第6章第1讲

人工神经网络

Artificial Neural Networks

向 世 明

smxiang@nlpr.ia.ac.cn

people.ucas.ac.cn/~xiangshiming

时空数据分析与学习课题组(STDAL)

中科院自动化研究所 模式识别国家重点实验室

助教： 张明亮(zhangmingliang2018@ia.ac.cn)

程真(chengzhen2019@nlpr.ia.ac.cn)

张姣(zhangjiao2019@ia.ac.cn)

内容提要

- 介绍
  - 发展历史
  - 网络结构
- 基本模型
  - 单层感知器、多层感知器、RBF网络
- 扩展模型
  - Hopfield网络、RBM、DNN、CNN、Autoencoder、RNN、LTSM等

第一节

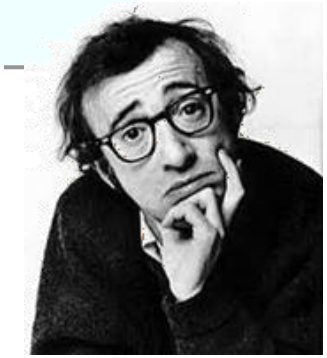
人工神经网络发展历程

6.1.1 概述

- 人脑

人是地球上具有最高智慧的动物，而人的指挥均来自大脑，人类靠大脑进行思考、联想、记忆和推理判断等，这些功能是任何被称为“电脑”的一般计算机所无法取代的。

“My brain: It's my second favorite organ.”



伍迪·艾伦 **(Woody Allen)**

6.1.1 概述

- 人脑

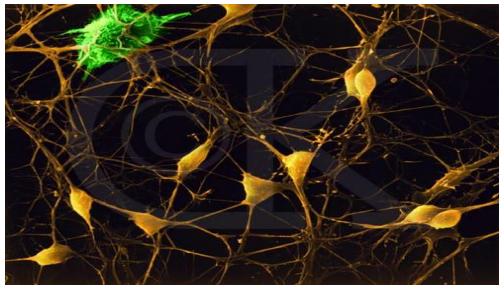
长期以来，科学家一直致力于人脑内部结构和功能的探讨和研究，并试图建立模仿人类大脑的计算机。

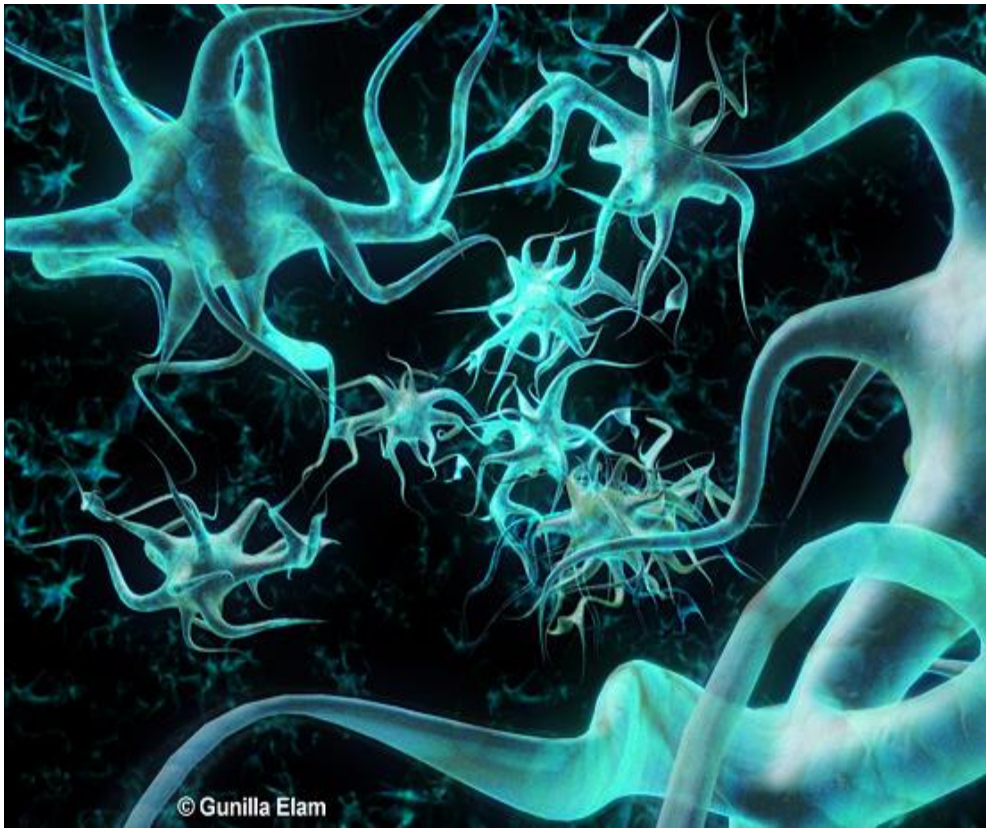
虽然到目前对大脑的内部工作机理还不甚完全清楚，但对其结构已有所了解。

6.1.1 概述

- 人脑

大脑里有许许多多（约800亿）神经细胞。每个神经细胞由两个部分组成：细胞体和突起





6.1.1 概述

- 人脑

每个神经元可看作是一个小的处理单元，这些神经元按某种方式连接起来，形成大脑内部的生理神经元网络。

神经元网络中各神经元之间联结的强弱，按外部的激励信号做自适应变化，而每个神经元又随着所接收到的多个接收信号的综合大小而呈现兴奋或抑制状态。

现已明确大脑的学习过程就是神经元之间连接强度随外部激励信息做自适应变化的过程，而大脑处理信息的结果则由神经元的状态表现出来。

6.1.2 什么是人工神经网络

- **Biological Inspiration**

- 神经元之间通过突触两两相连。
- 轴突记录了神经元间联系的强弱。
- 只有达到一定的兴奋程度，神经元才向外界传输信息。
- 每个神经元可抽象成一个激励函数（非线性处理）。

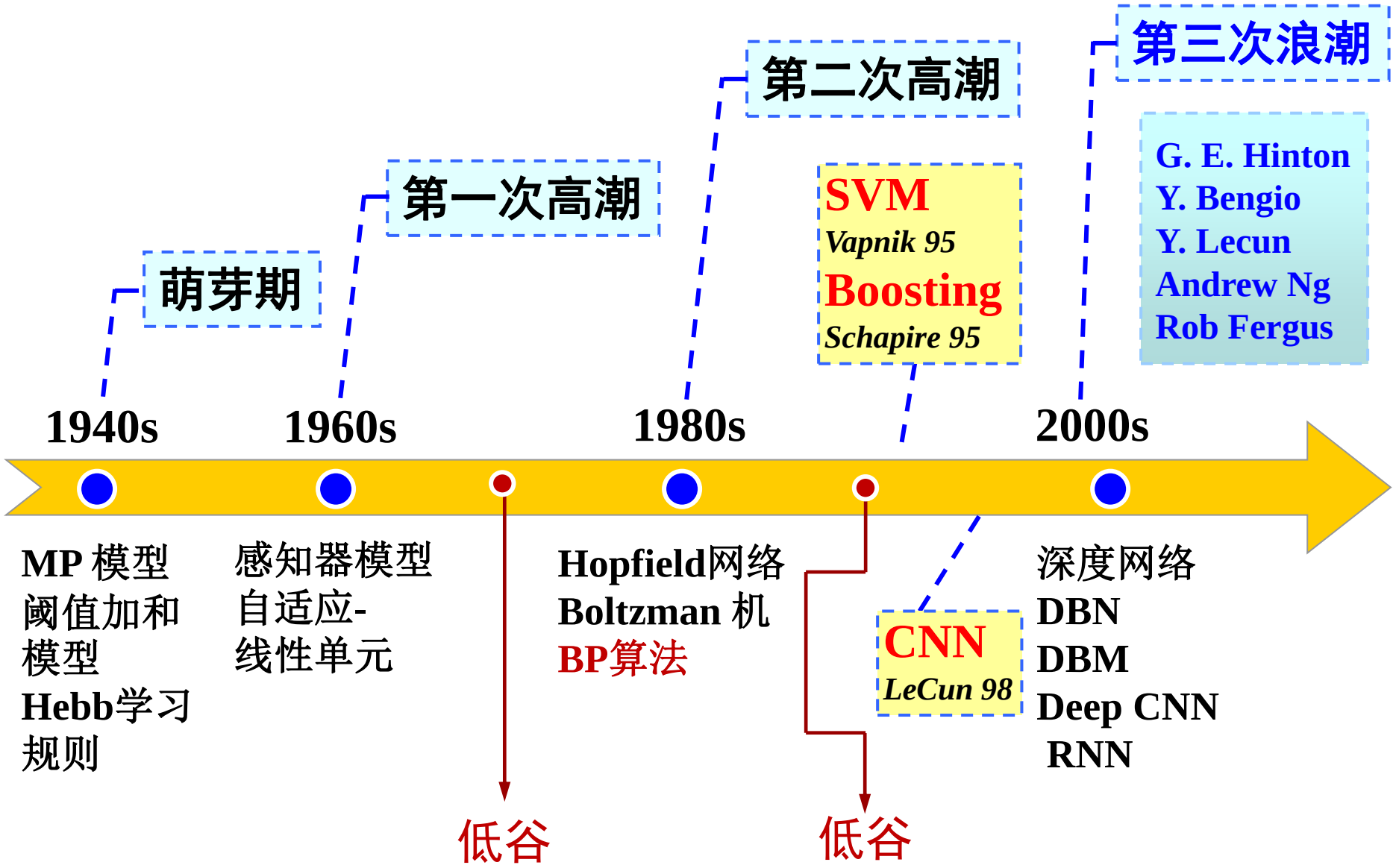
让计算机更加鲁棒、更加智能、且具有学习功能!

6.1.2 什么是人工神经网络

- **按美国神经网络学者Nielsen的定义:**
  - 人工神经网络是一个并行、分布式处理结构，由处理单元及其称为联接的无向通道互连而成。
  - 这些处理单元具有局部内存，可以完成局部操作，该操作由输入至该单元的信号值和存储在该单元中的信号值来确定。
  - 每个处理单元有一个单一的输出联接，输出信号可以是任何需要的数学模型。

是一种**模仿生物神经网络的结构和功能的数学模型或计算模型**

6.1.3 人工神经网络发展历程

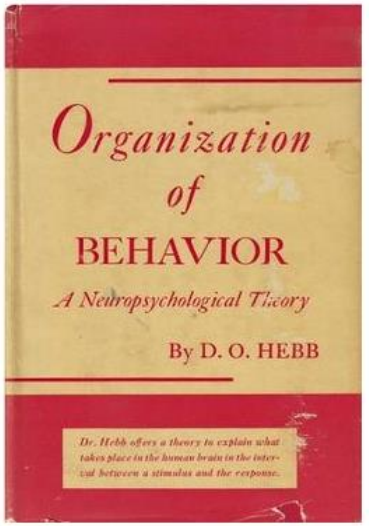


6.1.3 人工神经网络发展历程

- 萌芽期
  - **1943年**，心理学家 Warren. S. McCulloch 和数理逻辑学家 Walter Pitts 建立了神经网络和数学模型，称为 **MP (Machine Perception) 模型**：
    - 是一篇关于神经元如何工作的开拓性文章：“**A Logical Calculus of Ideas Immanent in Nervous Activity**”。
      - 指出：脑细胞的活动像**断/通开关**，脑细胞可以按各种方式相互结合，进行各种逻辑运算。
      - **用电路实现了简单的神经网络模型**，并预言大脑的所有活动最终将被解释清楚。
    - 虽然问题并非如此简单，但它给人们一个信念：**即大脑的活动是靠脑细胞的组合连接实现的**。

6.1.3 人工神经网络发展历程

- 萌芽期
  - **1949年，心理学家 Donala Hebb 写了一本书：“The Organization of Behavior”。**
    - 在该书中，他强调了心理学和生理学之间的联系和沟通，**指出脑细胞间的通路在参与某种活动时将被加强**，这就是后来的**Hebb学习规则**。
    - **提出了突触联系强度可变的设想。**
    - 目前有些神经网络模型仍然采用这种学习规则。



6.1.3 人工神经网络发展历程

- 萌芽期
  - 二十世纪50年代，随着计算机的发展和软硬件的进步，开始对部分神经系统功能的理论进行模拟，拓宽了研究的路子。
    - IBM的研究室在Hebb工作的基础上，对神经网络的模型进行了软件模拟。虽然开始时失败了，但在**“使得模型像人那样适应环境”**的实验上取得了一定程度的成功。
    - 在此情况下，人们开始酝酿**人工智能的项目**。

6.1.3 人工神经网络发展历程

- 萌芽期
  - 在1956 年，美国的明斯基(M. Minsky, 2016去世)、西蒙(H.Simon)以及麦卡锡(J. Mccarthy)等人在“达特茅斯夏季人工智能研究项目”**(Dartmouth Summer Research Project on Artificial Intelligence)**首次提出了人工智能的概念，使人工智能成为计算机科学的一个分支。
  - 人们提出两条研究思路：
    - **采用高级人工智能方法，试图建立描述智能机功能的计算机程序**；
    - 根据大脑对初级信号（比如视觉信号）处理方式构建成结构模型，以实现智能化。（使一部机器的瓜方式就象是一个在行动时所依据的智能）
  - **宣告了人工神经网络的诞生！**

6.1.3 人工神经网络发展历程

- 第一次高潮
  - **1957年**，计算机专家 **Frank Rosenblatt** 开始从事感知器研究，并制成硬件，被认为是最早的神经网络模型。
  - **1959年**，两位电机工程师 Bernard Widrow 和 Marcian Haff 开发出一种被称为自适应线性单元（ADALINE）的网络模型，并在其论文“Adaptive Switching Circuits”中描述了该模型和学习算法**（Widrow- Haff算法， 序列最小平方误差方法，见5章）**。
    - 该网络通过训练，可以成功用于抵消通信中的回波和噪声，也可用于天气预报，成为第一个用于实际问题的神经网络。

6.1.3 人工神经网络发展历程

- 第一次高潮
  - **1962年，Frank Rosenblatt** 出版了**一本书“The Principles of Neurodynamics”**，详述了感知器模型。
    - 该感知器具有**输入层、输出层和中间层**，通过实验可以模仿人的某些特性，并断言它可以学会任何它可以表示的功能。
  - **1967年**，Stephen Grossberg 通过对生理学的研究，开发了一种称作 Avalanche（**雪崩网**）的神经网络模型，可以执行连续语音识别和控制机器人手臂的运动。

6.1.3 人工神经网络发展历程

- 第一次高潮
  - 在这一时期，由于感知器的某些进展和对神经网络的宣传，**人们乐观地认为几乎已经找到了实现智能的关键**，许多部门开始大批地投入此项研究，希望尽快占领制高点，形成了研究人工神经网络的**第一次高潮**。
  - 由于当时对神经网络的乐观情绪的影响，**人们夸大了神经网络的潜力**（有人甚至担心制造机器人的人类会很快受到机器人的攻击）。

6.1.3 人工神经网络发展历程

- 反思期
  - M. Minsky 等仔细分析了以感知器为代表的神经网络系统的功能及局限后，于1969年出版了《Perceptron》一书，指出感知器不能解决高阶谓词问题，不能解决**异或**问题，**将Rosenblatt的感知器拉下神坛**。
  - 他们的论点极大地影响了神经网络的研究，加之当时**串行计算机和人工智能**所取得的成就，**掩盖了发展新型计算机和人工智能新途径的必要性和迫切性**，使人工神经网络的研究处于**低潮**。
  - 在此期间，仍有一些有影响力的工作：自适应谐振理论（ART网）、**自组织映射**、认知机网络，以及有关神经网络数学理论的研究。

