（请大家预习）

第6章第3讲

人工神经网络

Artificial Neural Networks

张 燕 明

ymzhang@nlpr.ia.ac.cn

people.ucas.ac.cn/~ymzhang

模式分析与学习课题组(PAL)

多模态人工智能系统实验室 中科院自动化所

助教： 杨 奇 (yangqi2021@ia.ac.cn)

张 涛 (zhangtao2021@ia.ac.cn)

内容回顾

- 神经网络模型的监督训练
  - 准则函数的选择
    - 回归：均方误差(MSE)；分类：交叉熵(Cross Entropy)
  - 模型的选择
    - 隐层数、隐层节点数、激活函数
  - 算法：梯度下降(BP)
    - 权重初始化、学习率、冲量项(momentum)
  - 训练中的问题
    - 过拟合：$L2$正则化(regularization, weight decay)、早停(early stopping)
    - 梯度消失：激活函数、权重初始化、BN层
    - 局部极小：引入随机性

内容回顾

- 径向基函数(RBF)网络

$$z_j(\mathbf{x}) = \sum_{h=1}^{H} w_{hj}\phi_h(\mathbf{x}), \quad j = 1, 2, ..., c$$

  - 由多个径向基函数{ $\phi_h(\mathbf{x})$ }线性组合而成
  - 每个RBF $\phi_h(\mathbf{x})$ 包含一个中心 $\mathbf{w}_h$ ； $\mathbf{w}_h$与x的维数相同，通常表示一个原型样本。
  - $\phi_h(\mathbf{x})$ 是$\| \mathbf{x} - \mathbf{w}_h \|$的单调减函数，反映 x 与 $\mathbf{w}_h$ 的相似度

$$\text{例如：} \phi_h(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{w}_h\|^2}{2\sigma^2}\right)$$

  - 确定一个RBF网络需要确定{ w_{ih} }, { w_{hj} }

内容回顾

- RBF vs MLP

RBF：
$$net_h = \|\mathbf{w}_h - \mathbf{x}\|^2,\ y_h = \exp(-\frac{net_h}{2\sigma^2}),\ z_j = \mathbf{w}_j^T\mathbf{x}$$

MLP：
$$net_h = \mathbf{w}_h^T\mathbf{x},\ y_h = \mathrm{ReLU}(net_h),\ z_j = \mathbf{w}_j^T\mathbf{x}$$

与MLP相似，RBF网络是全连接、包含一个隐层的前馈网络；其特殊之处在于激活函数和净输入计算方式。

内容回顾

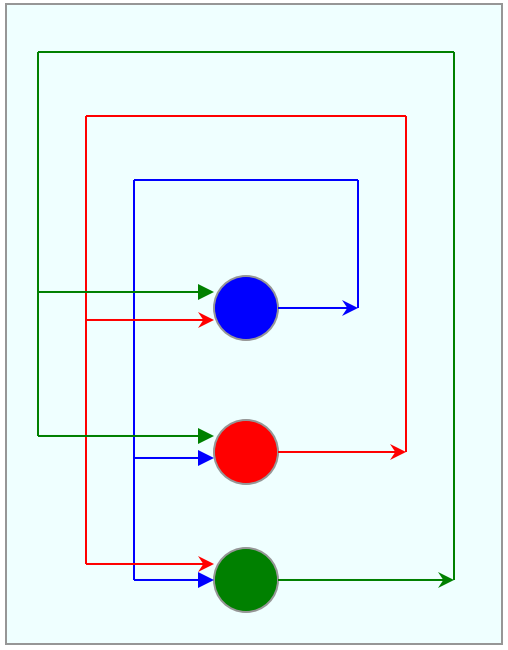
- **介绍**
  - 神经网络、深度学习的发展历史
  - 基本拓扑结构：前馈/反馈，全连接/局部连接
  - 学习算法：有监督(BP)/无监督(Hebb's rule, 竞争学习)
- **基本模型**
  - 神经元模型、单层感知器、多层感知器
- **扩展模型**
  - RBF网络、Hopfield网络、玻尔兹曼机、受限玻尔兹曼机、自组织映射网络
  - **Deep neural networks:** DBN, CNN, Autoencoder, RNN, LSTM, Transformer, GNN, etc.

