《机器学习》课件

3.2人工神经网络

背景

- 统计学习领域的重要分支，起源于感知机，现在比较新的神经网络应该是Deep Belief Network,加拿大
- 人工神经网络是集脑科学、神经心理学和信息科学等多学科的交叉研究领域。
- 它的研究目标是通过研究人脑的组成机理和思维方式，探索人类智能的奥秘，进而通过模拟人脑的结构和工作模式，使机器具有类似人类的智能。
- 它已在模式识别、机器学习、专家系统等多个方面得到应用，成为人工智能研究中的活跃领域。
- 人工智能领域的一个里程碑

3.2.1人工神经网络的提出

- 人工神经网络（Artificial Neural Networks，简记作ANN），是对人类大脑系统的一阶特性的一种描述。简单地讲，它是一个数学模型，可以用电子线路来实现，也可以用计算机程序来模拟，是人工智能研究的一种方法。

人工神经网络的提出

- 二、人工智能
- 人工智能：研究如何使类似计算机这样的设备去模拟人类的这些能力。
- 研究人工智能的目的
  - 增加人类探索世界，推动社会前进的能力
  - 进一步认识自己
- 三大学术流派
  - 符号主义（或叫做符号/逻辑主义）学派
  - 联接主义（或者叫做PDP）学派
  - 进化主义（或者叫做行动/响应）学派

人工神经网络的提出

- 联接主义观点
- 核心：智能的本质是联接机制。
- 神经网络是一个由大量简单的处理单元组成的高度复杂的大规模非线性自适应系统
- ANN力求从四个方面去模拟人脑的智能行为
  - 物理结构
  - 计算模拟
  - 存储与操作
  - 训练

历史回顾

- 人工神经网络研究的兴起与发展
- 人工神经网络的研究经历了不少的曲折，大体上可分为四个阶段:
  - 产生时期(20世纪50年代中期之前)
  - 高潮时期(20世纪50年代中期到20世纪60年代末期)
  - 低潮时期(20世纪60年代末到20世纪80年代初期)
  - 蓬勃发展时期(20世纪80年代以后)

历史回顾

- 萌芽期（20世纪40年代）
- 人工神经网络的研究最早可以追溯到人类开始研究自己的智能的时期，到1949年止。
- 1943年，心理学家McCulloch和数学家Pitts建立起了著名的阈值加权和模型，简称为M-P模型。发表于数学生物物理学会刊《Bulletin of Methematical Biophysics》
- 1949年，心理学家D. O. Hebb提出神经元之间突触联系是可变的假说——Hebb学习律。

第一高潮期（1950~1968）

- 以Marvin Minsky，Frank Rosenblatt，Bernard Widrow等为代表人物，代表作是单级感知器（Perceptron）。
- 可用电子线路模拟。
- 人们乐观地认为几乎已经找到了智能的关键。许多部门都开始大批地投入此项研究，希望尽快占领制高点。

反思期（1969~1982）

- M. L. Minsky和S. Papert，《Perceptron》，MIT Press，1969年
- 异或”运算不可表示
- 二十世纪70年代和80年代早期的研究结果
- 认识规律：认识——实践——再认识

第二高潮期（1983~1990）

- 1982年，J. Hopfield提出循环网络
  - 用Lyapunov函数作为网络性能判定的能量函数，建立ANN稳定性的判别依据
  - 阐明了ANN与动力学的关系
  - 用非线性动力学的方法来研究ANN的特性
  - 指出信息被存放在网络中神经元的联接上

$$V = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}s(x_i)s(x_j) - \sum_{i=1}^{n}\int_0^{x_i} s_i'(\theta_i)\beta_i(\theta_i)d\theta_i - \sum_{i=1}^{n}\sum_{j-1}^{n} w_{ij}$$

第二高潮期（1983~1990）

- 2）1984年， J. Hopfield设计研制了后来被人们称为Hopfield网的电路。较好地解决了著名的TSP问题，找到了最佳解的近似解，引起了较大的轰动。
- 3）1985年，UCSD的Hinton、Sejnowsky、Rumelhart等人所在的并行分布处理（PDP）小组的研究者在Hopfield网络中引入了随机机制，提出所谓的Boltzmann机。

第二高潮期（1983~1990）

- 4）1986年，并行分布处理小组的Rumelhart等研究者重新独立地提出多层网络的学习算法——BP算法，较好地解决了多层网络的学习问题。（Paker1982和Werbos1974年）
- 国内首届神经网络大会是1990年12月在北京举行的。

再认识与应用研究期（1991~）

- 问题：
- 1）应用面还不够宽
- 2）结果不够精确
- 3）存在可信度的问题

再认识与应用研究期（1991~）

- 研究：
- 1）开发现有模型的应用，并在应用中根据实际运行情况对模型、算法加以改造，以提高网络的训练速度和运行的准确度。
- 2）充分发挥两种技术各自的优势是一个有效方法
- 3）希望在理论上寻找新的突破，建立新的专用/通用模型和算法。
- 4）进一步对生物神经系统进行研究，不断地丰富对人脑的认识。

人工神经网络的概念

- 1、定义
- 1）Hecht—Nielsen（1988年）
- 人工神经网络是一个并行、分布处理结构，它由处理单元及其称为联接的无向讯号通道互连而成。这些处理单元（PE—Processing Element）具有局部内存，并可以完成局部操作。每个处理单元有一个单一的输出联接，这个输出可以根据需要被分枝成希望个数的许多并行联接，且这些并行联接都输出相同的信号，即相应处理单元的信号，信号的大小不因分支的多少而变化。

人工神经网络的概念

- （1）Hecht—Nielsen（1988年）（续）
- 处理单元的输出信号可以是任何需要的数学模型，每个处理单元中进行的操作必须是完全局部的。也就是说，它必须仅仅依赖于经过输入联接到达处理单元的所有输入信号的当前值和存储在处理单元局部内存中的值。

人工神经网络的概念

- （2） Rumellhart，McClelland，Hinton的PDP
- 1） 一组处理单元（PE或AN）；
- 2） 处理单元的激活状态（a_i）；
- 3） 每个处理单元的输出函数（f_i）；
- 4） 处理单元之间的联接模式；
- 5） 传递规则（$\sum w_{ij} o_i$）；
- 6） 把处理单元的输入及当前状态结合起来产生激活值的激活规则（F_i）；
- 7） 通过经验修改联接强度的学习规则；
- 8） 系统运行的环境（样本集合）。

人工神经网络的概念

- （3） Simpson（1987年）
- 人工神经网络是一个非线性的有向图，图中含有可以通过改变权大小来存放模式的加权边，并且可以从不完整的或未知的输入找到模式。

人工神经网络具有主要特征：

- 能较好的模拟人的形象思维。
  - 具有大规模并行协同处理能力。
  - 具有较强的学习能力。
    - 具有较强的容错能力和联想能力。
    - 是一个大规模自组织、自适应的非线性动力系统。

3.2.2 人工神经网络的组成

- **生物神经元的结构与功能特性**
- 1. 生物神经元的结构
- 神经细胞是构成神经系统的基本单元，称之为生物神经元，简称神经元。神经元主要由三部分构成：
- （1）细胞体；（2）轴突；（3）树突；

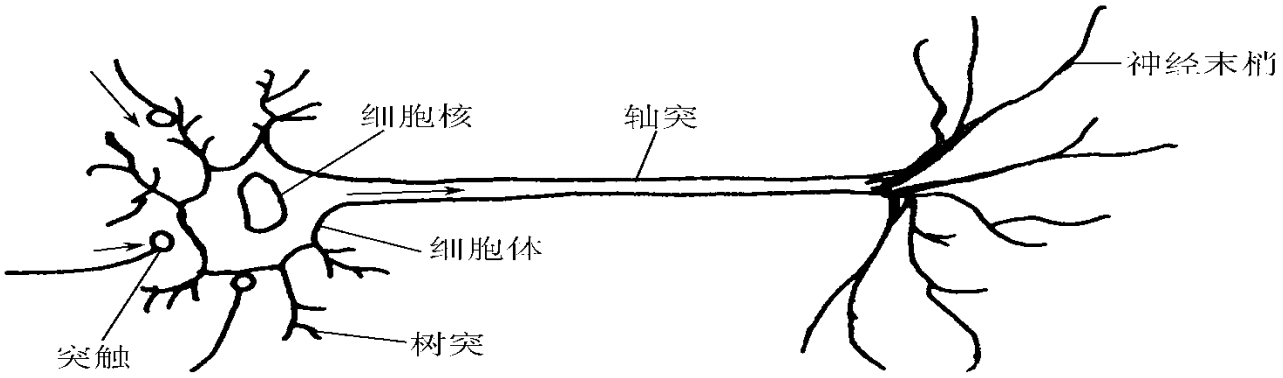
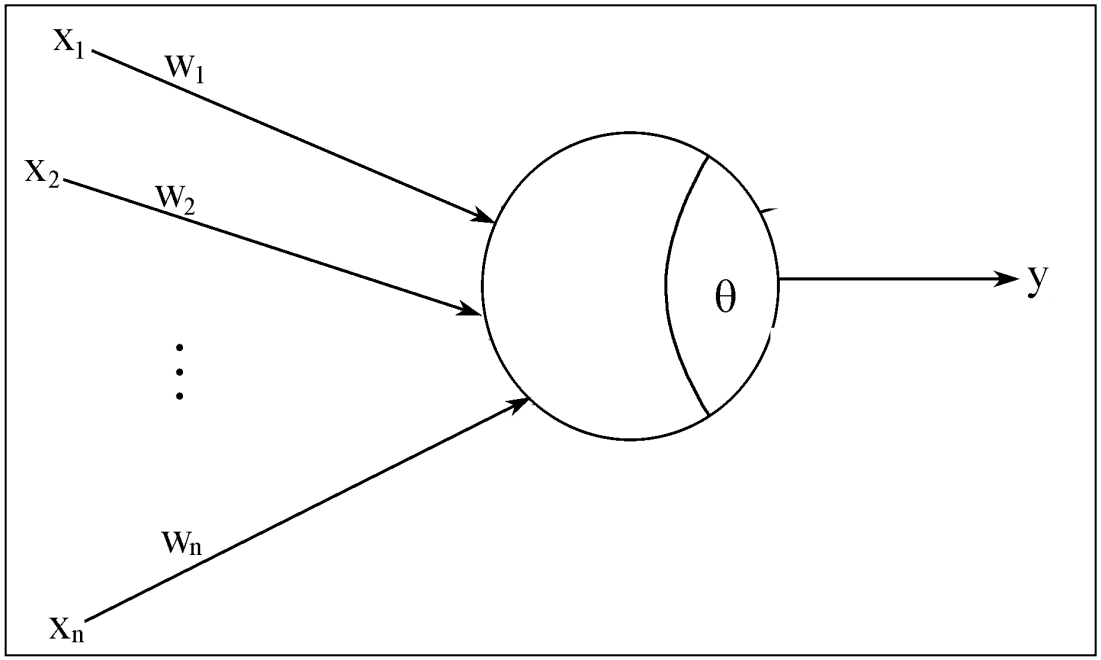