6.1.3 人工神经网络发展历程

- 反思期
  - 二十世纪70年代到80年代早期，仍有一些坚信神经网络的人坚持研究工作。比如：
    - 神经生理学家 James Anderson 开发的盒中脑模型（Brain-State-in-a-Box，BSB）。
    - 日本学者 Kunihik Fukushima 开发的用于视觉图形识别的认知器模型（Neocognitron）。
    - 电气工程师 Teuvo Kohonen 开发的与BSB类似的网络模型。
    - 以及 Grossberg，Rumelhart，McClelland，Marr，Amari和Cooper 等人的工作。

6.1.3 人工神经网络发展历程

- 第二次高潮
  - **1982年**，John Hopfield **向美国科学院递交了有关神经网络的报告**，主要内容就是建议收集和重视以前对神经网络的工作，其中特别强调了每种模型的实用性。
  - 根据对神经网络的数学分析和深入理解， Hopfield 揭示了以往的网络是如何工作的，可以做些什么，**并提出了他自己的模型**，能从失真的或不完善的数据图像中获得完整的数据图像，引起了美国军方的兴趣。
  - 当时，人工智能对自动制导车的研究失败，而利用神经网络有可能解决这个问题，使人们的注意力重新投向人工神经网络，导致了人工神经网络的**第二次高潮**。

6.1.3 人工神经网络发展历程

- 第二次高潮
  - J. Hopfield提出**Hopfield网络** (1984)
  - Hinton、 Sejnowsky、 Rumelhart等人提出了著名的**Boltzmann机** (1985)
  - Rumelhart等提出多层网络的学习算法—**BP算法** (1986)
    - 1970s – efficient error backpropagation, Linnainmaa

6.1.3 人工神经网络发展历程

- 第二次反思期(再认识与应用研究)
  - 二十世纪90年代后，神经网络的研究趋于平缓：
    - **应用领域不广、结果不够精确、存在可信度等问题**
  - **主要研究内容**
    - 开发现有模型的应用，改进模型和算法，提高训练速度和准确度。
    - 期望**在理论上**寻找新的突破，建立新的专用或通用模型和算法。
    - 进一步对生物神经系统进行研究，**不断丰富对人脑的认识**

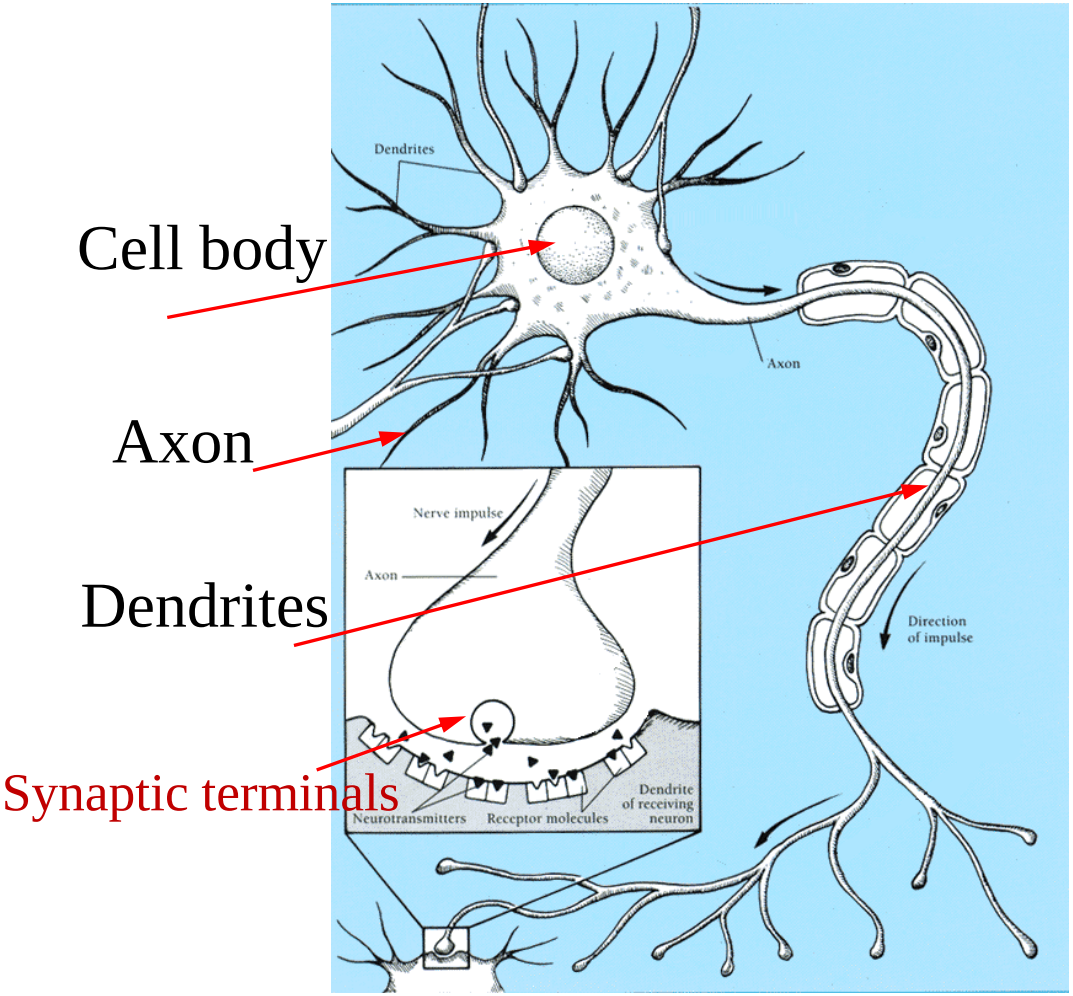
第二节

人工神经网络基础

6.2.1 神经元

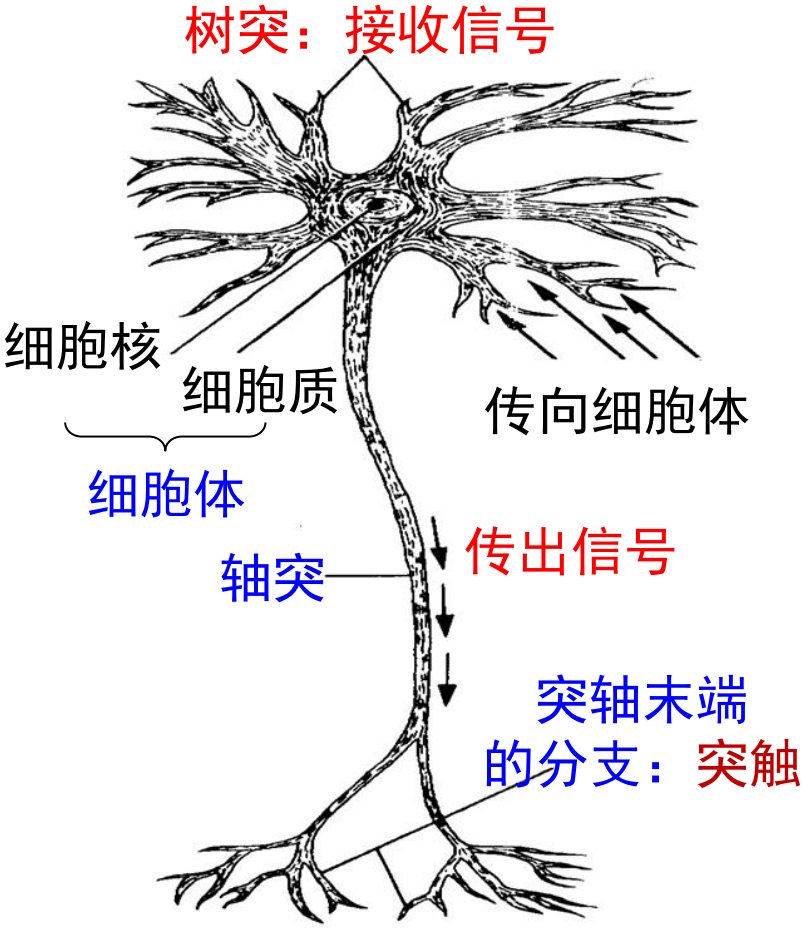
- 生物学原型

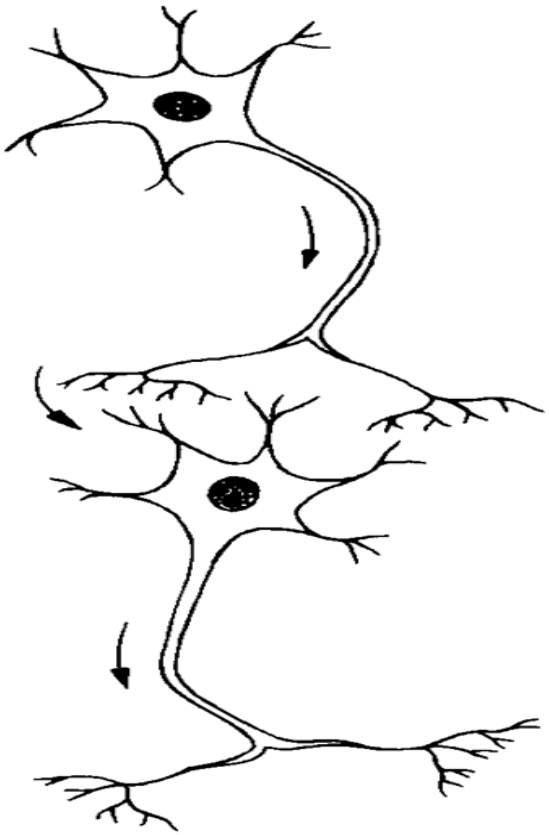
神经细胞由细胞体和突起组成。突起分为两种：短的叫树突，从其它神经细胞接收信号传递给 细胞体的；长的叫轴突，将信号传递给其它神经细胞的树突的。



神经元结构

- ✓ 神经元之间通过突触两两相连。
- ✓ 轴突记录了神经元间联系的强弱。
- ✓ 只有达到一定的兴奋程度，神经元才向外界传输信息。
- ✓ 每个神经元可抽象成一个激励函数（非线性处理）。





神经元间的信号**通过突触传递。**

6.2.1 神经元

细胞体：它是神经元的本体，内有细胞核和细胞质，完成普通细胞的生存功能。

树突：它有大量的分枝，多达10^3 数量级，长度较短（通常不超过1毫米），用以**接收**来自其它神经元的信号。

轴突：它用以**输出**信号，有些较长（可达1米以上），轴突的远端也有分枝，可与多个神经元相连。

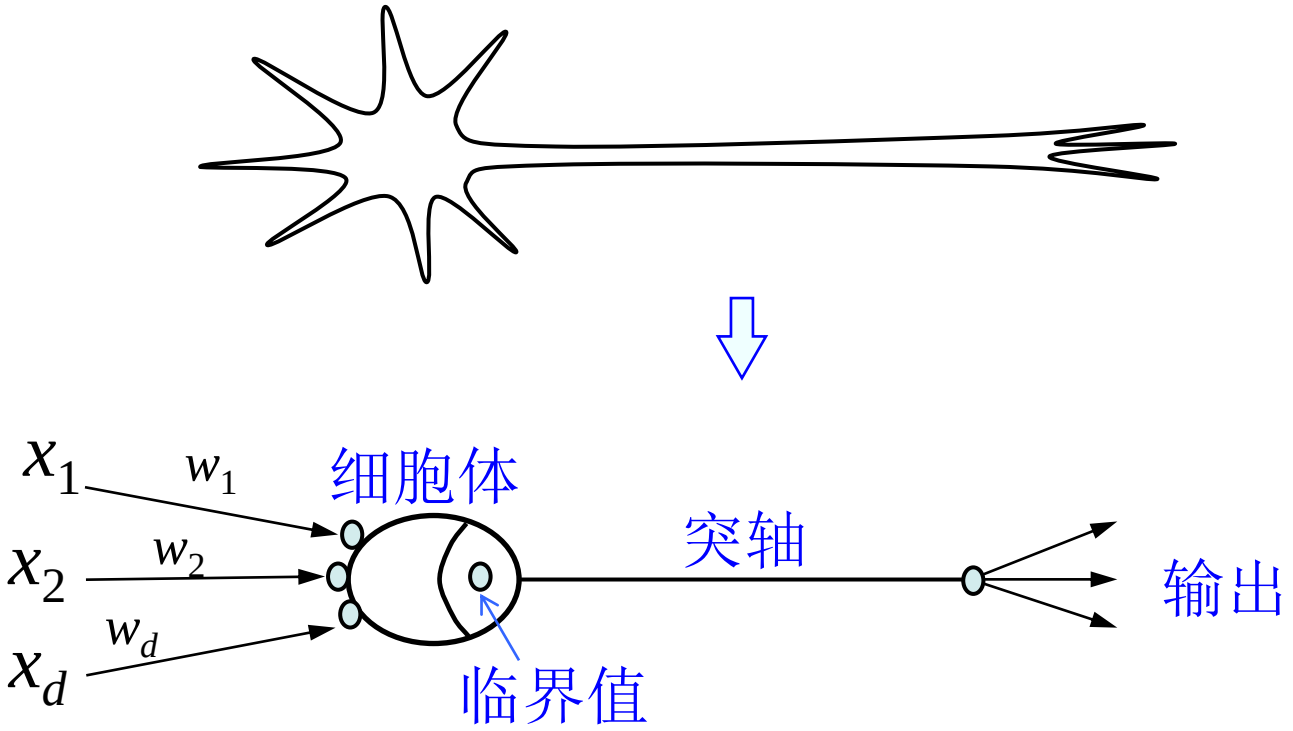
突触：它是一个神经元与另一个神经元相联系的特殊部位，通常是一个神经元轴突的端部靠化学接触或电接触将信号传递给下一个神经元的树突或细胞体。

6.2.1 神经元

- 神经元的基本工作机制
  - 一个神经元有**两种状态—兴奋和抑制**
  - 平时处于抑制状态的神经元，当接收到其它神经元经由突触传来的冲击信号时，多个输入在神经元中以代数和的方式叠加。
    - **进入突触的信号会被加权**，起兴奋作用的信号为正，起抑制作用的信号为负。
  - 如果叠加总量超过某个阈值，神经元就会被激发进入兴奋状态，发出输出脉冲，并由轴突的**突触传递给其它神经元**。

