目录 CONTENTS

中国石油大学(华东)
CHINA UNIVERSITY OF PETROLEUM

计算机科学与技术学院
College of Computer Science & Technology

第八章 人工神经网络与深度学习

人工智能课题组

智能科学系

目录 CONTENTS

概述

- 人工神经网络（Artificial Neural Network，ANN）是一个用大量简单处理单元经广泛连接而组成的人工网络。
- 能够发现高维数据中的复杂结构，取得比传统机器学习方法更好的结果，在图像识别、语音识别、机器视觉、自然语言理解等领域获得成功应用。
- 神经网络方法是一种***知识表示方法***和***推理方法***。
  - 神经网络知识表示是一种隐式的表示方法
  - **产生式框架等方法**是知识的显式表示：***产生式系统中，知识独立地表示为一条规则***

8.1 神经网络的发展历史

➢人工神经网络带有**一层隐藏节点**或没有隐层节点，又称为浅层学习（Shallow Learning，SL）。

- 1957年，康奈尔大学F.Rosenblatt提出了**感知器（perceptron）模型**

- 感知器模型结构
  - ✓由输入空间到输出空间的函数：$f(x) = sign(w \cdot x + b)$

- 第一次把神经网络研究从纯理论探讨推向工程实现，掀起了机器学习的第一次高潮。

- 但是感知器仅仅是线性模型，学习不了数学上很简单的异或问题。神经网络的研究陷入了低潮。

8.1 神经网络的发展历史

➢20世纪80年代，Rumelhart等人提出多层前向神经网络的BP学习算法。

- 神经网络的研究取得突破性发展，掀起了机器学习的新高潮。
- 但BP学习算法只适合训练浅层神经网络。

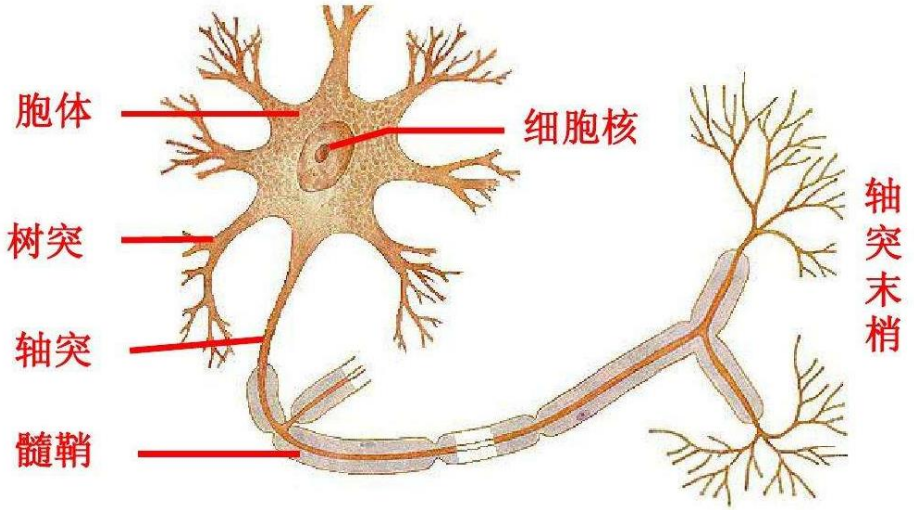
➢通过深度学习得到的深度网络结构是深层次的网络结构，称为深度神经网络（Deep Neural Network，DNN）。

- 2006年，加拿大多伦多大学Geofrey Hinton教授及其学生提出了深度学习，特别是计算机视觉、自然语言处理等多个领域中取得了突破性进展，再次掀起了神经网络的高潮。
- 与人工规则构造特征的方法相比，深度学习利用大数据来学习特征，能够发现大数据中的复杂结构。

8.2 神经元与神经网络

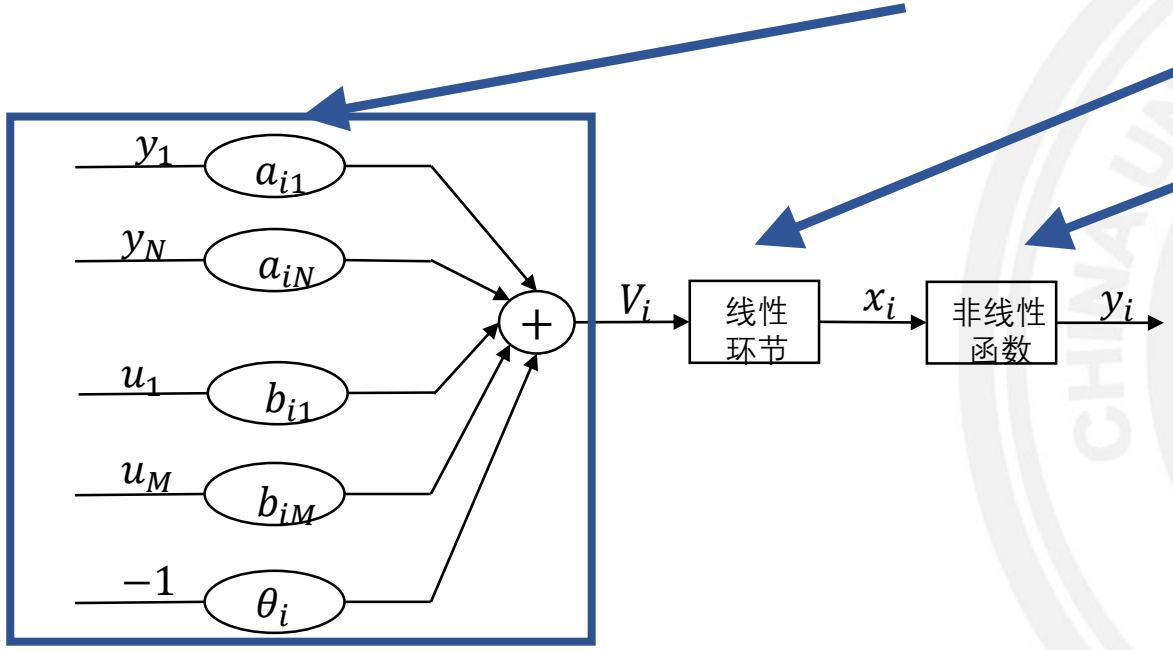
8.2.1 生物神经元结构

- 现代人的大脑内约有10^{11}个神经元，每个神经元与其他神经元之间约有1000个连接，所以大脑内约有10^{14}个连接。
- 神经元的主体部分为**细胞体**。细胞体由**细胞核**、**细胞质**、**细胞膜**等组成。神经元还包括树突和一条长的**轴突**。轴突末端部分有很多分支叫**轴突末梢**。**轴突**用来传递和输出信息。树突相当于细胞的输入端，接受神经元的冲动。
- 神经元具有两种常规工作状态：**兴奋**与**抑制**，满足“0-1”律。

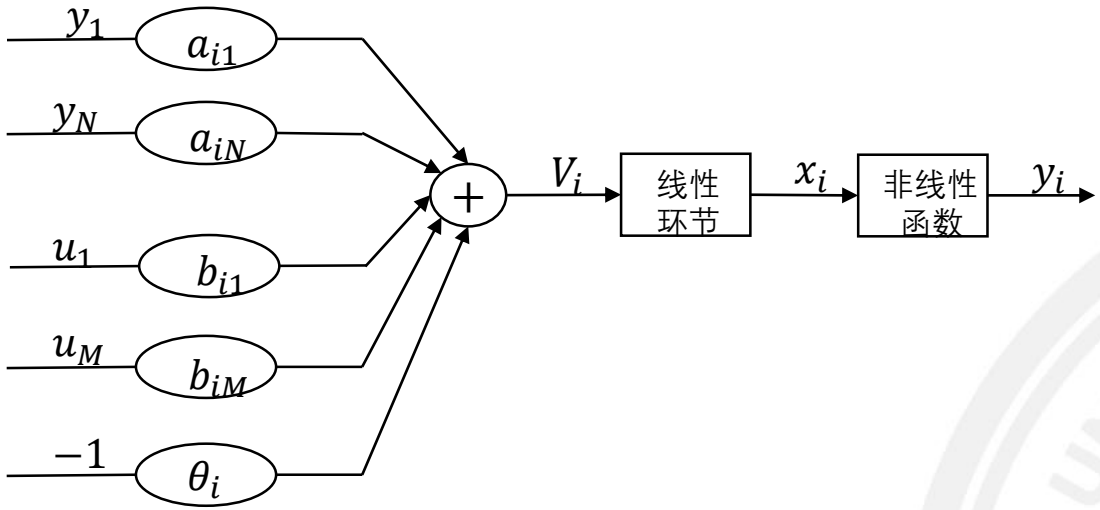


8.2.2 神经元数学模型

- 1943年，美国神经心理学家麦克洛奇和数学家皮兹提出神经元的数学模型（P-M模型）。此后，根据神经元的结构和功能不同，先后提出的神经元模型有几百种之多。
- 神经元的标准数学模型，由三部分组成，即**加权求和**、**线性动态系统**和**非线性函数映射**。

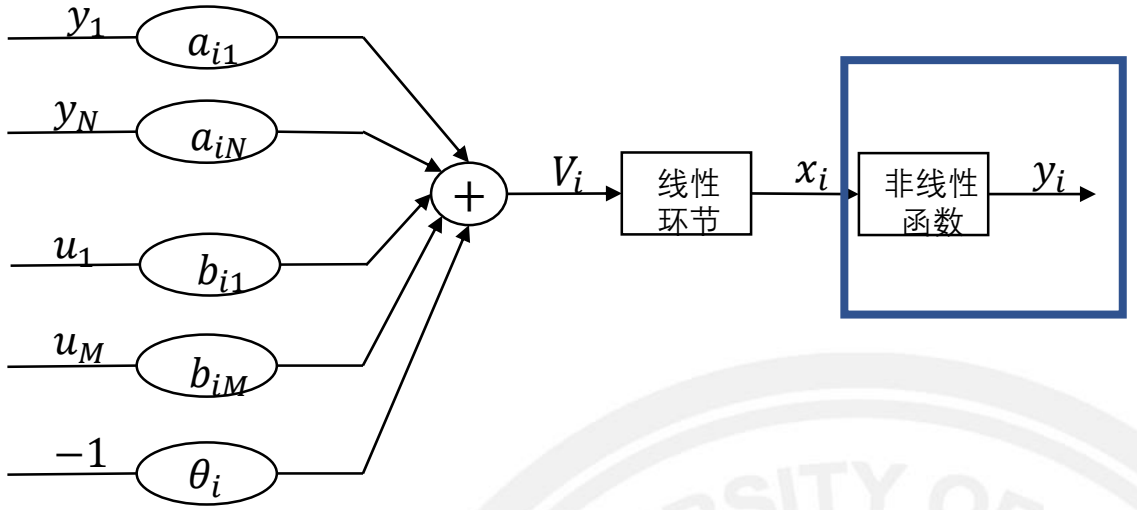


8.2.2 神经元数学模型



- **输入**：u_k为外部输入； a_{ij}， b_{ik}为权值。
- **输出**： y_i为第i个神经元的输出；θ_i为第i个神经元的阈值$(i = 1,2,\cdots,N)$；
- **加权求和**：加权求和把其他神经元的输出对第i个神经元的作用以及外部输入的作用求和再减去阈值θ_i：$V_i(t) = \sum_{j=1}^{N} a_{ij} y_j + \sum_{k=1}^{M} b_{ik}\, u_k - \theta_i$
- **线性环节**：最常见的是比例系数

8.2.2 神经元数学模型



- **非线性激励函数**：

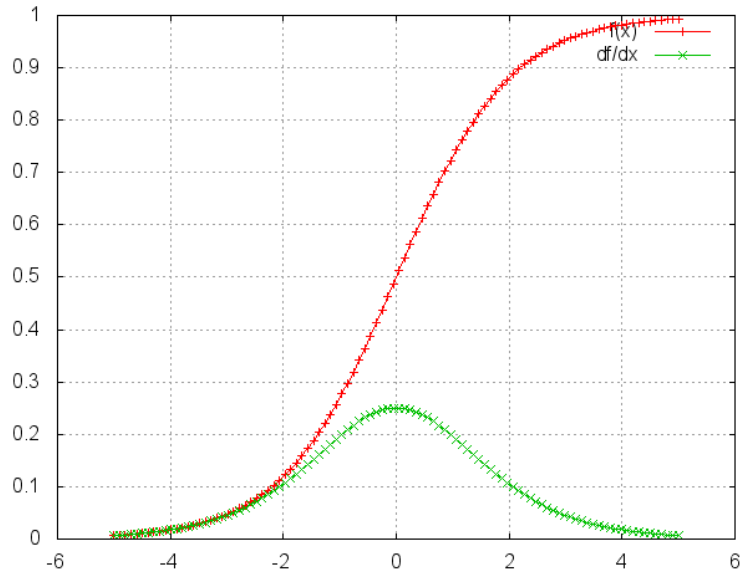
常用的有：

（1）阶跃函数

$$f(x_i)=\begin{cases}1 & x_i>0\\ 0 & x_i\le 0\end{cases}$$

（2）S型函数

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



8.2.3 神经网络的结构

- 神经网络由众多的神经元的轴突和其他神经元或者自身的树突相连接形成的一个网络。网络可以组成高度非线性动力学系统，表现出一般**复杂非线性系统**的特性，如不可预测性、不可逆性、多吸引子、可能出现混沌现象等。