图 生物神经元结构

人工神经网络的组成

- 突触是神经元之间相互连接的接口部分，即一个神经元的神经末梢与另一个神经元的树突相接触的交界面，位于神经元的神经末梢尾端。突触是轴突的终端。
- **2. 人工神经网络的组成**
  - 人工神经网络（简称ANN）是由大量处理单元经广泛互连而组成的人工网络，用来模拟脑神经系统的结构和功能。而这些处理单元我们把它称作**人工神经元。**
  - 人工神经网络（ANN）可看成是以人工神经元为节点，用有向加权弧连接起来的有向图。
  - 在此有向图中，人工神经元就是对生物神经元的模拟，而有向弧则是轴突—突触—树突对的模拟。
  - 有向弧的权值表示相互连接的两个人工神经元间相互作用的强弱。

神经网络的神经元模型



M－P神经元模型

人工神经元的工作过程

对于某个处理单元（神经元）来说，假设来自其他处理单元（神经元）i的信息为X_i，它们与本处理单元的互相作用强度即连接权值为W_i, i=0,1,…,n-1,处理单元的内部阈值为θ。那么本处理单元（神经元）的输入为

$$\sum_{i=0}^{n-1} W_i X_i \qquad (9.1.1)$$

而处理单元的输出为

$$y = f\left(\sum_{i=0}^{n-1} w_i x_i - \theta\right) \qquad (9.1.2)$$

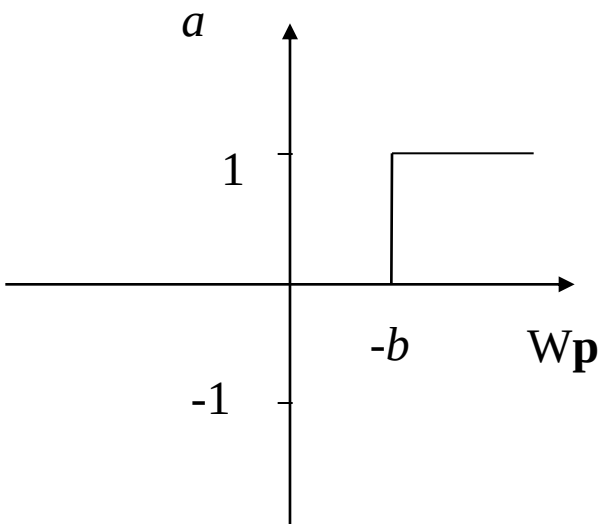
式中，x_i为第i个元素的输入，w_i为第i个处理单元与本处理单元的互联权重。f称为*激发函数*或作用函数，它决定节点（神经元）的输出。

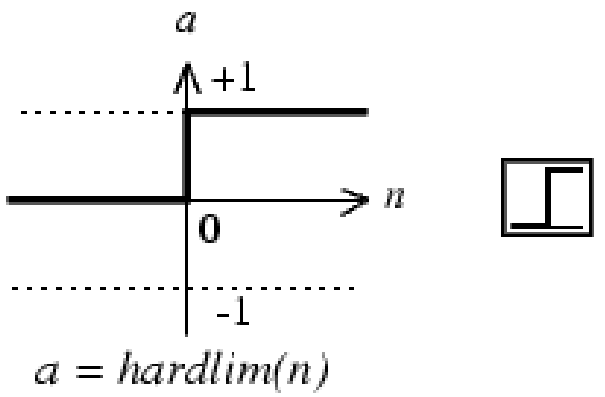
激发函数

- 阈值型函数又称阶跃函数，它表示激活值σ和其输出f(σ)之间的关系。
- 阈值型函数为激发函数的神经元是一种最简单的人工神经元，也就是我们前面提到的M-P模型。
- S型函数是一个有最大输出值的非线性函数，其输出值是在某个范围内连续取值的。以它为激发函数的神经元也具有饱和特性。

常用输出函数

- 阈值函数：

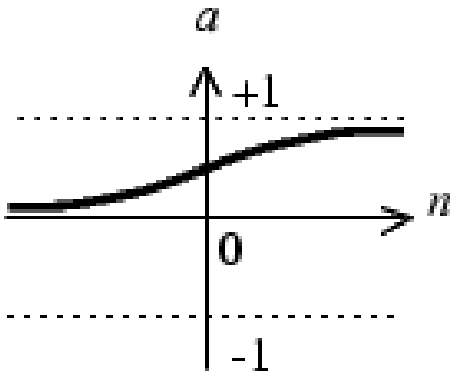




Hard-Limit Transfer Function

Sigmoid函数

- Sigmoid Function ：
- 特性：
  - 值域a∈(0,1)
  - 非线性，单调性
  - 无限次可微
  - |n|较小时可近似线性函数
  - |n|较大时可近似阈值函数