6.2.2 人工神经元

- 人工神经网络处理单元



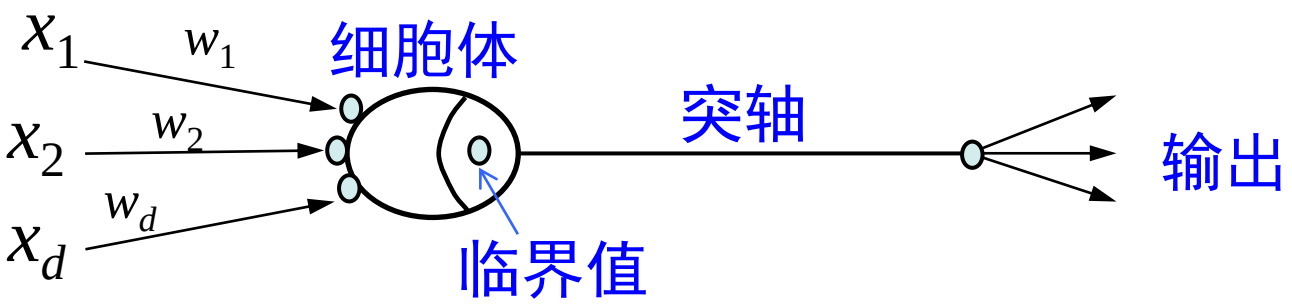
6.2.2 人工神经元

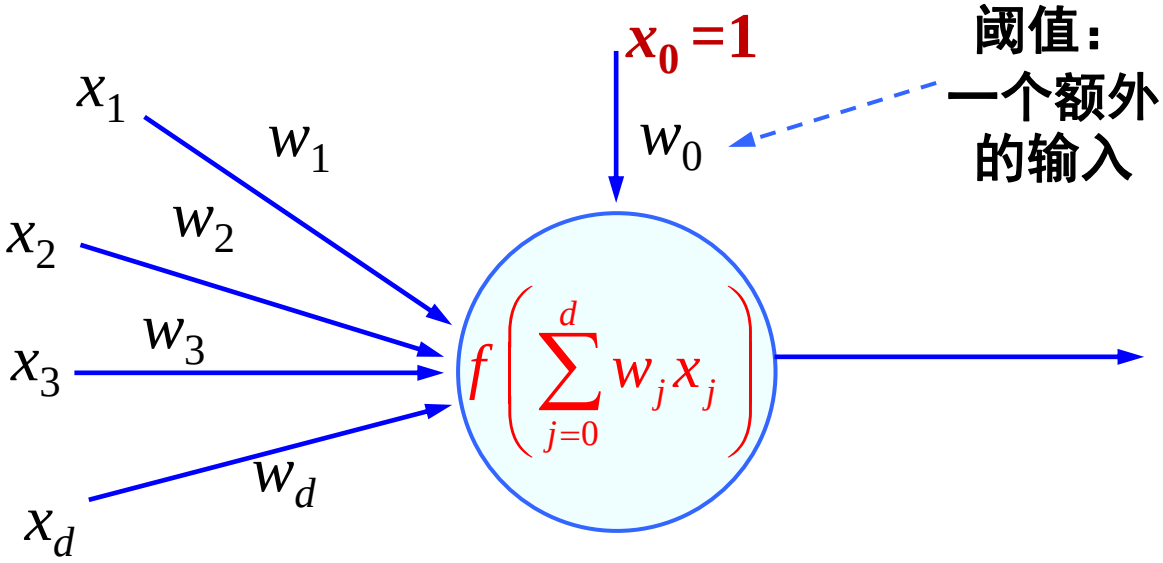
- 对生物神经元的模拟
  - 与生物神经元一样，处理单元**也有很多输入信号**（激励），并同时叠加到处理单元上。
  - 处理单元**以输出作为响应**。输出响应不但受输入信号的影响，同时也受内部其它因素的影响：
    - 内部因素：**内部阈值或一个额外输入（称为偏置项）**
  - **处理单元的每一个输入都经过相关的加权**，以影响输入的激励作用：
    - 类似于生物神经元中**突触的可变强度**，它确定了输入信号的强度，一般把它看作连接强度的测度。
    - 类似于生物神经元中的突触强度可受外界因素影响，权重可以调节。

6.2.2 人工神经元

- 处理单元的功能
  - 对每个输入信号进行加权处理（确定其强度）；
  - 确定所有输入信号的组合效果（求和）；
  - 确定其输出（转移特性，即激励特性）。

- 处理单元的基本结构和功能





三功能
加权
求和
激励

x: 收到信号　**w**: 信号的权重　$f(.)$: 激励函数

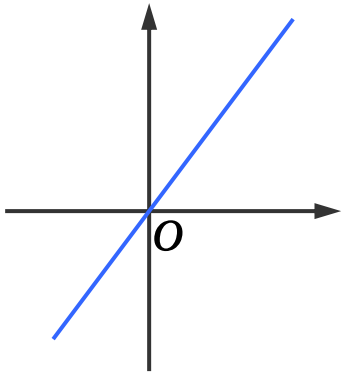
6.2.2 人工神经元

- 激活函数
  - 激活函数$f(\bullet)$，也称激励函数、转移函数、传输函数或限幅函数，其作用是将可能的无限域变换到指定的有限范围内进行输出（类似于生物神经元的非线性转移特性）。
  - **常用的**激活函数：
    - 线性函数、斜坡函数、阶跃函数、符号函数、Sigmoid函数、双曲正切函数

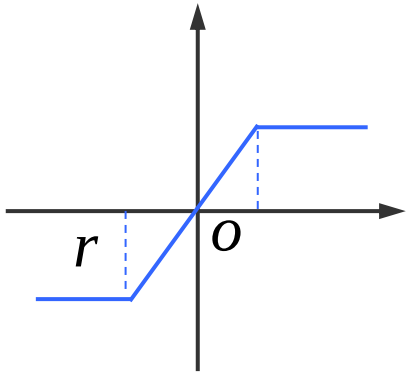
激活函数

- 目前，深度学习在计算机视觉领域取得了引人注目的成果。其中，激活函数的发展也起到了重要的作用。
- 新型激活函数ReLU在一定程度上克服了梯度消失，使得深度网络的直接监督式训练成为可能。
- Bengio 等在ICML2016的文章“Noisy Activation Functions”中有如下定义：
  - 激活函数是映射 h:R→R，且几乎处处可导。
- 软饱和函数性质：$\lim_{s\to\infty} f'(s) = 0$

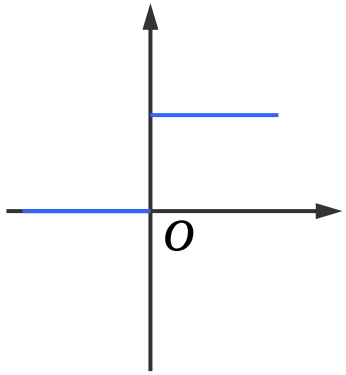
$$f(s)=\begin{cases} r, & s>r \\ s, & |s|\le r \\ -r, & s<-r \end{cases}$$



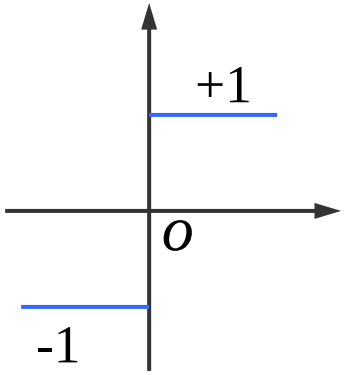
线性函数



斜坡函数

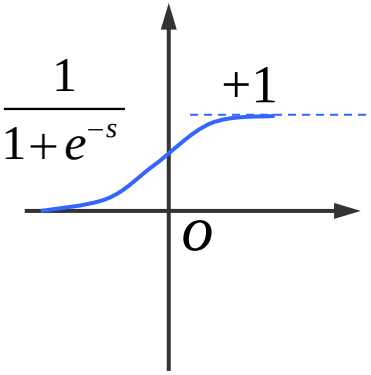


阶跃函数



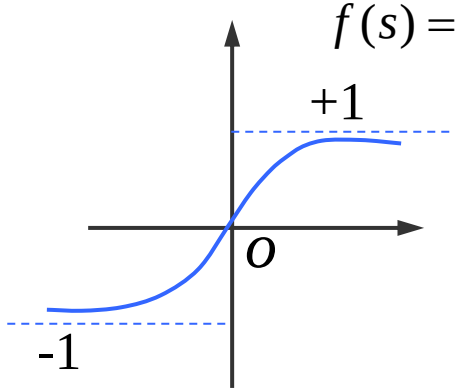
符号函数

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



Sigmoid函数

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

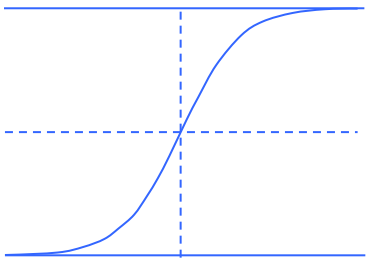


双曲正切函数(tanh)

激活函数

- 软饱和函数性质：$\lim_{s\to\infty} f'(s)=0$
  - **左饱和：** $\lim_{s\to-\infty} f'(s)=0$
  - **右饱和：** $\lim_{s\to+\infty} f'(s)=0$

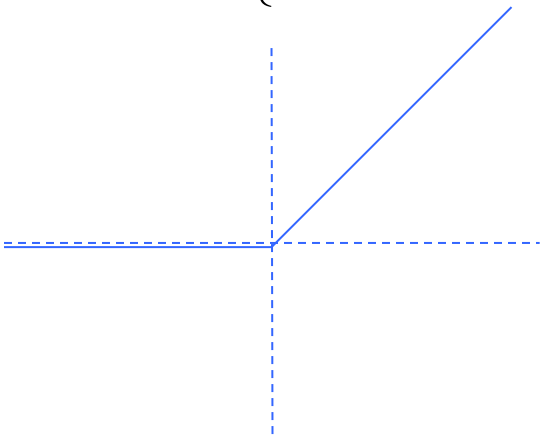
$$f(s)=\frac{1}{1+e^{-s}}$$



- Sigmoid 函数的软饱和性，使得深度神经网络在二三十年里一直难以有效的训练，是阻碍神经网络发展的重要原因。
- 硬饱和函数：与软饱和相对的是硬饱和激活函数，即：$f'(x)=0$，当 $|x|>c$，其中 c 为常数。同理，硬饱和也分为左饱和和右饱和。

激活函数

$$f(s)=\begin{cases} s & s \geq 0 \\ 0 & s < 0 \end{cases}$$



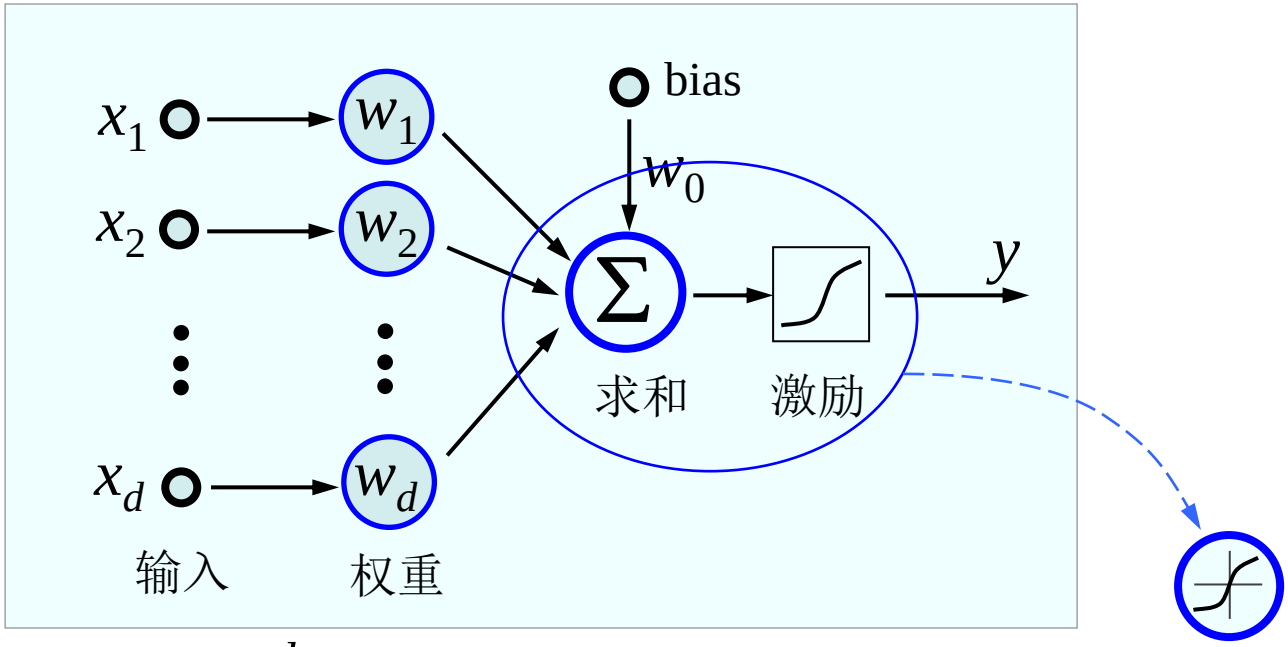
- ReLU
  - ReLU 在$x<0$ 时硬饱和。由于 $x>0$时导数为 1，所以，ReLU 能够在$x>0$时保持梯度不衰减，从而缓解梯度消失问题。
  - 导致网络稀疏性
  - 但随着训练的推进，部分输入会落入硬饱和区，导致对应权重无法更新。这种现象被称为“**神经元死亡**”。
  - ReLU还经常被“诟病”的一个问题是输出具有偏移现象，即输出均值恒大于零。
  - 偏移现象和神经元死亡会共同影响网络的收敛性。

6.2.3 拓扑结构

- 单层网络
  - 输入信号表示为向量：$\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$，其中每一分量通过加权连接到各个结点。
  - 每一个结点均可产生一个加权和。
  - 输入和结点间采用全连接，并且都是前馈连接。
    - 实际的人工神经网络和生物神经网络中有些连接可能不存在。