- 根据神经网络中神经元的连接方式，可划分为不同类型的结构。目前，人工神经网络主要由**前馈型**和**反馈型**两大类神经网络。

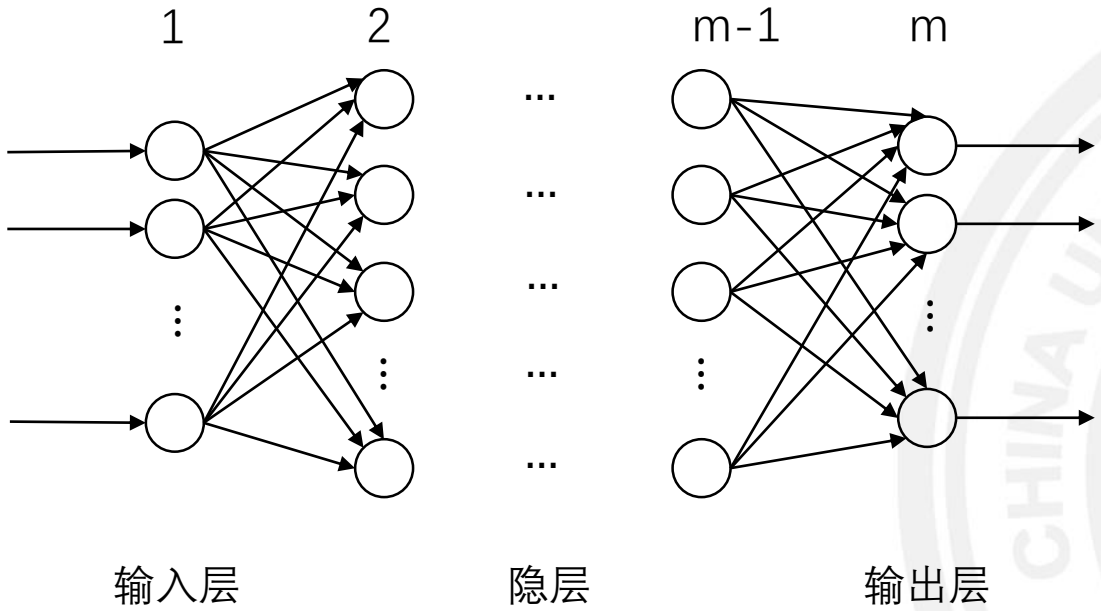
  1、**前馈型网络**中，各神经元接受前一层的输入并输出给下一层，没有反馈。前馈网络可分为不同层，第i层只与第$i-1$层输出相连，输入与输出神经元与外界相连。BP神经网络、卷积神经网络都是前馈型神经网络。

  2、**反馈型网络**中，存在一些神经元的输出经过若干个神经元后再反馈到这些神经元的输入端。典型的反馈型神经网络是Hopfield神经网络。

8.2.3 神经网络的结构

- **1. 前馈型神经网络**

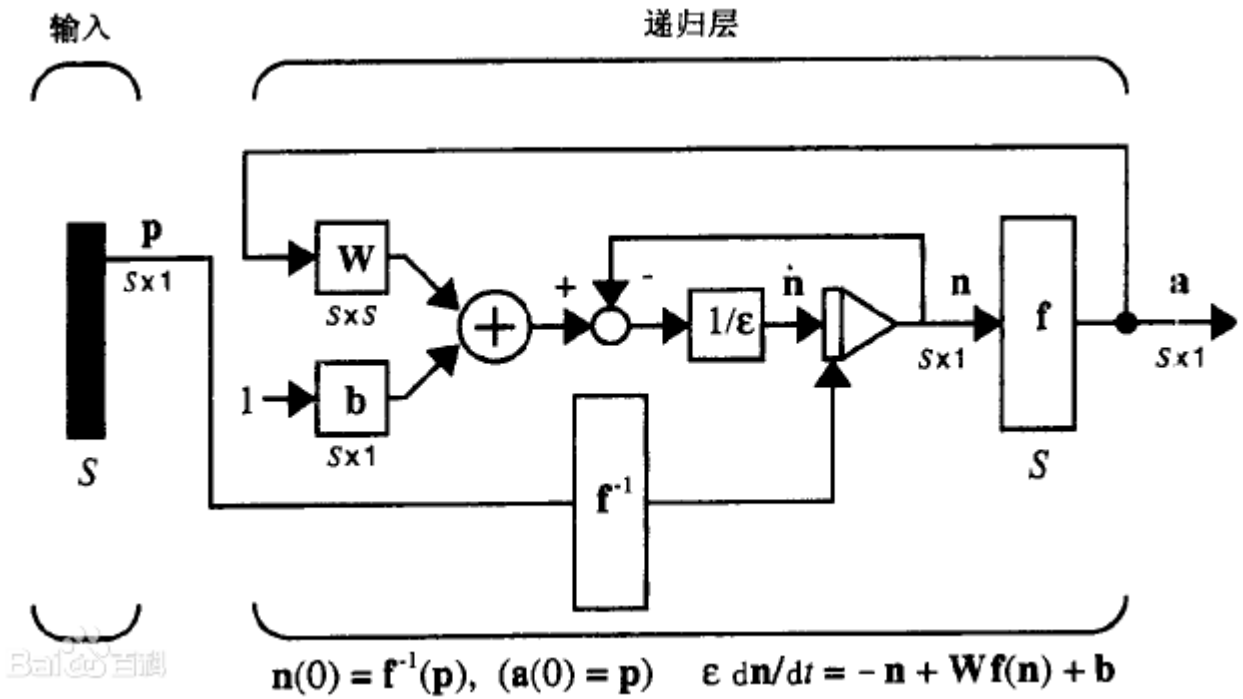
例如：反向传播或BP（Back-Propagation）神经网络



8.2.3 神经网络的结构

- **2. 反馈型神经网络**

例如：Hopfield神经网络



8.2.4 神经网络的学习

- 1944年，Hebb提出了改变神经元连接强度的Hebb学习规则。
  - 权重调整量：$delta(W_j) = k \times f(W_j^T X)X$
  - 权值调整量与输入输出的乘积成正比，经常出现的模式将对权向量有较大的影响.
  - Hebb学习规则是一个无监督学习规则，这种学习的结果是使网络能够**提取训练集的统计特性**，从而**把输入信息按照它们的相似性程度划分为若干类**。这一点与人类观察和认识世界的过程非常吻合，人类观察和认识世界在相当程度上就是在根据事物的统计特征进行分类。
  - 其基本思想很容易被接受，但是近年来神经科学发现这种规则没有准确反映神经元在学习过程中触突变化的基本规律。

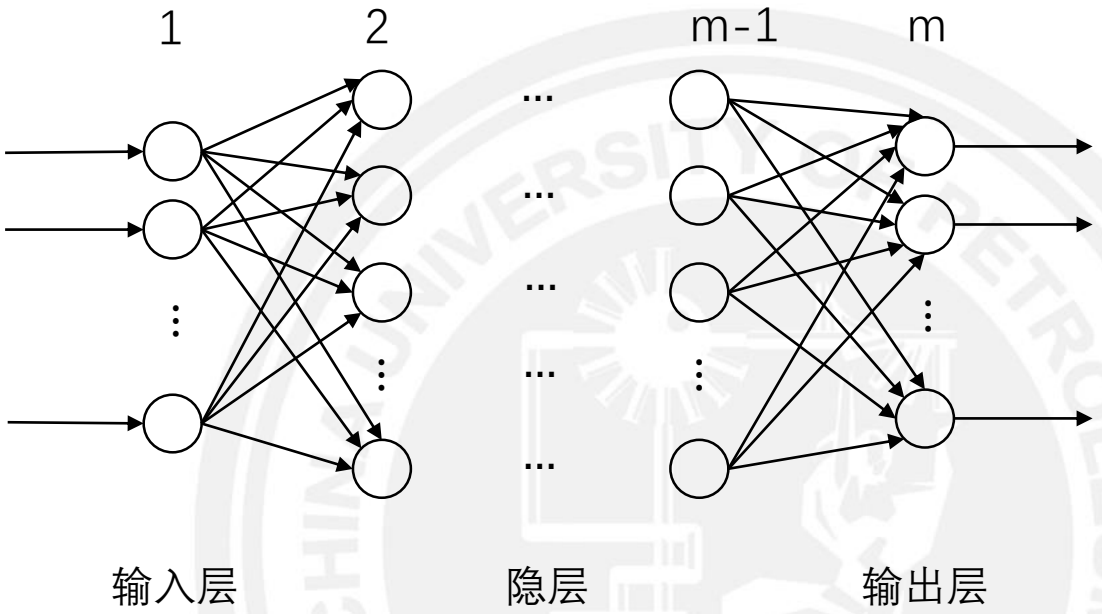
- 20世纪80年代，Rumelhart等人提出多层前向神经网络的BP学习算法。
  - 神经网络的研究取得突破性发展，掀起了机器学习的新高潮。
  - 但BP学习算法只适合训练浅层神经网络。

8.3 BP神经网络及其学习算法

8.3.1 BP神经网络的结构

BP神经网络是多层前向网络，其结构如图所示：

- 设BP神经网络有m层。
- 第一层为**输入层**，最后一层为**输出层**，中间层为**隐层**。
- 输入层输入输出关系一般是**线性函数**。
- 隐层中各个神经元的输入输出关系一般为**非线性关系**。

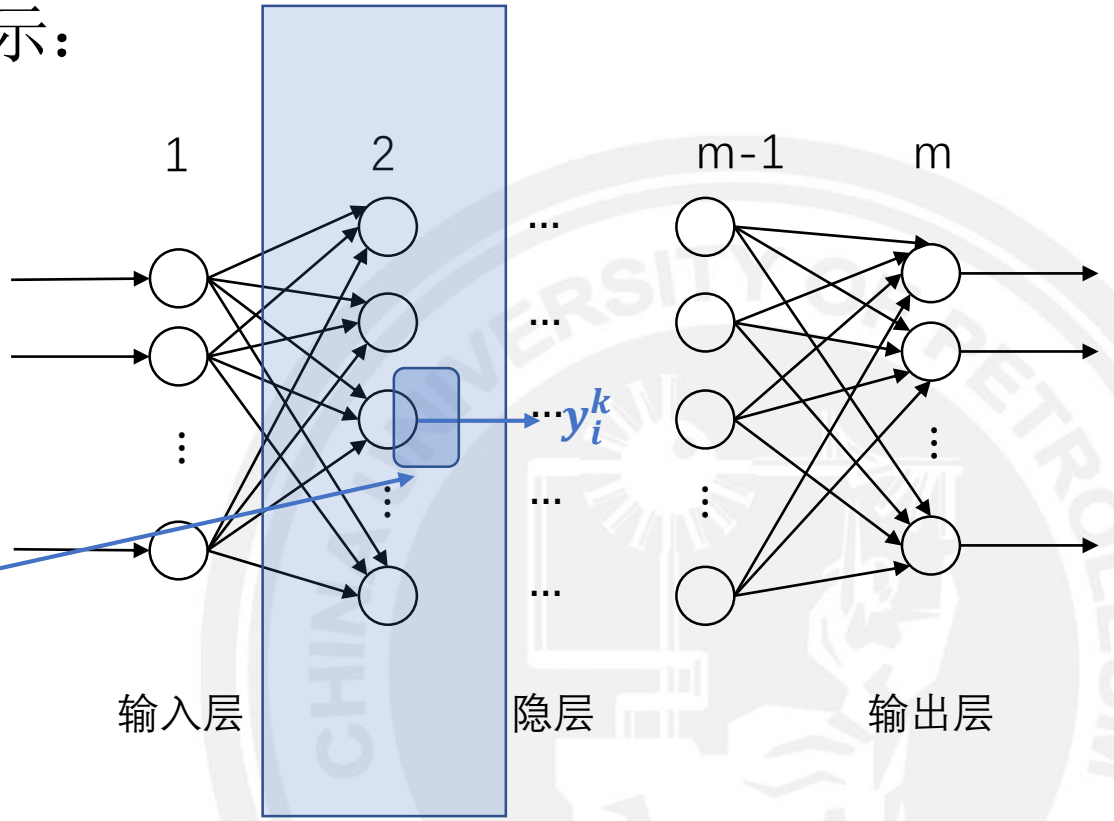


8.3.1 BP神经网络的结构

BP神经网络是多层前向网络，其结构如图所示：

- 隐层k中各个神经元的输入输出关系记为$f_k(k=2,\ldots,m)$，由第$k-1$层的第j个神经元到第k层的第i个神经元的连接权值为w_{ij}^{k-1}，并设第k层第i个神经元输入的总和为u_i^k、输出为y_i^k，则各变量之间的关系为：