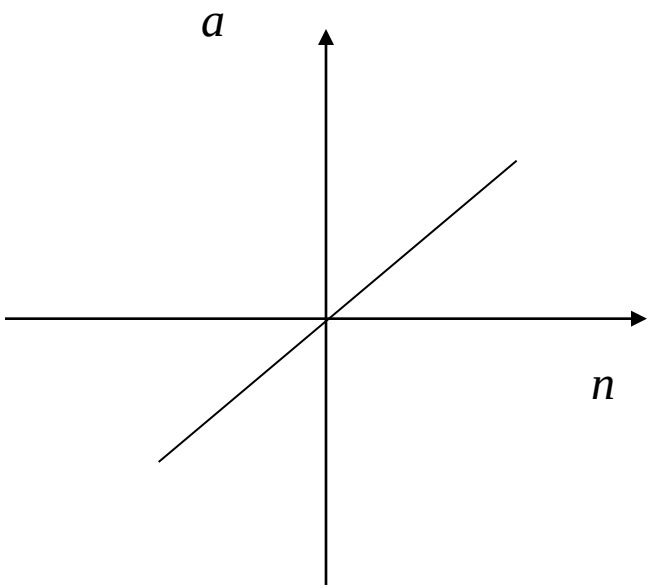
$a = logsig(n)$

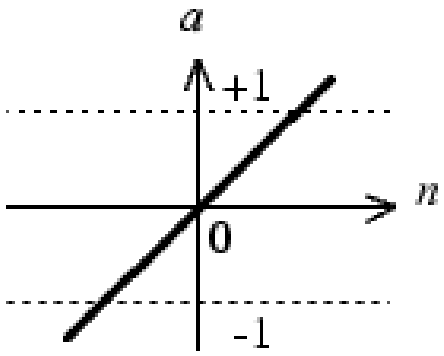
Log-Sigmoid Transfer Function

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

线性输出函数

- Purelin Transfer Function：





$a = purelin(n)$

Linear Transfer Function

损失函数

目标函数: $y(x) = wx + b$

参数: w, b

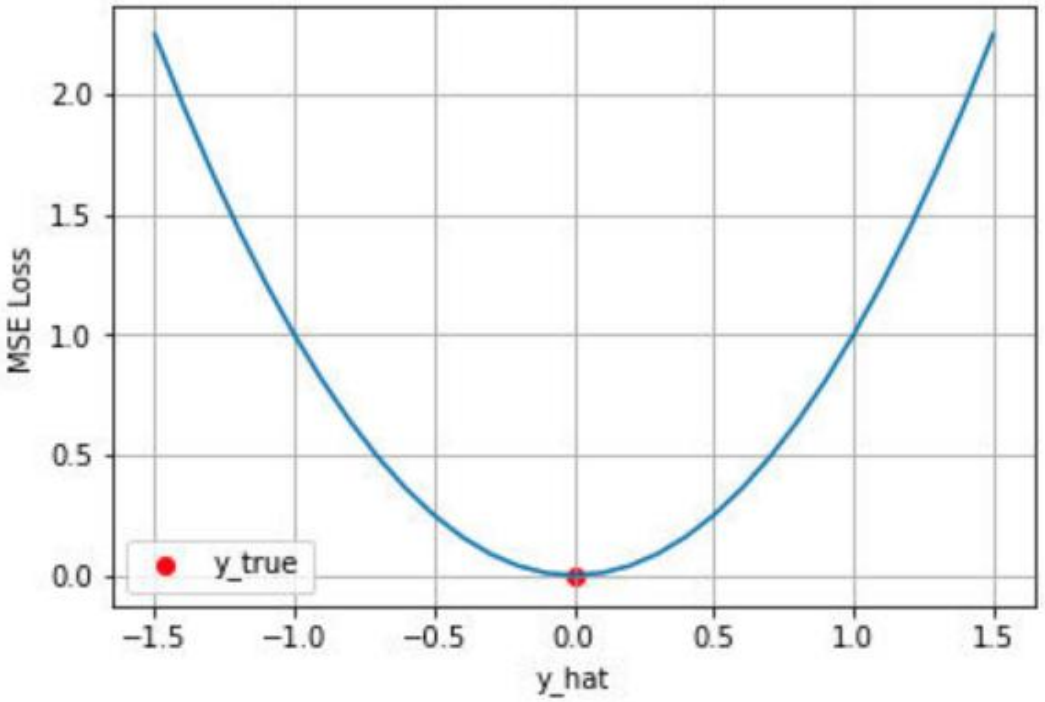
损失函数: $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(y(x_i) - y_i)^2$

目标: $minimize_{w,b}J(w,b)$

- 损失函数（loss function）或代价函数（cost function）：我们的目标是找到合适的w和b，使得损失函数值越小越好。我们通过梯度下降算法来达到这个目的。

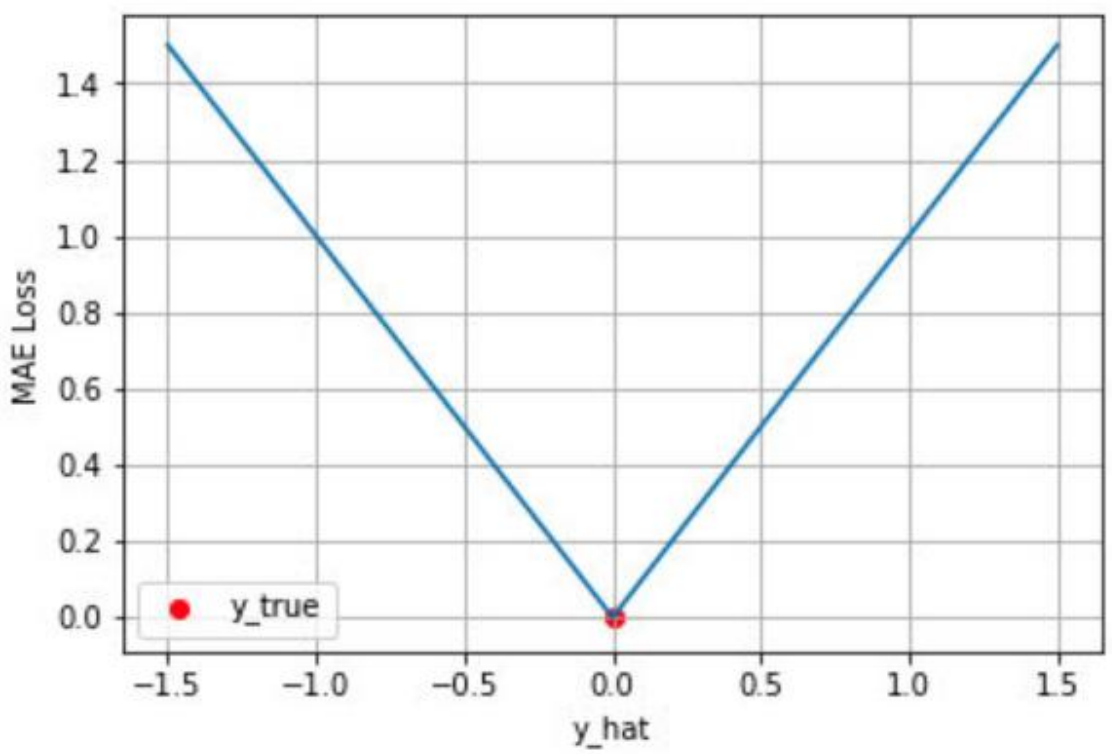
常用损失函数

- 均方差损失函数（Mean Squared Error, MSE），也称为 L2 Loss：$J = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y(x_i)\right)^2$
- 均方差 损失是机器学习、深度学习回归任务中最常用的一种损失函数。



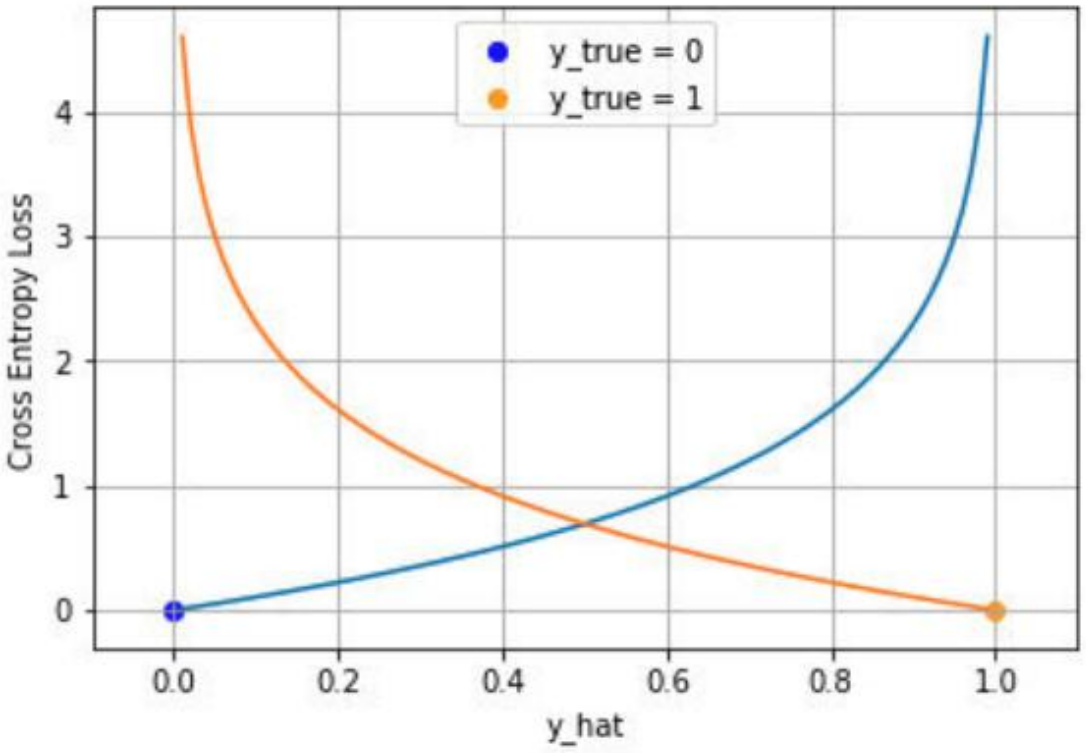
常用损失函数

- 平均绝对误差函数（Mean Absolute Error, MAE），也称为 L1 Loss：$J = \frac{1}{n}\sum_{i=1}^{n}|y_i - y(x_i)|$
- MSE 损失相比 MAE 通常可以更快地收敛，但 MAE 不易受到 out



常用损失函数

- 交叉熵损失函数（Cross Entropy Loss）：$J = -\frac{1}{n}\sum_x \left[y_i \ln y(x_i) + (1-y_i)\ln\left(1-y(x_i)\right)\right]$
- 当激活函数为sigmoid函数时，交叉熵一般都是更好的选



监督

- 监督就是对每一个输入X_i，都假定我们已经知道它的期望输出Y_i，这个Y_i可以理解为监督信号，也叫“教师信号”。
- 每一个输入X_i及期望输出Y_i，就构成了一个训练案例。

监督学习与非监督学习

- 在监督学习中，假定我们知道每一输入对应的期望输出，并利用学习系统的误差，不断校正系统的行为。
- 在非监督学习中，我们不知道学习系统的期望输出。

神经网络的分类

神经网络的分类有多种方法，常用如下分类：

- 按网络结构分为：前馈网络和反馈网络；
- 按学习方式分为：监督学习和非监督学习。

3.2.3 BP神经网络

- BP (Back Propagation) 网络1985年由Rumelhart和M从Celland提出。
  BP神经网络算法是一种用于前向多层的反向传播学习算法—起源于感知机

感知器模型及其学习算法

- 感知器模型
- 感知器模型是美国学者罗森勃拉特（Rosenblatt）为研究大脑的存储、学习和认知过程而提出的一类具有自学习能力的神经网络模型，它把神经网络的研究从纯理论探讨引向了从工程上的实现。
- Rosenblatt提出的感知器模型是一个只有单层计算单元的前向神经网络，称为单层感知器。

感知器模型及其学习算法

- 单层感知器模型的学习算法
- 算法思想：首先把连接权和阈值初始化为较小的非零随机数，然后把有n个连接权值的输入送入网络，经加权运算处理，得到的输出如果与所期望的输出有较大的差别，就对连接权值参数按照某种算法进行自动调整，经过多次反复，直到所得到的输出与所期望的输出间的差别满足要求为止。
- 为简单起见，仅考虑只有一个输出的简单情况。设$x_i(t)$是时刻t感知器的输入（i=1,2,.....,n），$\omega_i(t)$是相应的连接权值，y(t)是实际的输出，d(t)是所期望的输出，且感知器的输出或者为1，或者为0