6.2.3 拓扑结构

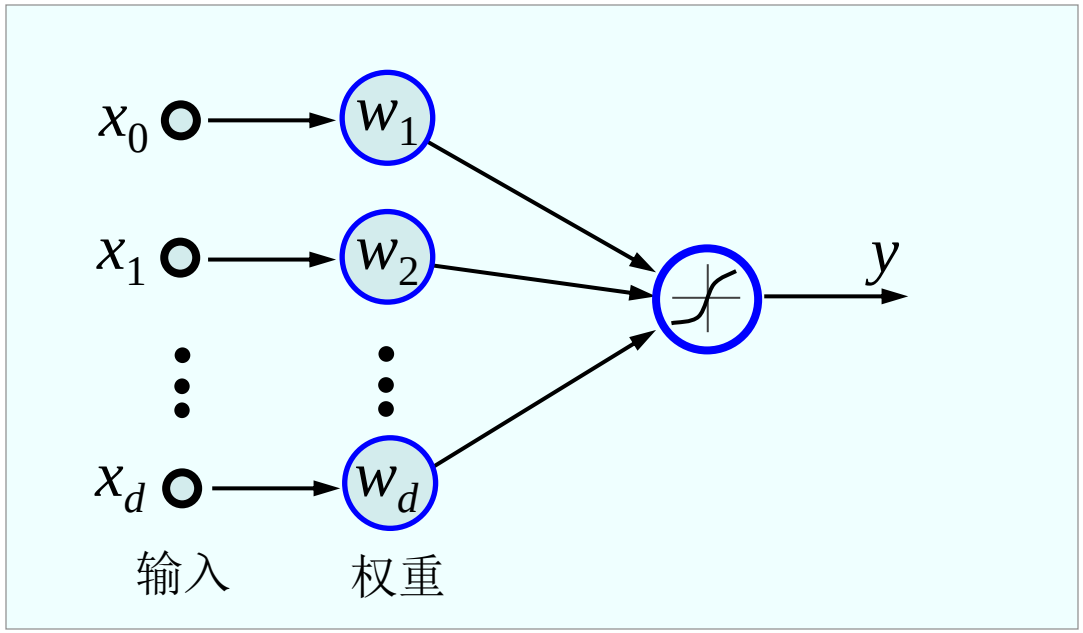
- 单层网络



$$net = \sum_{i=1}^{d} x_i w_i + w_0, \quad y = f(net)$$

6.2.3 拓扑结构

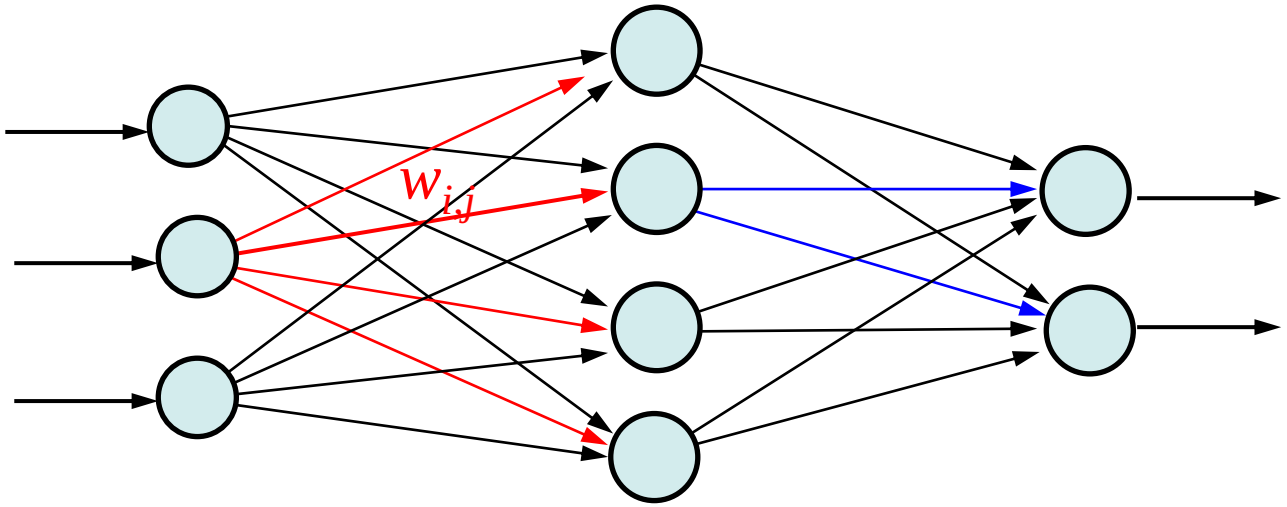
- 单层网络——将偏移（阈值）当成权值



$$net = \sum_{i=0}^{d} x_i w_i, \quad x_0 = 1, \quad y = f(net)$$

6.2.3 拓扑结构

- 多层网络
  - 虽然目前有很多网络模型，但它们的结点基本上都是按层排列的。这一点模仿了大脑皮层中的网络模块。
  - **多层网络是由单层网络进行级联构成的**，即上一层的输出作为下一层的输入。



6.2.3 拓扑结构

- 多层网络
  - 接收输入信号的层称为输入层。
  - 产生输出信号的层称为输出层。
  - 中间层称为隐含层，不直接与外部环境打交道。
    - 隐含层的层数可从零到若干层。
    - 实际情况中，层与层之间可能有部分连接的情况。

6.2.3 拓扑结构

- 多层网络
  - **转移函数应是非线性的**，否则多层网络的计算能力并不比单层网络强：

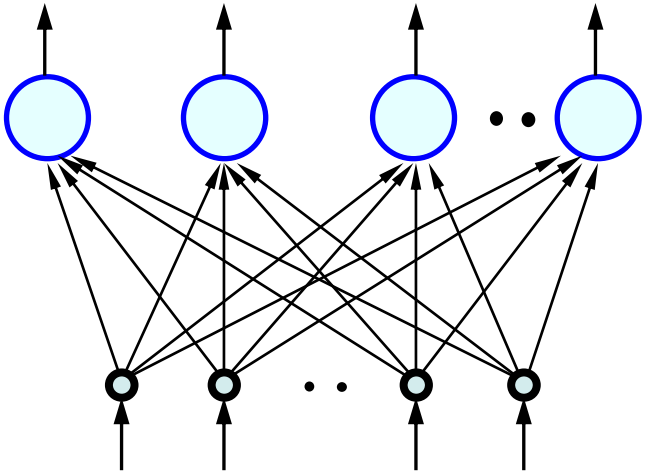
  - 对线性转移函数：第一层的输出$\mathbf{x}^T\mathbf{W}_1$作为第二层的输入，通过第二个加权矩阵得到网络的输出:

$$\mathbf{y} = \mathbf{W}_2(\mathbf{W}_1\mathbf{x}) = (\mathbf{W}_2\mathbf{W}_1)\mathbf{x} = \mathbf{W}\mathbf{x}$$

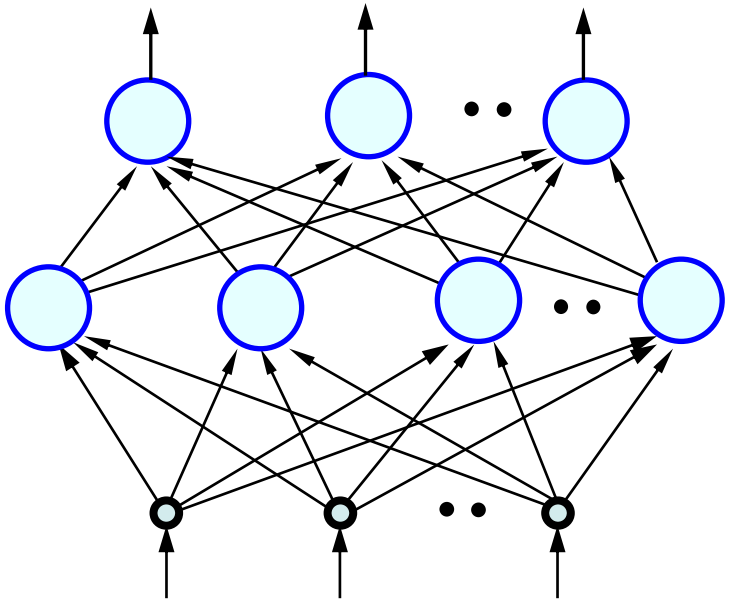
  - 可见：两层线性网络等效于单层网络，只是后者的加权矩阵为两个加权矩阵的乘积。

6.2.3 拓扑结构

- 前馈神经网络



单层感知器

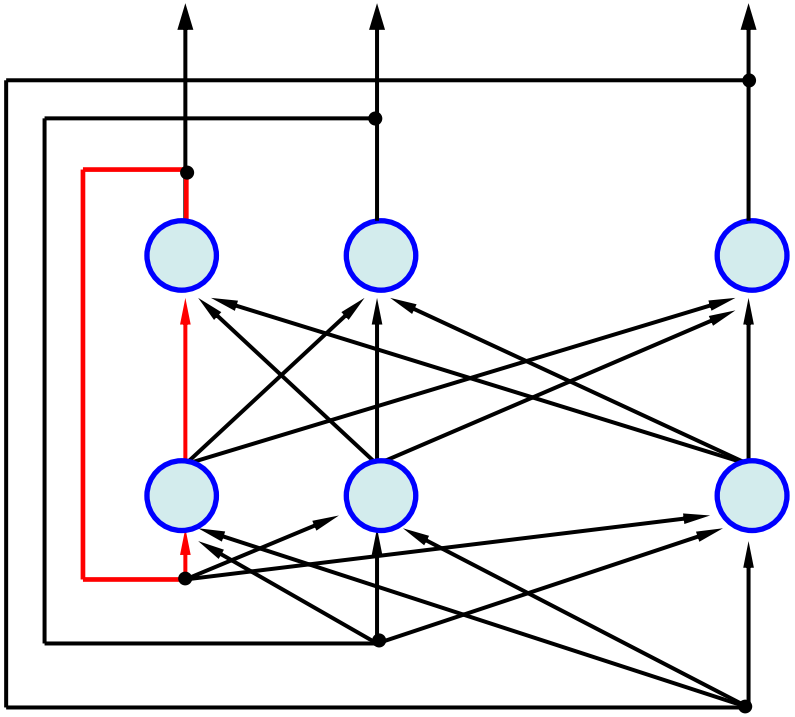


多层感知器

没有层内联接，各结点前馈联接到下一层所有结点

6.2.3 拓扑结构

- 反馈网络



网络有回路，信息可以沿回路流动

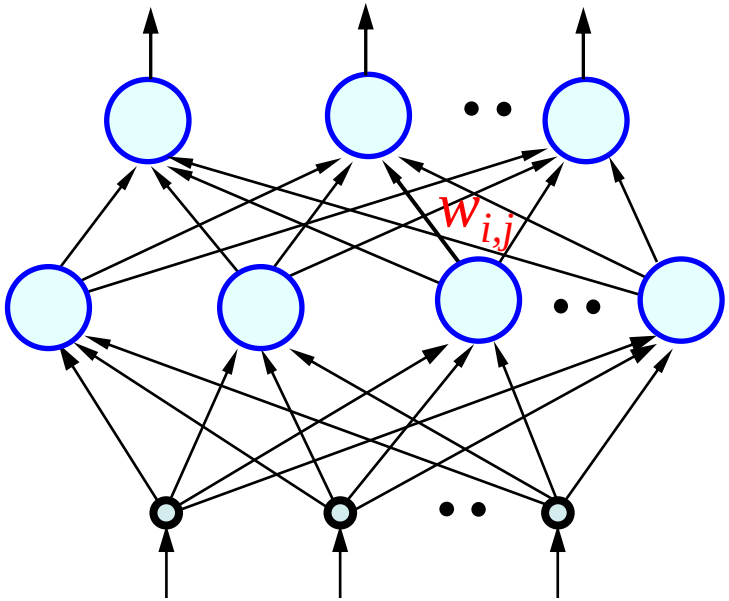
结点的输出：依赖于当前的输入，也依赖于自己以前的输出.

6.2.3 拓扑结构

- 前馈网络与反馈网络的比较
  - 前馈型网络“不存储记忆”，结点的输出仅仅是当前输出的加权和（再加激励）。
  - 在反馈网络中，**要将以前的输出循环返回到输入**。
  - 反馈网络类似于“**人类的短期记忆”，即网络的输出状态部分取决于以前的输入**，是一类广泛应用的网络。

6.2.4 网络训练

- 任务：
  - 给定 n 个观测数据 $\{\mathbf{x}_k\}$，训练网络的各层结点之间的连接权重 w_{ij} (含偏置项)
  - 训练：相继加入训练样本，**并按预定规则调整网络各权重。**



6.2.4 网络训练

- 有监督的训练
  - 希望训练这样一个网络，对每个训练数据，通过该网络计算之后**能尽可能输出其原先给定的值**。
- 无监督训练
  - 无监督的训练不要求有目标向量，网络通过自身的“**经历**”来学会某种功能。
  - 其目的是训练一个网络，使其产生的输出**具有某种可理解的规律性**。
  - 从本质上讲，该训练过程是**抽取样本所隐含的统计特征。**

6.2.4 网络训练

Hebb认为：如果源和目的神经元都被激活或抑制，则它们之间的联系会加强。

- Hebb训练方法（经验方法）
  - **基本思路（突触修正假设）**：如果两个相互联接的处理单元（结点）处于相同的激励电平**（即输出值具有相同的符号）**，则突触传导增强（权重增强）。
  - **训练策略**：**数学上，**两结点 i 和 j 的连接权重将按它们的输出值之积来改变：

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \eta y_i z_j$$

权重更新　　训练速率　　结点 i, j 的输出

  - Hebb学习规则需预先设置权饱和值，以防止输出和输出正负始终一致时出现权重无约束增长。
  - Hebb学习规则是前馈、无指导学习规则。

6.2.4 网络训练

- **δ 训练方法（分析方法）**
  - **基本思路：**按差值（δ值）最小准则连续地修正各连接权重的强度。
    - 差值最小：是指**处理单元所要求的输出与当前实际输出间的差值**，通过调节各权重值以达到最小。
  - **训练策略：**梯度下降法
    - 普遍采用的一种有效的训练方法：

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

梯度增量

与训练样本相关的能量

6.2.4 网络训练

- **随机训练方法**
  - **基本思路**：采用概率和能量关系来调节权重
  - **训练策略**：随机改变一个权重，计算改变后产生的最终能量，并按如下准则来确定是否接受此改变：
    - 若改变后，网络的能量降低，则接受这一改变。
    - 若改变后，能量没有降低，则根据一个预定的概率分布来保留（接受）这一改变。
    - 否则，拒绝这一改变，使权值恢复到原来的值。
  - 典型的随机训练算法：**模拟退火算法**

6.2.4 网络训练

- Kohonen训练方法
  - **基本思路：**
    - 在训练过程中结点（处理单元）参与彼此竞争，具有最大输出的结点为获胜者。
    - 获胜的结点具有抑制其竞争者的能力和激活其近邻结点的能力，只有获胜者和其近邻结点的权重才被允许调节。
    - 获胜者的近邻结点的范围在训练中是可变的。
  - **自组织竞争型神经网络**

芬兰 Teuvo Kohonen 教授受生物系统的启发而提出，是无监督训练方法。

第三节

单层前馈神经网络

（单层感知器）

6.3.1 感知器的结构

- 单层感知器网络描述
  - 该网络由 d+1个输入结点（包含一个 bais 结点）和一个含有 c 个结点的输出层构成，没有隐蔽层。
  - 输入向量：$\mathbf{x}=[1, x_1, x_2,\ldots, x_d]^T$，（齐次坐标）
  - 输出向量：$\mathbf{z}=[z_1, z_2,\ldots, z_c]^T$
  - 权重集合：$\{w_{ij}\}$
  - 对于**输出层结点 j**，其输入加权和为：

$$net_j = \sum_{i=0}^{d} w_{ij} x_i = \mathbf{w}_j^T \mathbf{x}, \quad (x_0 = 1)$$

其中，$\mathbf{w}_j=[w_{0j}, w_{1j}, w_{1j},\ldots, w_{dj}]^T$

6.3.1 感知器的结构

- 单层感知器网络描述
  - 对于输出结点 j，其输出值 z_j 为：

$$z_j = f(net_j) = f\left(\sum_{i=0}^{d} w_{ij} x_i\right) \quad \text{（经过激励）}$$

  - 其中 $f(net_j)$ 为结点 j 的激励（转移）函数，如**符号函数、线性函数和非线性函数**等。
  - 具有不同激励（转移）函数的感知器，具有不同的功能。

6.3.2 感知器的训练

- 学习任务
  - 任务：确定网络各个联接权重$\{w_{ij}\}$：
    - 给定训练样本 $\{\mathbf{x}_k\}$及其目标值（导师信号） $\{\mathbf{t}_k\}$，当网络训练完成后（即权重均确定），期望：

$$\mathbf{z}_k \approx \mathbf{t}_k,\ k=1,2,\cdots,\ （\mathbf{z}_k 为\ \mathbf{x}_k 的网络输出值）$$

