第四章 神经网络基础

§ 4.1 神经元模型

§ 4.2 多层感知机

§ 4.3 反向传播算法

§4.1 神经元模型

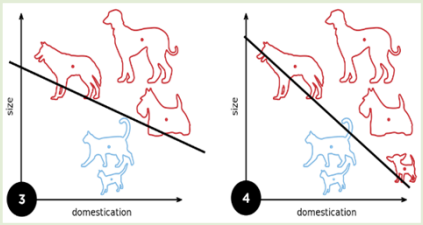
一、发展历史

二、概念定义

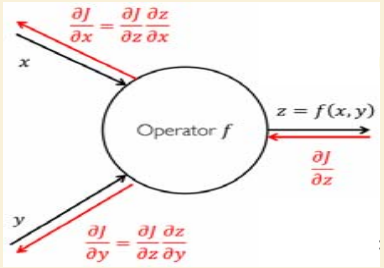
三、M-P神经元模型

神经网络发展历史

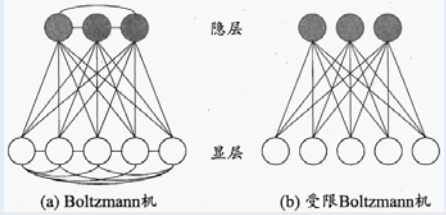
- 神经网络的发展历史伴随着人工智能的发展历史三起三落



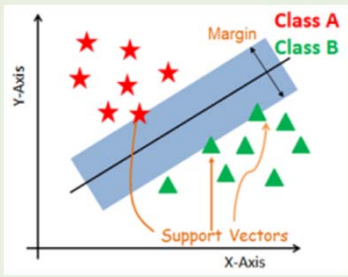
Rosenblatt提出感知机



BP算法



受限玻尔兹曼机



SVM流行

1958 1969 1974 1970s 1986 1989 1995 2012

感知机不能处理异或问题

人工智能寒冬

Yann LeCun将CNN应用于手写数字识别

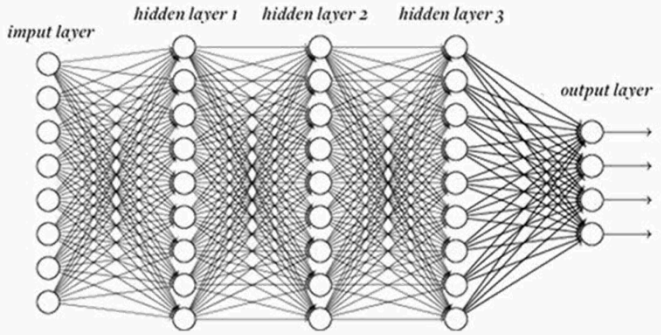
AlexNet在ImageNet竞赛中获胜

概念定义

神经网络的定义

- **定义：**神经网络是由**具有适应性简单单元**组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体和环境所做出的交互反映
- 由Kohonen于1988年在Neural Networks创刊号上给出
- 本课程中提及的“人工神经网络”并非生物学意义的神经网络

生物神经网络



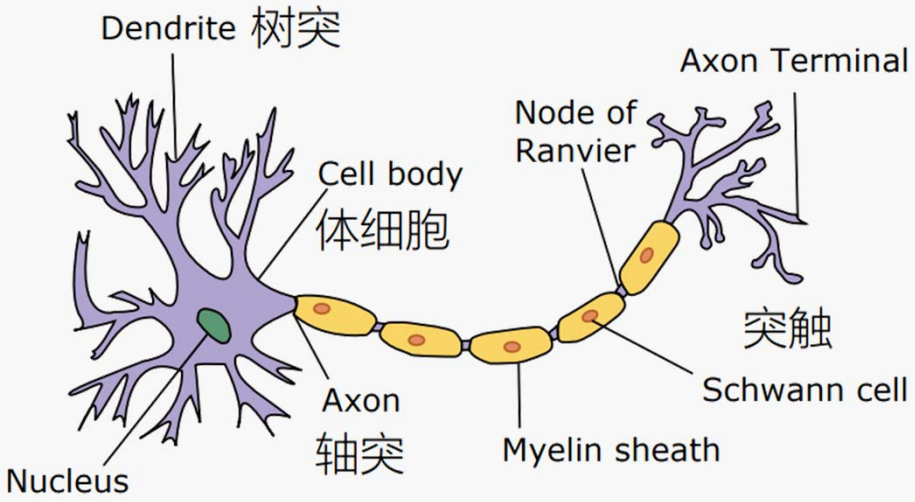
人工神经网络

如何建模神经网络？从基本单元入手！

概念定义

神经元的定义

- **定义：**神经元是组成神经网络的“简单单元”
- **生物学联系：**生物神经网络中，每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过一个“阈值”，它就会被激活，即“兴奋”起来，向其他神经元发送化学物质

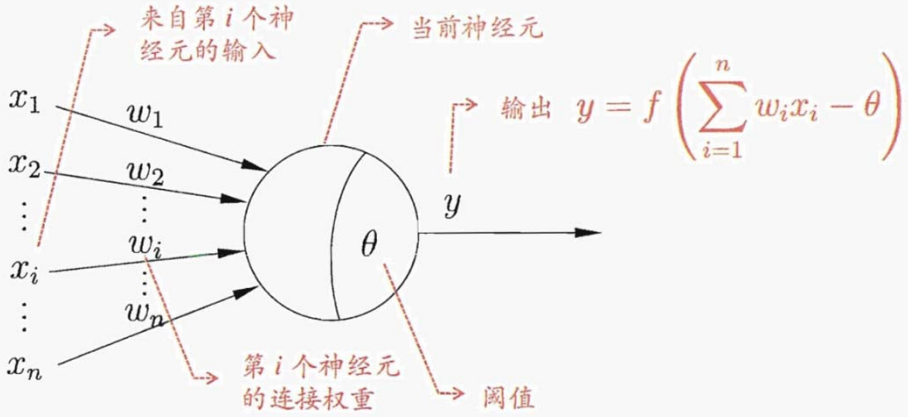


如何建立神经元的数学模型？
M-P神经元模型！

M-P神经元模型

M-P神经元模型

- **提出：**1943年，神经生理学家McCulloch和数学家Pitts将生物神经元抽象为简单数学模型，即沿用至今的“M-P”神经元模型
- **机制：**神经元接收来自其它n个神经元传递过来的**输入信号**，这些输入信号通过**带权重的连接**进行传递，神经元接收到的总输入值将与**神经元的阈值**进行比较，通过**“激活函数”**处理以产生神经元的输出
- 将许多个这样的神经元按一定的层次结构连接起来，就得到了神经网络

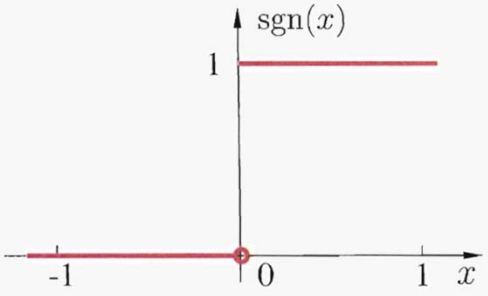


激活函数如何选择？
阶跃函数！

M-P神经元模型

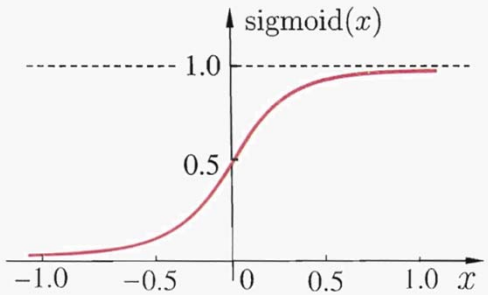
激活函数

- **阶跃函数**：理想的激活函数，将输入值映射为输出值为“0”或“1”，“1”对应于神经元兴奋，“0”对应于神经元抑制，但阶跃函数具有不连续、不光滑等不太好的数学性质
- **Sigmoid函数**：实际常用的激活函数，将可能在较大范围内变化的输入值挤压到（0,1）输出值范围内，因此有时也称为“挤压函数”



