机器学习

第6章 人工神经网络

朱桂祥 (9120201070@nufe.edu.cn)
南京财经大学信息工程学院
江苏省电子商务重点实验室
电子商务信息处理国家级国际联合研究中心
电子商务交易技术国家地方联合工程实验室

https://github.com/zgx881205/Machine-Learning/tree/main

本章目录

01 发展历史

02 感知机算法

03 BP算法

1.人工神经网络发展历史

01 发展历史

02 感知机算法

03 BP算法

1.人工神经网络发展历史

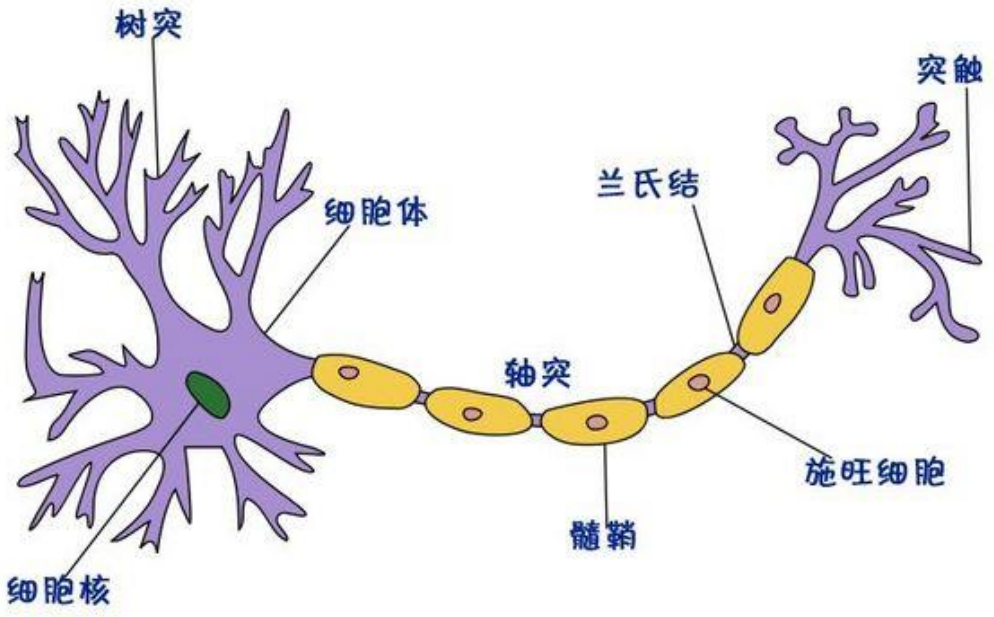
第一阶段

- 1943年, McCulloch和Pitts 提出第一个神经元数学模型, 即M-P模型, 并从原理上证明了人工神经网络能够计算任何算数和逻辑函数
- 1949年, Hebb 发表《The Organization of Behavior》一书, 提出生物神经元学习的机理, 即Hebb学习规则
- 1958年, Rosenblatt 提出感知机网络（Perceptron）模型和其学习规则
- 1960年, Widrow和Hoff提出自适应线性神经元（Adaline）模型和最小均方学习算法
- 1969年, Minsky和Papert 发表《Perceptrons》一书, 指出单层神经网路不能解决非线性问题, 多层网络的训练算法尚无希望. 这个论断导致神经网络进入低谷