$$y_i^k = f_k(u_i^k)$$



8.3.1 BP神经网络的结构

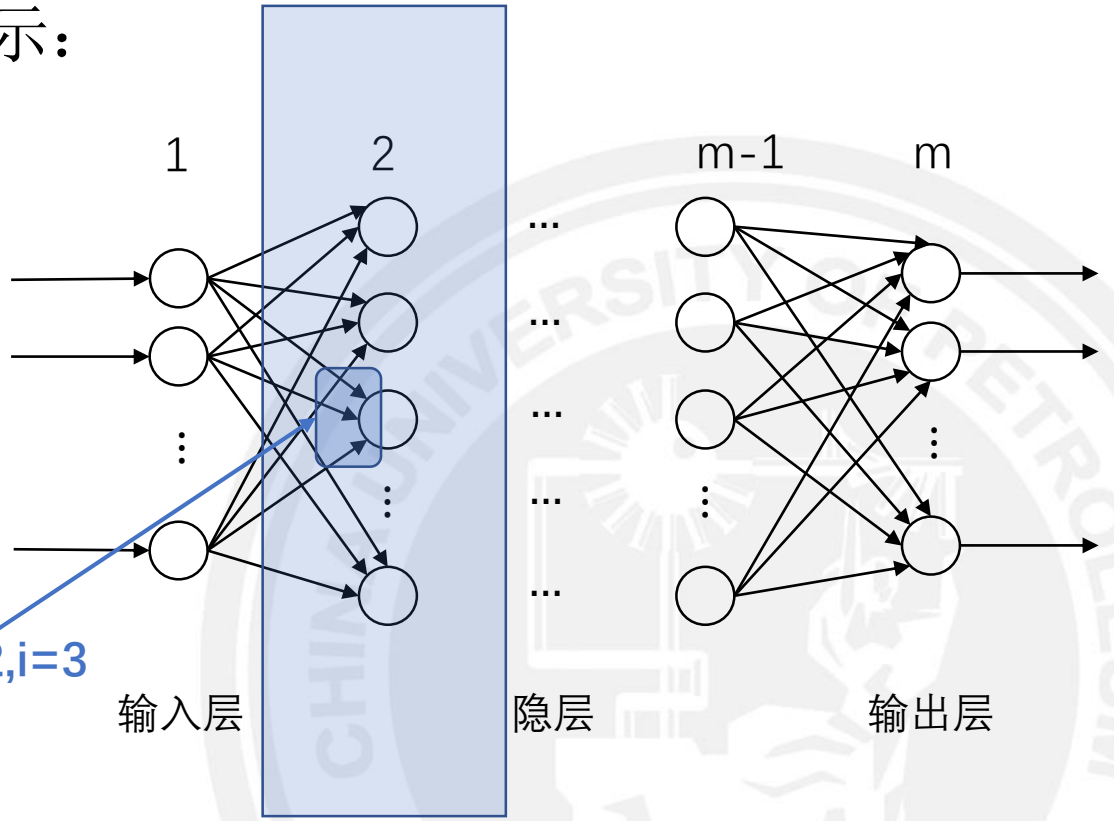
BP神经网络是多层前向网络，其结构如图所示：

- 隐层k中各个神经元的输入输出关系记为$f_k(k=2,\ldots,m)$，由第$k-1$层的第j个神经元到第k层的第i个神经元的连接权值为w_{ij}^{k-1}，并设第k层第i个神经元输入的总和为u_i^k、输出为y_i^k，则各变量之间的关系为：

$$y_i^k = f_k\left(u_i^k\right)$$

$$u_i^k = \sum_j w_{ij}^{k-1} y_j^{k-1} + b_i^k$$

$$K = 2,\ldots,m$$



8.3.1 BP神经网络的结构

- 当BP神经网络输入数据$X=\left[x_1\ \ x_2\ \ \ldots\ \ x_{p_1}\right]^T$（设输入层有$p_1$个神经元），则从输入层依次经过各隐层节点可得到输出数据$Y=\left[y_1^m\ \ y_2^m\ \ \ldots\ \ y_{p_m}^m\right]^T$（设输出层有$p_m$个神经元）。

- 可以把BP神经网络看成是一个从输入到输出的非线性映射。

- 根据Kolmogorov定理，**一个三层的BP神经可以任意精度逼近一个任意给定的连续函数f。**

8.3.2 BP学习算法

BP神经网络的**学习问题**：

给定N组输入输出样本为$\{X_{si}, Y_{si}\}, i = 1,2,\cdots,N$，$s$是向量的维数。如何调整BP神经网络的权值，使BP神经网络输入为样本X_{si}时，神经网络的输出为样本Y_{si}？

- BP学习算法通过**反向学习**过程使误差最小，即**选择神经网络权值使期望输出Y_{si}与神经网络实际输出Y_i^m之差的平方和最小**。

- 利用非线性规划中的“**最快下降法**”，**使权值沿目标函数的负梯度方向改变**。

8.3.2 BP学习算法

- Chain rule

$$\frac{\partial E_{total}}{\partial \omega} = \frac{\partial E_{total}}{\partial out} \cdot \frac{\partial out}{\partial net} \cdot \frac{\partial net}{\partial \omega}$$

E_{total} :总误差

out：神经元输出f（x）

net：网络输入wx+b

8.3.2 BP学习算法

- 网络参数的优化方法：如果神经元的非线性函数取S型函数，则参数按照如下公式更新：

$$\Delta w_{ij}^{k-1} = -\varepsilon d_i^k y_j^{k-1}$$，其中，ε为学习步长，一般小于0.

输出层权值调整公式

  - $k = m$时，即对输出层，$d_i^m = y_i^m(1 - y_i^m)(y_i^m - y_{si})$
  - 对其他隐层，$d_i^k = y_i^k(1 - y_i^k)\sum_l d_l^{k+1} w_{li}^k \quad (k = m-1, \cdots, 2)$

隐层权值调整公式

    - 从公式可以看出，求第k层的误差信号d_i^k需要上一层

- 链式法则：$\frac{\partial E_{total}}{\partial \omega} = \frac{\partial E_{total}}{\partial out} \cdot \frac{\partial out}{\partial net} \cdot \frac{\partial net}{\partial \omega}$　　$y_i^k = f_k(u_i^k) \quad u_i^k = \sum_j w_{ij}^{k-1} y_j^{k-1}$
- Sigmoid函数求导结果：$\sigma'(x) = \sigma(x)(1 - \sigma(x))$

8.3.3 BP学习算法的实现

BP学习算法的设计：

（1）隐层数及隐层**神经元数**的确定，**目前尚无理论指导**。

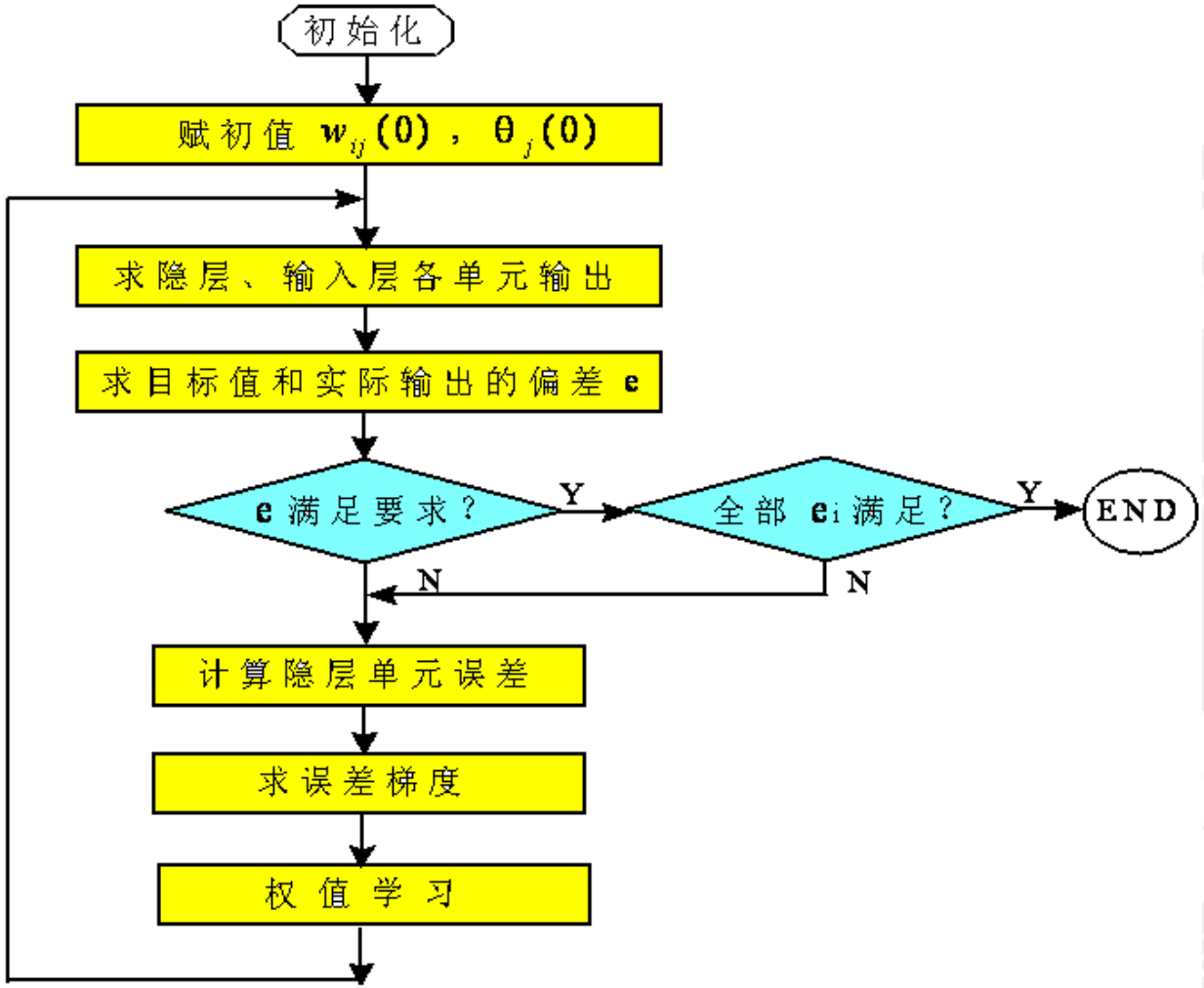
（2）**初始权值**的设置：一般以一个均值为0的随机分布设置网络的初始权值。

（3）训练**数据预处理**：线性的特征比例变换，将所有的特征变换到[0，1]或者[-1，1]区间内，使得在每个训练集上，每个特征的均值为0，并且具有相同的方差。