感知器模型及其学习算法

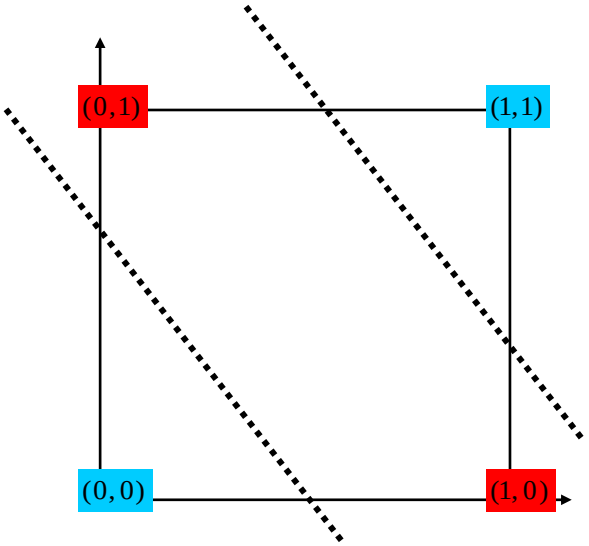
- **线性不可分问题**
- 单层感知器不能表达的问题被称为线性不可分问题。
  - 1969年，明斯基证明了“异或”问题是线性不可分问题：
  - “异或”（XOR）运算的定义如下：

$$y(x_1, x_2) = \begin{cases} 0, & \text{if } x_1 = x_2 \\ 1, & \text{其他} \end{cases}$$

  - 其相应的逻辑运算真值表如表9-1所示。（见教材）

非线性变换 —异或问题

- 异或问题在二维空间线性不可分



x	y		C
0	**0**		**0**
1	**0**		**1**
0	**1**		**1**
1	**1**		**0**

感知器模型及其学习算法

如果“异或”（XOR）问题能用单层感知器解决，则由XOR的真值表，可知，w_1、 w_2 和θ 必须满足如下方程组：

$$w_1+w_2-\theta<0$$
$$w_1+0-\theta\geq 0$$
$$0+0-\theta<0$$
$$0+w_2-\theta\geq 0$$

显然，该方程组是无解，这就说明单层感知器是无法解决异或问题的。

感知器模型及其学习算法

- 异或问题是一个只有两个输入和一个输出，且输入输出都只取1和0两个值的问题，分析起来比较简单。对于比较复杂的多输入变量函数来说，到底有多少是线性可分的？多少是线性不可分的呢？相关研究表明，线性不可分函数的数量随着输入变量个数的增加而快速增加，甚至远远超过了线性可分函数的个数。
- 也就是说，单层感知器不能表达的问题的数量远远超过了它所能表达的问题的数量。
- 这也难怪当Minsky给出单层感知器的这一致命缺陷时，会使人工神经网络的研究跌入漫长的黑暗期。

delta法则

- 不要求训练数据线性可分
- 关键思想：梯度下降
- 遍历所有不同类型连续参数化假设空间
- 考虑：简单的线性单元（无阈值感知器）

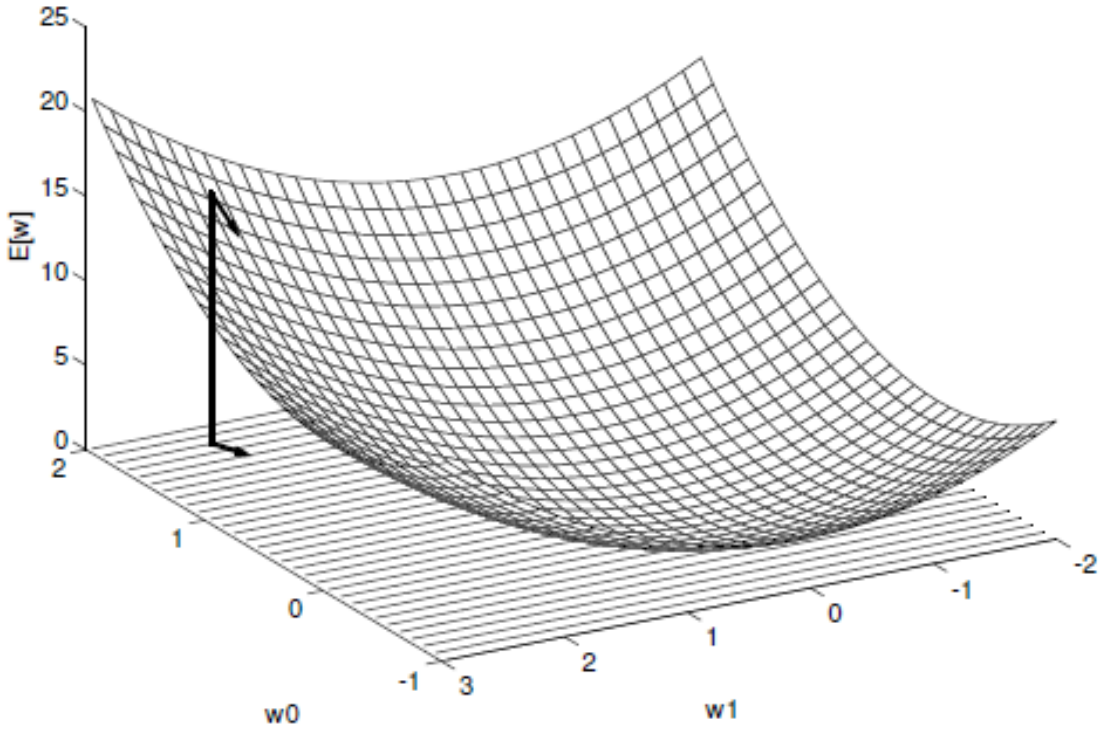
$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- 定义训练误差：
- D---训练样例集

delta法则

- 可视化假设空间



delta法则

- 梯度下降推导

方向导数

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

训练规则：

梯度下降

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

基本的BP算法

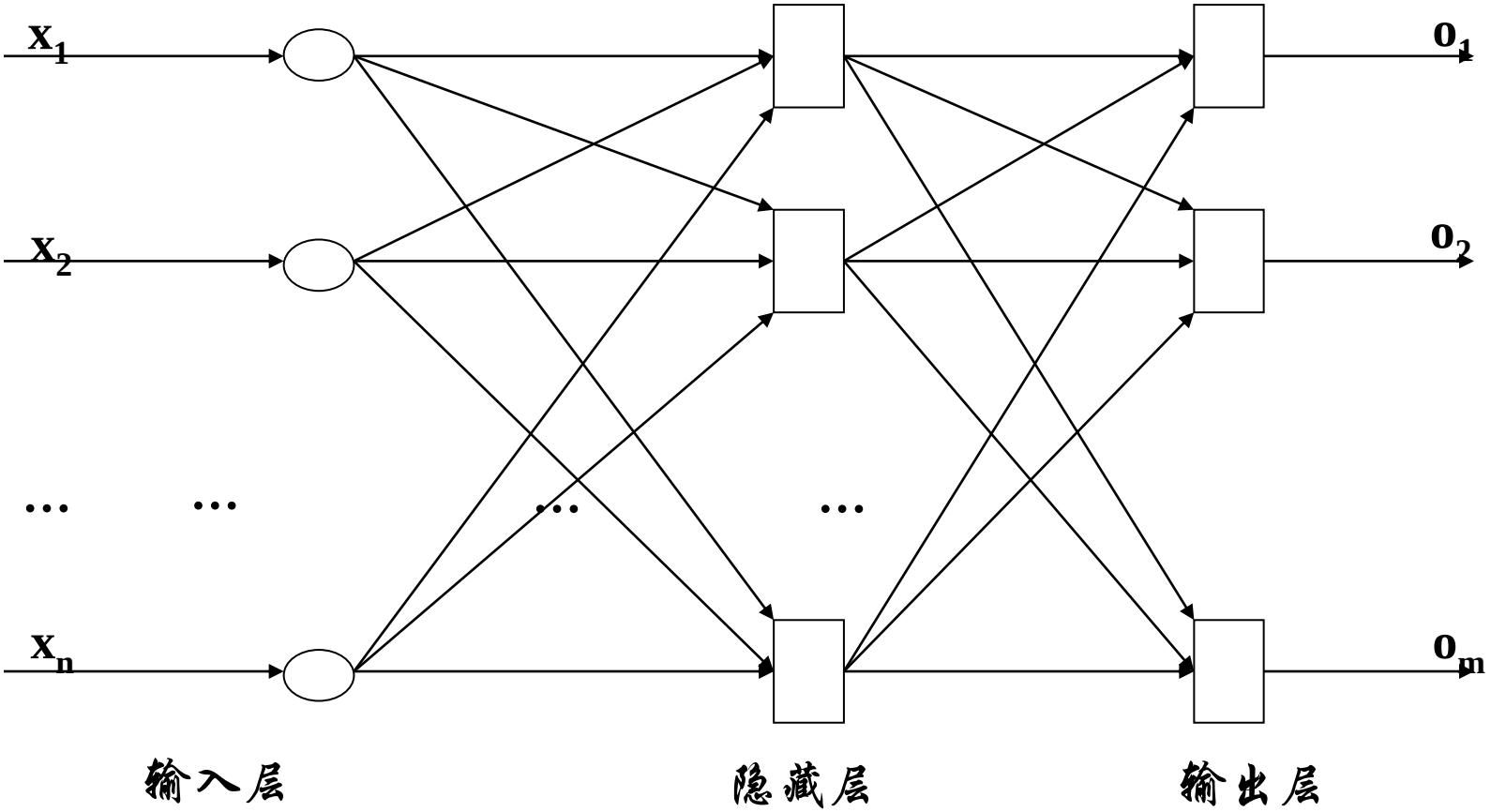
- 反向传播模型及其网络结构
  - 反向传播模型也称B-P模型，是一种用于前向多层的反向传播学习算法。之所以称它是一种学习方法，是因为用它可以对组成前向多层网络的各人工神经元之间的连接权值进行不断的修改，从而使该前向多层网络能够将输入它的信息变换成所期望的输出信息。
  - 之所以将其称作为反向学习算法，是因为在修改各人工神经元的连接权值时，所依据的是该网络的实际输出与其期望的输出之差，将这一差值反向一层一层的向回传播，来决定连接权值的修改。