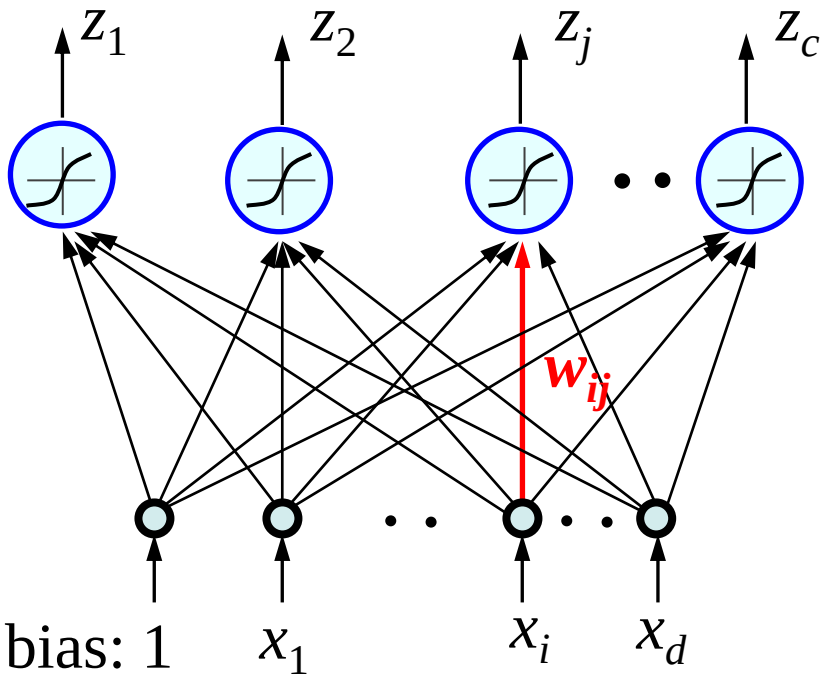
  - **对分类任务：**
    - 两类：输出层只需一个结点。目标值为+1或–1。
    - 多类： t_j 可以为一个类别标签向量中的第 j 个元素。$t_j = 1$, 如果 $\mathbf{x}_j$ 属于第 j 类，否则为0。

6.3.2 感知器的训练

- 学习任务

i: 联系输入层各结点
j: 联系输出层各结点

期望： $z_1 \approx t_1 \quad z_2 \approx t_2 \quad z_j \approx t_j \quad z_c \approx t_c$



6.3.2 感知器的训练

- 线性单元
  - 具有连续转移函数的感知器，转移函数是线性函数，且处处可微的。
    - **转移函数：** ***f (net) = net.***
  - 训练：设输入端点加上偏置端点共有d+1个，训练样本总数为n，c 为输出层结点个数。
  - 要求“输出值＝期望值”：

$$z_j^k = t_j^k,\ k = 1, \cdots, n$$

其中，$z_j^k = w_{0j}x_0^k + w_{1j}x_1^k + \cdots + w_{d+1\,j}x_d^k$

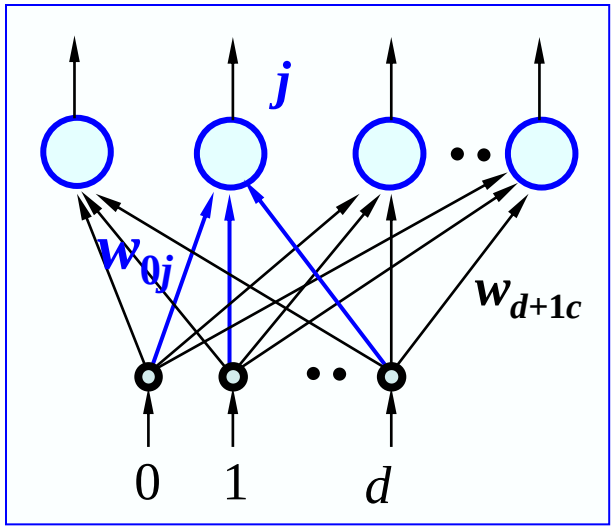
$\left(x_0^k = 1\right)$

上标 k 联系第 k 个样本

6.3.2 感知器的训练

- 线性单元训练： 直接加权

z_j^1：$\mathbf{x}_1$在结点j的输出值



$$\begin{cases} w_{0j}x_0^1 + w_{1j}x_1^1 + \cdots + w_{d+1j}x_d^1 = t_j^1 \\ w_{0j}x_0^2 + w_{1j}x_1^2 + \cdots + w_{d+1j}x_d^2 = t_j^2 \\ \cdots \\ w_{0j}x_0^n + w_{1j}x_1^n + \cdots + w_{d+1j}x_d^n = t_j^n \end{cases}$$

$$j = 1, 2, \cdots, c$$

（一共要解c个方程组）

6.3.2 感知器的训练

- **线性单元训练：δ规则**

***i*:** 联系输入层各结点
***j*:** 联系输出层各结点
***k*:** 联系各训练样本

误差函数：
$$E(W)=\frac{1}{2}\sum_{k,j}(t_j^k-z_j^k)^2=\frac{1}{2}\sum_{k,j}(t_j^k-\sum_i w_{ij}x_i^k)^2$$

梯度：
$$\Delta w_{ij}=-\eta\frac{\partial E}{\partial w_{ij}}=\eta\sum_k(t_j^k-z_j^k)x_i^k$$

单个样本的贡献：
$$\Delta w_{ij}=\eta(t_j^k-z_j^k)x_i^k=\eta\delta_j^k x_i^k,\quad(\delta_j^k=t_j^k-z_j^k)$$

称为δ规则训练、**Widrow-Hoff**（最小平方误差**MSE**）算法

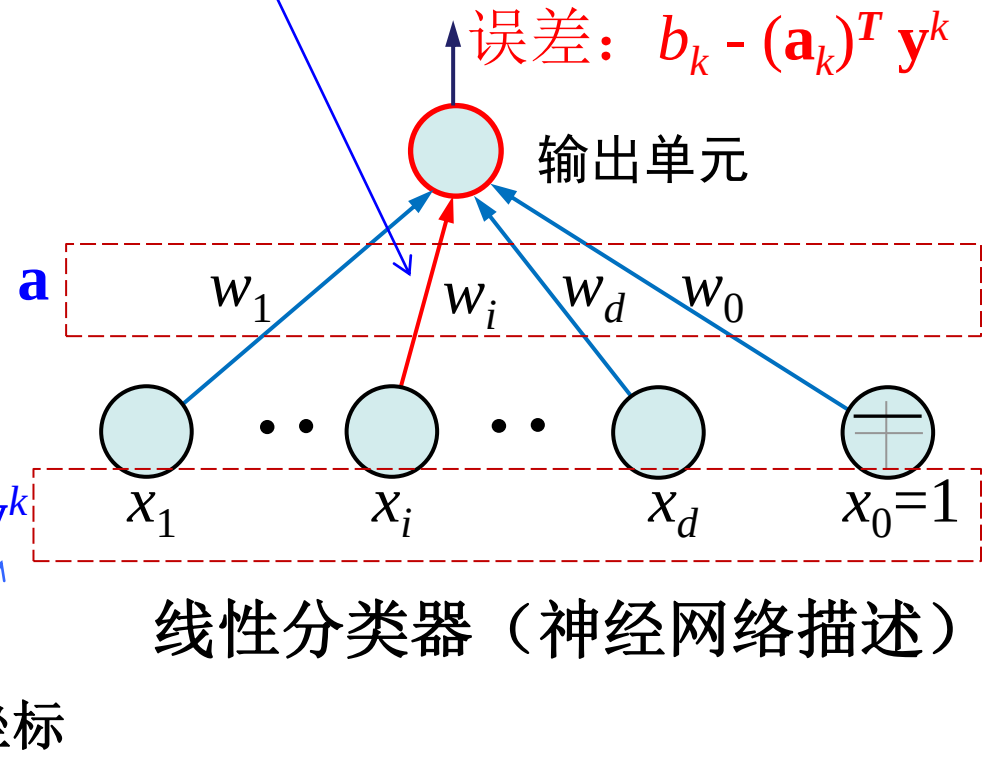
从任意起始点开始调调整网络的权重，当网络的实际输出值与要求的目标值的差值达到最小时，权重最佳。

- **回顾上一章的Widrow-Hoff方法**

梯度更新：$\mathbf{a} = \mathbf{a} + \eta_k (b_k - (\mathbf{a}_k)^T \mathbf{y}^k) \mathbf{y}^k$

第i个分量：$[\boldsymbol{a}]_i = [\boldsymbol{a}]_i + \eta_k (b_k - (\mathbf{a}_k)^T \mathbf{y}^k) [\mathbf{y}^k]_i$

- ✓ 误差乘以样本分量。
- ✓ 从神经网络的角度：边的起始端的输入 x_i 乘以边的指向端收集的误差 “$b_k - (\mathbf{a}_k)^T \mathbf{y}^k$”



6.3.2 感知器的训练

- 非线性单元
  - 转移函数是非线性函数，且处处可微：

误差函数：

$$E(\mathbf{w})=\frac{1}{2}\sum_{k,j}(t_j^k-z_j^k)^2=\frac{1}{2}\sum_{k,j}[t_j^k-f(net_j^k)]^2,\ net_j^k=\sum_i w_{ij}x_i^k$$

梯度：

$$\frac{\partial E}{\partial w_{ij}}=-\sum_k\frac{\partial E}{\partial net_j^k}\frac{\partial net_j^k}{\partial w_{ij}}=-\sum_k[t_j^k-f(net_j^k)]\,f'(net_j^k)\,x_i^k$$

单个样本的贡献：

$$\Delta w_{ij}=-\eta\frac{\partial E}{\partial w_{ij}}=[t_j^k-z_j^k]\,f'(net_j^k)\,x_i^k,\ \ z_j^k=f(net_j^k)$$

$$\Delta w_{ij}=\eta\delta_j^k x_i^k,\ \delta_j^k=\frac{\partial E}{\partial net_j^k}=f'(net_j^k)[t_j^k-z_j^k]\ \text{(δ规则)}$$

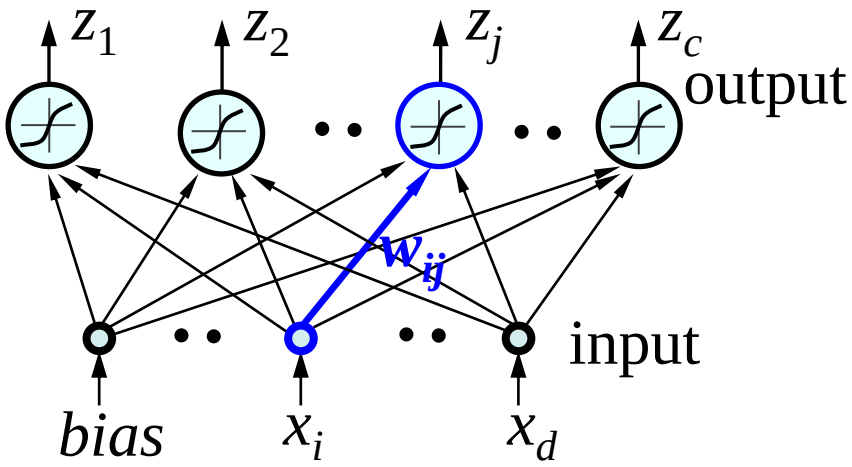
6.3.2 感知器的训练

- 非线性单元
  - 转移函数是非线性函数，且处处可微。

(δ规则)

单个样本的贡献：

$$\Delta w_{ij} = \eta \delta_j^k x_i^k, \quad \delta_j^k = \frac{\partial E}{\partial net_j^k} = f'(net_j^k)[t_j^k - z_j^k]$$



边的起始结点的输出乘以边的指向结点“经导数缩放之后的误差”

6.3.2 感知器的训练

- 非线性单元
  - Sigmoid 函数

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

$$f'(s)=\frac{e^{-s}}{(1+e^{-s})^2}=\frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right)=y(1-y),\quad y=f(s)$$

$$\delta_j^k=y_j^k(1-y_j^k)[t_j^k-y_j^k]$$

6.3.2 感知器的训练

- 单样本随机更新算法

Stochastic Updating

1 begin initialize: $\mathbf{w}$, η, criterion θ, $k=0$

2 do $k \leftarrow k+1 \pmod{n}$

3 $\mathbf{x}^k$, randomly chose a sample (pattern)

4 $w_{ij} \leftarrow w_{ij} + \eta \delta_j^k x_i^k$

5 until $\| \nabla J(\mathbf{w}) \| < \theta$

6 return $\mathbf{w}$

7 end

适用：超大规模数据

- 批量更新算法

Batch Upadating

1 begin initialize: **w,** η, criterion θ, r=0

2 do $r \leftarrow r+1$ (increment epoch)

3 $k = 0$, $\Delta w_{ij} = 0$

4 do $k \leftarrow k+1 \pmod{n}$

5 $\mathbf{x}^k$, select a sample (pattern)

6 $\Delta w_{ij} \leftarrow \Delta w_{ij} + \eta \delta_j^k x_i^k$

7 until $k = n$

8 $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$ // 所有样本完成之后再更新

9 until $\| \nabla J(\mathbf{w}) \| < \theta$

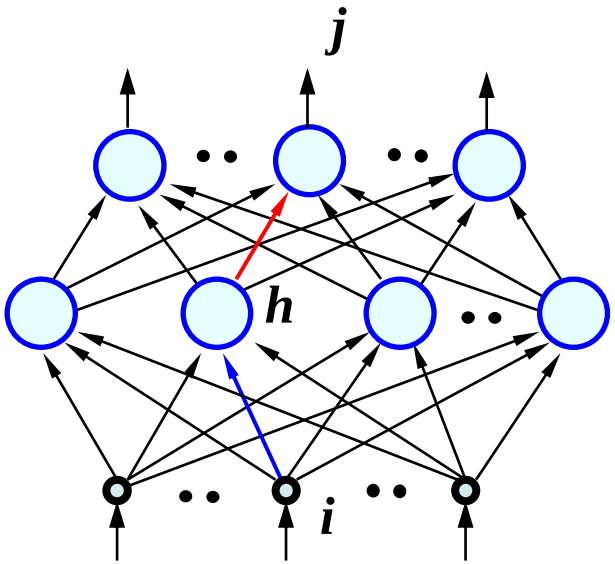
10 return **w**

11 end

第四节　多层感知器

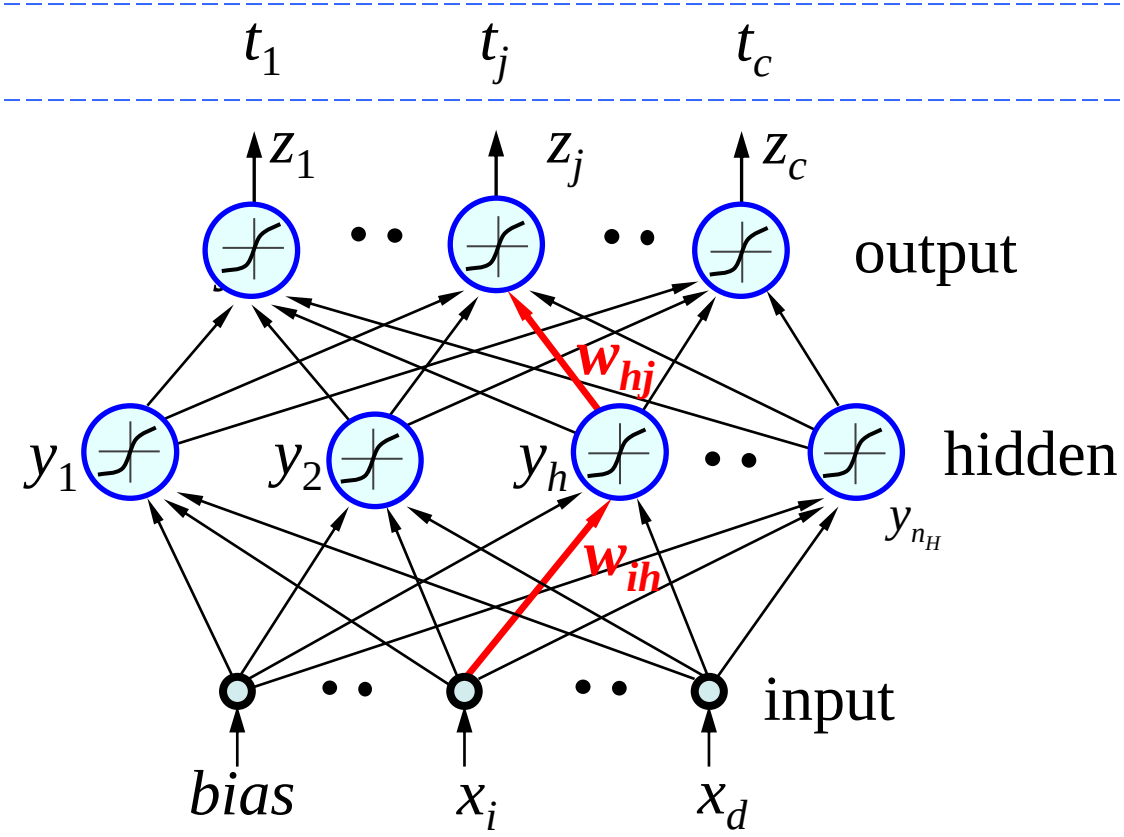
6.4.1 多层感知器

- 三层网络的描述
  - 训练数据输入输出对：$\{x_i^k, t_j^k\}$
  - 输出层结点的输出：z_j^k
  - 隐含层结点的输出：y_h^k
  - 输入信号：x_i^k
  - 输入端点数目：$d+1$
  - 输入层结点 i 至隐含层结点 h 的权重：w_{ih}
  - 隐含层结点 h 至输出层结点 j 的加权表示：w_{hj}
  - 上标 k 表示训练对的序号，$k=1, 2, \ldots, n$