（4）**后处理**过程：当应用神经网络进行分类操作时，通常将输出值编码成所谓的名义变量，具体的值对应类别标号。

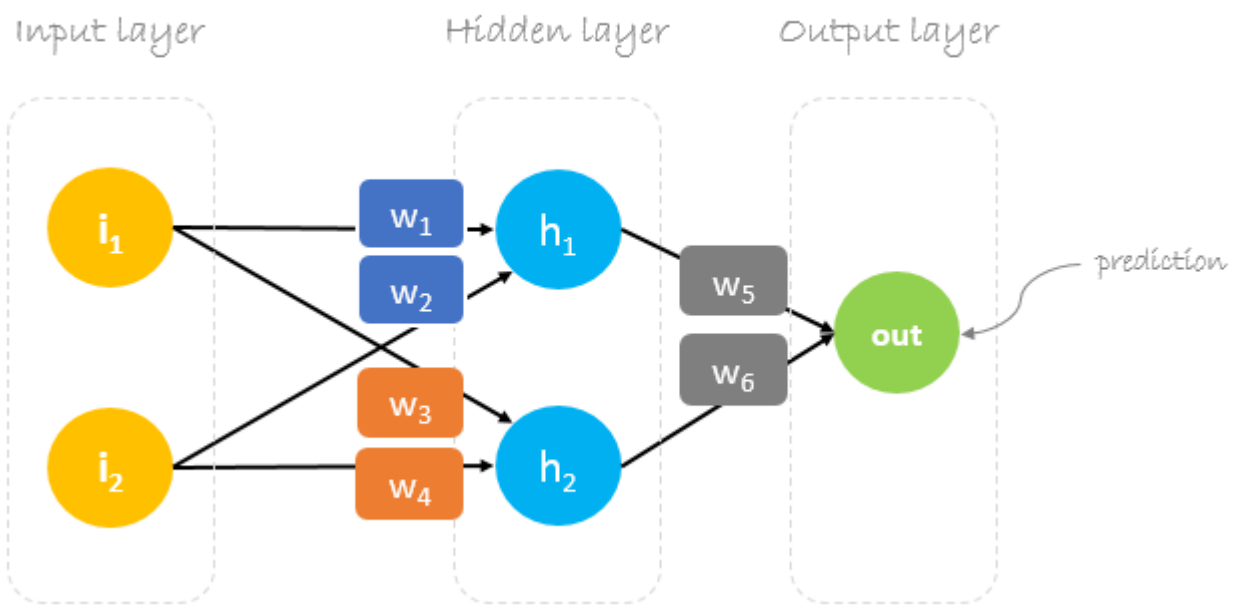
8.3.3 BP学习算法的实现

BP学习算法的程序框架：



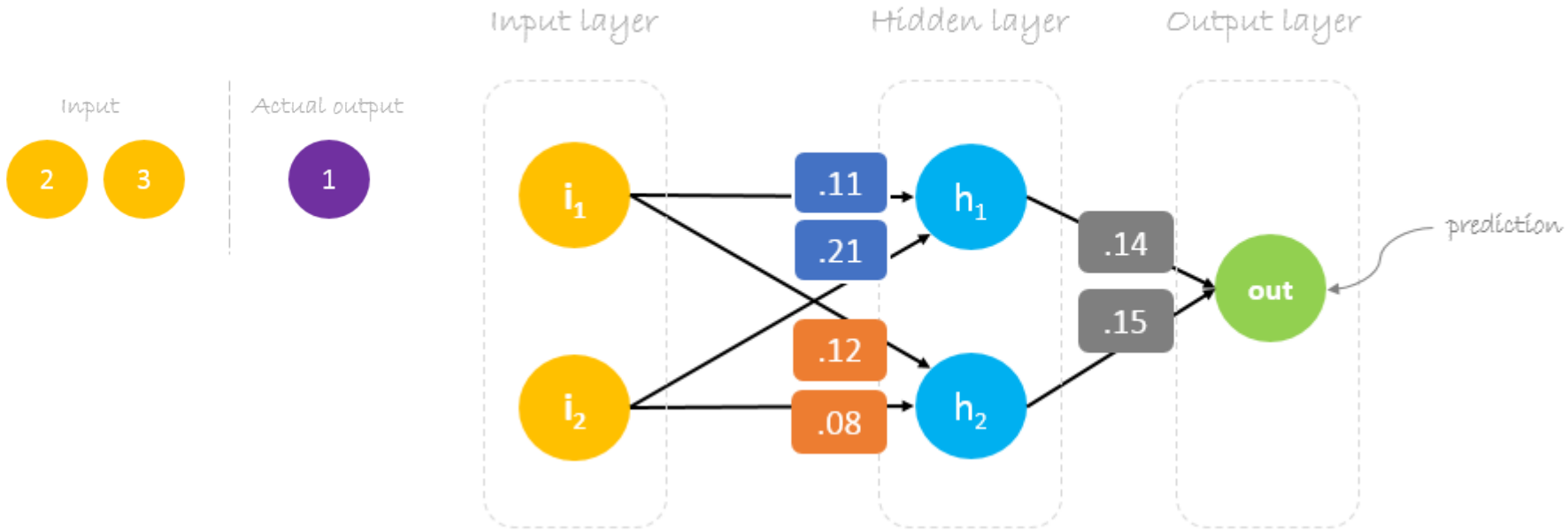
8.3.3 BP学习算法的实现

BP学习算法的实例：



8.3.3 BP学习算法的实现

BP学习算法的实例：



$$u_i^k = \sum_j w_{ij}^{k-1} y_j^{k-1} + b_i^k$$

$$y_i^k = f_k(u_i^k)$$

prediction = out

prediction = $(h_1)\, w_5 + (h_2)\, w_6$

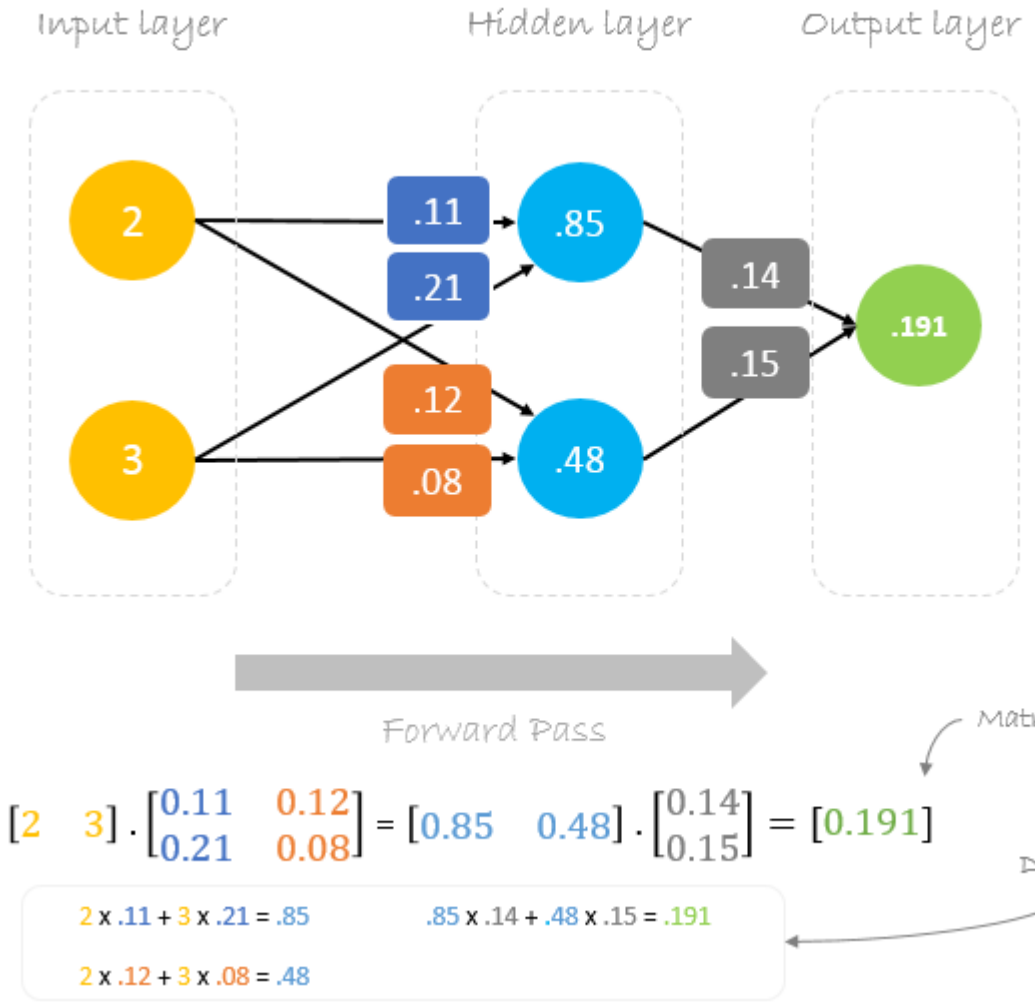
$h_1 = i_1 w_1 + i_2 w_2$

$h_2 = i_1 w_3 + i_2 w_4$

prediction = $(i_1 w_1 + i_2 w_2)\, w_5 + (i_1 w_3 + i_2 w_4)\, w_6$

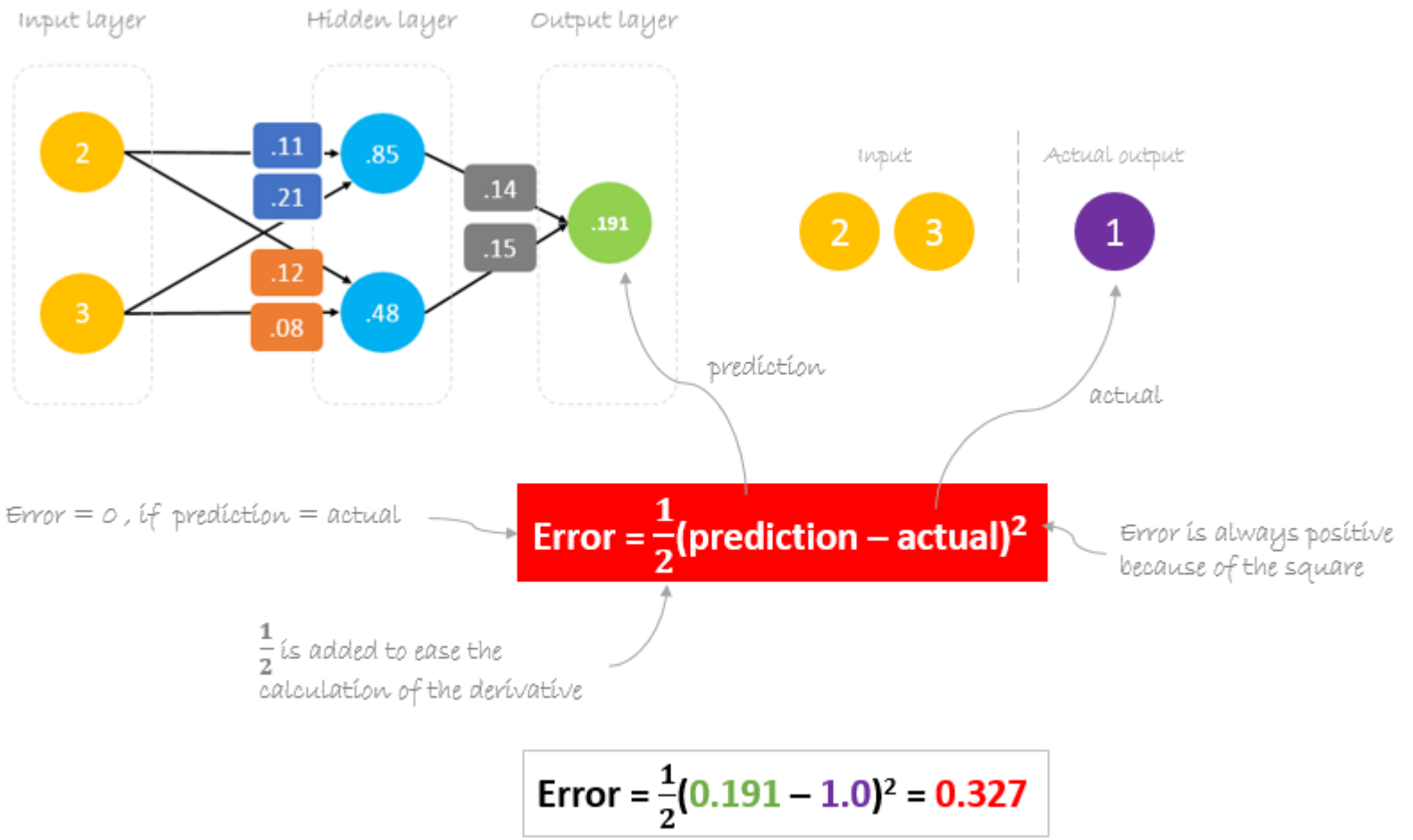
8.3.3 BP学习算法的实现

BP学习算法的实例：



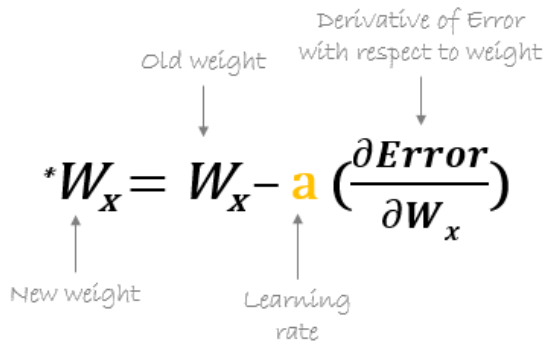
8.3.3 BP学习算法的实现

BP学习算法的实例：



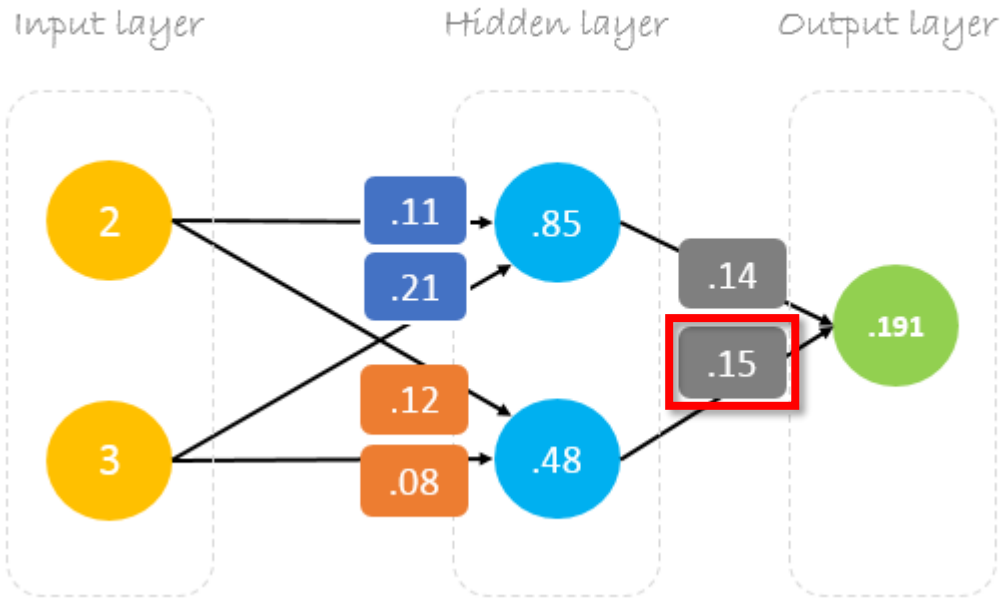
8.3.3 BP学习算法的实现

BP学习算法的实例：



$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

$$^{*}W_6 = W_6 - a\left(\frac{\partial Error}{\partial W_6}\right)$$



$$\frac{\partial Error}{\partial W_6} = \frac{\partial Error}{\partial prediction} * \frac{\partial prediction}{\partial W_6}$$ chain rule