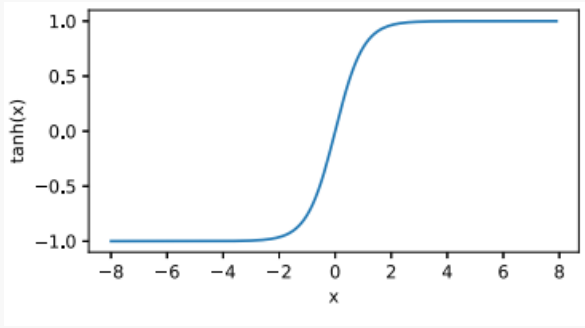
$$\mathrm{sgn}(x)=\begin{cases}1, & x\geqslant 0;\\ 0, & x<0.\end{cases}$$

阶跃函数

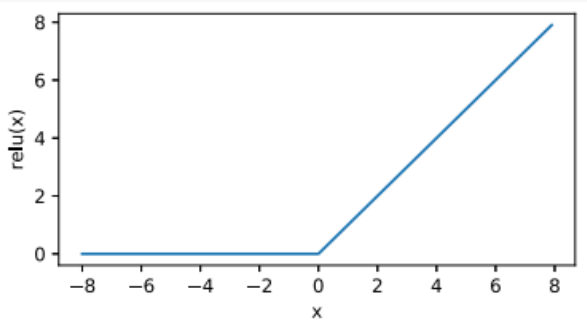


$$\mathrm{sigmoid}(x)=\frac{1}{1+e^{-x}}$$

Sigmoid函数



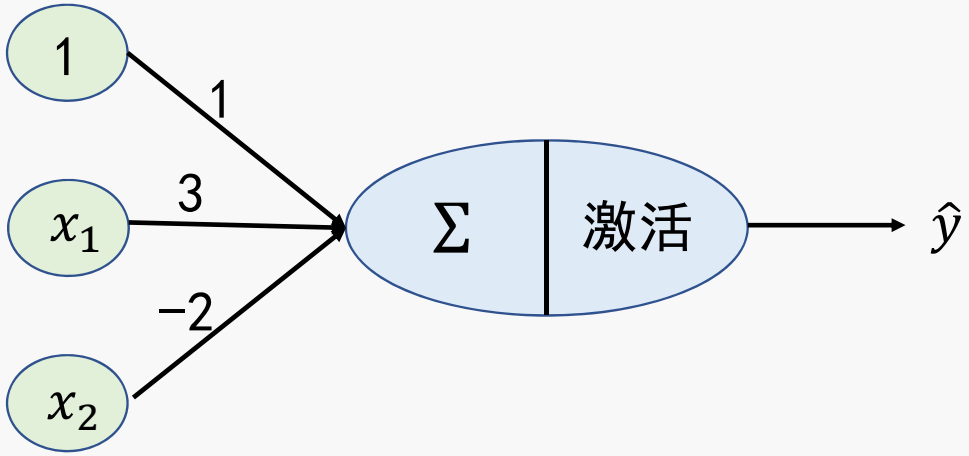
$$\mathrm{ReLU}(z)=\max(0,z)$$



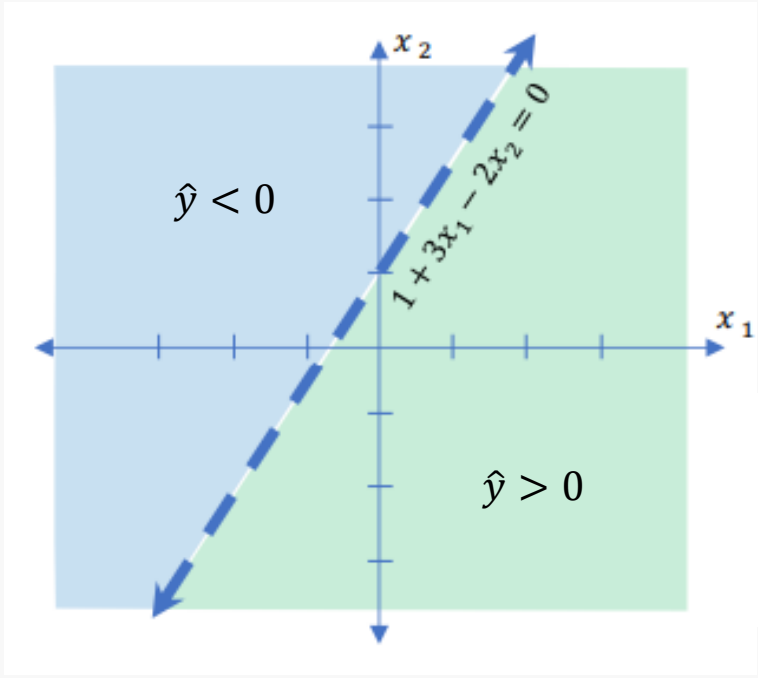
$$\tanh(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

M-P神经元模型

计算举例



$$\hat{y} = \mathrm{sgn}(1 + 3x_1 - 2x_2)$$



M-P神经元模型

- 其他神经元模型
  - M-P神经元模型使用最广泛，但还有一些其他神经元模型也受到关注，例如考虑电位脉冲发放时间而不是电位积累的**脉冲神经元模型**
- 生物视角
  - 从计算机科学的角度看，不考虑神经网络是否真的模拟了生物神经网络，只需将神经网络视为**包含了许多参数的数学模型**
  - 有效的神经网络学习算法大多以数学证明为支撑
- 思考题
  - 10个神经元两两连接，有（ ）个连接权重？
  - 答案：有向完全图，90个权重

§4.2 多层感知机

一、感知机模型

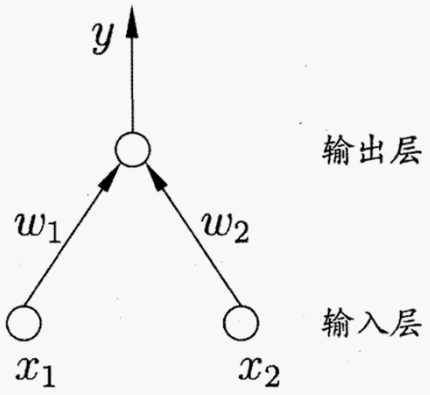
二、模型参数调整

三、多层前馈网络

感知机模型

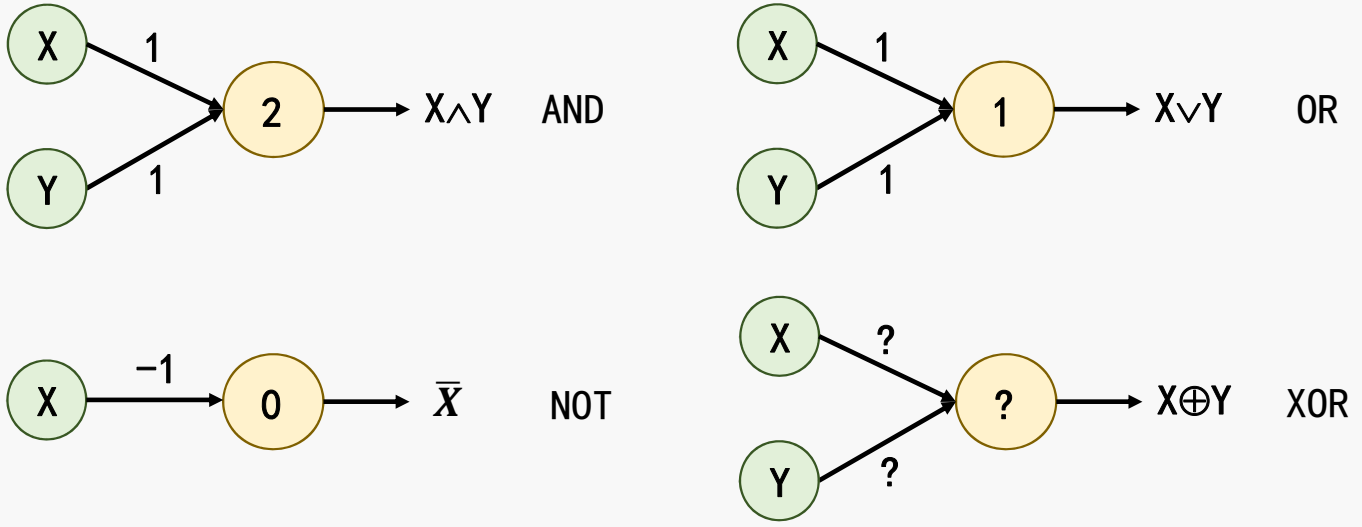
感知机模型

- **定义：**感知机由**两层神经元**组成，输入层接收外界输入信号后传递给输出层，输出层是M-P神经元
- **功能：**感知机能容易地实现逻辑与、或、非运算