6.4.1 多层感知器

- **Hope:** $z_1 \approx t_1, \cdots, z_c \approx t_c$, **for all samples:** $J(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{c}(t_j - z_j)^2 \approx 0$



- 网络描述-每个样本所经历的计算

上标 k 联系
第 k 个样本

对第 k 个样本，隐含层 h 结点的输入加权和为：

$$net_h^k = \sum_i w_{ih} x_i^k$$

经过激励，隐含层 h 结点的输出：

$$y_h^k = f(net_h^k) = f\left(\sum_i w_{ih} x_i^k\right)$$

输出层 j 结点的输入加权和为：

$$net_j^k = \sum_h w_{hj} y_h^k = \sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)$$

经过激励，输出层 j 结点的输出：

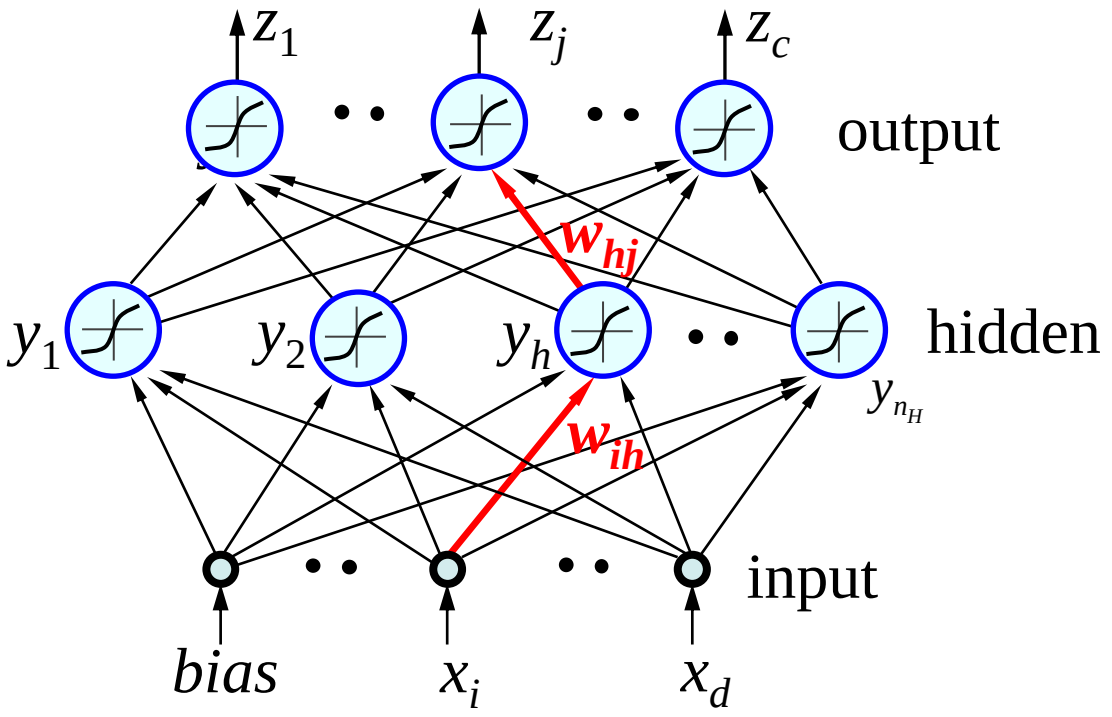
$$z_j^k = f(net_j^k) = f\left(\sum_h w_{hj} y_h^k\right) = f\left(\sum_h w_{hj} f(\sum_i w_{ih} x_i^k)\right)$$

- 网络描述–每个样本所经历的计算

上标k：第 k 个样本

$$z_j^k = f\left(\sum_h w_{hj} y_h^k\right)$$

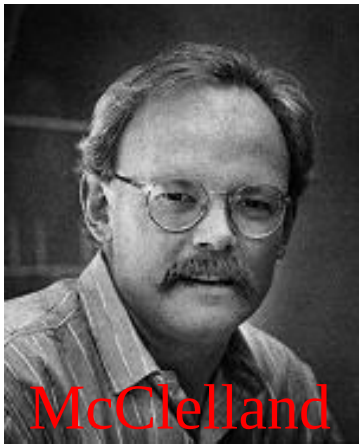
$$y_h^k = f\left(\sum_i w_{ih} x_i^k\right)$$



$$z_j^k = f(net_j^k) = f\left(\sum_h w_{hj} y_h^k\right) = f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)$$

6.4.2 误差反向传播(BP)算法

- D. Rumelhart，J. McClelland于1985年提出了误差反向传播(Back Propagation, BP)学习算法



- 基本原理
  - 利用输出后的误差来估计**输出层的前一层**的误差，再用这个误差估计更前一层的误差，如此一层一层地反传下去，从而获得所有其它各层的误差估计

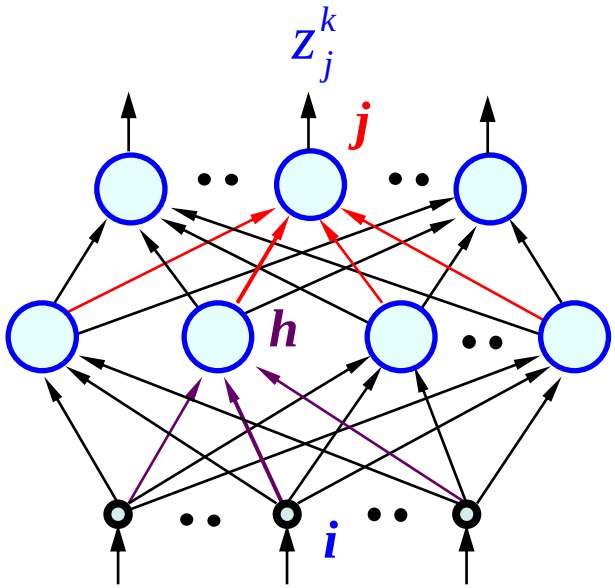
6.4.2 BP算法

- 误差反向传播训练算法
  - 属于监督学习算法，**通过调节各层的权重，使网络学会**由“输入-输出对”组成的训练组。
  - BP算法核心是梯度下降法。
  - 权重先从输出层开始修正，再依次修正各层权重
    - 首先修正：**“输出层至最后一个隐含层”**的连接权重
    - 再修正：**“最后一个隐含层至倒数第二个隐含层”**的连接权重，….
    - 最后修正：**“第一隐含层至输入层”**的连接权重。

学习的本质：对网络各连接权重作动态调整！

- 误差函数—单个样本

上标 k 联系
第 k 个样本



$$z_j^k = f\left(net_j^k\right) = f\left(\sum_h w_{hj} y_h^k\right)$$

$$y_h^k = f\left(net_h^k\right) = f\left(\sum_i w_{ih} x_i^k\right)$$

$$E(\mathbf{w})^k = J(\mathbf{w})^k = \frac{1}{2}\sum_j (t_j^k - z_j^k)^2$$

$$= \frac{1}{2}\sum_j \left(t_j^k - f(net_j^k)\right)^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} y_h^k\right)\right\}^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} f\left(net_h^k\right)\right)\right\}^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)\right\}^2$$

- **复合函数求导数**

设 $f(x)=g_1(g_2(g_3(\cdots g_n(x))))$,

令 $h_2=g_2(g_3(\cdots g_n(x)))$,

$h_3=g_3(g_4(\cdots g_n(x))),\cdots$

$$\frac{\partial f(x)}{\partial x}$$

$$=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2}{\partial x}$$

$$=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2(h_3)}{\partial h_3}\frac{\partial h_3}{\partial x}$$

$$=\cdots\cdots$$

$$=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2(h_3)}{\partial h_3}\cdots\frac{\partial g_n(x)}{\partial x}$$

设 $f(x)=g_1(g_2(g_3(\cdots g_n(x)))+t_2(t_3(\cdots t_m(y))))$,

令 $H_2=g_2(g_3(\cdots g_n(x)))+t_2(t_3(\cdots t_m(y)))$,

$h_2=g_2(g_3(\cdots g_n(x)))$,

$h_3=g_3(g_4(\cdots g_n(x))),\cdots$

$$\frac{\partial f(x)}{\partial x}=\frac{\partial g_1(H_2)}{\partial H_2}\frac{\partial H_2}{\partial x}$$

$$=\frac{\partial g_1(H_2)}{\partial H_2}\left(\frac{\partial h_2(h_3)}{\partial h_3}\frac{\partial h_3}{\partial x}+0\right)$$

$$=\cdots\cdots$$

$$=\frac{\partial g_1(H_2)}{\partial H_2}\frac{\partial h_2(h_3)}{\partial h_3}\cdots\frac{\partial g_n(x)}{\partial x}$$

- 网络训练：隐含层－输出层

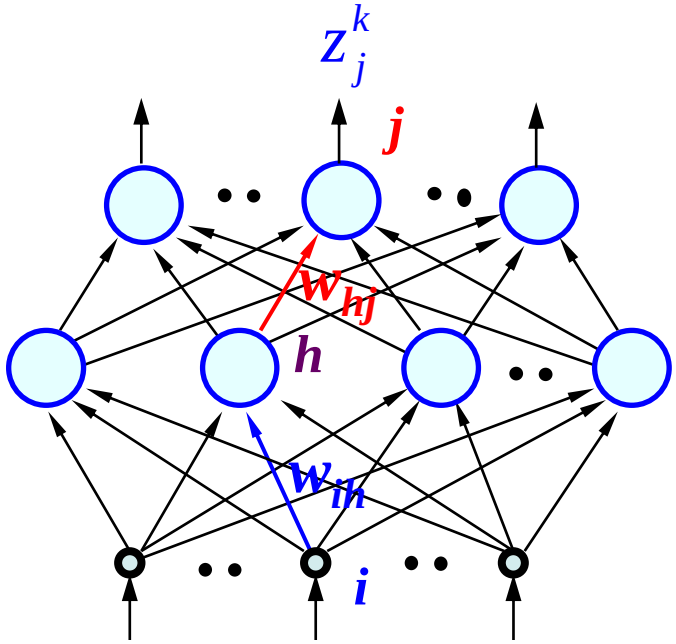
隐含层到输出层的连接权重调节量：

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}} = -\eta \sum_k \frac{\partial E}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{hj}}$$

$$= \eta \sum_k (t_j^k - z_j^k) f'(net_j^k) y_h^k$$

$$= \eta \sum_k \delta_j^k y_h^k$$

(δ规则)



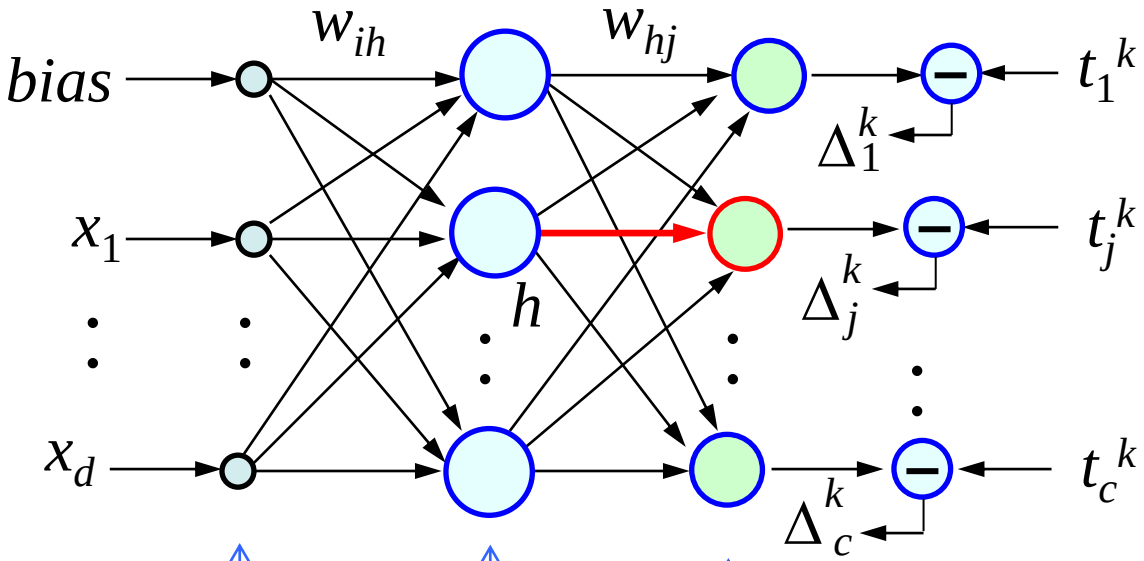
$$\delta_j^k = \frac{-\partial E}{\partial net_j^k} = f'(net_j^k)(t_j^k - z_j^k) = f'(net_j^k)\Delta_j^k,$$

$$\Delta_j^k = t_j^k - z_j^k$$

边的指向结点的误差信号（局部梯度）

- 隐含层－输出层，**准备好输出层的误差：** $\Delta_j^k = t_j^k - z_j^k$

样本→：　　x_i^k → y_h^k → z_j^k － t_j^k



注：上标 k 联系第 k 个样本

- 隐含层—输出层：第 k 个训练样本对权重 w_{hj} 的贡献

$h \to j$, for sample k:

δ规则： $$\Delta w_{hj}\big|_{\text{sample } k} = \eta \delta_j^k y_h^k$$

权重所联边的起始结点（隐含结点 h）的输出

权重所联边的指向结点（输出结点 j）收集到的误差信号

$$\delta_j^k = f'(net_j^k)\Delta_j^k, \quad \Delta_j^k = t_j^k - z_j^k$$

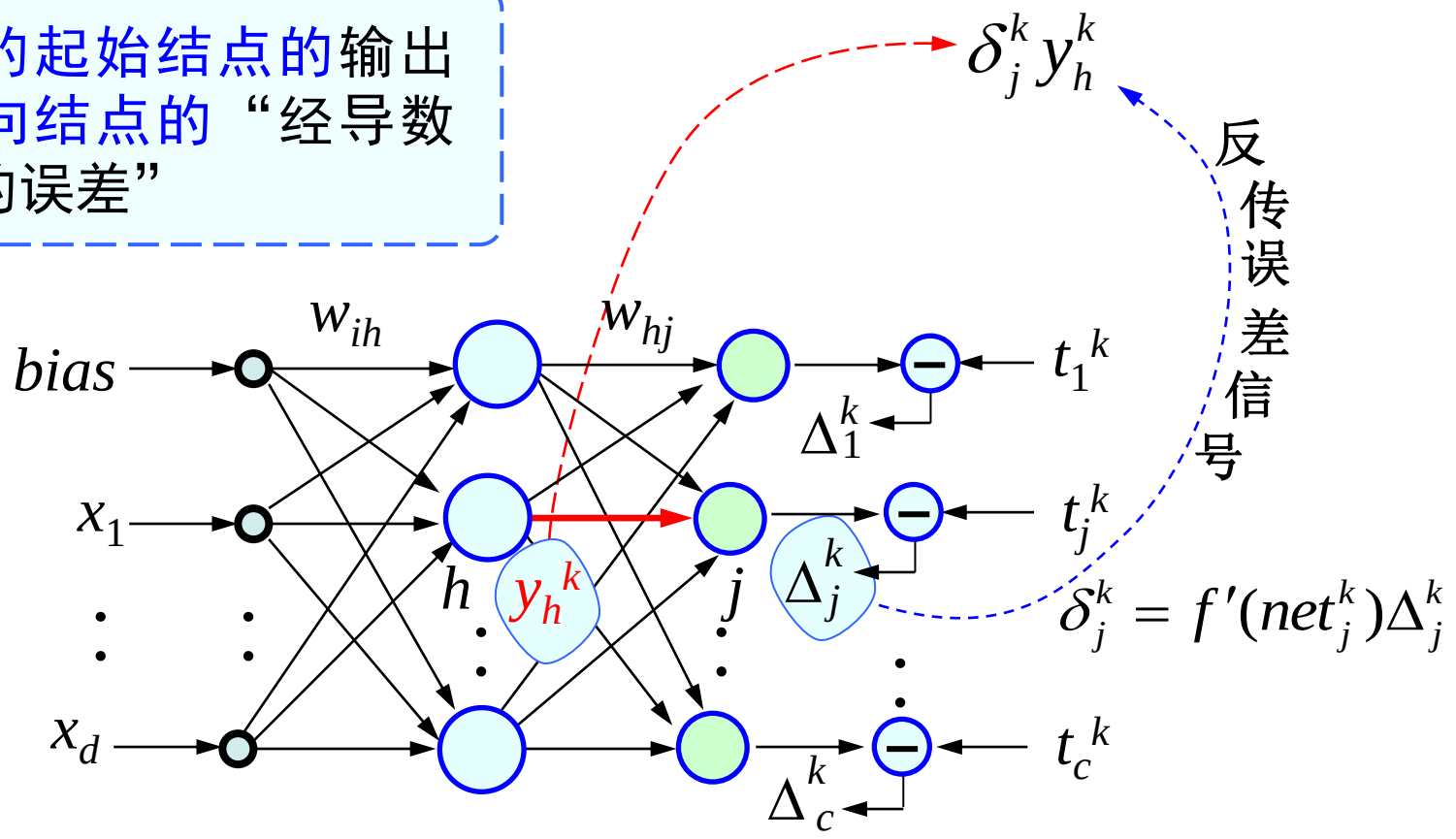
误差**在权重所联边的指向结点处**计算。

误差大小等于：该结点收集到的误差乘以激励函数对“该结点加权和”的导数。

误差反传与权重更新：

所联边的起始结点的输出乘以指向结点的“经导数缩放后的误差”

第 k 个样本对权重更新的贡献

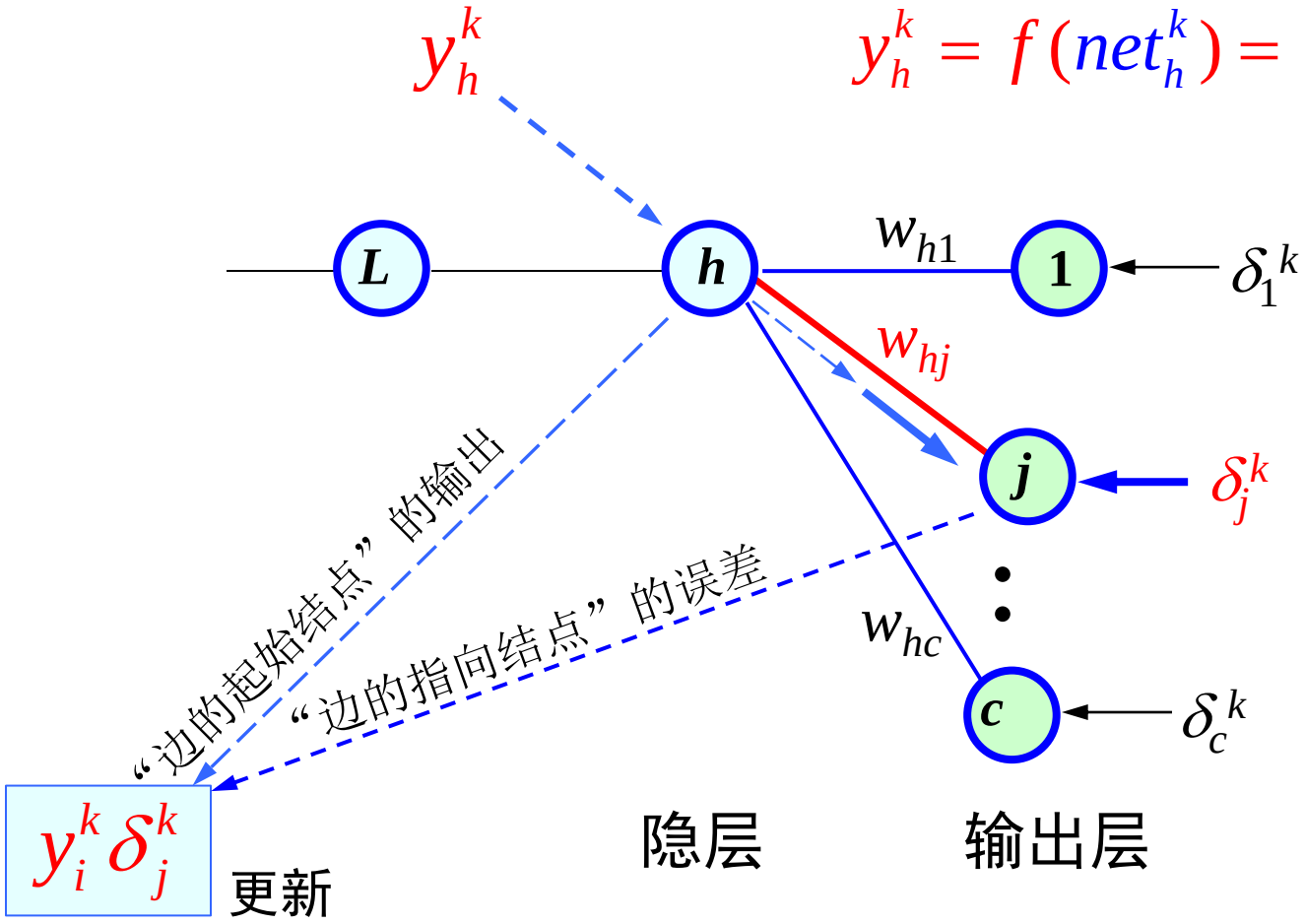


下标联系的结点 **i** **h** **j**

隐层-输出权重更新示意：

w_{hj}所连接的边的起始结点向外传递的信号值

$$y_h^k = f(net_h^k) = f\left(\sum_i w_{ih} x_i^k\right)$$



w_{hj}所连接的边的指向结点的误差

6.4.2 BP算法

- 激励函数采用sigmoid函数(最常用)：

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

$$f'(s)=\frac{e^{-s}}{(1+e^{-s})^2}=\frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right)=z(1-z),\quad z=f(s)$$

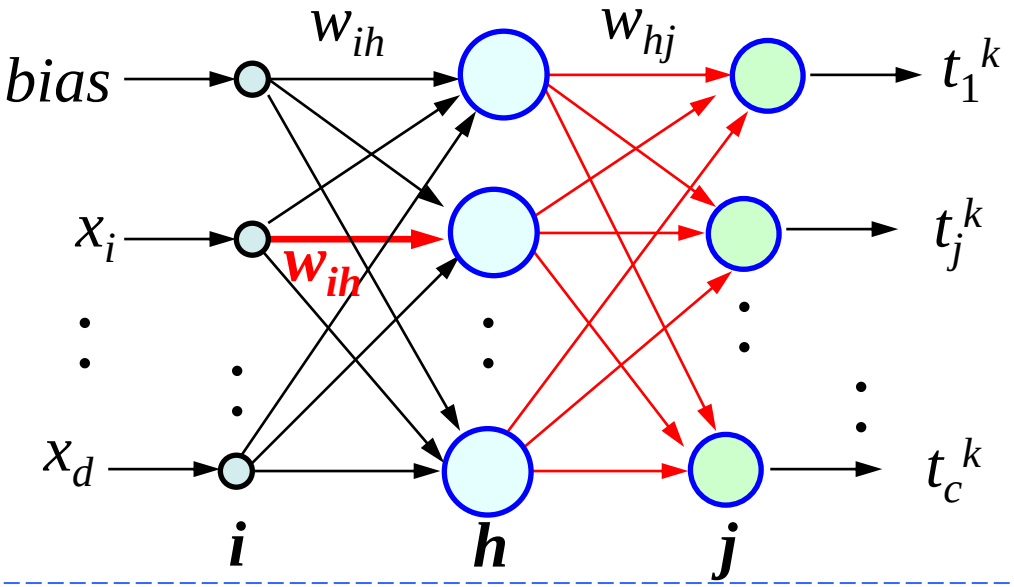
对输出层结点 j，我们有：

$$\delta_j^k=\frac{-\partial E}{\partial net_j^k}=f'\left(net_j^k\right)\left(t_j^k-z_j^k\right)=z_j^k\left(1-z_j^k\right)\left(t_j^k-z_j^k\right)$$

对于**输入层到隐含层**结点连接的边的权重修正量 Δw_{ih}，必须考虑将 $E(\mathbf{w})$ 对 w_{ih} 求导，需利用分层链路法。

输入层至隐含层权重更新：

$$E(\mathbf{w})=\sum_k J(\mathbf{w})=\frac{1}{2}\sum_{k,j}(t_j^k-z_j^k)^2=\frac{1}{2}\sum_{k,j}\left\{t_j^k-f\left(\sum_h w_{hj}\,f\left(\sum_i w_{ih}x_i^k\right)\right)\right\}^2$$



样本→：　$x_i^k \rightarrow net_h^k \rightarrow y_h^k \rightarrow net_h^k \rightarrow z_j^k$　　t_j^k

输入层至隐含层权重更新：

用别名$\tilde{h}$代替h

$$E(\mathbf{w})=\frac{1}{2}\sum_{k,j}(t_j^k-z_j^k)^2=\frac{1}{2}\sum_{k,j}\left\{t_j^k-f\left(\sum_{\tilde{h}}w_{\tilde{h}j}f\left(\sum_{i}w_{i\tilde{h}}x_i^k\right)\right)\right\}^2$$

$$net_{\tilde{h}}^k=\sum_i w_{i\tilde{h}}x_i^k,\quad y_{\tilde{h}}^k=f(net_{\tilde{h}}^k)$$

$$net_j^k=\sum_{\tilde{h}} w_{\tilde{h}j}y_{\tilde{h}}^k,\quad z_j^k=f(net_j^k)$$

z_j^k

net_j^k

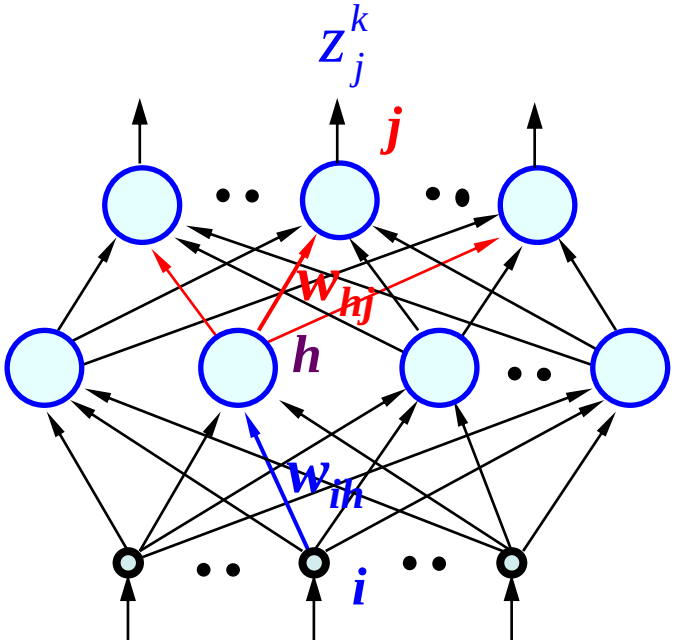
$y_{\tilde{h}}^k$

$net_{\tilde{h}}^k$

包含权重w_{ih}

待更新权重w_{ih}的增量：

$$\Delta w_{ih} = -\eta \frac{\partial E}{\partial w_{ih}} = -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial w_{ih}}$$



$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{ih}}$$

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

(链式法则)

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

待更新权重的增量（具体化）

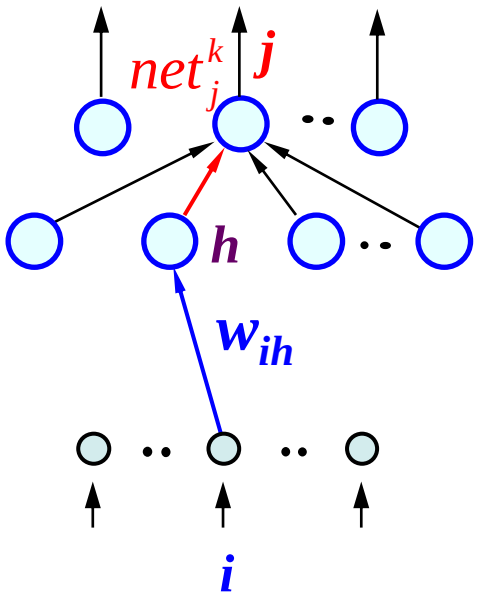
$$\Delta w_{ih} = -\eta \frac{\partial E}{\partial w_{ih}} = -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \cdot \frac{\partial z_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) \frac{\partial z_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) f'\left(net_j^k \right) \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) f'\left(net_j^k \right) w_{hj} \frac{\partial y_h^k}{\partial w_{ih}}$$