Error = $\frac{1}{2}$(prediction − actual)²

prediction = $(i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6$

$$\frac{\partial Error}{\partial W_6} = \frac{\frac{1}{2}(predictoin - actula)^2}{\partial prediciton} * \frac{\partial (i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6}{\partial W_6}$$

$$\frac{\partial Error}{\partial W_6} = 2 * \frac{1}{2}(predictoin - actula) \frac{\partial (predictoin - actula)}{\partial prediciton} * (i_1 w_3 + i_2 w_4)$$ $h_2 = i_1 w_3 + i_2 w_4$

$$\frac{\partial Error}{\partial W_6} = (predictoin - actula) * (h_2)$$

Δ = prediction − actual ← delta

$$\frac{\partial Error}{\partial W_6} = \Delta h_2$$

8.3.3 BP学习算法的实现

$$\frac{\partial Error}{\partial W_6} = \frac{\partial Error}{\partial prediction} * \frac{\partial prediction}{\partial W_6}$$

chain rule

$$Error = \frac{1}{2}(prediction - actual)^2$$

$$prediction = (i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6$$

$$\frac{\partial Error}{\partial W_6} = \frac{\frac{1}{2}(predictoin - actula)^2}{\partial prediciton} * \frac{\partial \left(i_1 w_1 + i_2 w_2\right) w_5 + \left(i_1 w_3 + i_2 w_4\right) w_6}{\partial W_6}$$

$$\frac{\partial Error}{\partial W_6} = 2 * \frac{1}{2}(predictoin - actula) \frac{\partial (predictoin - actula)}{\partial prediciton} * (i_1 w_3 + i_2 w_4)$$

$$h_2 = i_1 w_3 + i_2 w_4$$

$$\frac{\partial Error}{\partial W_6} = (predictoin - actula) * (h_2)$$

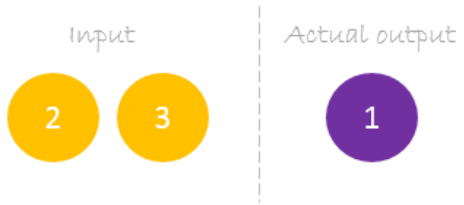
$$\Delta = prediction - actual$$

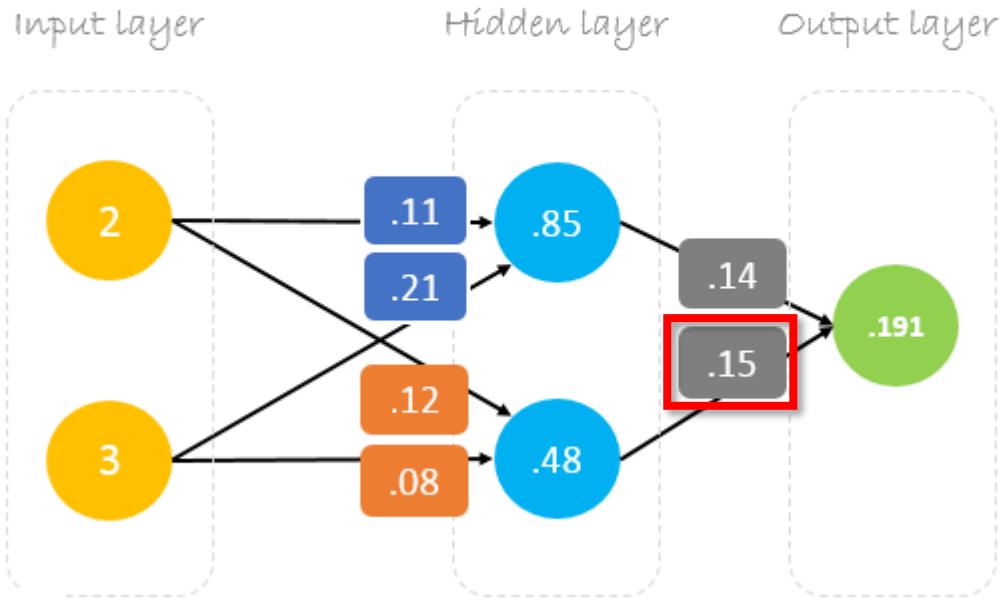
delta

$$\frac{\partial Error}{\partial W_6} = \Delta h_2$$

8.3.3 BP学习算法的实现

BP学习算法的实例：





$$\frac{\partial Error}{\partial W_6} = \frac{\partial Error}{\partial prediction} * \frac{\partial prediction}{\partial W_6}$$

$$\frac{\partial Error}{\partial W_6} = \frac{\frac{1}{2}(predictoin-actula)^2}{\partial prediciton} * \frac{\partial (i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6}{\partial W_6}$$

$$\frac{\partial Error}{\partial W_6} = 2 * \frac{1}{2}(\mathbf{predictoin - actula}) \frac{\partial (predictoin-actula)}{\partial prediciton} * (i_1 w_3 + i_2 w_4)$$

$$\frac{\partial Error}{\partial W_6} = (\mathbf{predictoin - actula}) * (h_2)$$

$$\frac{\partial Error}{\partial W_6} = \Delta h_2$$

$$^*W_6 = W_6 - a\left(\frac{\partial Error}{\partial W_6}\right)$$

$$^*w_6 = w_6 - a\,(h_2 \,.\, \Delta) \qquad ^*w_5 = w_5 - a\,(h_1 \,.\, \Delta)$$

$$\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - a\,\Delta \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - \begin{bmatrix} a h_1 \Delta \\ a h_2 \Delta \end{bmatrix}$$

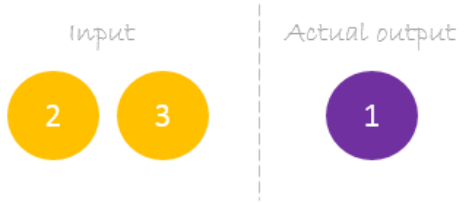
$\Delta = 0.191 - 1 = -0.809$ ← **Delta** = prediction - actual

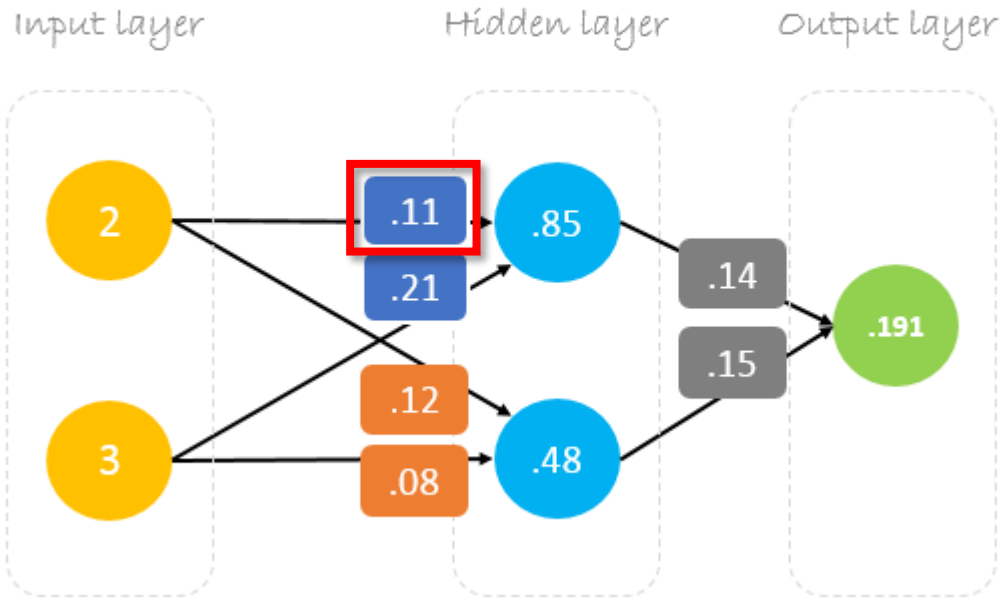
$a = 0.05$ ← **Learning rate**, we smartly guess this number

$$\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 0.85 \\ 0.48 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - \begin{bmatrix} -0.034 \\ -0.019 \end{bmatrix} = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

8.3.3 BP学习算法的实现

BP学习算法的实例：





$$\frac{\partial Error}{\partial W_1} = \frac{\partial Error}{\partial prediction} * \frac{\partial prediction}{\partial h_1} * \frac{\partial h_1}{\partial W_1}$$

$$\frac{\partial Error}{\partial W_1} = \frac{\partial \frac{1}{2}(predictoin-actula)^2}{\partial prediciton} * \frac{\partial (h_1)\, w_5 + (h_2)\, w_6}{\partial h_1} * \frac{\partial\, i_1 w_1 + i_2 w_2}{\partial w_1}$$

$$\frac{\partial Error}{\partial W_1} = 2 * \frac{1}{2}(\mathbf{predictoin - actula}) \frac{\partial (predictoin-actula)}{\partial prediciton} * (w_5) * (i_1)$$

$$\frac{\partial Error}{\partial W_1} = (\mathbf{predictoin - actula}) * (w_5 i_1)$$

$$\frac{\partial Error}{\partial W_1} = \Delta\, w_5 i_1$$

$$^*W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

$$^*w_4 = w_4 - a\,(i_2 \,.\, \Delta w_6)$$
$$^*w_3 = w_3 - a\,(i_1 \,.\, \Delta w_6)$$
$$^*w_2 = w_2 - a\,(i_2 \,.\, \Delta w_5)$$
$$^*w_1 = w_1 - a\,(i_1 \,.\, \Delta w_5)$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - a\,\Delta \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} . [w_5 \quad w_6] = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - \begin{bmatrix} a\, i_1 \Delta w_5 & a\, i_1 \Delta w_6 \\ a\, i_2 \Delta w_5 & a\, i_2 \Delta w_6 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 2 \\ 3 \end{bmatrix} . [0.14 \quad 0.15] = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - \begin{bmatrix} -0.011 & -0.012 \\ -0.017 & -0.018 \end{bmatrix} = \begin{bmatrix} .12 & .13 \\ .23 & .10 \end{bmatrix}$$

8.3.3 BP学习算法的实现

$$\frac{\partial Error}{\partial W_1} = \frac{\partial Error}{\partial prediction} * \frac{\partial prediction}{\partial h_1} * \frac{\partial h_1}{\partial W_1}$$

chain rule

$$\frac{\partial Error}{\partial W_1} = \frac{\partial \frac{1}{2}(predictoin - actula)^2}{\partial prediciton} * \frac{\partial (h_1)\, w_5 + (h_2)\, w_6}{\partial h_1} * \frac{\partial\, i_1 w_1 + i_2 w_2}{\partial w_1}$$

$$\frac{\partial Error}{\partial W_1} = 2 * \frac{1}{2}(\mathbf{predictoin - actula}) \frac{\partial (predictoin - actula)}{\partial prediciton} * (w_5) * (i_1)$$

$$\frac{\partial Error}{\partial W_1} = (\mathbf{predictoin - actula}) * (w_5 i_1)$$

$$\frac{\partial Error}{\partial W_1} = \Delta\, w_5 i_1$$

$\mathbf{Error = \frac{1}{2}(prediction - actual)^2}$

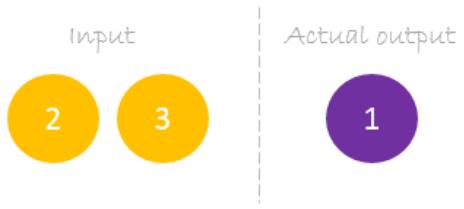
$\mathbf{prediction} = (h_1)\, w_5 + (h_2)\, w_6$