两个输入神经元的
感知机示意图

使用阶跃激活函数：



模型参数调整

模型参数调整

- 给定训练数据集，权重$w_i(i=1,2,\cdots,n)$以及阈值θ可通过学习得到
- 阈值θ可看作一个固定输入为-1.0的“哑结点”所对应的连接权重w_{n+1}
- **调整规则：**对训练样例(x,y)，若当前感知机的输出为$\hat{y}$，则感知机权重调整如下：

$$w_i \leftarrow w_i + \Delta w_i$$
$$\Delta w_i = \eta(y-\hat{y})x_i$$

其中$\eta \in (0,1)$为学习率，x_i是x对应于第i个输入神经元的分量

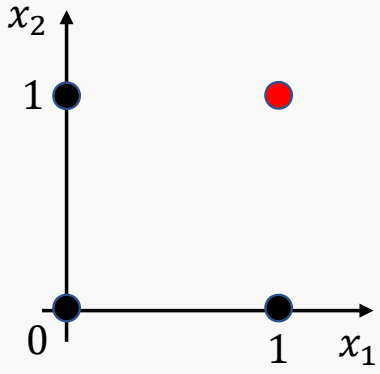
- **调整分析：**若感知机对训练样例预测正确，即$\hat{y}=y$，则感知机不发生变化，否则将根据错误的程度进行权重调整

模型参数调整

- 用感知机解决AND问题
  - 数据集：

$$\{((x_1, x_2), y)\} = \{((0,0),0),((1,0),0),((0,1),0),((1,1),1)\}$$

注：将(x_1, x_2)增广为$(x_1, x_2, -1)$以考虑阈值



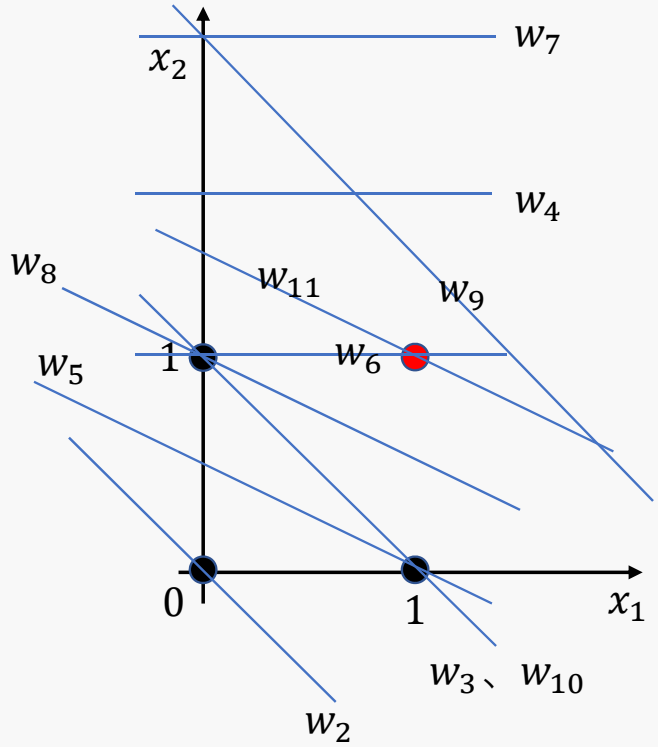
  - 参数：初始权重$w_0 = (0,0,0)$，学习率$\eta = 1$，使用阶跃激活函数
  - 1、考虑$((0,0),0)$，$\hat{y} = 1 \neq y$，更新权重为$w_1 = w_0 - (0,0,-1) = (0,0,1)$
  - 2、考虑$((1,0),0)$，$\hat{y} = y = 0$，不更新权重
  - 3、考虑$((0,1),0)$，$\hat{y} = y = 0$，不更新权重
  - 4、考虑$((1,1),1)$，$\hat{y} = 0 \neq y$，更新权重为$w_2 = w_1 + (1,1,-1) = (1,1,0)$
  - 5、考虑$((0,0),0)$，$\hat{y} = 1 \neq y$，更新权重为$w_3 = w_2 - (0,0,-1) = (1,1,1)$
  - 6、考虑$((1,0),0)$，$\hat{y} = 1 \neq y$，更新权重为$w_4 = w_3 - (1,0,-1) = (0,1,2)$
  - 7、考虑$((0,1),0)$，$\hat{y} = y = 0$，不更新权重
  - 8、考虑$((1,1),1)$，$\hat{y} = 0 \neq y$，更新权重为$w_5 = w_4 + (1,1,-1) = (1,2,1)$

模型参数调整

- 举例：用感知机解决AND问题
  - 9、考虑$((0,0),0)$，$\hat{y}=y=0$，不更新权重
  - 10、考虑$((1,0),0)$，$\hat{y}=1\neq y$，更新权重为$w_6=w_5-(1,0,-1)=(0,2,2)$
  - 11、考虑$((0,1),0)$，$\hat{y}=1\neq y$，更新权重为$w_7=w_6-(0,1,-1)=(0,1,3)$
  - 12、考虑$((1,1),1)$，$\hat{y}=0\neq y$，更新权重为$w_8=w_7+(1,1,-1)=(1,2,2)$
  - 13、考虑$((0,0),0)$，$\hat{y}=y=0$，不更新权重
  - 14、考虑$((1,0),0)$，$\hat{y}=y=0$，不更新权重
  - 15、考虑$((0,1),0)$，$\hat{y}=1\neq y$，更新权重为$w_9=w_8-(0,1,-1)=(1,1,3)$
  - 16、考虑$((1,1),1)$，$\hat{y}=0\neq y$，更新权重为$w_{10}=w_9+(1,1,-1)=(2,2,2)$
  - 17、考虑$((0,0),0)$，$\hat{y}=y=0$，不更新权重
  - 18、考虑$((1,0),0)$，$\hat{y}=1\neq y$，更新权重为$w_{11}=w_{10}-(1,0,-1)=(1,2,3)$
  - 可验证$\hat{y}=\mathrm{sgn}(w_{11}\cdot x)$能将4个样本全部正确分类