基本的BP算法

- 样本集： $S=\{(X_1,Y_1),(X_2,Y_2),\ldots,(X_s,Y_s)\}$
- 基本思想：
  - 逐一地根据样本集中的样本(X_k,Y_k)计算出实际输出O_k和误差测度E_1，对$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$各做一次调整，重复这个循环，直到$\sum E_p<\varepsilon$。
  - 用输出层的误差调整输出层权矩阵，并用此误差估计输出层的直接前导层的误差，再用输出层前导层误差估计更前一层的误差。
  - 如此获得所有其它各层的误差估计，并用这些估计实现对权矩阵的修改。形成将输出端表现出的误差沿着与输入信号相反的方向逐级向输入端传递的过程

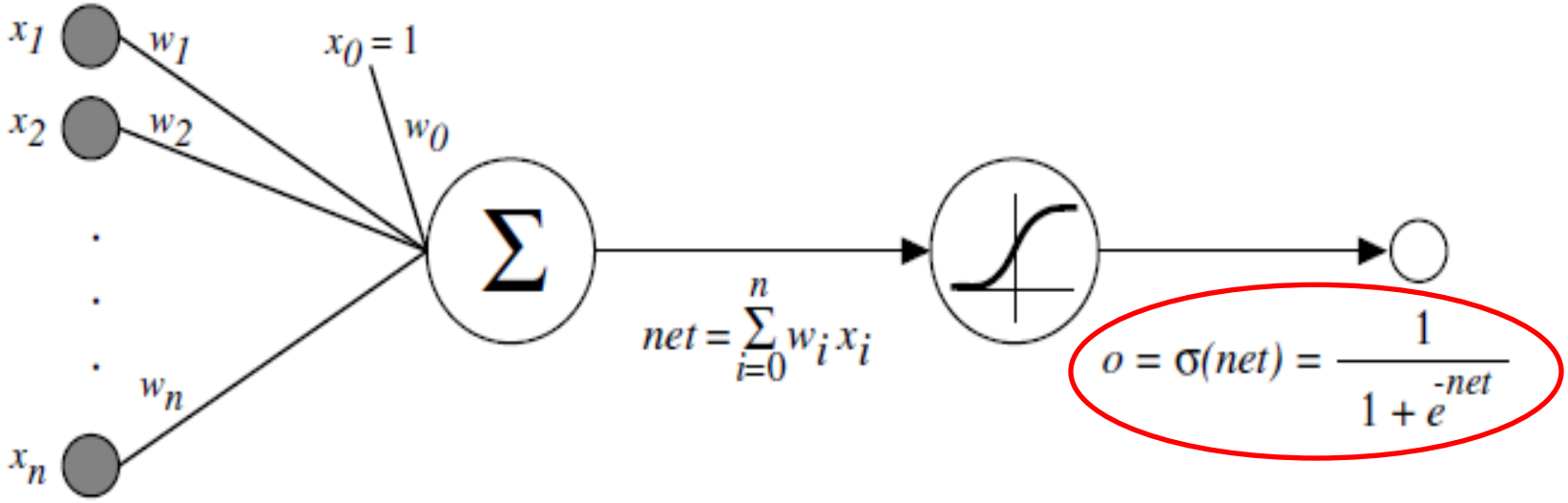
反向传播模型及其学习算法

- B-P算法的网络结构是一个前向多层网络，如图所示。



Sigmoid 单元

- 构建多层网络的单元基础：非线性、可微
- 选择Sigmoid单元



Sigmoid函数

反向传播模型及其学习算法

- B-P算法的学习过程如下：
  - （1）选择一组训练样例，每一个样例由输入信息和期望的输出结果两部分组成。
  - （2）从训练样例集中取一样例，把输入信息输入到网络中。
  - （3）分别计算经神经元处理后的各层节点的输出。
  - （4）计算网络的实际输出和期望输出的误差。
  - （5）从输出层反向计算到第一个隐层，并按照某种能使误差向减小方向发展的原则，调整网络中各神经元的连接权值。
  - （6）对训练样例集中的每一个样例重复（3）—（5）的步骤，直到对整个训练样例集的误差达到要求时为止。

反向传播模型及其学习算法

在以上的学习过程中，第（5）步是最重要的，如何确定一种调整连接权值的原则，使误差沿着减小的方向发展，是B-P学习算法必须解决的问题。其相关的讨论请参见教材。

反向传播模型及其学习算法

- B-P算法的优缺点：
  - 优点：理论基础牢固，推导过程严谨，物理概念清晰，通用性好等。所以，它是目前用来训练前向多层网络较好的算法。
  - 缺点：
    - （1）该学习算法的收敛速度慢；
    - （2）网络中隐节点个数的选取尚无理论上的指导；
    - （3）从数学角度看，B-P算法是一种*梯度最速下降法*，这就可能出现局部极小的问题。当出现局部极小时，从表面上看，误差符合要求，但这时所得到的解并不一定是问题的真正解。
    - 所以B-P算法是不完备的。

反向传播模型及其学习算法

反向传播计算的举例

设图是一个简单的前向传播网络，用B-P算法确定其中的各连接权值时，δ的计算方法如下：

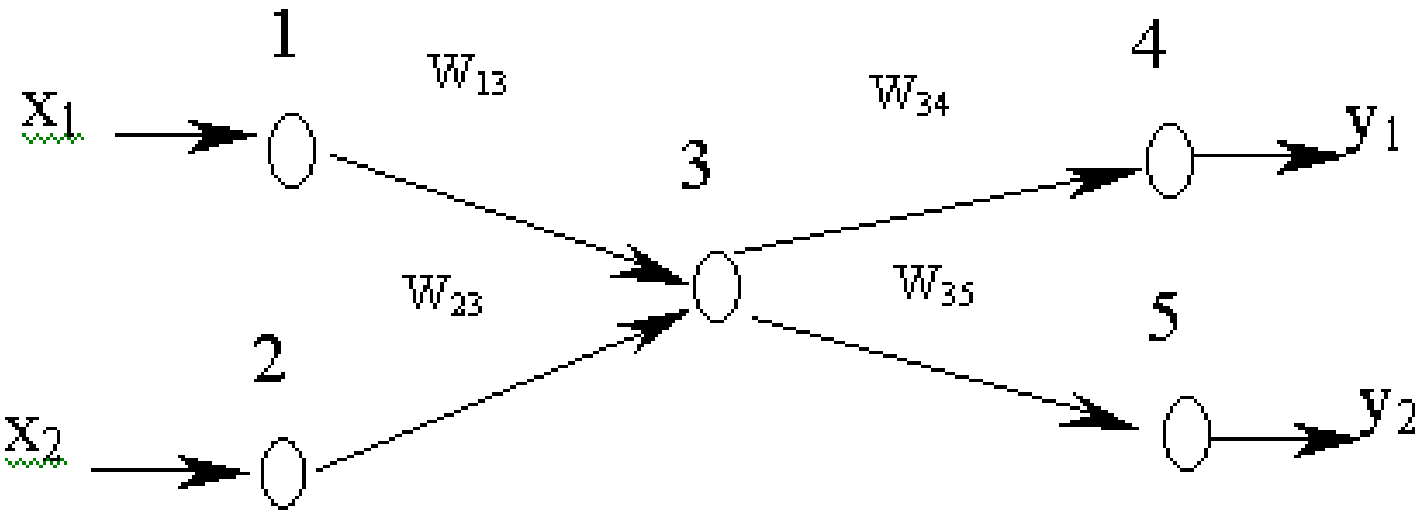


图 9.12　一个简单前向传播网络

反向传播模型及其学习算法

$$I_3 = W_{13}x_1 + W_{23}x_2 \qquad O_3 = f(I_3)$$

$$I_4 = W_{34}O_3 \qquad O_4 = y_1 = f(I_4)$$

$$I_5 = W_{35}O_3 \qquad O_5 = y_2 = f(I_5)$$

$$e = \frac{1}{2}\left[(y_1' - y_1)^2 + (y_2' - y_2)^2\right]$$

反向传播模型及其学习算法

反向传输时计算如下：

（1）计算 $\frac{\partial e}{\partial W}$

$$\frac{\partial e}{\partial W_{13}} = \frac{\partial e}{\partial I_3} \cdot \frac{\partial I_3}{\partial W_{13}} = \frac{\partial e}{\partial I_3} x_1 = \delta_3 x_1$$

$$\frac{\partial e}{\partial W_{23}} = \frac{\partial e}{\partial I_3} \cdot \frac{\partial I_3}{\partial W_{23}} = \frac{\partial e}{\partial I_3} x_2 = \delta_3 x_2$$

$$\frac{\partial e}{\partial W_{34}} = \frac{\partial e}{\partial I_4} \cdot \frac{\partial I_4}{\partial W_{34}} = \frac{\partial e}{\partial I_4} O_3 = \delta_4 O_3$$

$$\frac{\partial e}{\partial W_{35}} = \frac{\partial e}{\partial I_5} \cdot \frac{\partial I_5}{\partial W_{35}} = \frac{\partial e}{\partial I_5} O_3 = \delta_5 O_3$$

