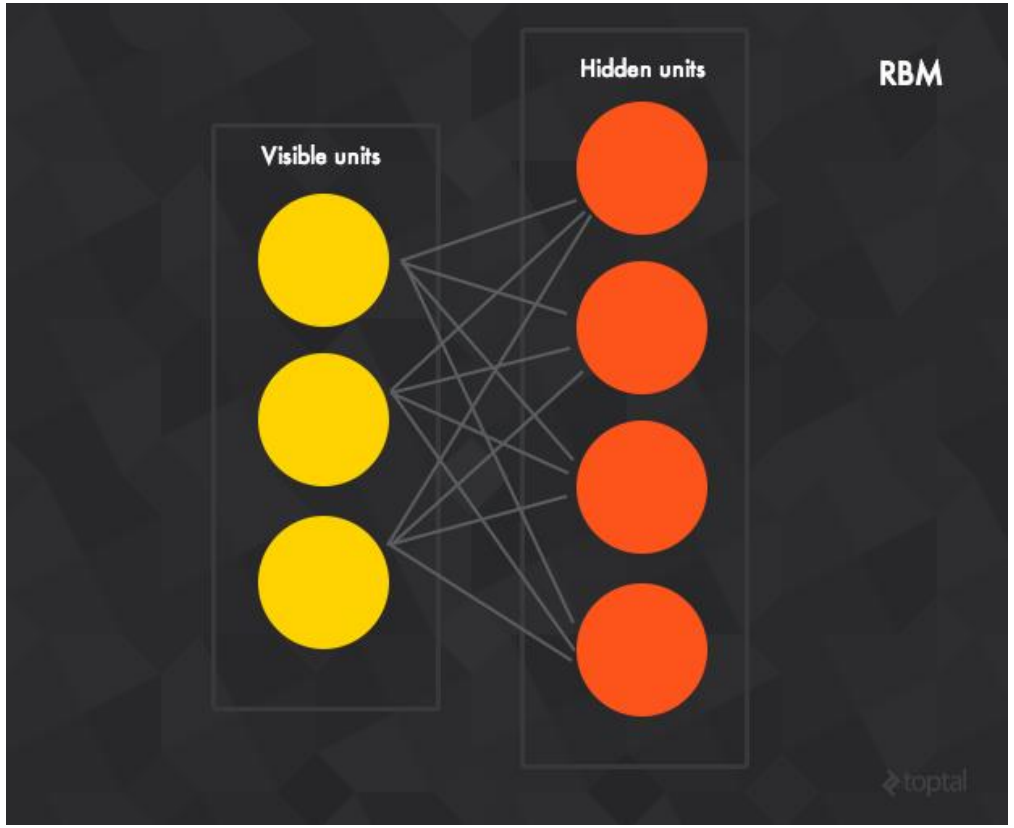


(a) 一般BM是一个全互联的网络



(b) RBM是只有可见结点与隐藏结点互联的BM

玻尔兹曼机的拓扑结构



- 当BM的**W**，**L**和**R**均不为0，BM即为一般BM。
- 当BM的**W**和**L**不为0，**R**=0时，BM演化为半受限BM。
- 当BM的$\mathbf{W} \neq 0$，**L**和**R**均为0时，BM就是目前应用较为广泛的受限玻尔兹曼机（Restricted BM，简称RBM）

BM的学习过程

- **BM的学习过程分为两个阶段：**
  - **约束学习阶段：**将可见单元状态值取为训练样本值，采样得到隐藏单元。可见单元向量$\boldsymbol{v}_\alpha$出现第α个状态的概率$p^+(\boldsymbol{v}_\alpha)$由训练样本集决定。
  - **自由学习阶段：**网络自由运行，系统稳定时可见单元向量$\boldsymbol{v}_\alpha$出现第α个状态的概率$p^-(\boldsymbol{v}_\alpha)$由模型决定。

BM的学习过程

BM学习是通过调整连接权矩阵，使模型定义的概率分布$p^-(\boldsymbol{v}_\alpha)$尽可能地与训练样本集定义的概率分布$p^+(\boldsymbol{v}_\alpha)$相一致。用K-L离差（Kullback-Leibler divergence）度量这两个概率分布的接近程度。

学习步骤如下：

1、随机设定网络的初始连接权值$w_{ij}(0)$及初始温度T;

2、根据已知概率$p(\boldsymbol{v}_\alpha)$依次给定训练样本，在训练样本的约束下按照SA算法运行网络，直到平衡状态。

3、统计出各个p_{ij}^+，在无约束条件下按同样的步骤运行网络相同的次数，统计出各个p_{ij}^-；

修改每个权值w_{ij}：

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}$$

4、重复上面的步骤，直到$p_{ij}^+ - p_{ij}^-$小于某个预设值。

总结

01 神经元

02 感知机

03 前馈神经网络

04 BP神经网络

05 玻尔兹曼机

作业

例 2.1 如图 2.2 所示的训练数据集，其正实例点是 $x_1 = (3,3)^{\mathrm{T}}$，$x_2 = (4,3)^{\mathrm{T}}$，负实例点是 $x_3 = (1,1)^{\mathrm{T}}$，试用感知机学习算法的原始形式求感知机模型 $f(x) = \mathrm{sign}(w \cdot x + b)$。这里，$w = (w^{(1)}, w^{(2)})^{\mathrm{T}}$，$x = (x^{(1)}, x^{(2)})^{\mathrm{T}}$。