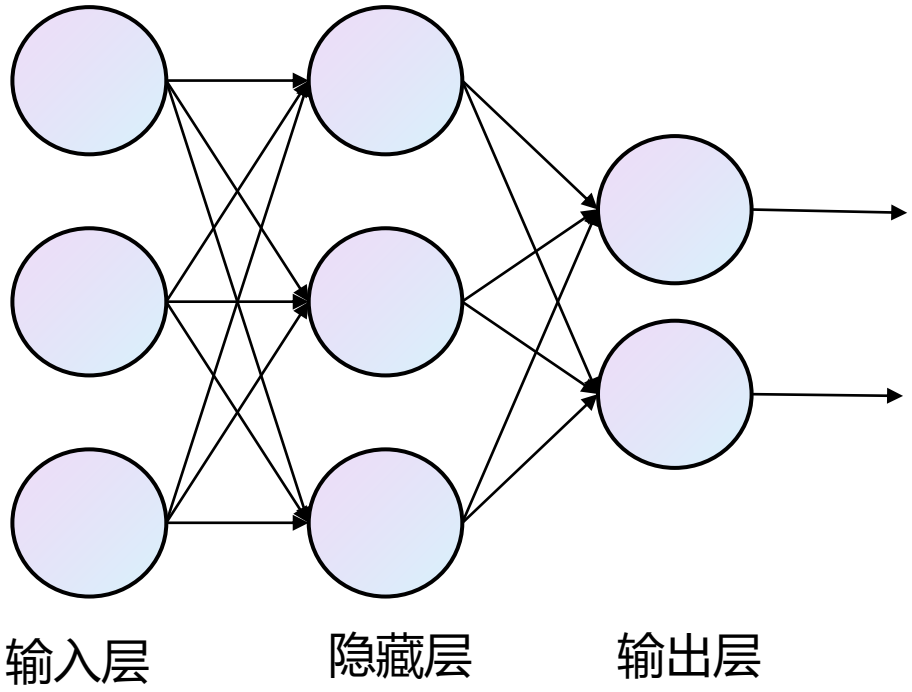
1.人工神经网络发展历史

发展历史

1943年，心理学家McCulloch和逻辑学家Pitts建立神经网络的数学模型，MP模型



神经元生理结构

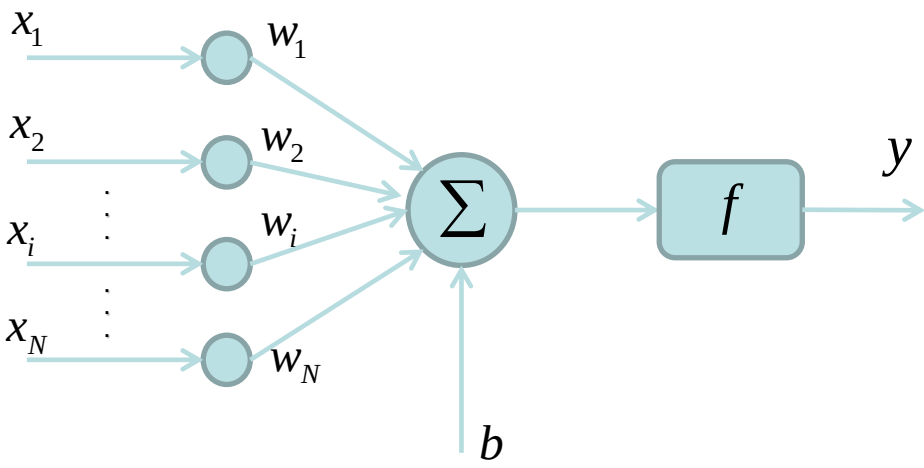


神经元数学模型

1.人工神经网络发展历史

1960年代，人工网络得到了进一步地发展感知机和自适应线性元件等被提出。

M.Minsky仔细分析了以感知机为代表的神经网络的局限性，指出了感知机不能解决非线性问题，这极大影响了神经网络的研究。



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

单层感知机的数学模型

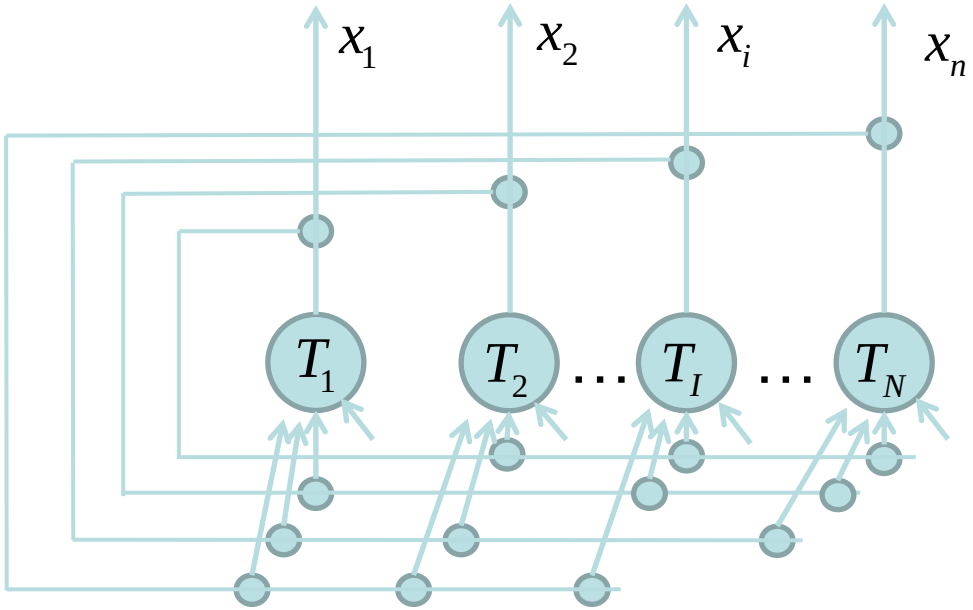
1.人工神经网络发展历史

第二阶段

- 1982年, 物理学家Hopfield提出了一种具有联想记忆、优化计算能力的递归网络模型, 即Hopfield 网络
- 1986年, Rumelhart 等编辑的著作《Parallel Distributed Proceesing:Explorations in the Microstructures of Cognition》报告了反向传播算法
- 1987年, IEEE 在美国加州圣地亚哥召开第一届神经网络国际会议（ICNN）
- 90年代初, 伴随统计学习理论和SVM的兴起, 神经网络由于理论不够清楚, 试错性强, 难以训练, 再次进入低谷

1.人工神经网络发展历史

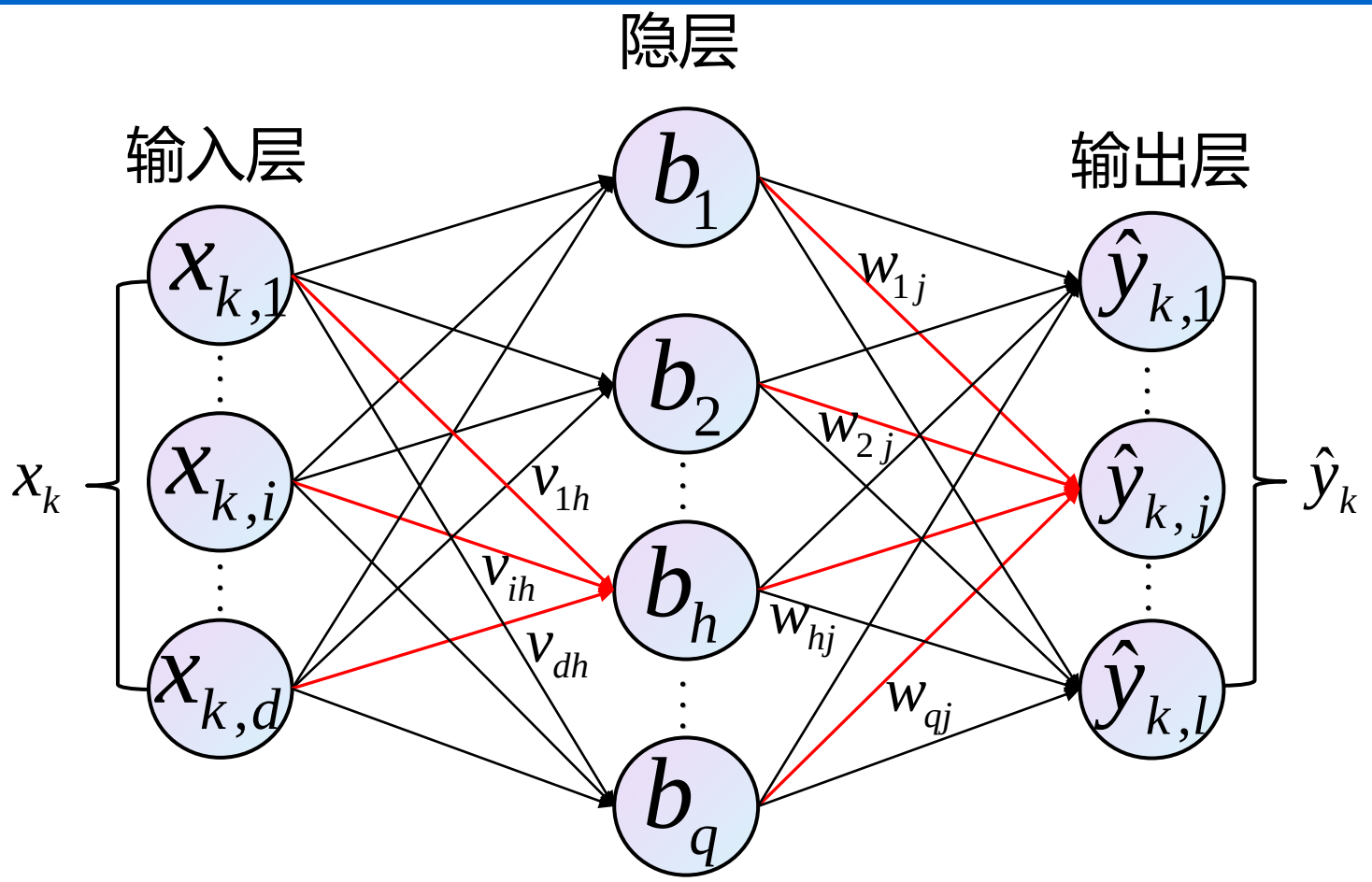
1982年，加州理工学院J.J.Hopfield教授提出了Hopfield神经网络模型，引入了计算能量概念，给出了网络稳定性判断。



离散Hopfield神经网络模型

1.人工神经网络发展历史

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。

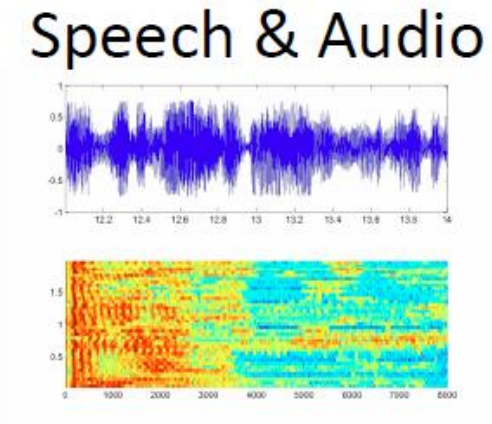


BP神经网络模型

1.人工神经网络发展历史

第三阶段

- 2006年, Hinton提出了深度信念网络(DBN), 通过“预训练+微调”使得深度模型的最优化变得相对容易
- 2012年, Hinton 组参加ImageNet 竞赛, 使用 CNN 模型以超过第二名10个百分点的成绩夺得当年竞赛的冠军
- 伴随云计算、大数据时代的到来，计算能力的大幅提升，使得深度学习模型在计算机视觉、自然语言处理、语音识别等众多领域都取得了较大的成功