模型参数调整

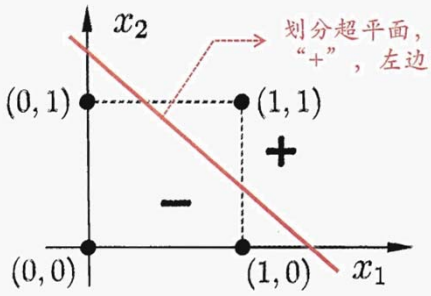
- 更新过程动画演示



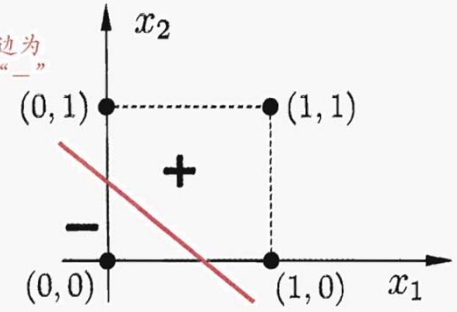
多层前馈网络

感知机的局限性

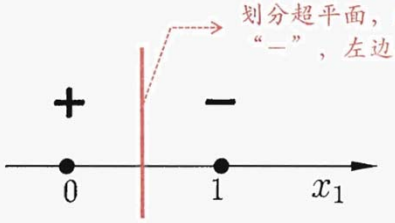
- **结构限制**：感知机只有输出层神经元进行激活函数处理，即只有一层功能神经元，其学习能力非常有限
- **线性可分**：对两类模式，若存在一个**线性超平面**将它们分开，则我们称其为线性可分的。**与、或、非问题**都是线性可分的问题，**异或问题**线性不可分
- 可以证明：若两类模式线性可分，则感知机的学习过程一定会收敛而求得权向量；否则感知机学习过程将会发生振荡



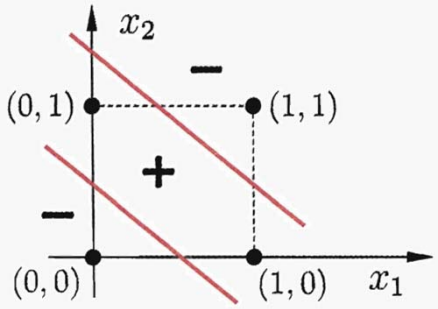
“与”运算



“或”运算



“非”运算

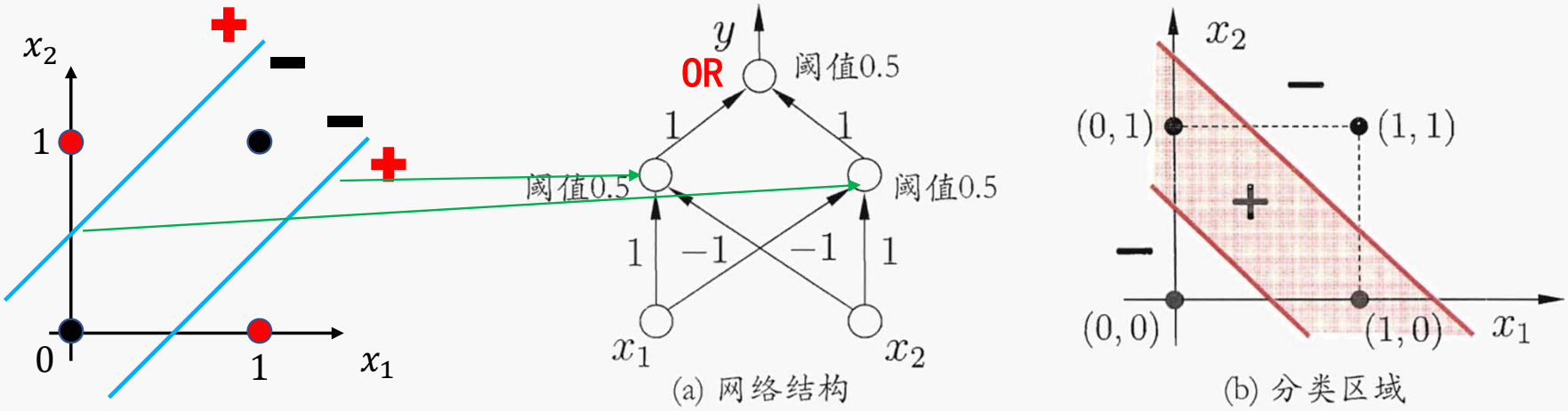


“异或”运算

多层前馈网络

多层前馈网络的提出

- **建模非线性可分问题：**要解决非线性可分问题，需要考虑使用多层功能神经元构成神经网络。例如可以设计两层神经网络解决“异或”问题
- **隐含层：**输出层与输入层之间的一层神经元，被称为隐层或隐含层，隐含层和输出层神经元都是拥有激活函数的功能神经元

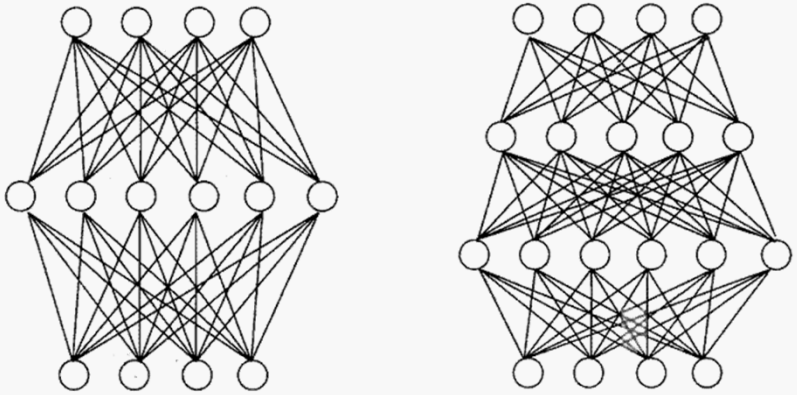


能解决异或问题的两层感知机

多层前馈网络

多层前馈网络

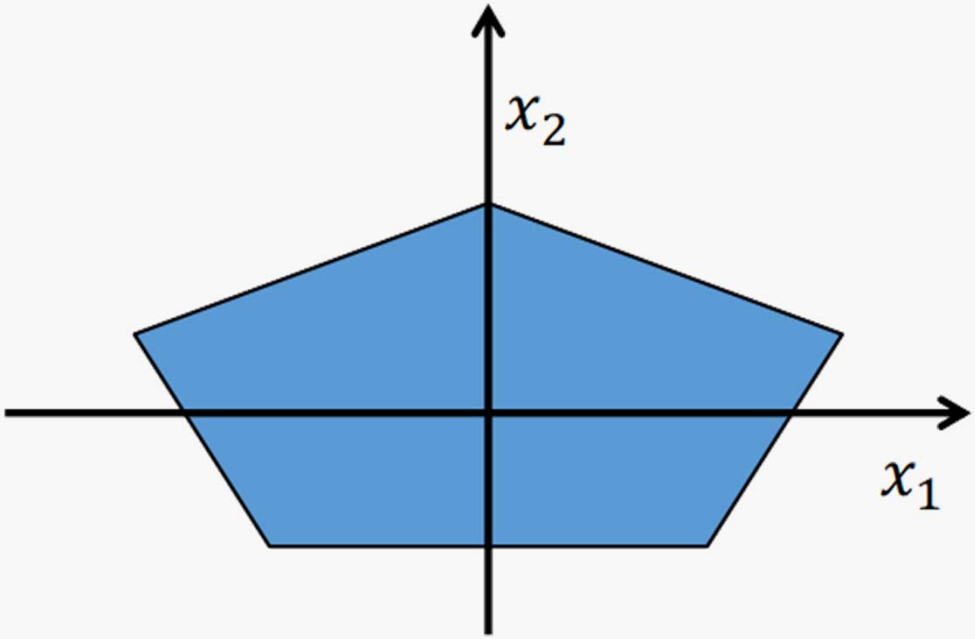
- **多层前馈神经网络：**每层神经元与下一层神经元全互连，神经元之间不存在同层连接，也不存在跨层连接
- **机制：**输入层神经元接收外界输入，隐层与输出层神经元对信号进行加工，最后结果由输出层神经元输出
- 两层神经网络又称单隐层网络，只需包含隐层，即可称为多层网络。“前馈”并不意味着网络信号不能向后传，而是指网络拓扑结构上不存在环或回路