反向传播模型及其学习算法

（2）计算δ

$$\delta_4 = \frac{\partial e}{\partial I_4} = (y_1 - y_1')f'(I_4)$$

$$\delta_5 = \frac{\partial e}{\partial I_5} = (y_2 - y_2')f'(I_5)$$

$$\delta_3 = (\delta_4 W_{34} + \delta_5 W_{35})f'(I_3)$$

也就是说，δ_3的计算要依赖于与它相邻的上层节点的δ_4和δ_5的计算。

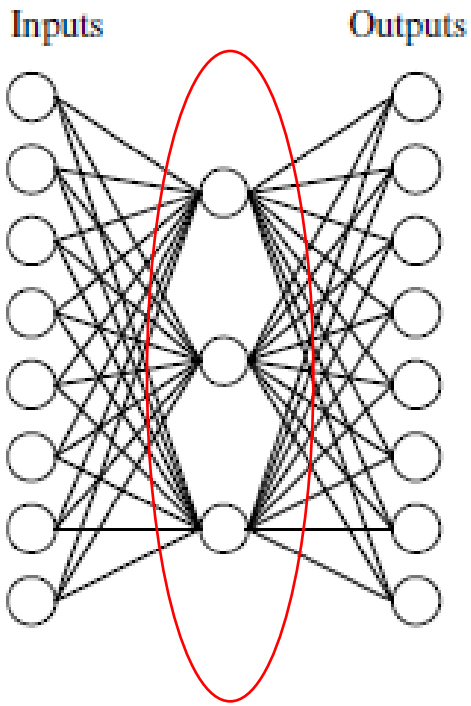
前馈网络的表征能力

- 布尔函数：任何布尔函数可以用两层网络准确表示。**隐层单元数随输入数增加呈指数增长。**
- 连续函数：每个有界连续函数可以用两层网络以任意小的误差逼近。**隐层使用Sigmoid函数。**
- 任意函数：任意函数可以用三层网络以任意精度逼近。**输出层线性单元，隐层Sigmoid单元。**