1.人工神经网络发展历史

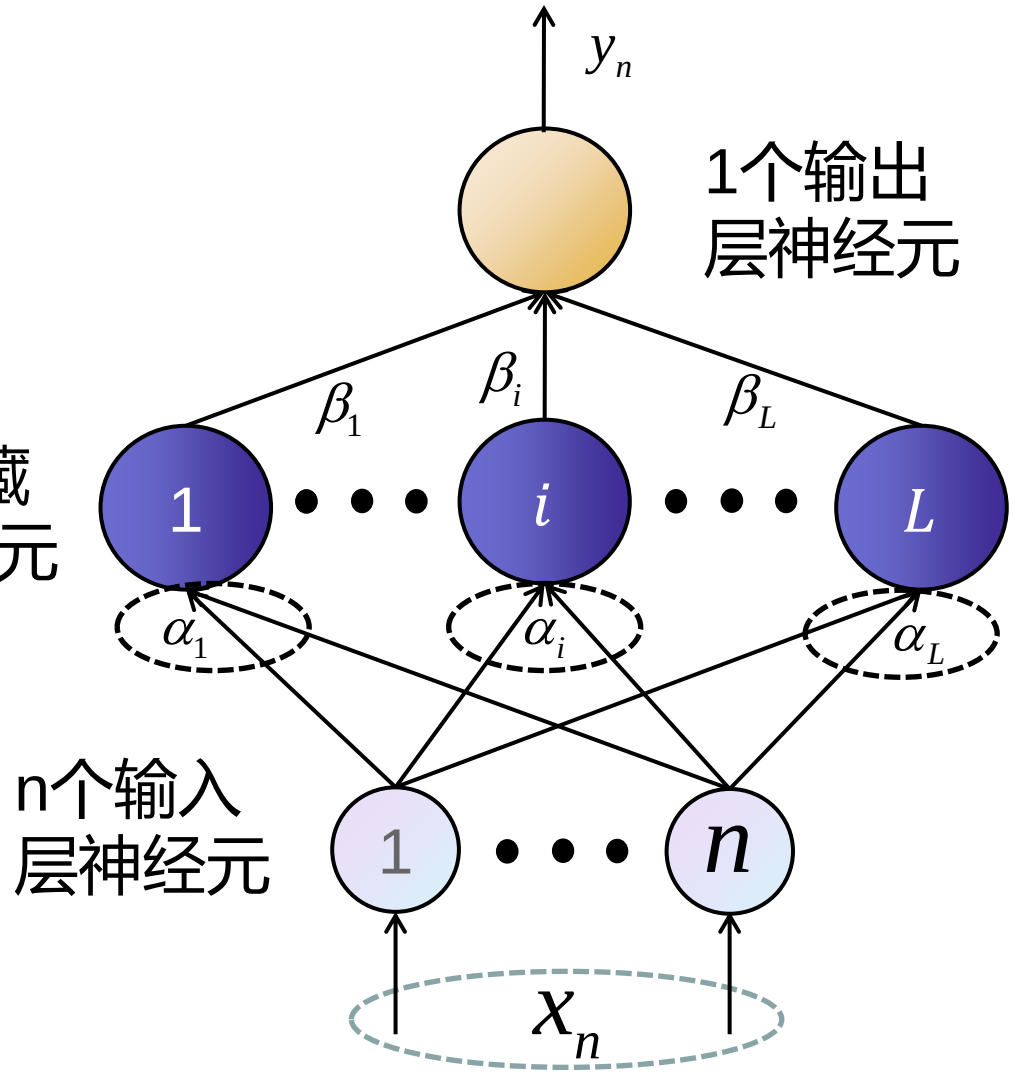
极限学习机(Extreme Learning Machine, ELM)，是由黄广斌提出的用于处理单隐层神经网络的算法

随机初始化输入权重α_i和偏置，只求解输出权重值β_i。

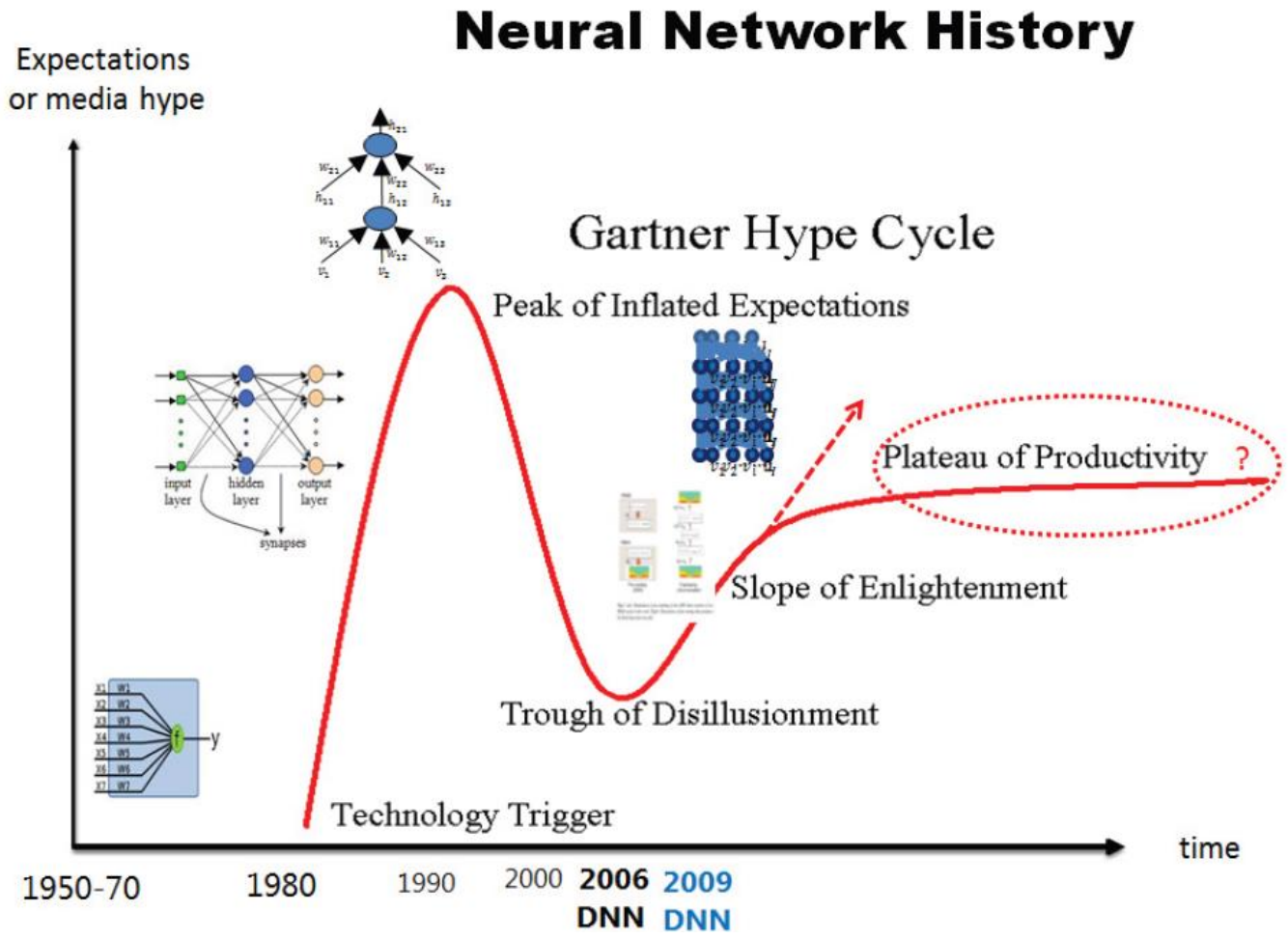
优点：

1.学习精度有保证

2.学习速度快



1.人工神经网络发展历史



2.感知器算法

01 发展历史

02 感知机算法

03 BP算法

2.感知机算法

M-P 神经元模型 [McCulloch and Pitts, 1943]