(a) 单隐层前馈网络　　(b) 双隐层前馈网络

多层前馈网络

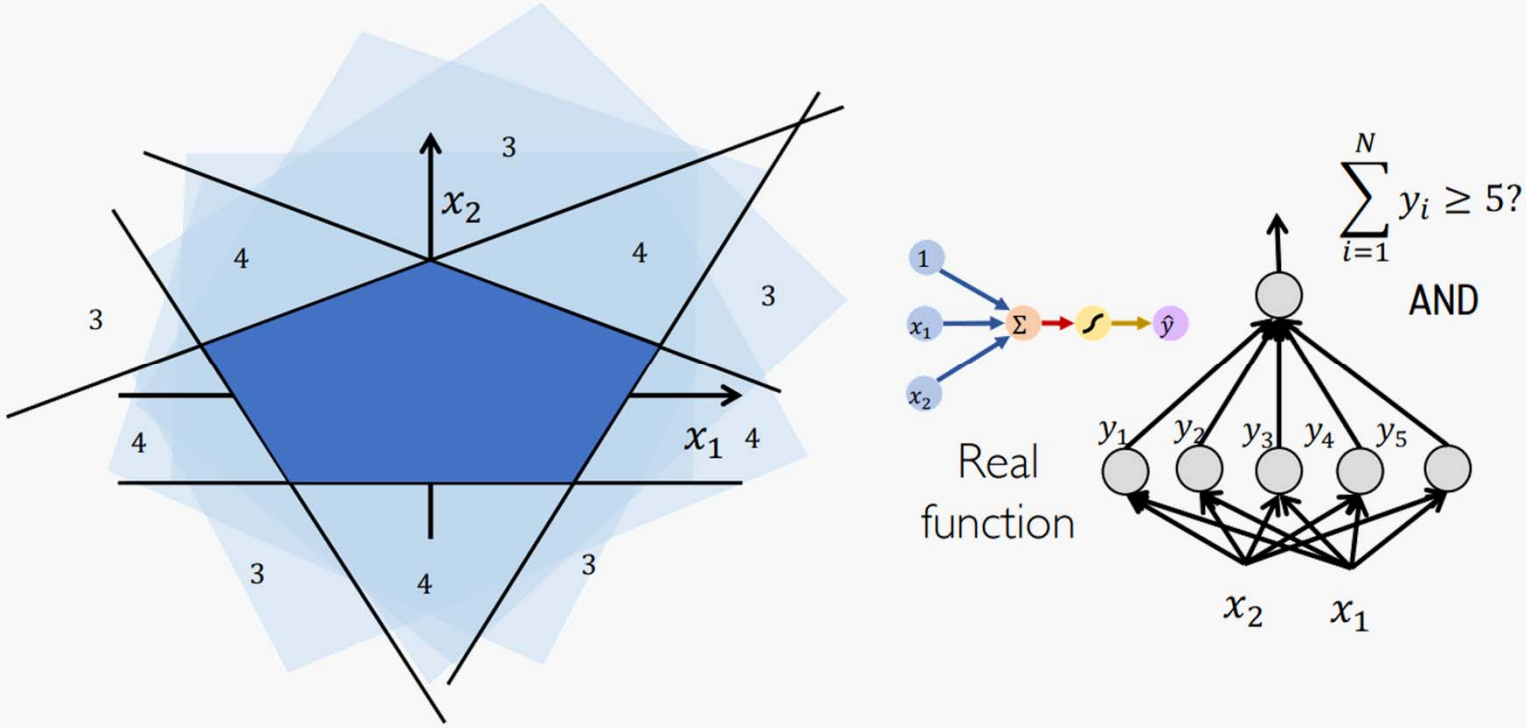
- 思考题
  - 如何建立一个输出层神经元个数为1的多层感知机（MLP）实现下面区域的二分类建模？



多层前馈网络

思考题解答

- 如何建立一个输出层神经元个数为1的多层感知机（MLP）实现下面区域的二分类建模？



§4.3 反向传播算法

一、算法背景

二、算法内容

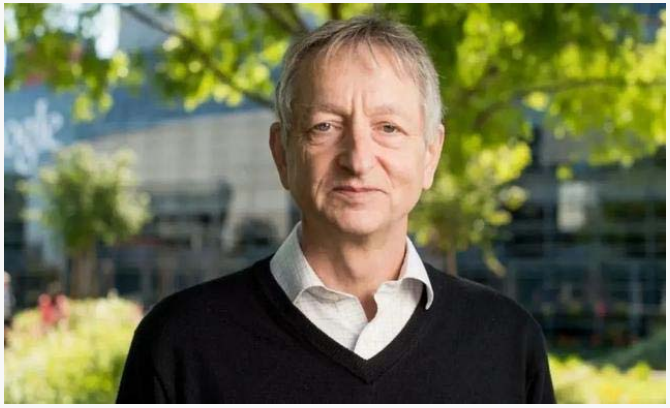
三、算法性质

四、最小值与极小值

算法背景

提出背景

- 若需要训练多层网络，简单感知机的学习规则难以胜任，需要更强大的学习算法。**反向传播算法**（或误差逆传播算法）是诸多学习算法中最杰出的代表，也是迄今最成功的神经网络学习算法，以下简称为BP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，值得指出的是，BP算法不仅可用于多层前馈神经网络，还可用于其他类型的神经网络，例如训练递归神经网络



Geoffrey Hinton

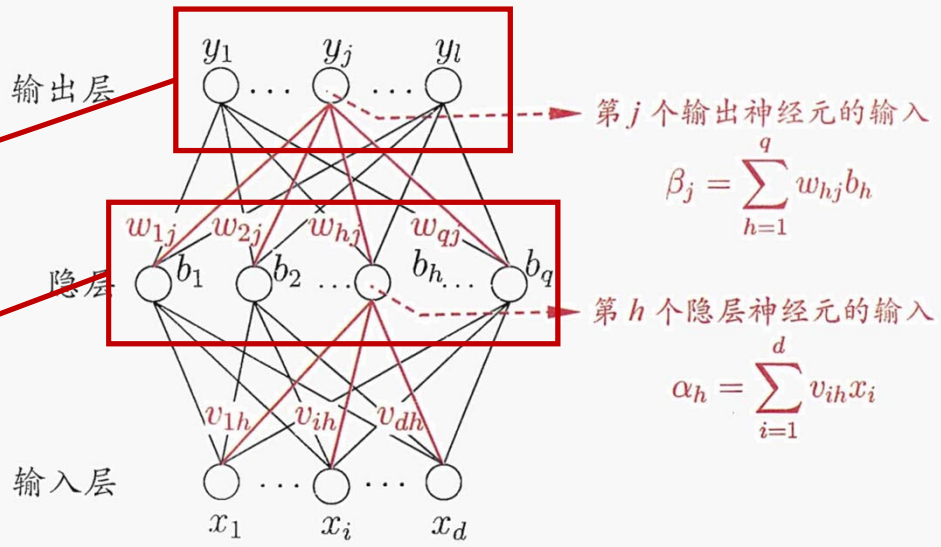
加拿大多伦多大学计算机科学系荣誉退休教授
2018年图灵奖获得者
深度学习之父
谷歌公司副总裁
给人工智能带来了重大突破，这些突破使深度神经网络成为智能计算的关键组成部分

算法内容

- 符号约定
  - **输入数据：**给定训练集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}, x_i\in R^d, y_i\in R^l$，即输入示例（样本）由$d$个属性描述，输出$l$维实值向量
  - **中间变量：**给定拥有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馈网络结构
    - 输出层第j个神经元的阈值θ_j
    - 隐层第h个神经元的阈值γ_h