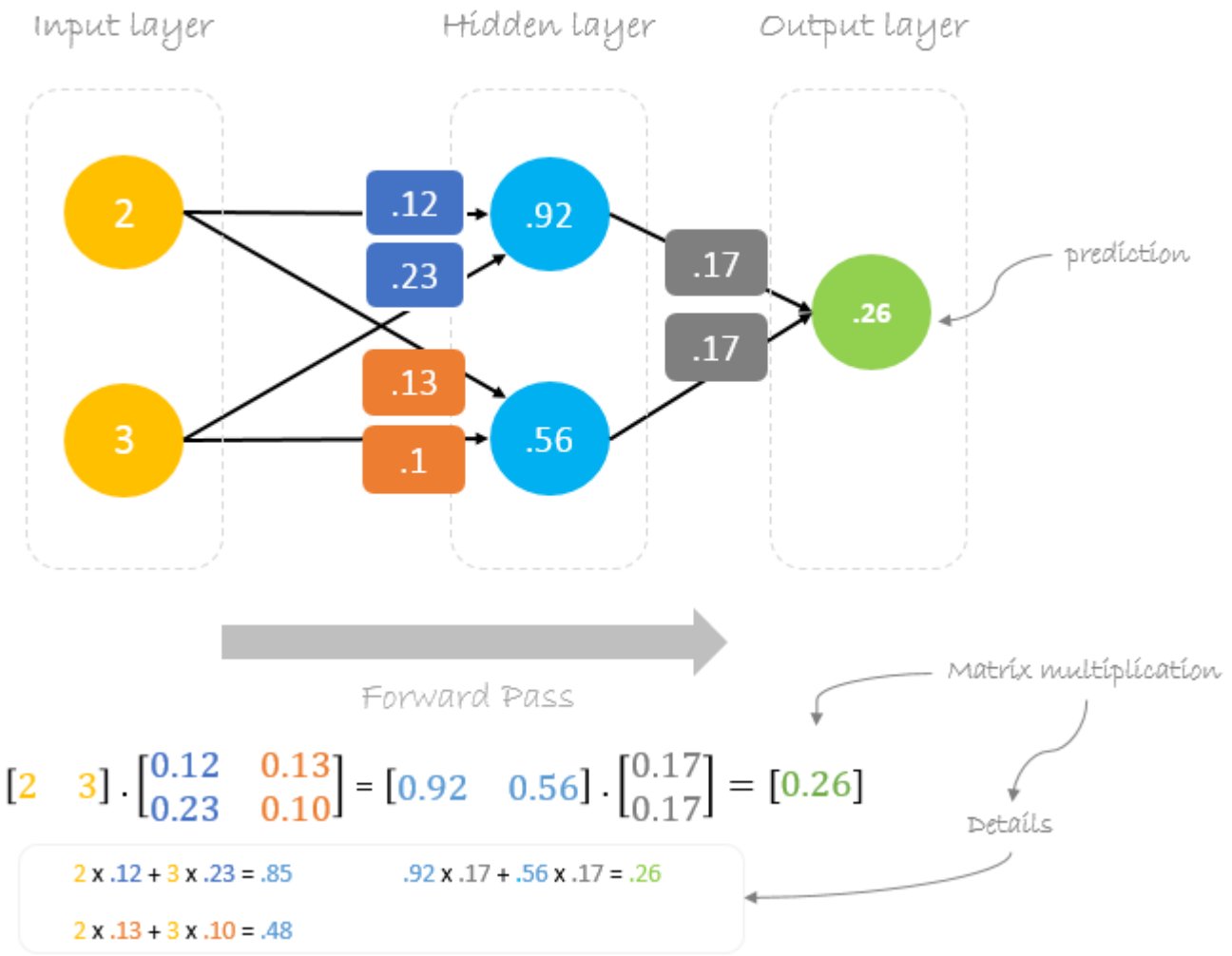
$h_1 = i_1 w_1 + i_2 w_2$

Δ = prediction − actual ← delta

8.3.3 BP学习算法的实现

BP学习算法的实例：



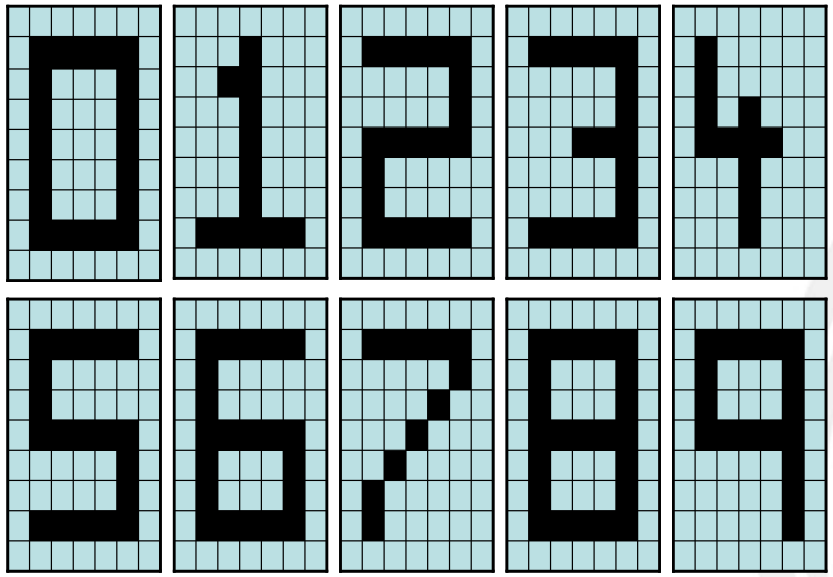


8.3.4 BP神经网络在模式识别中的应用

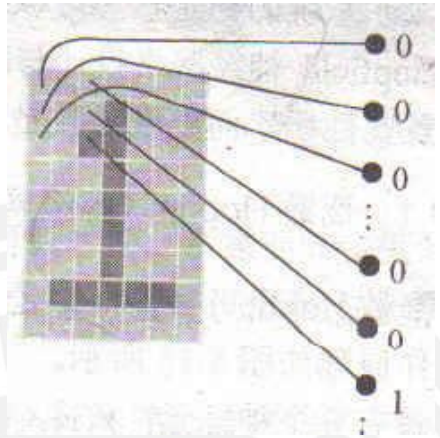
- 模式识别研究用计算机模拟生物、人的感知，对模式信息，如图像、文字、语音等，进行识别和分类。
- 传统人工智能的研究部分地显示了人脑的归纳、推理等智能。但是，对于人类底层的智能，如视觉、听觉、触觉等方面，现代计算机系统的信息处理能力还不如一个幼儿园的孩子。
- 神经网络模型模拟了人脑神经系统的特点：处理单元的广泛连接；并行分布式信息储存、处理；自适应学习能力等。
- 神经网络模式识别方法具有**较强的容错能力**、**自适应学习能力**、**并行信息处理能力**。

8.3.4 BP神经网络在模式识别中的应用

例：设计一个三层BP网络对数字0至9进行分类。



8.3.4 BP神经网络在模式识别中的应用



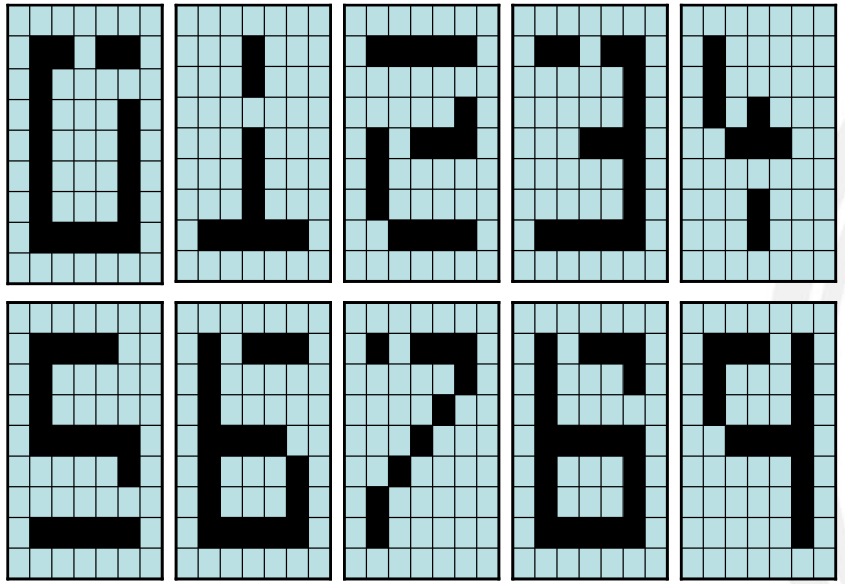
例：设计一个三层BP网络对数字0至9进行分类。

- 每个数字用9×7的网格表示，灰色像素代表0，黑色像素代表1。将每个网格表示为0，1的长位串。位映射由左上角开始向下直到网格的整个一列，然后重复其他列。
- 选择BP网络结构为63-6-9。97个输入结点，对应上述网格的映射。9个输出结点对应10种分类。
- 使用的学习步长为0.3。训练600个周期，如果输出结点的值大于0.9，则取为ON，如果输出结点的值小于0.1，则取为OFF。

8.3.4 BP神经网络在模式识别中的应用

例：设计一个三层BP网络对数字0至9进行分类。

- 当训练成功后，对如图所示测试数据进行测试。测试数据都有一个或者多个位丢失。



8.3.4 BP神经网络在模式识别中的应用

例：设计一个三层BP网络对数字0至9进行分类。

- 测试结果表明：除了8以外，所有被测的数字都能够被正确地识别。
- 对于数字8，神经网络的第6个结点的输出值为0.53，第8个结点的输出值为0.41，表明第8个样本是模糊的，可能是数字6，也可能是数字8，但也不完全确信是两者之一。

8.4 卷积神经网络

8.4.1 人脑视觉机理

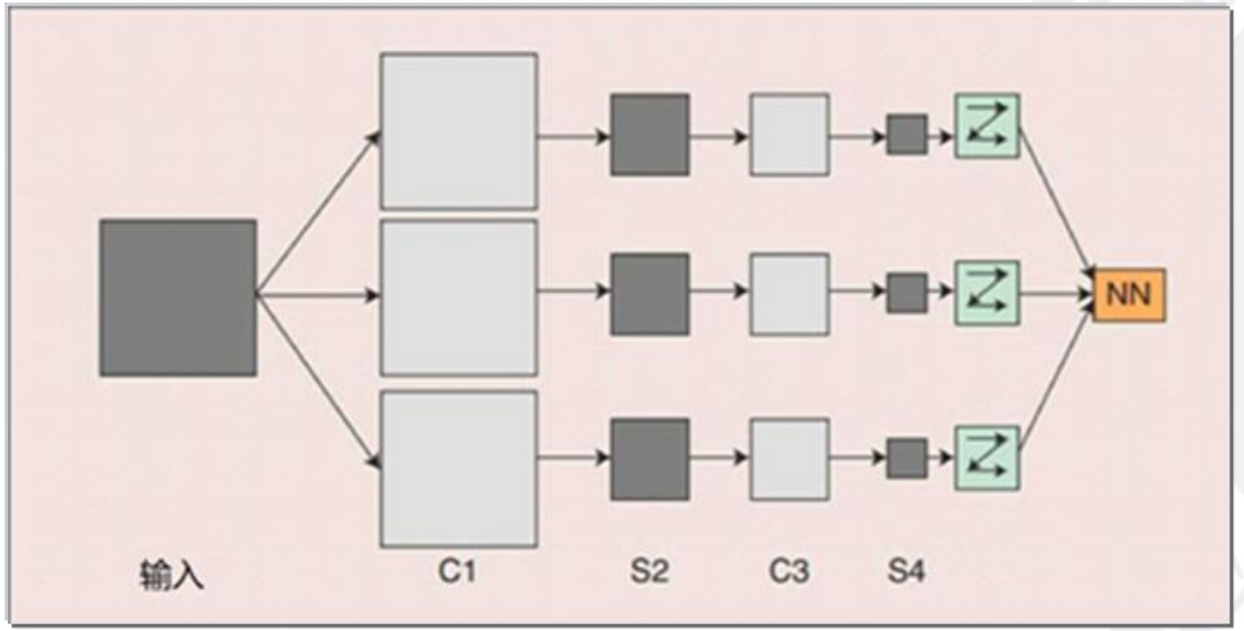
- 1962年Hubel和Wiesel通过对猫视觉皮层细胞的研究，提出了**感受野**(receptive field)的概念。视觉皮层的神经元就是**局部接受信息**的，**只受某些特定区域刺激的响应，而不是对全局图像进行感知**。

- 1984年日本学者Fukushima基于感受野概念提出**神经认知机**(neocognitron)。神经认知机可通过无监督学习来学习识别简单的图像。

- CNN可看作是神经认知机的推广形式。最早开发出 CNN 的 AI 研究者 Yann LeCun 明确表明他们的开发根基于神经认知机，参阅：https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf。

8.4.2 卷积神经网络的结构

- CNN是一个多层的神经网络，每层由多个二维平面组成，而每个平面由多个独立神经元组成。
  1、C层为**特征提取层**（**卷积层**）
  2、S层是**特征映射层**（**下采样层**）
  - CNN中的每一个C层都紧跟着一个S层。