- 输入：来自其他 n 个神经云传递过来的输入信号
- 处理：输入信号通过带权重的连接进行传递，神经元接受到总输入值将与神经元的阈值进行比较
- 输出：通过激活函数的处理以得到输出

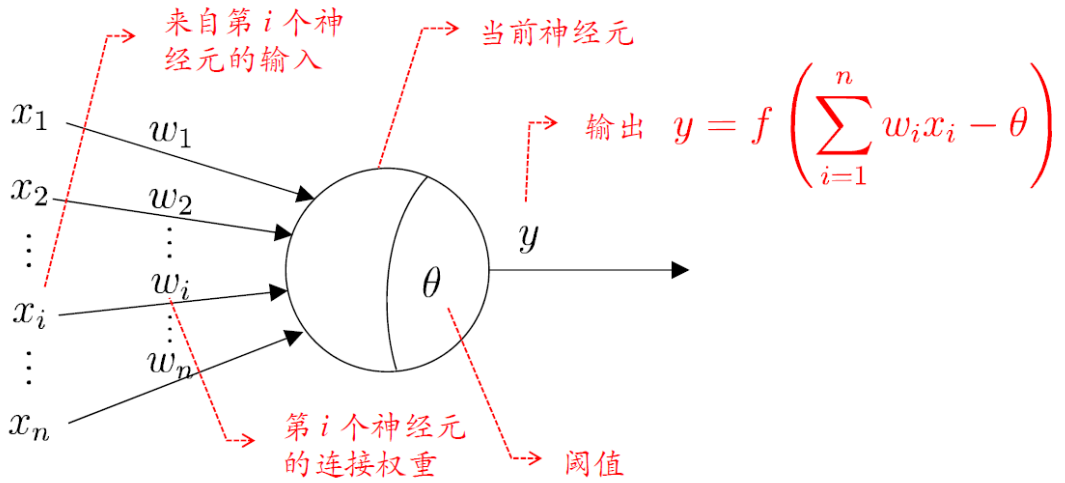
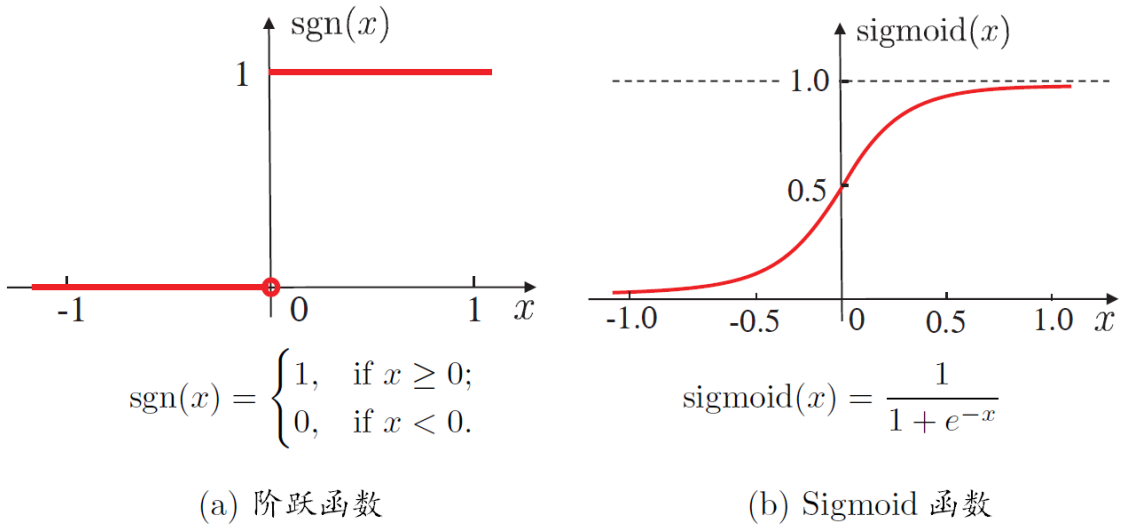


图 5.1　M-P 神经元模型

2.感知机算法

激活函数



(a) 阶跃函数　　(b) Sigmoid 函数

图 5.2　典型的神经元激活函数

- 理想激活函数是阶跃函数, 0表示抑制神经元而1表示激活神经元
- 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