假设空间搜索

- 反向传播算法假设空间：n个网络权值的n维欧氏空间（空间连续）

隐层表达



Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

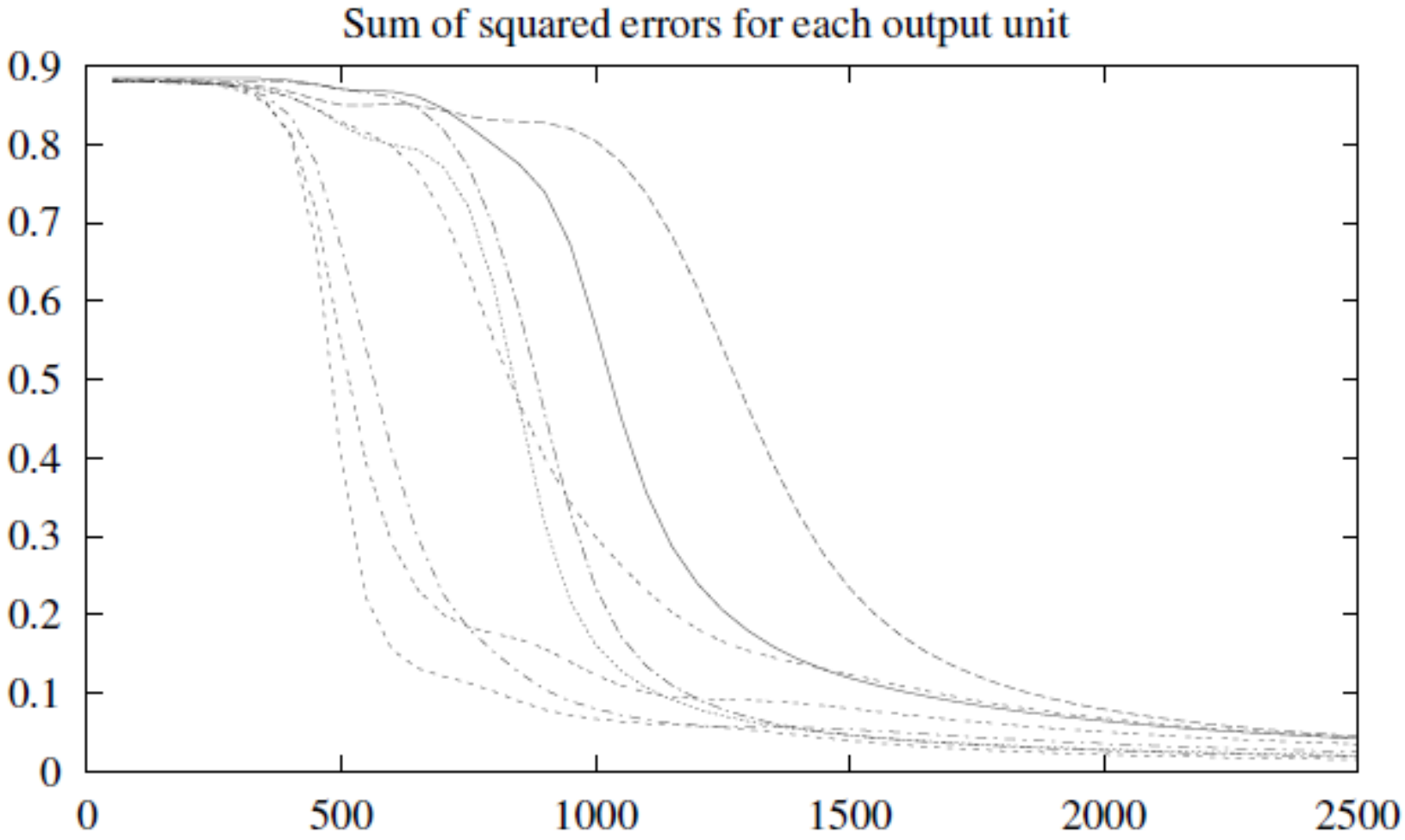
可以学到什么？

隐层表达

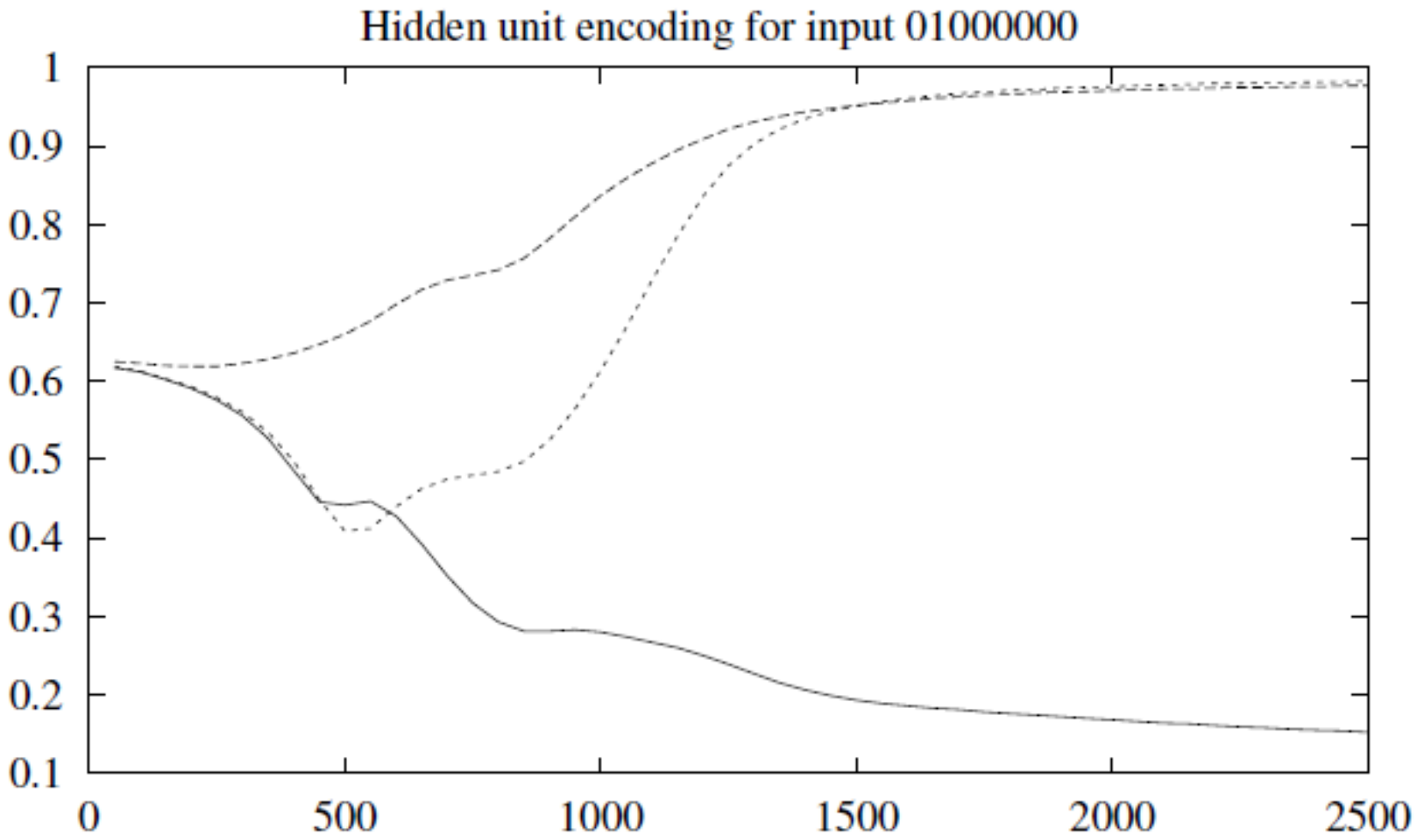
- 学到的隐层表达：

Input		Hidden Values				Output	
10000000	→	.89	.04	.08	→	10000000	100
01000000	→	.01	.11	.88	→	01000000	001
00100000	→	.01	.97	.27	→	00100000	010
00010000	→	.99	.97	.71	→	00010000	111
00001000	→	.03	.05	.02	→	00001000	000
00000100	→	.22	.99	.99	→	00000100	011
00000010	→	.80	.01	.98	→	00000010	101
00000001	→	.60	.94	.01	→	00000001	

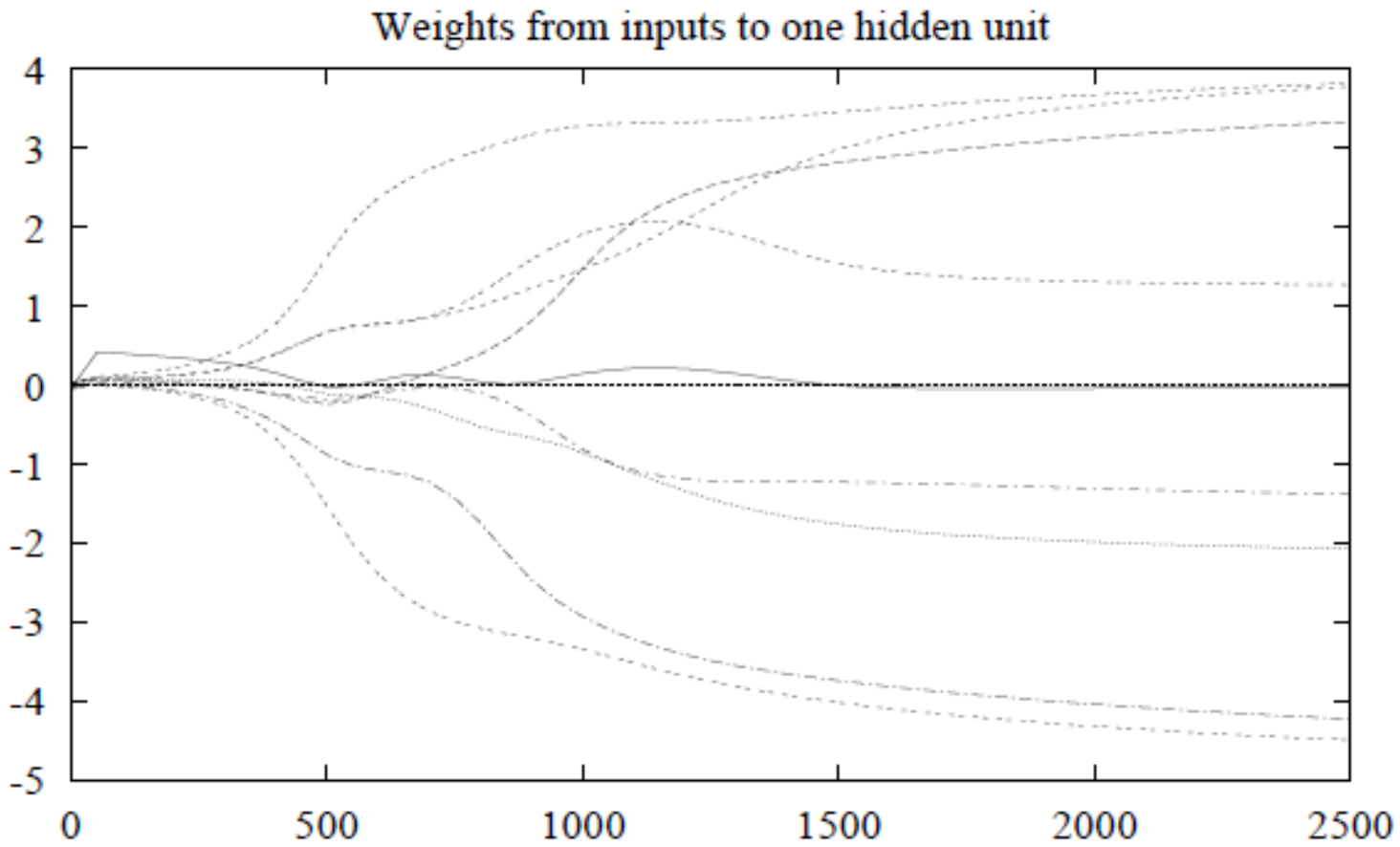
训练


Sum of squared errors for each output unit
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500

训练


Hidden unit encoding for input 01000000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500

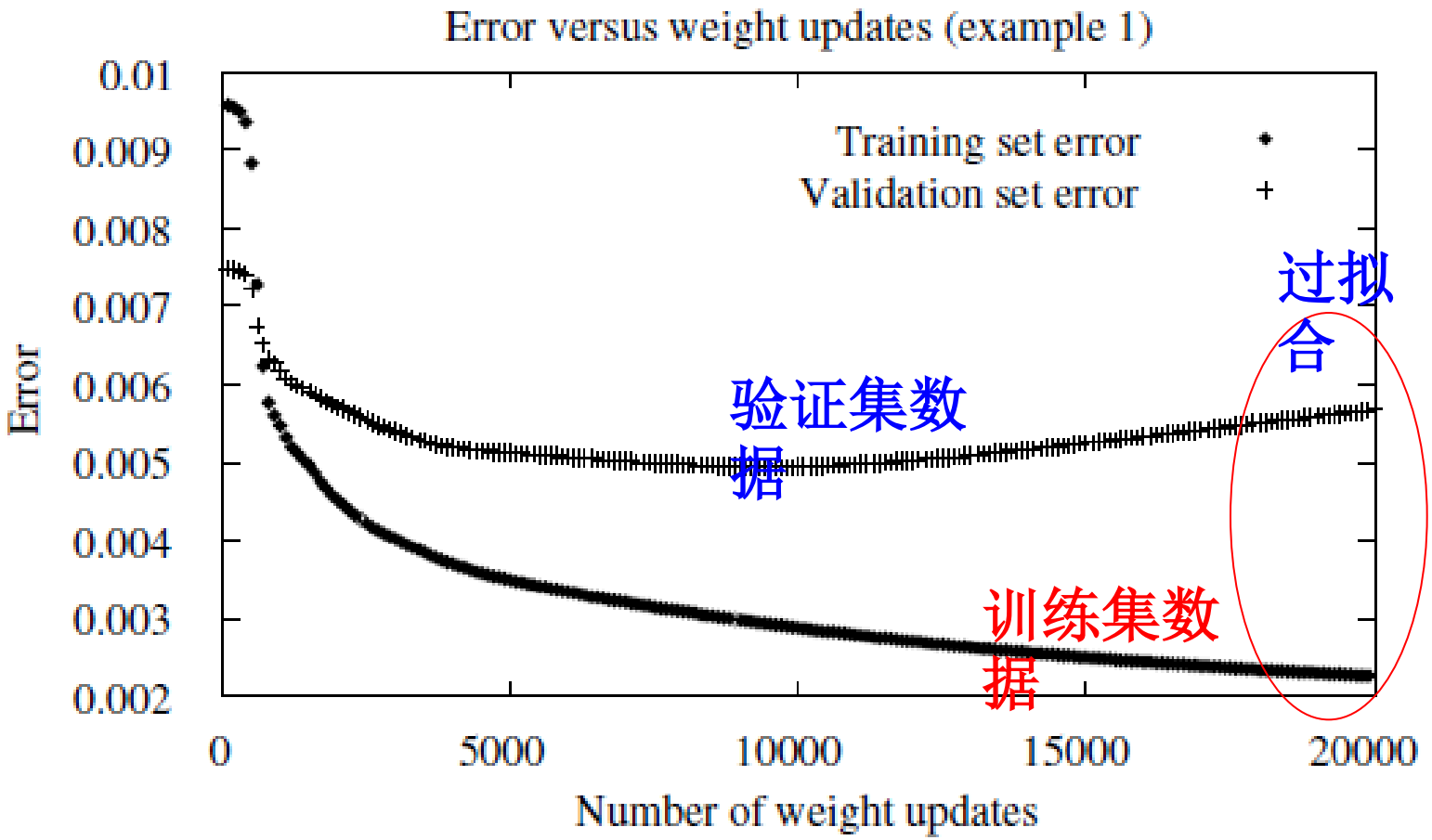
训练


Weights from inputs to one hidden unit
4
3
2
1
0
-1
-2
-3
-4
-5
0
500
1000
1500
2000
2500