6.9.2 深度概率模型

- **本小节主要内容**
  - Hopfield网络
  - 玻尔兹曼机　　(Boltzmann Machine)
  - 受限玻尔兹曼机　(Restricted Boltzmann Machine，RBM)
  - 深度信念网络　　(Deep Belief Network，DBN)

Hopfield network

- **结构**
  - 单层全互连、对称权值的反馈网络
  - 神经元状态：$v_j \in \{-1, +1\}$



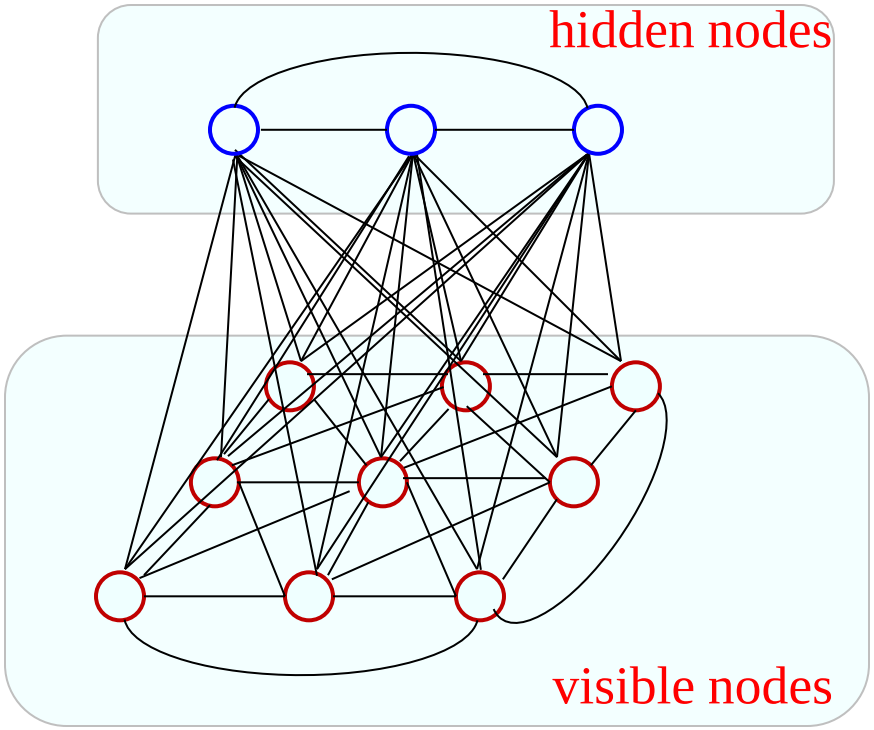
Hopfield网络按动力学方式运行，其工作过程为状态的演化过程，即从初始状态按能量减小的方向进行演化，直到达到稳定状态。稳定状态即为网络的输出：$v(t+1) = v(t)$。

$$\begin{cases} v_j(0) = x_j \\ v_j(t+1) = \mathrm{sgn}\left(\sum_{i=1}^{d} w_{ij} v_i(t)\right) \end{cases}$$

$$w_{ji} = w_{ij},\ w_{ii} = 0$$

Boltzmann machine

- BM是一种随机的Hopfield网络，是具有隐单元的反馈互联网络



- ✓ BM中一部分神经元与外部相连，可以起到网络的输入、输出功能，或者说可以受到外部条件的约束。另一部分神经元则不与外部相连，属于隐单元。
- ✓ 神经元的状态为0或1的概率取决于相应的输入。