2.感知机算法

感知机（Perceptron）感知机由两层神经元组成，输入层接受外界输入信号传递给输出层，输出层是M-P神经元（阈值逻辑单元），是二分类问题的线性分类模型。

用 $X \in R^{n \times d}$ 表示数据集，用 Y 表示标签。

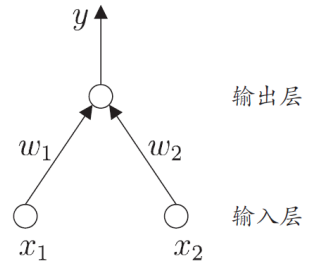


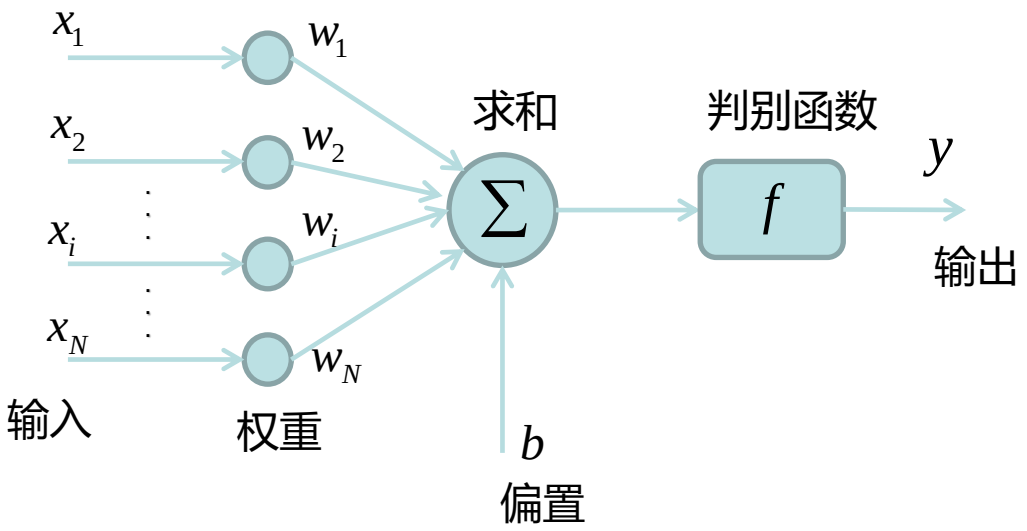
图 5.3　两个输入神经元的感知机网络结构示意图

需要学习的目标函数是

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

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

从一堆输入输出中学习模型参数w和b。



2.感知机算法

感知机算法（Perceptron Algorithm）：

随机选择模型参数的(w_0, b_0)初始值。

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

选择一个训练样本(x_n, y_n)。

若判别函数$w^T x_n + b > 0$，且$y_n = -1$，则$w = w - x_n$，$b = b - 1$。

若判别函数$w^T x_n + b < 0$，且$y_n = +1$，则$w = w + x_n$，$b = b + 1$。

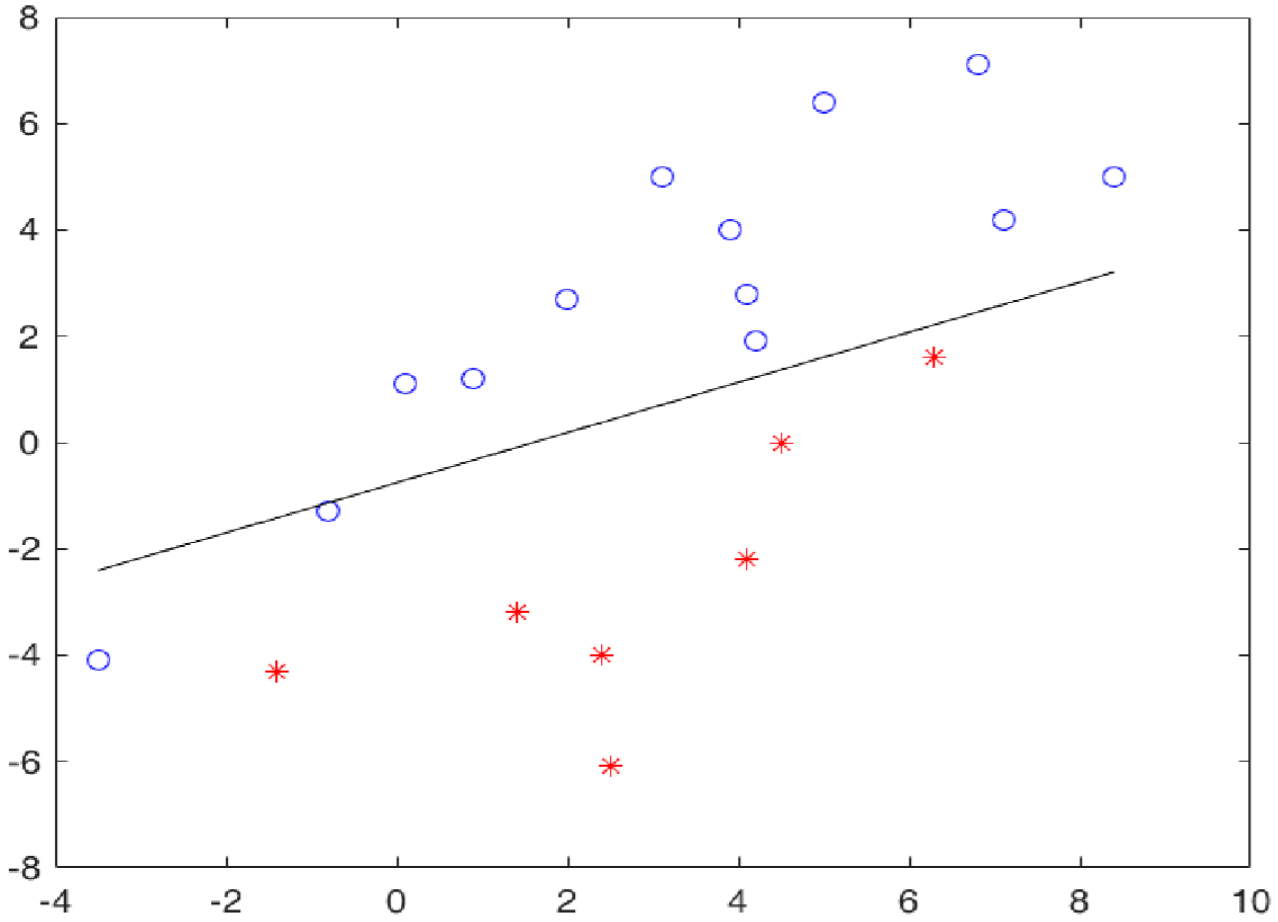
再选取另一个训练样本(x_m, y_m)，回到2。

终止条件：直到所有数据的输入输出对都不满足2中的(i)和(ii)中之一，则退出循环。

2.感知机算法

算法演示　分类问题

单层感知机只拥有一层功能神经元(functional neuron)，其学习能力有限，只能处理线性问题，无法处理非线性问题！！



2.感知器算法

01 发展历史

02 感知机算法

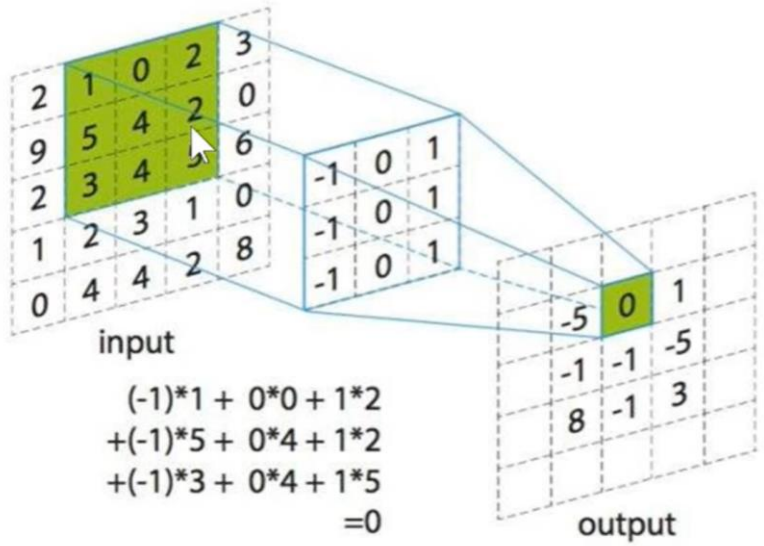
03 BP算法

3.BP算法

反向传播（Back Propagation, BP）算法是 "误差反向传播" 的简称，也称为backprop，允许来自代价函数的信息通过网络向后流动，以便计算梯度。

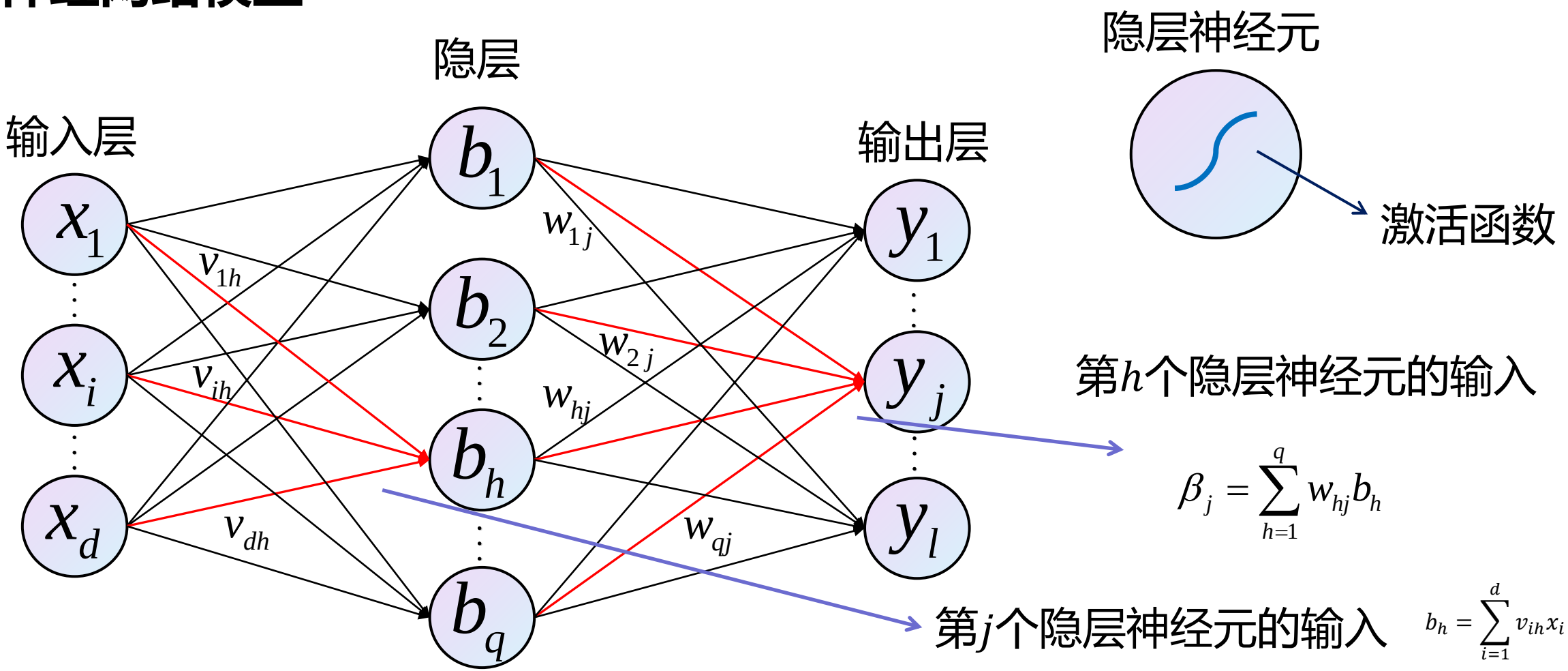
反向传播是一种与最优化方法（如梯度下降法）结合使用的，用来训练人工神经网络的常见方法。该方法对网络中所有权重计算损失函数的梯度。这个梯度会反馈给最优化方法，用来更新权值以最小化损失函数。