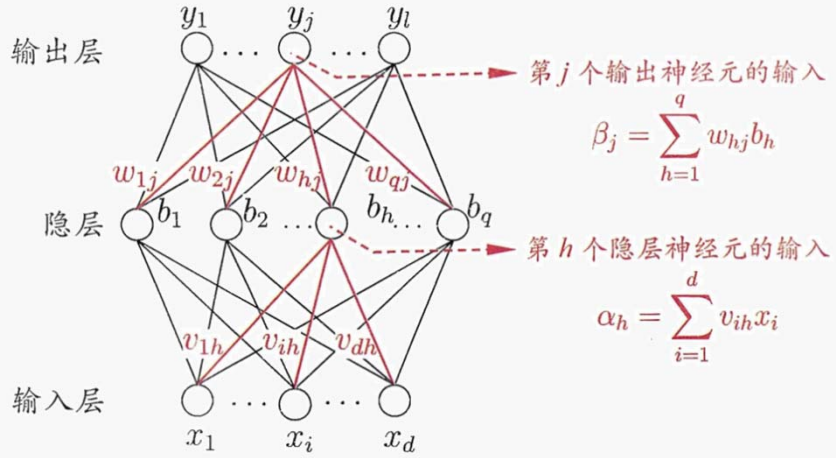
$f(\beta_j-\theta_j)$

$f(\alpha_j-\gamma_j)$



算法内容

- 符号约定
  - **中间变量：**
    - 输入层第i个神经元与隐层第h个神经元之间的连接权v_{ih}
    - 隐层第h个神经元与输出层第j个神经元之间的连接权w_{hj}
    - 记隐层第h个神经元接收到的输入为$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$，输出层第$j$个神经元接收到的输入为$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$，其中$b_h$为隐层第$h$个神经元的输出



算法内容

模型设定

- **输入数据：**若输入属性为离散值则需要先进行处理。若属性值间存在“序”关系则可以进行连续化，否则通常转化为k维向量，k为属性值数
- **激活函数：**假设隐层和输出层神经元使用Sigmoid函数。对训练示例(x_k, y_k)，假定神经网络的输出为$\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \cdots, \hat{y}_l^k)$，即

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

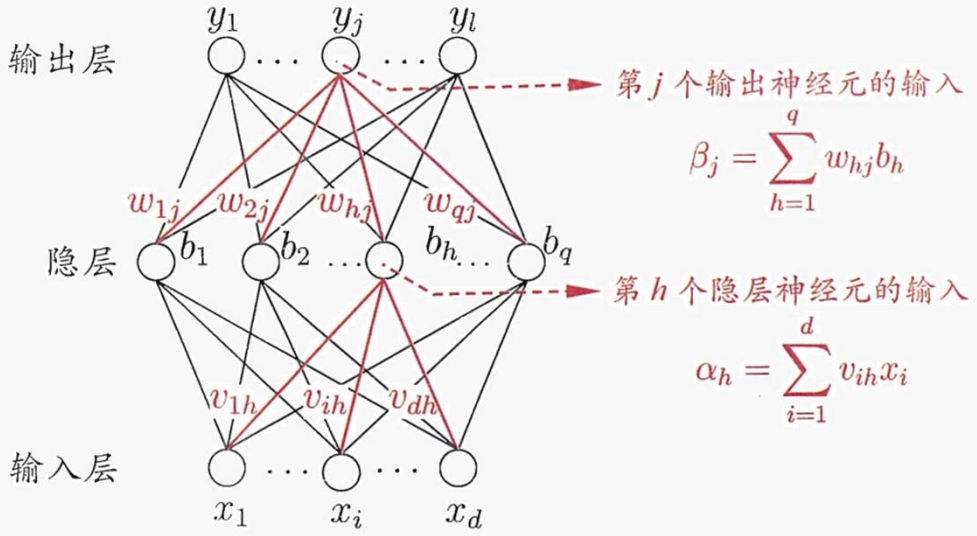
- **损失函数：**网络在(x_k, y_k)上的均方误差为

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

算法内容

模型设定

- **参数个数：**图中网络中有$(d+l+1)q+l$个参数需确定。输入层到隐层的$d\times q$个权值、隐层到输出层的$q\times l$个权值、q个隐层神经元的阈值、l个输出层神经元的阈值



算法内容

更新策略

- BP算法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，任意参数v的更新估计公式为

$$\mathbf{v} \leftarrow \mathbf{v} + \Delta\mathbf{v}$$

- 对于BP算法，更新估计公式中的Δv利用优化目标对模型参数的梯度$\frac{\partial E}{\partial v}$计算得到

算法内容

梯度计算

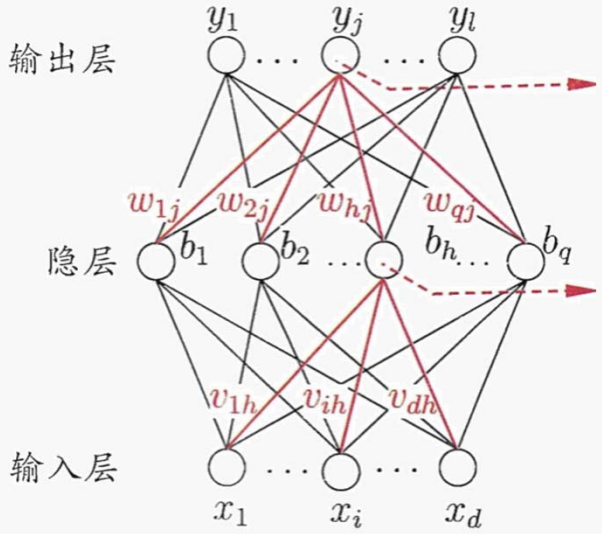
- **梯度下降**：BP算法基于梯度下降策略，以目标的负梯度方向对参数进行调整。例如对于误差E_k，给定学习率η，有

$$\Delta w_{hj} = \eta \frac{\partial E_k}{\partial w_{hj}}$$

- **链式法则**：注意到w_{hj}先影响到第j个输出层神经元的输入值β_j，再影响到其输出值$\hat{y}_j^k$，然后影响到E_k，有

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

根据β_j的定义，有$\frac{\partial \beta_j}{\partial w_{hj}} = b_h$



算法内容

- 梯度计算
  - （接上页）链式法则：对于使用Sigmoid函数作为激活函数的网络，该函数具有很好的性质：$f'(x)=f(x)\big(1-f(x)\big)$。因此根据链式法则，有

$$g_j=-\frac{\partial E_k}{\partial \hat{y}_j^k}\cdot\frac{\partial \hat{y}_j^k}{\partial \beta_j}=-\big(\hat{y}_j^k-y_j^k\big)f'\big(\beta_j-\theta_j\big)=\hat{y}_j^k\big(1-\hat{y}_j^k\big)\big(y_j^k-\hat{y}_j^k\big)$$

综上可以得到BP算法中关于w_{hj}的更新公式

$$\Delta w_{hj}=\eta g_j b_h$$

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

$$f\big(\beta_j-\theta_j\big)$$

算法内容

- 梯度计算
  - （接上页）链式法则：类似可得

$$\Delta\theta_j = -\eta g_j$$
$$\Delta v_{ih} = \eta e_h x_i$$
$$\Delta\gamma_h = -\eta e_h$$