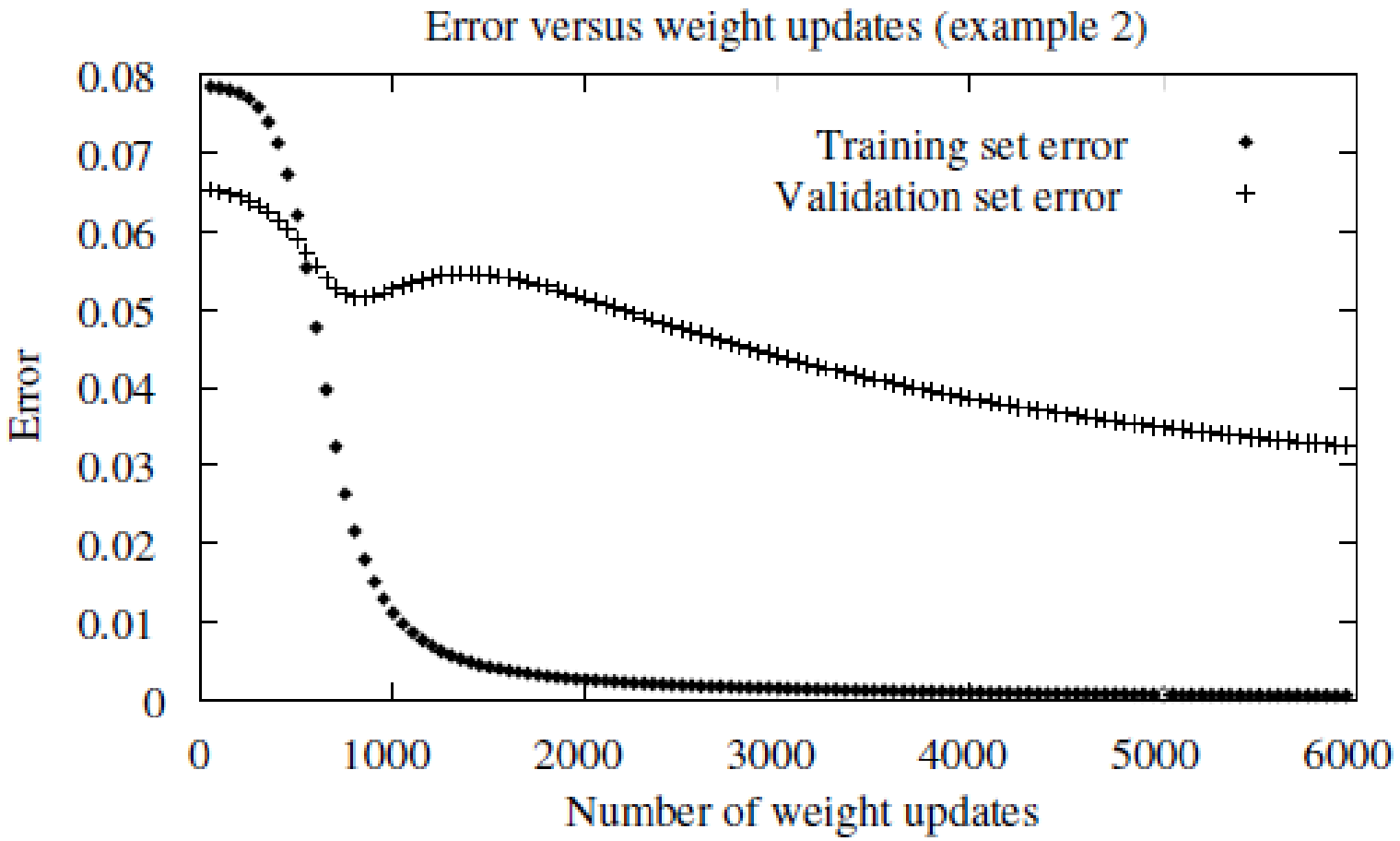
过拟合与欠拟合

- 过拟合（overfitting）：隐含层节点数目越大，网络学习能力越强，但不能保证预测能力好。
- 欠拟合（underfitting）：隐含层节点过少，网络不能构建复杂决策面，网络学习能力低。

过拟合


Error versus weight updates (example 1)
Training set error
Validation set error
过拟合
验证集数据
训练集数据
Error
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0
5000
10000
15000
20000
Number of weight updates

解决过拟合---交叉验证


Error versus weight updates (example 2)
Training set error
Validation set error
Error
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
1000
2000
3000
4000
5000
6000
Number of weight updates

K-fold方法

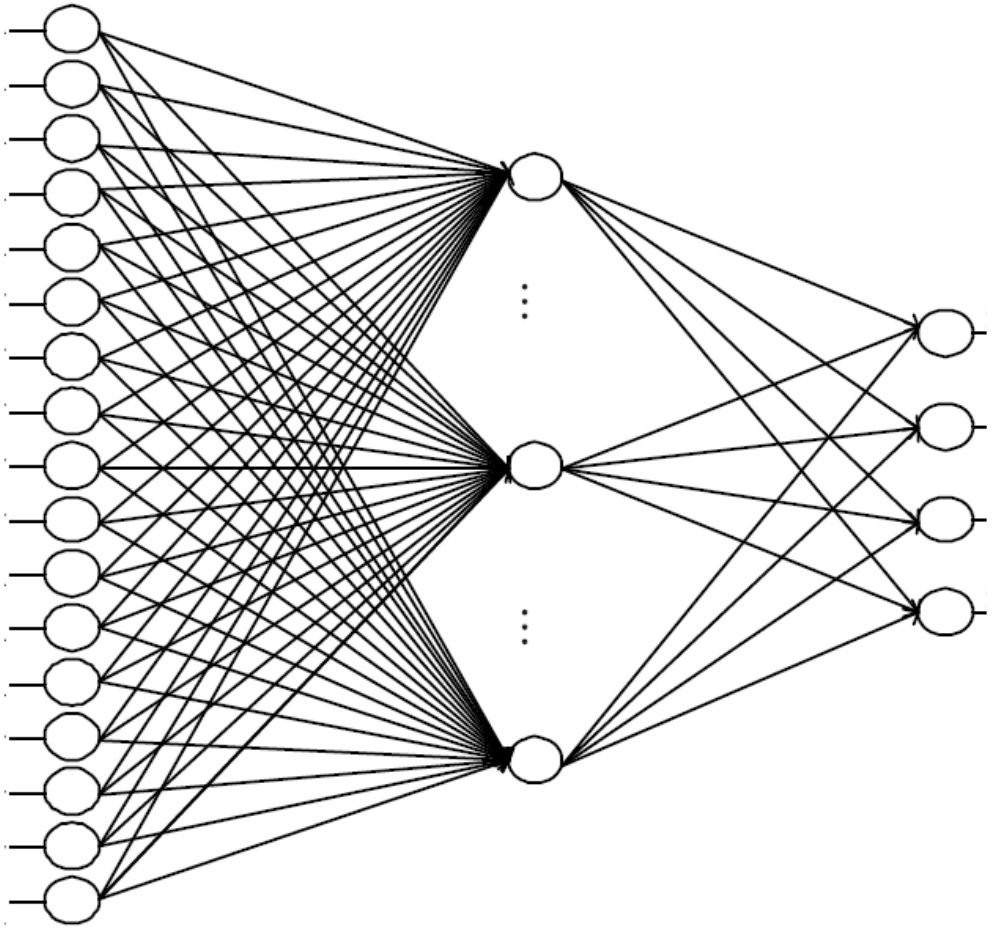
- 分m个样例为k个不相交子集，每个子集样例数为：m/k
- K次交叉验证过程：每次使用不同子集作为验证集合，合并其余（k-1）个子集为训练集合
- 使用验证集与训练集结果没有过拟合出现的最佳迭带次数i
- 计算i的均值 $\bar{i}$，训练m实例迭带 $\bar{i}$ 次

3.2.3 径向基函数网络

- 径向基函数：radial basis function, RBF
- 只有一个隐层，隐层单元采用径向基函数。隐层把原始的非线性可分的特征空间变换到另一个空间（通常是高维空间），使之可以线性可分
- 输出为隐层的线性加权求和。采用基函数的加权和来实现对函数的逼近
- 径向基函数：径向对称的标量函数$k(\|\mathbf{x}-\mathbf{x}_c\|)$，最常用的RBF是高斯核函数

$$k(\|\mathbf{x}-\mathbf{x}_c\|)=\exp\left(-\frac{(\mathbf{x}-\mathbf{x}_c)^T(\mathbf{x}-\mathbf{x}_c)}{2\sigma^2}\right)$$

➢ 径向基函数网络结构



与BP网络比较：

- RBF网络的输出是隐单元输出的线性加权和，学习速度加快。
- BP网络使用sigmoid()函数作为激活函数，这样使得神经元有很大的输入可见区域。
- 径向基神经网络使用径向基函数（一般使用高斯函数）作为激活函数，神经元输入空间区域很小，因此需要更多的径向基神经元。

（1）把网络可以看成f(x)的逼近器，任何函数都可以表示成一组基函数的加权和，相当于隐层单元的输出函数构成一组基函数来逼近

（2）在RB网络中基函数相当于非线性映射，参数分别为中心、方差、输出单元的权值

（3）参数可以通过梯度下降来计算

思考题：

- BP算法是否完美？
- 如果模型变得更新-深度学习，数据量大，BP算法改进