BP算法是迄今最成功的神经网络学习算法,现实任务中使用神经网络时,大多是在使用BP算法进行训练,包括最近炙手可热的深度学习概念下的卷积神经网络(CNNs)。



3.BP算法

神经网络模型

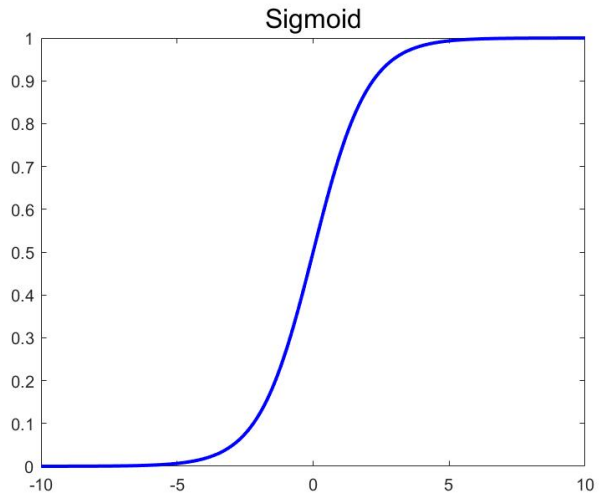


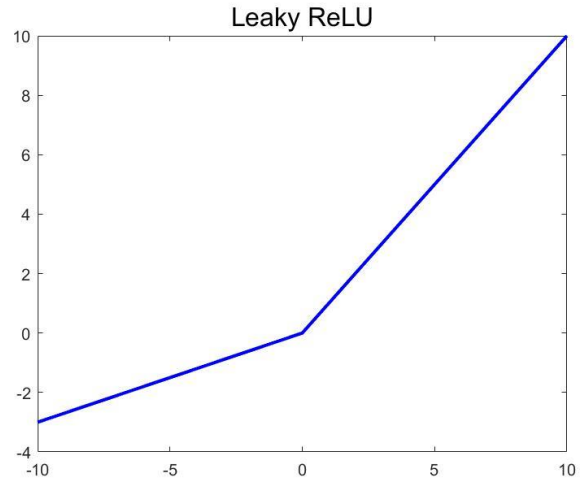
3.BP算法

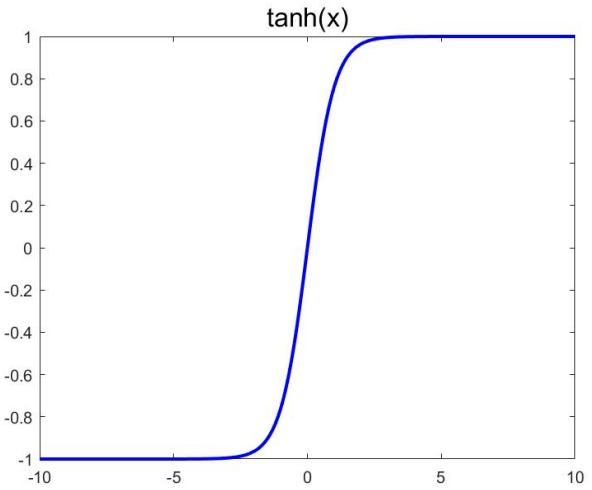
激活函数

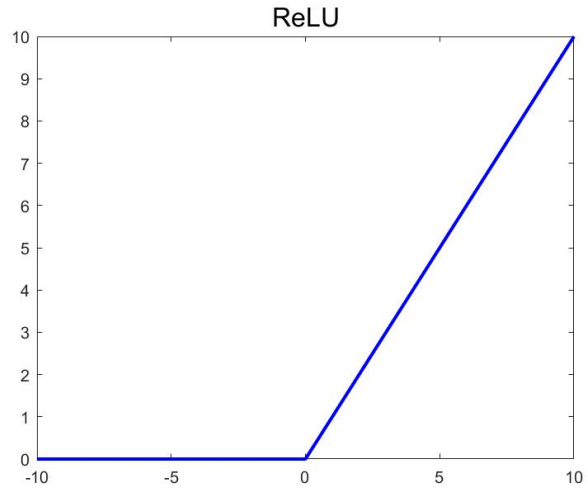
常见激活函数选择：

sigmoid 函数
tanh 函数
ReLU 函数
Leaky ReLU函数









3.BP算法

最常用Sigmoid函数的优缺点：

优点：
1.函数处处连续，便于求导
2.可将函数值的范围压缩至[0,1]，可用于压缩数据，且幅度不变
3.便于前向传输

缺点：
1.在趋向无穷的地方，函数值变化很小，容易出现梯度消失，不利于深层神经的反馈传输
2.幂函数的梯度计算复杂
3.收敛速度比较慢

3.BP算法

主要步骤

第一步，对样本明确预测输出值与损失函数

第二步，明确参数调整策略

第三步，计算输出层阈值的梯度

第四步，计算隐层到输出层连接权值的梯度

第五步，计算隐层阈值的梯度

第六步，计算输入层到隐层连接权值的梯度

第七步，引出归纳结论

3.BP算法

误差逆传播算法（Error BackPropagation, 简称BP）是最成功的训练多层前馈神经网络的学习算法.

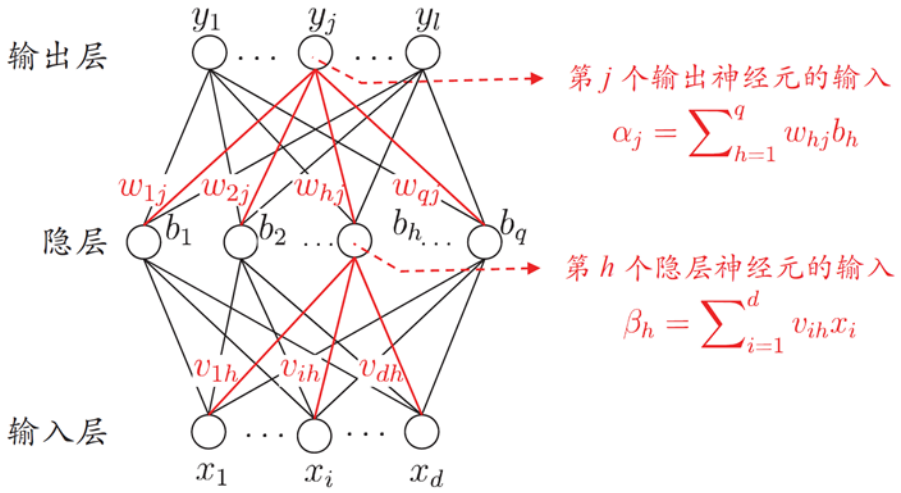
- 给定训练$D=\{(\boldsymbol{x}_i,y_i)\},\boldsymbol{x}_i\in R^d,y_i\in R^l,(i=1,2,\cdots,m)$, 即输入示例由$d$个属性描述, 输出 l 维实值向量.
- 为方便讨论, 给定一个拥有 d个输入神经元, l个输出神经元,q个隐层神经元的多层前向前馈网络结构.
- 记号：