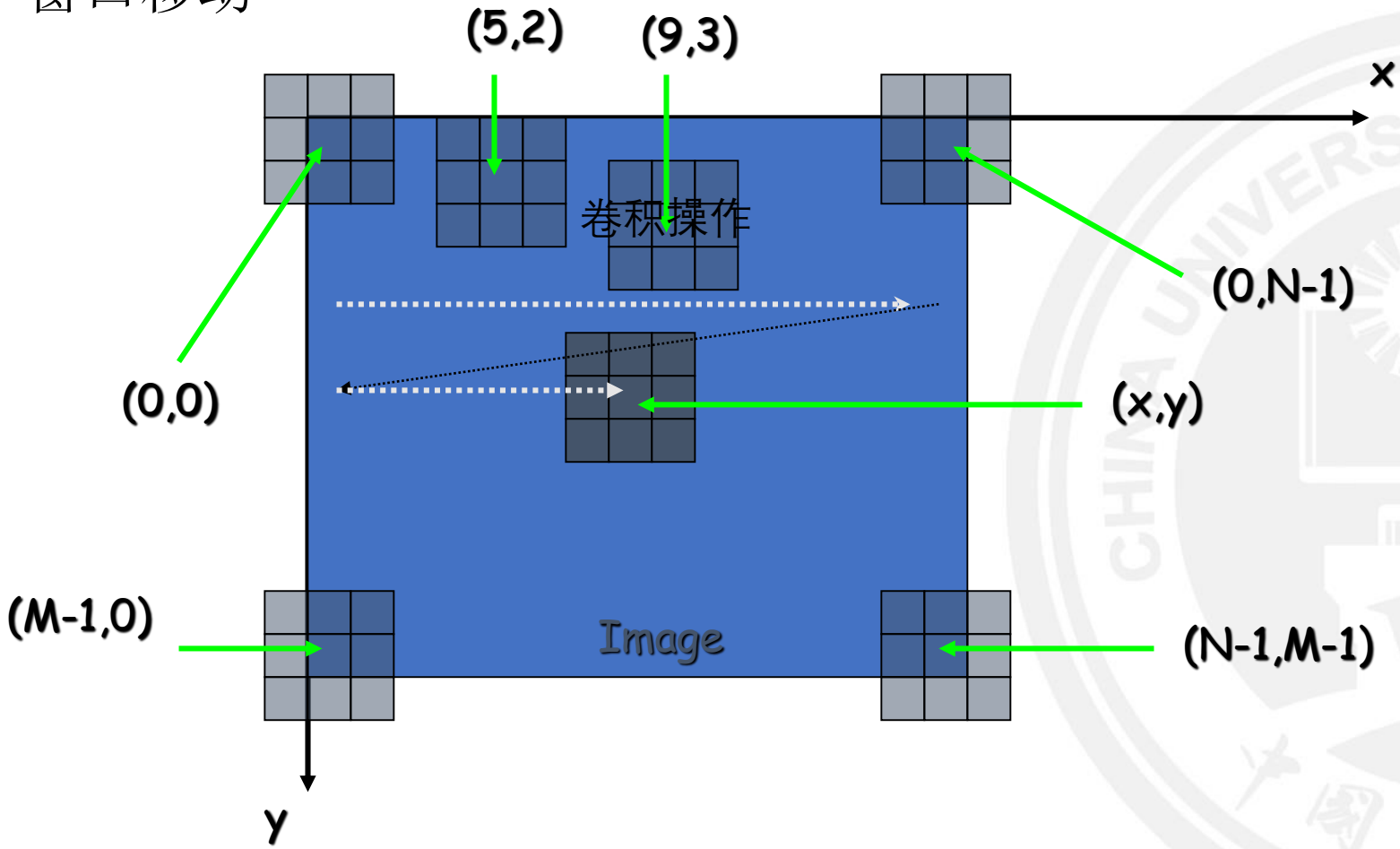


8.4.2 卷积神经网络的结构

- 特征提取层（卷积层）——C层（Convolution layer）
  - 对图像用一个卷积核进行卷积运算，实际上是一个**滤波**的过程。每个卷积核都是一种特征提取方式，就像是一个筛子，将图像中符合条件的部分筛选出来。
  - 卷积运算是一种用邻域点按一定权重去重新定义该点值的运算。卷积实际上提供了一个**权重模板**。

8.4.2 卷积神经网络的结构

- 特征提取层（卷积层）——C层（Convolution layer）
  - 卷积：窗口移动



8.4.2 卷积神经网络的结构

- 特征提取层（卷积层）——C层（Convolution layer）
  - 卷积：以图像进行卷积处理

-1	-1	-1
0	0	0
1	1	1

卷积核

0	1	3	2	1	3	2	1
0	5	7	6	2	5	7	6
1	6	0	6	1	6	3	1
2	6	7	5	3	5	6	5
3	2	2	7	2	6	1	6
2	6	5	0	2	3	5	2
1	2	3	2	1	2	4	2
3	1	2	3	1	2	0	1

原图f(x,y)

8.4.2 卷积神经网络的结构

- 特征提取层（卷积层）——C层（Convolution layer）
  - 卷积：以图像进行卷积处理

-1	-1	-1
0	0	0
1	1	1

卷积核

0	1	3	2	1	3	2	1
0	5	7	6	2	5	7	6
1	6	0	6	1	6	3	1
2	6	7	5	3	5	6	5
3	2	2	7	2	6	1	6
2	6	5	0	2	3	5	2
1	2	3	2	1	2	4	2
3	1	2	3	1	2	0	1

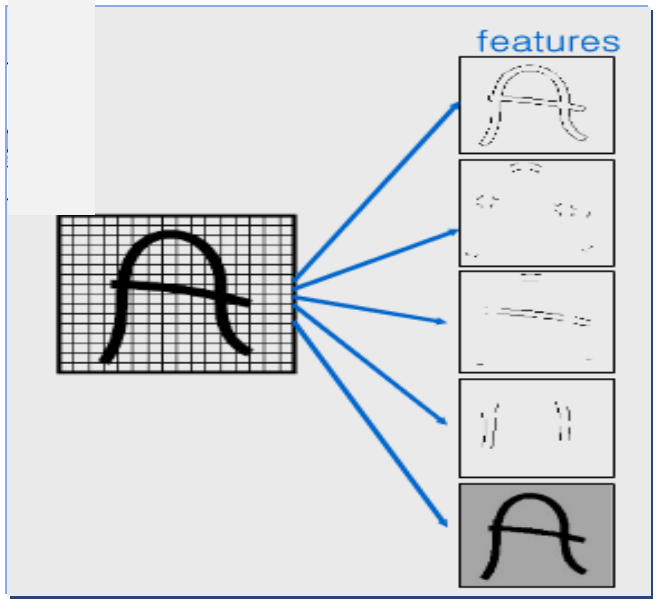
原图f(x,y)

3	6	1	7	4	4
3	0	0	0	0	2
0	1	4	2	1	3
2	7	8	8	4	6
1	4	5	10	2	5
7	5	1	1	7	7

处理后的结果g(x,y)

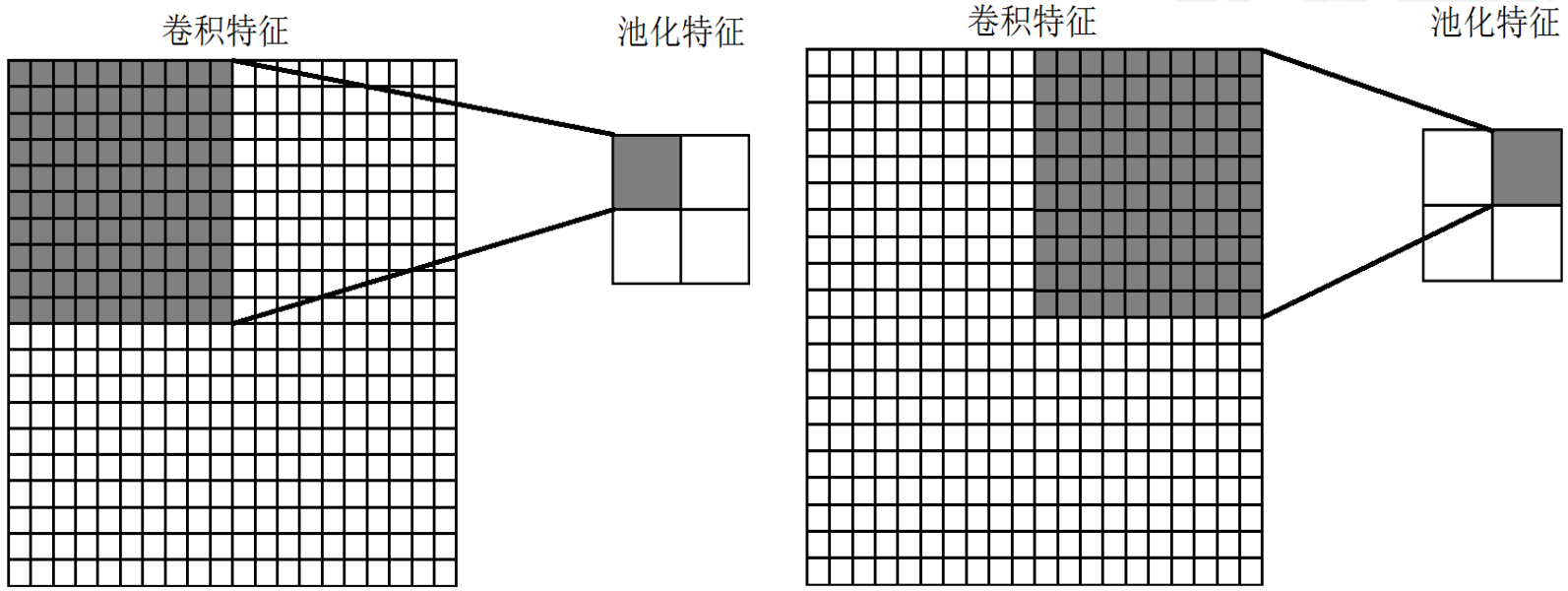
8.4.2 卷积神经网络的结构

- 特征提取层（卷积层）——C层（Convolution layer）
  - 大部分的特征提取都依赖于卷积运算。
  - 利用卷积算子对图像进行滤波，可以得到显著的**边缘特征**。



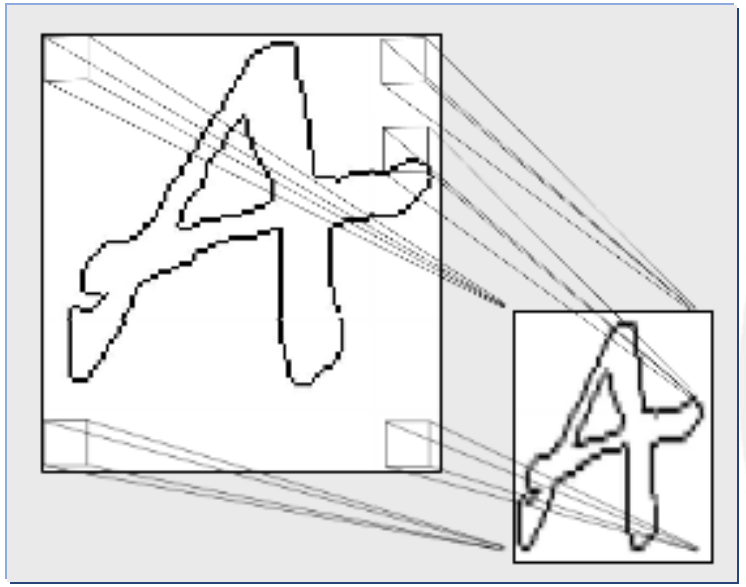
8.4.2 卷积神经网络的结构

- 特征映射层（下采样层）——S层（ Subsampling layer）
  - 池化的目的是降低特征空间的维度，只抽取局部最显著的特征，这些特征出现的具体位置也被忽略
  - 降低了计算量，对噪声具有健壮性
  - 最常用的有最大池化、平均池化