其中

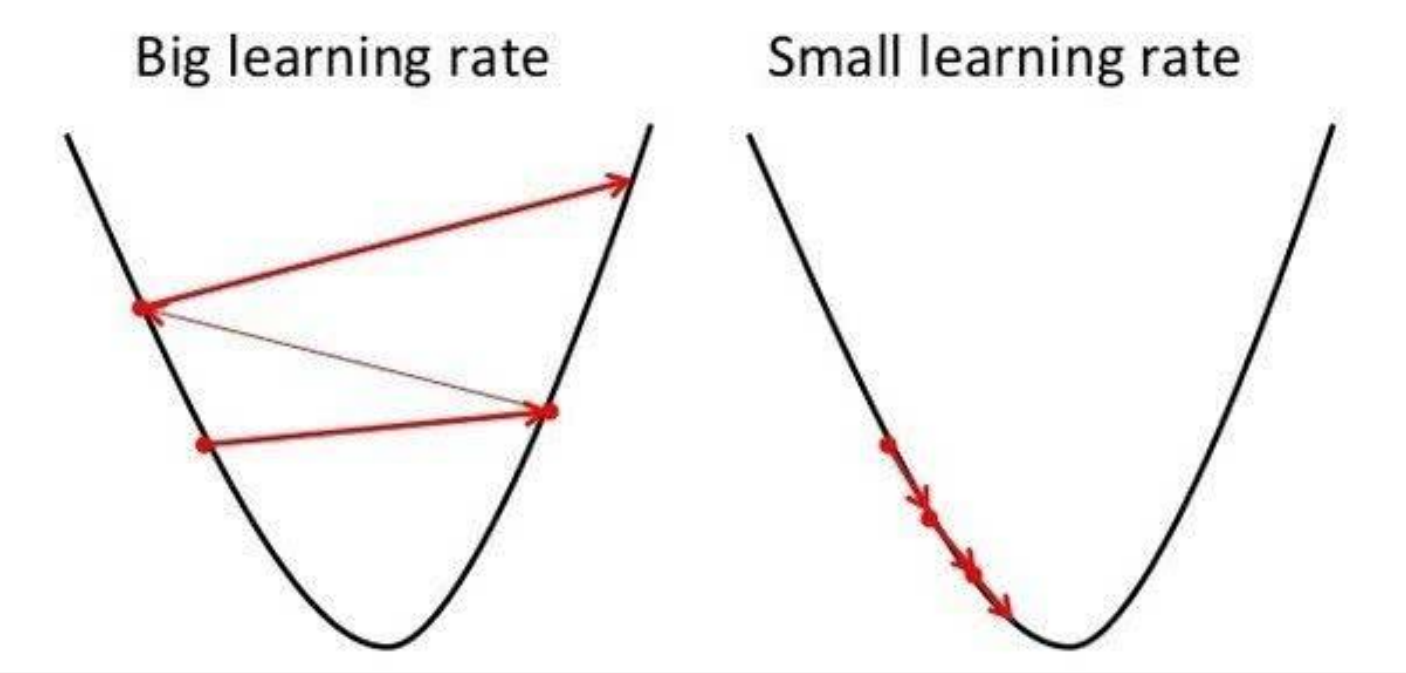
$$e_h = -\frac{\partial E_k}{\partial b_h}\cdot\frac{\partial b_h}{\partial \alpha_h} = -\sum_{j=1}^{l}\frac{\partial E_k}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h)$$

$f(\alpha_h - \gamma_h)$

$$= \sum_{j=1}^{l} w_{hj} g_j f'(\alpha_h - \gamma_h) = b_h(1-b_h)\sum_{j=1}^{l} w_{hj} g_j$$

算法内容

- 梯度计算
  - **学习率设定：**学习率$\eta \in (0,1)$控制着算法每一轮迭代中的更新步长，若太大则容易振荡，太小则收敛速度又会过慢。有时为了做精细调节，可以对不同参数使用不同大小的步长



算法内容

算法框图

- **进行与终止：**对每个训练样例，BP算法先将输入示例提供给输入层神经元，然后逐层将信号前传，直到产生输出层的结果；然后计算输出层的误差，再将误差逆向传播值隐层神经元，最后根据隐层神经元的误差来调整参数。该迭代过程循环进行，直到达到**停止条件**，例如训练误差达到一个很小的值

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 (x_k, y_k) ∈ D do
        根据当前参数和式(5.3) 计算当前样本的输出 ŷ_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

算法内容

- 累积BP算法
  - **定义**：上述BP算法的目标是最小化训练集D上的累积误差

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k$$

前面介绍的“标准BP算法”每次仅针对一个训练样例更新参数。利用链式法则我们也可以类似地推导出基于累计误差最小化的更新规则

  - **术语**：读取训练集一遍称为进行了“一轮”（one round，亦称one epoch）学习

算法内容

- 两种BP算法对比
  - **标准BP算法：**一般来说，标准BP算法每次更新只针对单个样例，参数更新非常频繁，而且对不同样例进行更新的效果可能出现“抵消”现象。因此，为了达到同样的累积误差极小点，标准BP算法往往需进行更多次数的迭代
  - **累积BP算法：**累积BP算法直接针对累积误差最小化，它在读取整个训练集D一遍后才对参数进行更新，其参数更新的频率低得多。但在很多任务中，累积误差下降到一定程度之后，进一步下降会非常缓慢，这时标准BP往往会更快获得较好的解，尤其是在训练集D非常大时更明显

算法实例

反向传播算法推导

- 给定训练样本$\{x_i, y_i\}_{i=1}^n, x_i = \left(x_{i,1}, \cdots, x_{i,n_x}\right)^T \in R^{n_x}, y_i = \left(y_{i,1}, \cdots, y_{i,n_y}\right)^T \in R^{n_y}$。我们实现两层全连接网络如下

$$z_{1,i} = W^{(1)} x_i + b^{(1)} \in R^{n_1}, W^{(1)} \in R^{n_1 \times n_x}, b^{(1)} \in R^{n_1}$$

$$h_{1,i} = Sigmoid\left(z_{1,i}\right) \in R^{n_1}$$

$$z_{2,i} = W^{(2)} h_{1,i} + b^{(2)} \in R^{n_y}, W^{(2)} \in R^{n_y \times n_1}, b^{(2)} \in R^{n_y}$$

$$\hat{y}_i = Softmax\left(z_{2,i}\right)$$

$$f\left(\{x_i, y_i\}, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\right) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{n_y} y_{i,k} \log \hat{y}_{i,k}$$

试推导$\frac{\partial f}{\partial W^{(1)}}, \frac{\partial f}{\partial b^{(1)}}, \frac{\partial f}{\partial W^{(2)}}, \frac{\partial f}{\partial b^{(2)}}$。

提示：Softmax函数是Sigmoid函数的多值版本

算法实例

- 反向传播算法推导
  - 利用链式法则，对第i个样本有

$$\frac{\partial f}{\partial \hat{y}_{i,j}} = -\frac{1}{n} y_{i,j} \cdot \frac{1}{\hat{y}_{i,j}}$$

$$\frac{\partial \hat{y}_{i,j}}{\partial z_{2,i,k}} = \begin{cases} \hat{y}_{i,j}\left(1 - \hat{y}_{i,k}\right), j = k \\ -\hat{y}_{i,j} \cdot \hat{y}_{i,k}, j \neq k \end{cases}$$

$$\frac{\partial z_{2,i,k}}{\partial b_k^{(2)}} = 1, \frac{\partial z_{2,i,k}}{\partial W_{k,l}^{(2)}} = h_{1,i,l}$$

$$\frac{\partial h_{1,i,l}}{\partial z_{1,i,l}} = h_{1,i,l}(1 - h_{1,i,l})$$

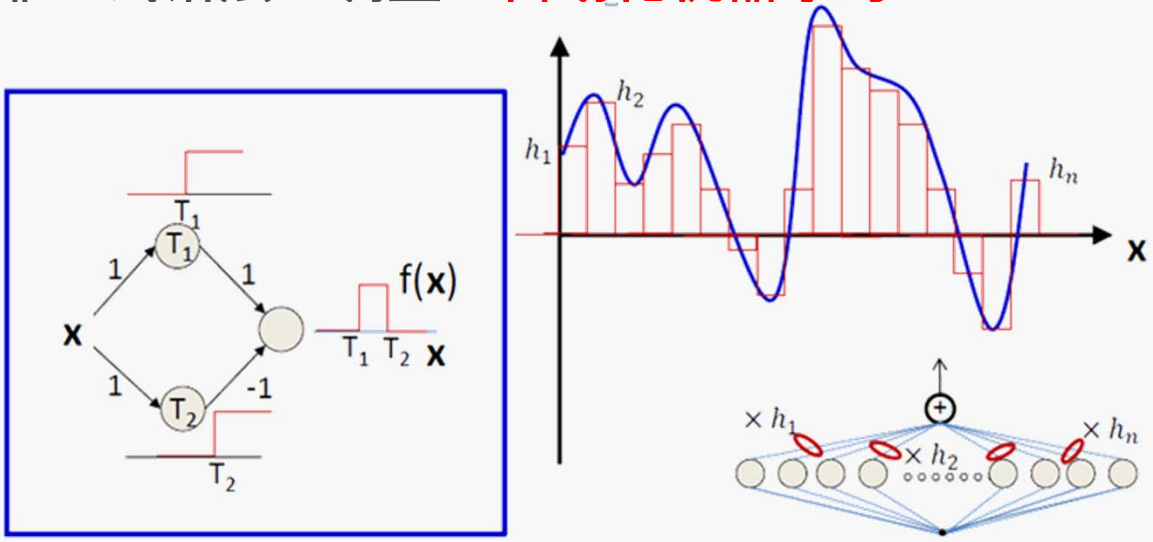
$$\frac{\partial z_{1,i,l}}{\partial b_l^{(1)}} = 1, \frac{\partial z_{1,i,l}}{\partial W_{l,s}^{(1)}} = x_{i,s}$$