θ_j： 输出层第 j 个神经元阈值；

γ_h ：隐含层第 h 个神经元阈值；

v_{ih}： 输入层与隐层神经元之间的连接权重；

w_{hj}： 隐层与输出层神经元之间的连接权重；



3.BP算法

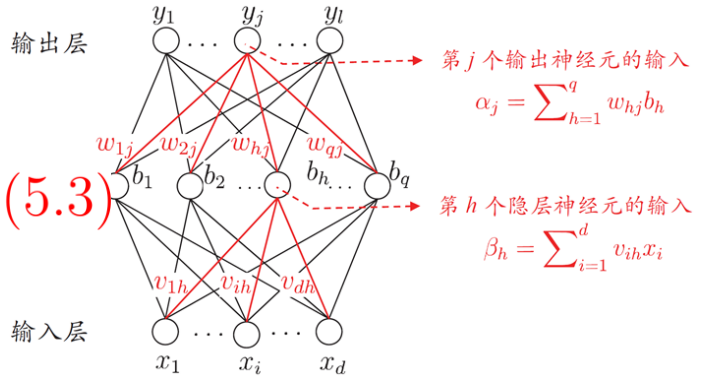
对于样例 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$，假设网络的实际输出为 $\hat{\boldsymbol{y}}_k$

- 前向计算

step1: $b_h = f(\beta_h - \gamma_h), \beta_h = \sum_{i=1}^{d} v_{ih} x_i$

step2: $\hat{y}_j^k = f(\alpha_j - \theta_j), \alpha_h = \sum_{i=1}^{q} w_{hj} b_h$

step3: $E_k = \frac{1}{2} \sum_{j=1}^{l} (\hat{y}_j^k - y_j^k)^2$ (5.3)



- 参数数目

这里选择$\frac{1}{2}$是为了后续求导方便

权重：v_{ih}，w_{hj} 阈值：θ_j，γ_h $(i = 1, \cdots, d, h = 1, , \cdots, q, j = 1, \cdots, l)$

因此网络中需要 $(d + l + 1)q + l$ 个参数需要优化

- 参数优化

BP是一个迭代学习算法，在迭代的每一轮中采用广义的感知机学习规则对参数进行更新估计，任意的参数v的更新估计式为

$$v \leftarrow v + \triangle v.$$

3.BP算法

BP 学习算法

- BP算法基于梯度下降策略, 以目标的负梯度方向对参数进行调整. 对误差 E_k ， 给定学习率 η

$$\triangle w_{hj} = -\eta \frac{\partial E_k}{\partial w_{jk}}$$

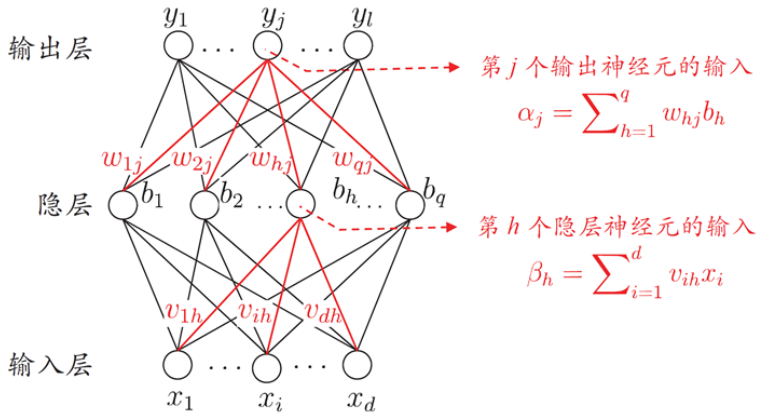
$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial_{hj}}$$

$$g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$$

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

$$= \hat{y}_j^k (1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k) \quad (5.10)$$

$$\triangle w_{hj} = \eta b_j g_h \quad (5.11)$$



3.BP算法

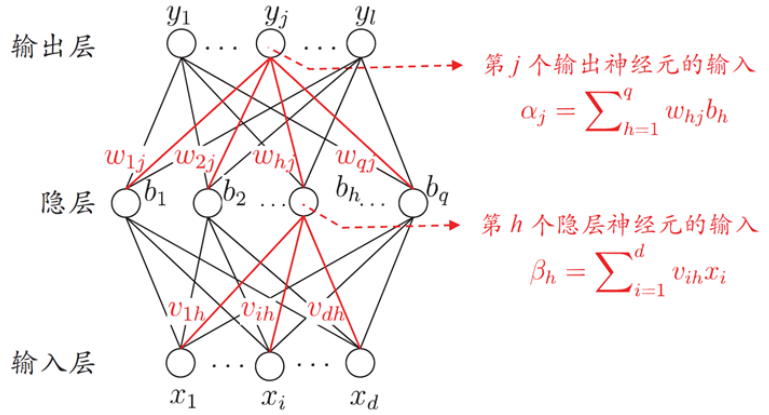
类似的可以推导出：

$$\triangle\theta_j = -\eta g_j, \quad (5.12)$$

$$\triangle v_{ih} = \eta e_h x_i, \quad (5.13)$$

$$\triangle\gamma_h = -\eta e_h, \quad (5.14)$$

其中



$$\begin{aligned} e_h &= -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \\ &= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} \partial b_h f'(\alpha_h - \gamma_h) \\ &= \sum_{j=1}^{h} w_{hj} g_j f'(\alpha_h - \gamma_h) \\ &= b_h(1-b_h) \sum_{j=1}^{h} w_{hj} g_j. \quad (5.15) \end{aligned}$$

- 学习率 $\eta \in (0,1)$ 控制着算法每一轮迭代中的更新步长, 若太长则让容易震荡, 太小则收敛速度又会过慢.