（接前一页）

$$\Delta w_{ih} = \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) w_{hj} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) w_{hj} \frac{\partial y_h^k}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) w_{hj} f'\left(net_h^k\right) x_i^k$$

$$= \eta \sum_{k,j} \delta_j^k w_{hj} f'(net_h^k) x_i^k$$

$$\delta_j^k = f'\left(net_j^k\right)\left(t_j^k - z_j^k\right)$$

$$= \eta \sum_{k} \left(f'(net_h^k) \sum_j \delta_j^k w_{hj} \right) x_i^k$$

$$= \eta \sum_{k} \delta_h^k x_i^k$$

$$\delta_h^k = \frac{-\partial E}{\partial net_h^k} = f'(net_h^k) \sum_j w_{hj} \delta_j^k = f'(net_h^k) \Delta_h^k, \quad \Delta_h^k = \sum_j w_{hj} \delta_j^k$$

输入-隐层：第 k 个训练样本对权重 w_{ih} 的贡献

$i \to h$, for sample k:

δ规则：

$$\Delta w_{ih}\,|_{sample\ k} = \eta \delta_h^k x_i^k$$

w_{ih}所连接的边的起始结点（输入层结点 i）的输出（此时即为样本第 i 个分量）

w_{ih}所连接的边的指向结点（隐含结点 h）收集到的误差信号

输入-隐层：第 k 个训练样本对权重 w_{ih} 的贡献

$$\delta_h^k = \frac{-\partial E}{\partial net_h^k} = f'(net_h^k) \sum_j w_{hj}\delta_j^k$$

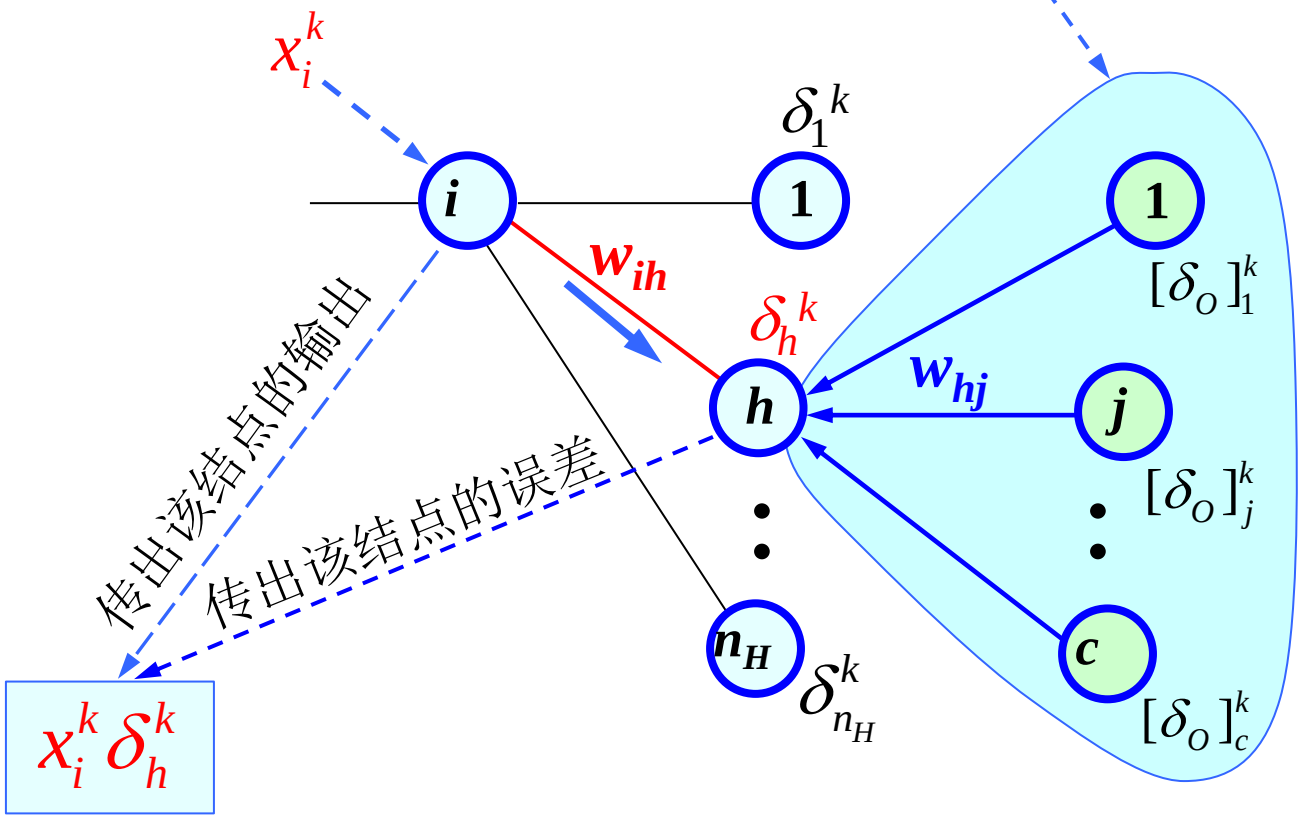
从前一层收集误差：加权和

误差**在权重所联边的指向结点处**计算。

误差大小等于：该结点收集到的误差乘以激励函数对“该结点加权和”的导数。

输入—隐层权重更新示意：

误差收集: $\delta_h^k = f'(net_h^k)\sum_j w_{hj}[\delta o]_j^k$

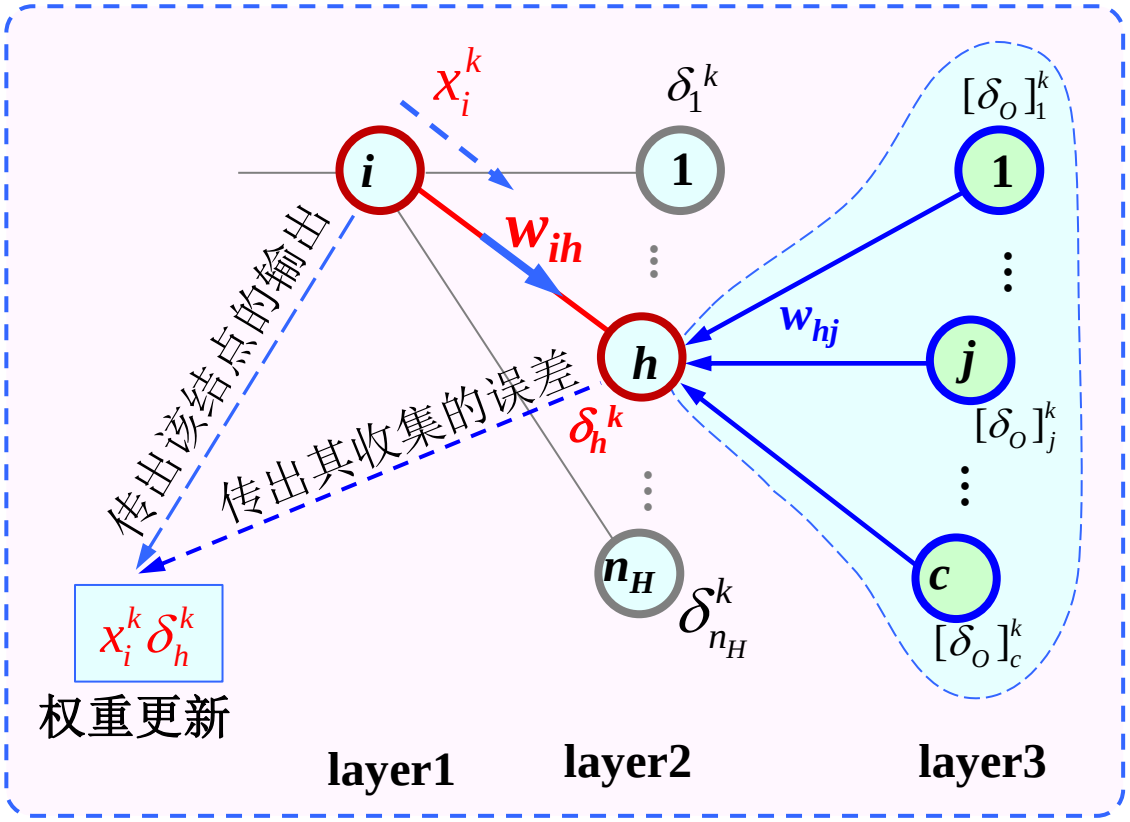


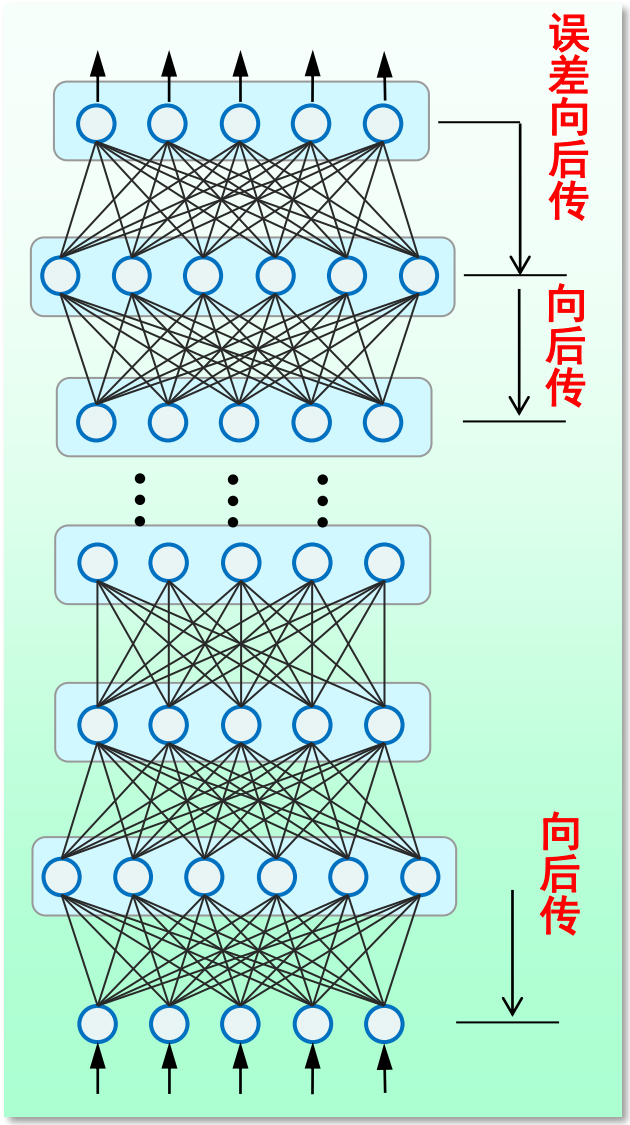
注：已将上页中的 δ 更为 δ_O (对输出层)

x_i^k ： 边*i-h* 起点的输出，即向外传递的信号值

神经网络基础

- w_{ih} 权重更新





w_{ih} 的更新量：

$$\Delta w_{ih} = \eta x_i^k \delta_h^k,\ \delta_h^k = f'(net_h^k)\sum_j w_{hj}[\delta o]_j^k$$

6.4.2 BP算法

- 网络训练
  - BP算法对任意层的加权修正量的一般形式：

$$\Delta w_{in\to o} = \eta \sum_{all\ samples} \delta_o y_{in}$$

  - 单个训练样本的贡献：

$$\Delta w_{in\to o} = \eta \cdot \delta_o \cdot y_{in} = \eta \cdot \left(\sum_h w_{o\to h} [\delta_o]_h \right) \cdot y_{in}$$

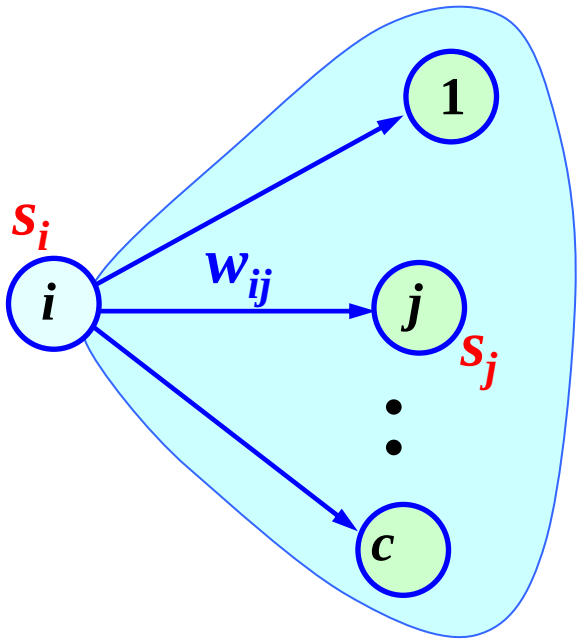
从后一层各结点h收集误差

下标 in 和 o 分别指**“待更新权重”**所连边的起始结点和指向结点，y_{in}代表起始结点的实际输出，δ_o表示指向结点的误差 (由后一层收集得到)。

- 网络训练
  - 从更一般的角度来认识网络
    - 目标函数不是误差平方损失，比如交叉熵、softmax、hinge loss等，对于权重更新是否有上述同样的文字表述？
  - 目标函数对某一层结点 i 的输出 s_i 的梯度（见右上）。
  - 目标函数对权重的梯度(导数)：

$$\nabla_{w_{ij}} E = \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_j}\frac{\partial s_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial s_i} = \sum_{j \in N(i)} \frac{\partial E}{\partial s_j}\frac{\partial s_j}{s_i}$$



$j \in$ {结点 i 指向的所有结点}

6.4.2 BP算法

- 随机更新

Stochastic Backpropagation

1 begin initialize: n_H, **w**, η, criterion θ, $k=0$

2 do $k \leftarrow k+1 \pmod{n}$

3 $\mathbf{x}^k$, randomly chosen a sample (pattern)

4 $w_{hj} \leftarrow w_{hj} + \eta \delta_j^k y_h^k, \quad w_{ih} \leftarrow w_{ih} + \eta \delta_h^k x_i^k$

5 until $\| \nabla J(\mathbf{w}) \| < \theta$

6 return **w**

7 end

适用：超大规模数据

- 批量更新算法

Batch Backpropagation

1 begin initialize: n_H, **w**, η, criterion θ, $r=0$

2 do $r \leftarrow r+1$ (increment epoch)

3 $k = 0,\ \Delta w_{ih} = 0, \Delta w_{hj} = 0$

4 do $k \leftarrow k+1 \pmod{n}$

5 $\mathbf{x}^k$, selected a sample (pattern)

6 $\Delta w_{hj} \leftarrow \Delta w_{hj} + \eta \delta_j^k y_h^k, \quad \Delta w_{ih} \leftarrow \Delta w_{ih} + \eta \delta_h^k x_i^k$

7 until $k = n$

8 $w_{hj} \leftarrow w_{hj} + \Delta w_{hj}, \quad w_{ih} \leftarrow w_{ih} + \Delta w_{ih}$

9 until $\| \nabla J(\mathbf{w}) \| < \theta$ // 所有样本完成之后再更新

10 return **w**

11 end

下次课内容

- 多层前向网络
  - BP算法讨论
- 径向基函数网络
- 反馈神经网络基础
- 自组织映射
- 深度学习综述

Thank All of You!

(Questions?)

向世明

smxiang@nlpr.ia.ac.cn

people.ucas.ac.cn/~xiangshiming

时空数据分析与学习课题组(STDAL)

中科院自动化研究所· 模式识别国家重点实验室