$\hat{y}_i$的每个元素都与$z_{2,i}$的每个元素有关联！故

$$\frac{\partial f}{\partial z_{2,i,k}} = \sum_{j=1}^{n_y} \frac{\partial f}{\partial \hat{y}_{i,j}} \cdot \frac{\partial \hat{y}_{i,j}}{\partial z_{2,i,k}}$$

算法性质

- 理论研究
  - **万有近似定理**：Hornik等人于1989年证明，只需一个包含足够多神经元的隐层，多层前馈网络就能以任意精度逼近任意复杂度的连续函数
  - **局限性**：但是如何设置隐层神经元的个数仍然是未解决的问题，实际应用中通常依靠“试错法”调整—**自动化机器学习**

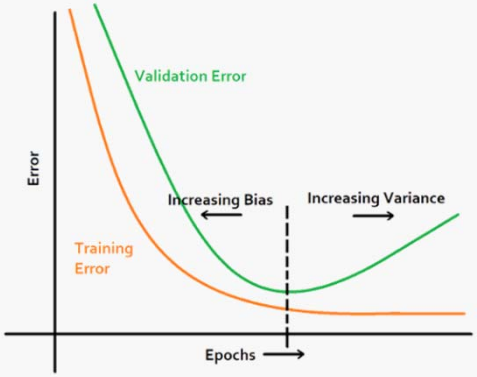


万有近似定理

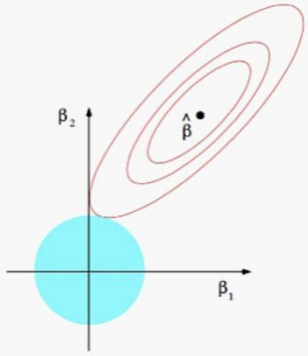
算法性质

过拟合及其缓解

- **过拟合现象**：由于其强大的表示能力，BP神经网络经常遭遇过拟合，其训练误差持续降低，但测试误差却可能上升
- **（缓解策略1）早停**：将数据分成训练集和验证集，训练集用于梯度更新，验证集用于估计误差，若训练集误差降低但验证集误差升高，则停止训练
- **（缓解策略2）正则化**：基本思想是在误差目标函数中增加一个用于描述网络复杂度的部分，例如连接权与阈值的平方和



早停



正则化

算法性质

- 过拟合及其缓解
  - （接上页）正则化：令E_k表示第k个训练样例上的误差，w_i表示连接权和阈值，则误差目标函数改变为

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

其中$\lambda \in (0,1)$用于对经验误差与网络复杂度这两项进行折中，常通过交叉验证法来估计

  - 理解：增加连接权与阈值平方和后，训练过程将会偏好比较小的连接权和阈值，使网络输出更加“光滑”，从而对过拟合有所缓解

梯度下降法能否求得全局最小值？
需要区分最小值与极小值！

最小值与极小值

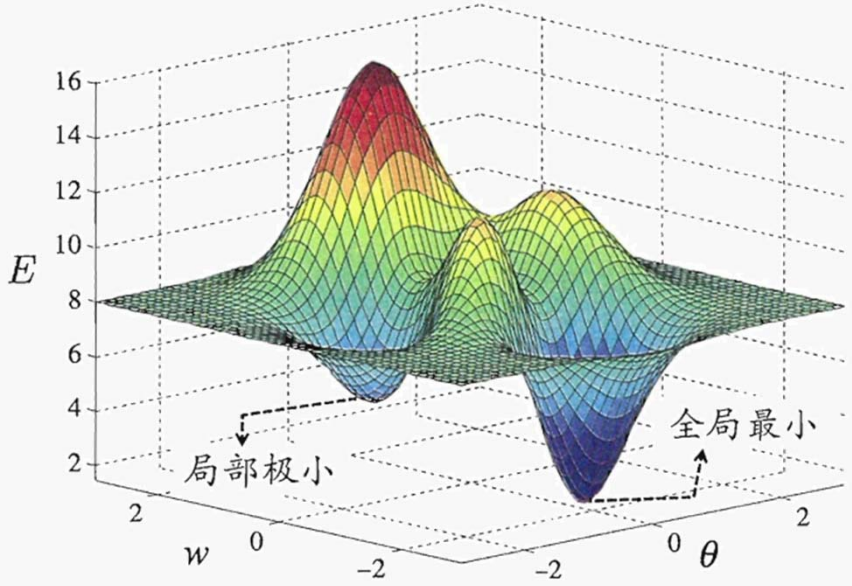
概念介绍

- **提出背景：**若用E表示神经网络在训练集上的误差，则它显然是关于连接权w和阈值θ的函数。此时，神经网络的训练过程可看作参数寻优的过程，即在参数空间中，寻找一组最优参数使得E最小
- **局部极小：**对w^*和θ^*，若存在$\epsilon > 0$使得$\forall (w;\theta) \in \{(w;\theta)| ||(w;\theta) - (w^*;\theta^*)|| \leq \epsilon\}$都有$E(w;\theta) \geq E(w^*;\theta^*)$成立，则$(w^*;\theta^*)$为局部极小解
- **全局最小：**若对参数空间中的任意$(w;\theta)$都有$E(w;\theta) \geq E(w^*;\theta^*)$，则$(w^*;\theta^*)$为全局最小解

最小值与极小值

概念介绍

- **理解：**直观来看，局部极小解释参数空间中的某个点，其邻域点的误差函数值均不小于该点的函数值；全局最小解则是指参数空间中所有点的误差函数值均不小于该点的误差函数之，两者对应的$E(\boldsymbol{w}^*;\boldsymbol{\theta}^*)$分别称为误差函数的局部极小值和全局最小值



最小值与极小值

- 参数寻优方法
  - **基于梯度的搜索：**该类方法是使用最为广泛的参数寻优方法。此类方法从某些初始解出发，迭代寻找最优参数值。每次迭代中，先计算误差函数在当前点的梯度，然后根据梯度确定搜索方向。例如，由于负梯度方向是函数值下降最快的方向，因此梯度下降法就是沿着负梯度方向搜索最优解
  - **搜索停止条件：**若误差函数在当前点的梯度为零，则可能已达到局部极小，更新量将为零，意味着参数的迭代更新将停止。显然，如果误差函数仅有一个局部极小，那么此时找到的局部极小就是全局最小；否则不能保证找到全局最小解

如何避免陷入局部极小点？
利用一些跳出极小点的策略！

最小值与极小值

- 跳出局部极小点的策略
  - **多组初始化**：以多组不同参数值初始化多个神经网络，按标准方法训练后，取其中误差最小的解作为最终参数
  - **模拟退火**：模拟退火在每一步都以一定的概率接受比当前解更差的结果，从而有助于“跳出”局部极小。在每步迭代过程中，接受“次优解”的概率要随着时间的推移而逐渐降低，从而保证算法稳定
  - **随机梯度下降**：与标准梯度下降法精确计算梯度不同，随机梯度下降法在计算梯度时加入了随机因素。于是，即便陷入局部极小点，它计算出的梯度仍可能不为零，这样就有机会跳出局部极小继续搜索
  - **说明**：上述用于跳出局部极小的技术大多是启发式，理论上目前缺乏保障