3.BP算法

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

过程：

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

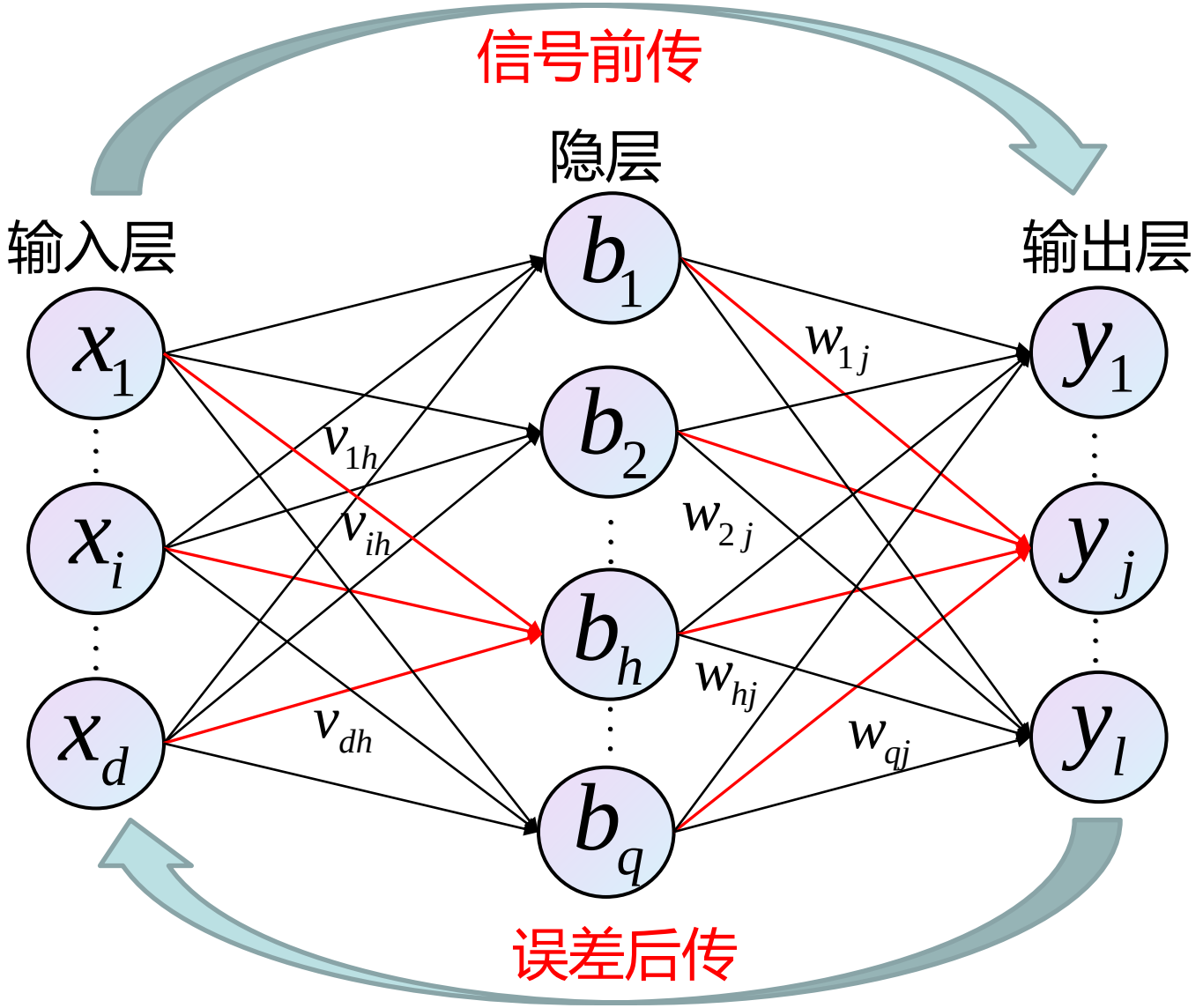
输出： 连接权与阈值确定的多层前馈神经网络

图 5.8 误差逆传播算法

3.BP算法

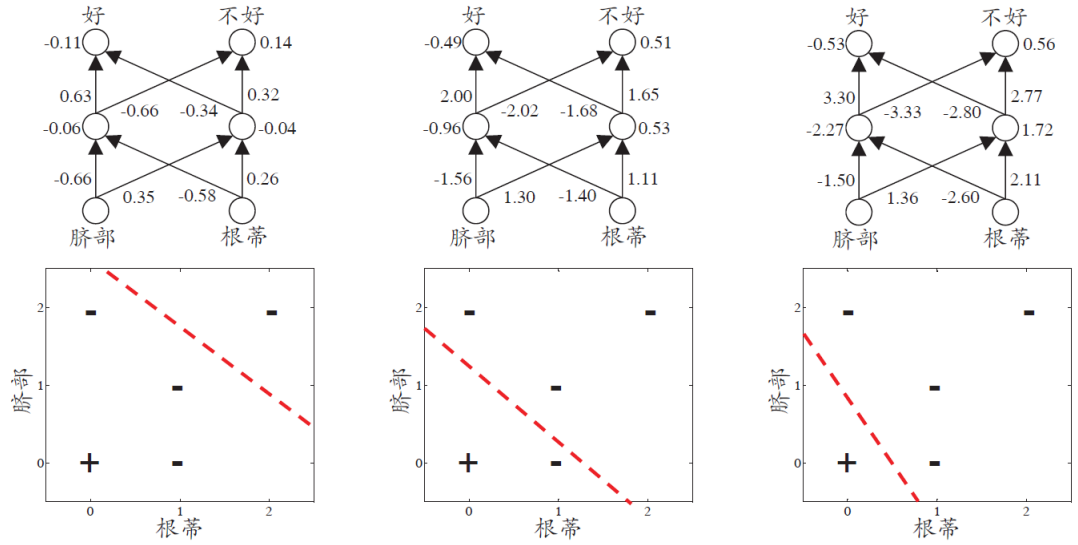
算法流程回顾：

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



3.BP算法

BP 算法实验



(a) 第 25 轮　(b) 第 50 轮　(c) 第 100 轮

图 5.9　在 2 个属性、5 个样本的西瓜数据上, BP网络参数更新和分类边界的变化情况

3.BP算法

优点：

1.能够自适应、自主学习。BP可以根据预设参数更新规则，通过不断调整神经网络中的参数，已达到最符合期望的输出。

2.拥有很强的非线性映射能力。

3.误差的反向传播采用的是成熟的链式法则，推导过程严谨且科学。

4.算法泛化能力很强。

缺点：

1.BP神经网络参数众多，每次迭代需要更新较多数量的阈值和权值，故收敛速度比较慢。

2.网络中隐层含有的节点数目没有明确的准则，需要不断设置节点数字试凑，根据网络误差结果最终确定隐层节点个数

3.BP算法是一种速度较快的梯度下降算法，容易陷入局部极小值的问题。

3.BP算法

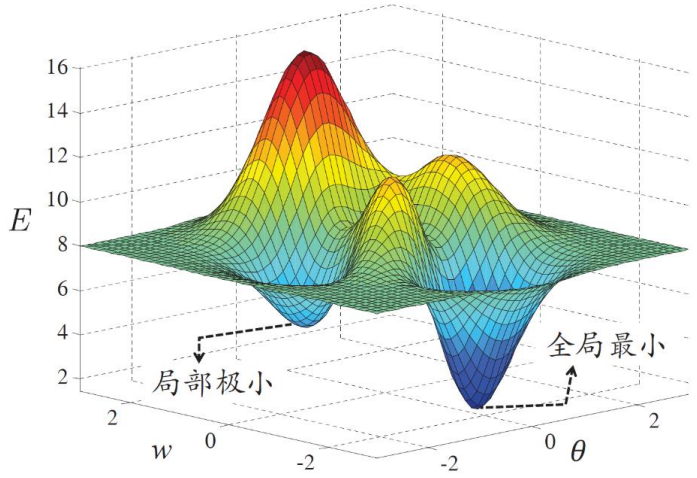
全局最小与局部极小

- 对 $\boldsymbol{w}^*$ 和 θ^*, 若存在 $\epsilon > 0$ 使得

$$\forall(\boldsymbol{w};\theta) \in \{\|(\boldsymbol{w};\theta) - (\boldsymbol{w}^*;\theta^*)\| \leq \epsilon\},$$

都有 $E(\boldsymbol{w};\theta) \geq E(\boldsymbol{w}^*;\theta^*)$ 成立, 则 $(\boldsymbol{w}^*;\theta^*)$ 为局部极小解; 若度参数空间中任意的 $(\boldsymbol{w};\theta)$, 都有 $E(\boldsymbol{w};\theta) \geq E(\boldsymbol{w}^*;\theta^*)$, 则 $(\boldsymbol{w}^*;\theta^*)$ 为全局最小解. 两者对应的 $E(\boldsymbol{w}^*;\theta^*)$ 分别称为误差函数的局部最小解和全局最小值.

- 显然参数空间梯度为零的点, 只要气误差函数值小于邻点的误差函数值, 就是局部极小点
- 可能存在多个局部极小值, 但却只会有一个全局极最小值



参考文献

1. 《统计学习方法》，清华大学出版社，李航著，2019年出版
2. 《机器学习》，清华大学出版社，周志华著，2016年出版
3. Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag, 2006

谢谢观赏 下节课见