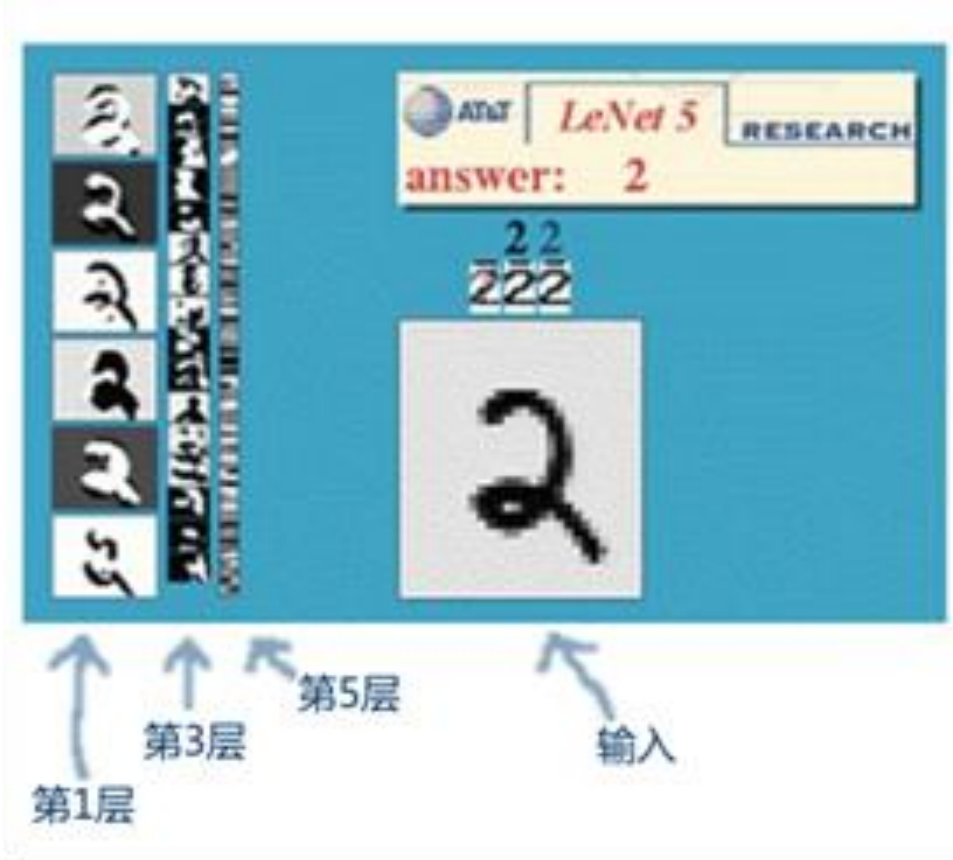
8.4.2 卷积神经网络的结构

- 特征映射层（下采样层）——S层（ Subsampling layer）
  - 下采样层降低了每个特征图的空间分辨率。



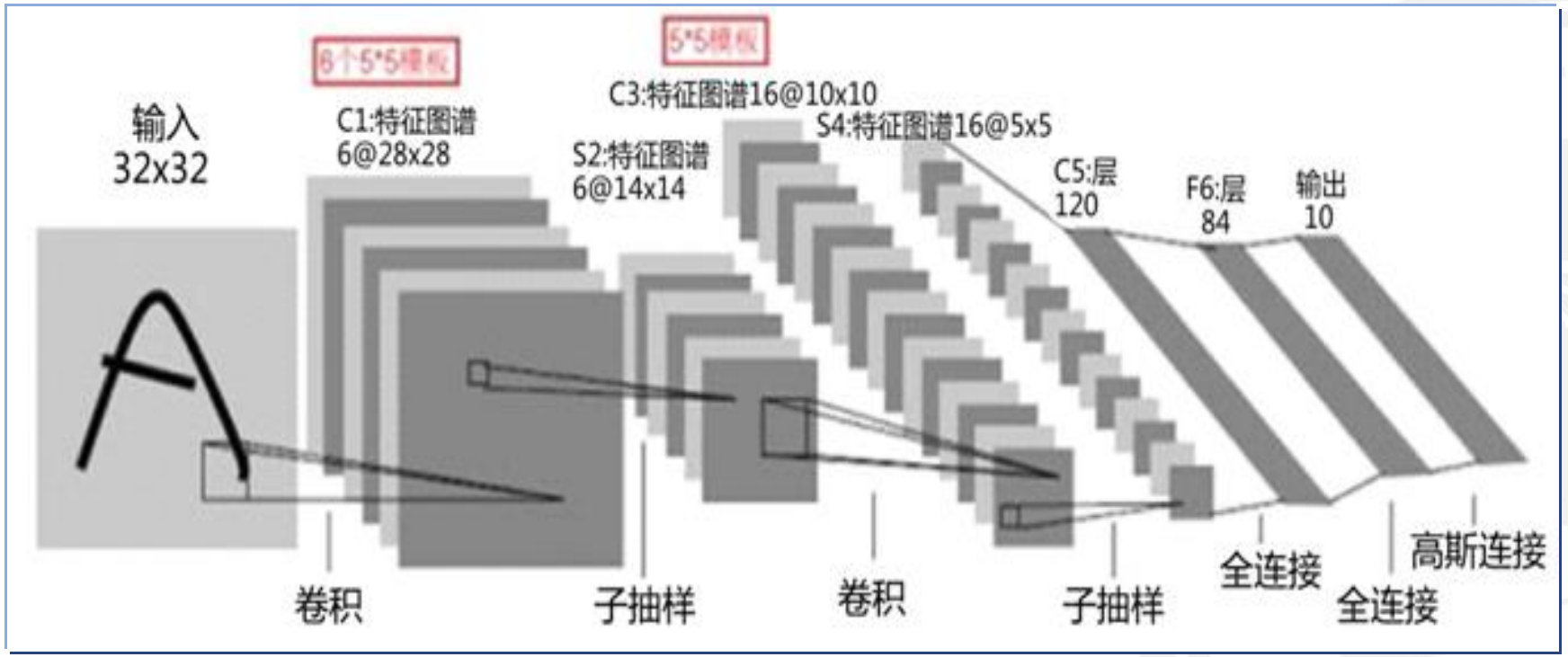
8.4.3 卷积神经网络实例

一种典型的用来识别数字的卷积网络是LeNet-5。美国大多数银行当年用它识别支票上面的手写数字，达到了商用地步，说明该算法具有很高的准确性



8.4.3 卷积神经网络实例

- LeNet-5是一个数字手写系统，不包含输入层，共有7层，每层都包含可训练参数（连接权重）。输入图像为32*32大小。



8.5 生成对抗网络

8.5.1 生成对抗网络的基本原理

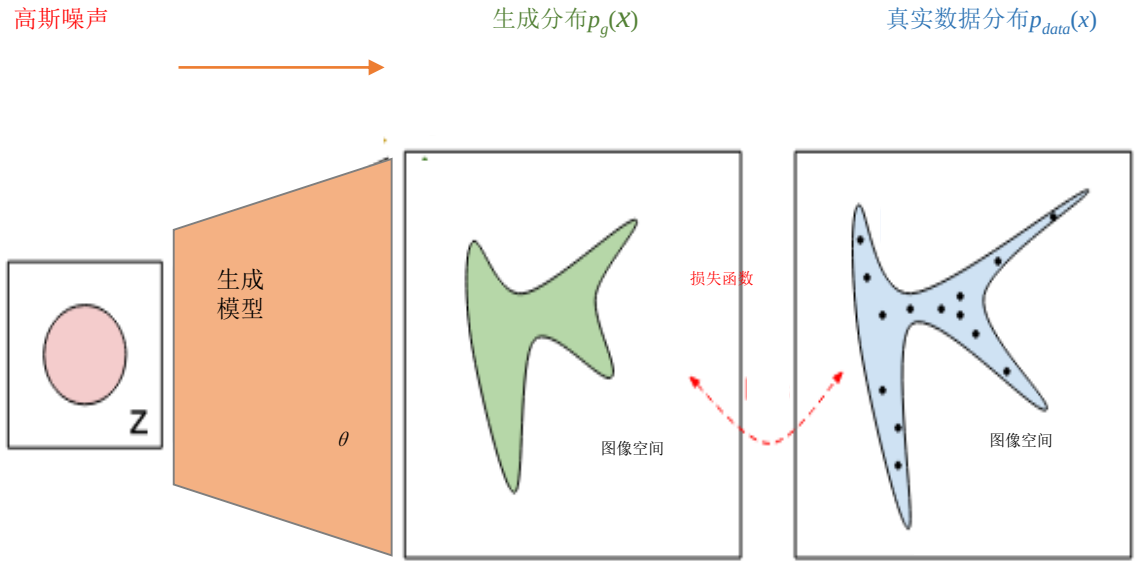
8.5.2 生成对抗网络的结构

8.5.1 生成对抗网络的基本原理

- 深度学习的模型可大致分为**判别式模型**和**生成式模型**
  - Goodfellow等于2014年提出一种新型生成式模型——**生成对抗网络**（GAN, generative adversarial network），使用对抗训练机制对两个神经网络进行训练。
- 目前深度学习取得的成果主要集中在**判别式模型**：**即将一个高维的感官输入映射为一个类别标签。**
- **生成方法**是学习方法中的一个重要分支，涉及对数据显式或隐式变量的分布假设和对分布参数的学习，基于学习得到的模型采样出新样本。
- **生成式模型**：**通过生成式方法学习得到的模型。**
  - 首先，对真实世界进行建模需要大量先验知识，建模的好坏直接影响生成式模型的性能；
  - 其次，真实世界的数据往往非常复杂，拟合模型所需计算量往往非常庞大，甚至难以承受。

8.5.1 生成对抗网络的基本原理

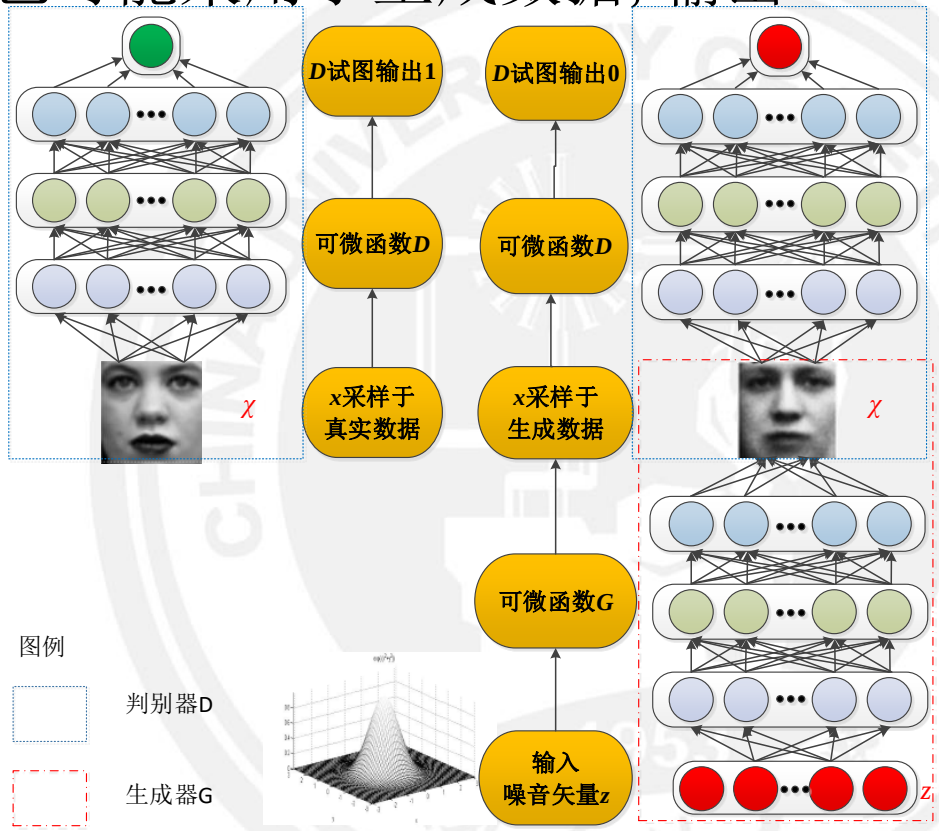
- 生成式模型的概念图：
  - 每个黑点分别表示采样于真实数据分布的一张图像。
  - 蓝色区域表示以高概率包含真实图像的图像空间。
  - 参数化生成模型将一个高斯噪声矢量z影射为一个生成概率分布p_g。并通过优化目标函数调整参数θ，使生存概率分布p_g，尽可能逼近真实数据分布p_{data}，从而准确解释真实数据。



8.5.2 生成对抗网络的结构

- **生成器**的输入是一个来自常见概率分布的随机噪声适量，输出是计算机生成的伪数据。
- **判别器**的输入是图片，可能采用与真实数据，也可能采用于生成数据，输出是一个标量用来代表真实图片的概率。

- 判别器和生成器不断优化，当判别器无法正确区分数据来源时，可以认为生成器捕捉到了真实数据样本的分布。



8.6 深度学习的应用

- 深度学习在博弈中的应用
  - 2016年3月。AlphaGo以4:1战胜韩国棋手李世石，成为第一个击败人类职业围棋选手的电脑程序。
  - 2017年5月AlphaGo在乌镇以3:0完胜柯洁。
- 深度学习在医学影像识别中的应用
  - 黑色素瘤识别中，人的精度为84%，计算机的精度可达到97%。
- 生成对抗网络在图像处理中的应用
  - 生成各种图像、数字、人脸、图像风格迁徙、图像翻译、图像修复、图像上色、人脸图像编辑以及视频生成，构成各种逼真的室内外场景，从物体轮廓恢复物体图像等。
- 生成对抗网络在语言处理中的应用
  - 文本生成；从文本生成图像，即给计算机输入一段文字描述，计算机自动生成与文字描述相近的图片。

THANKS