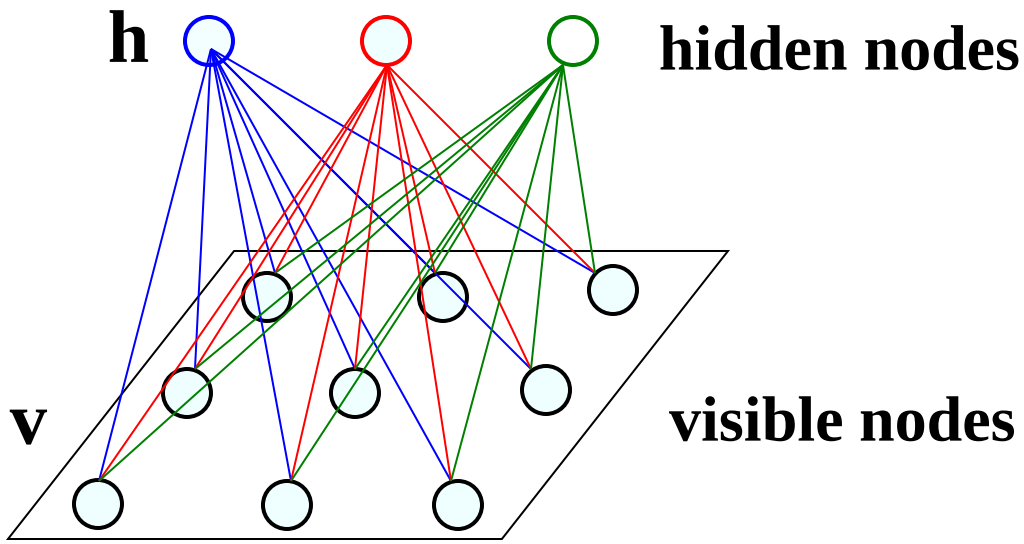
$$p_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{Z(\theta)}\exp(-E(\mathbf{v},\mathbf{h};\theta))$$

Restricted Boltzmann Machine, RBM

- RBM是对BM的简化：层间相连，层内结点不相连，信息可双向流动
- RBM中的条件独立性
  - 隐变量状态已知时，各观测变量相互独立；观测变量状态已知时，各隐变量相互独立

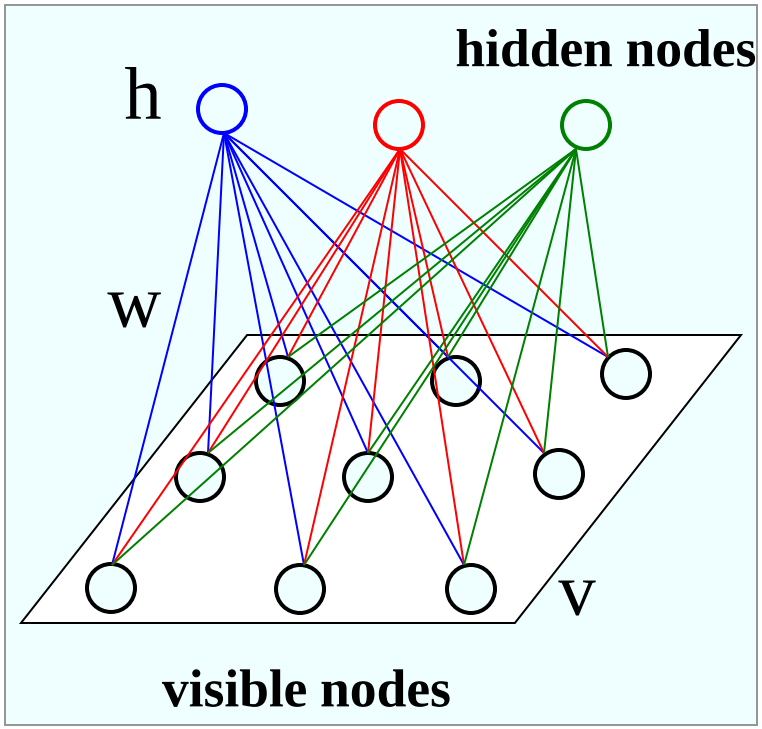
$$P(v_1,\ldots,v_d \mid h_1,\ldots,h_p)=\prod_{i=1}^{d} P(v_i \mid h_1,\ldots,h_p)$$

$$P(h_1,\ldots,h_p \mid v_1,\ldots,v_d)=\prod_{j=1}^{p} P(h_j \mid v_1,\ldots,v_d)$$



RBM

结构 (Bipartite Structure)



Classical structure

Stochastic binary visible variables $\mathbf{v} \in \{0,1\}^d$ are connected to stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^p$.

The energy of the **joint configuration**:

$$E(\mathbf{v},\mathbf{h};\theta) = -\sum_{ij} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j$$

θ={w,a,b}—model parameters

Probability of the joint configuration is given by the Boltzman distribution:

$$p_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{z(\theta)} \exp(-E(\mathbf{v},\mathbf{h};\theta)) = \frac{1}{z(\theta)} \prod_{ij} e^{w_{ij} v_i h_j} \prod_i e^{b_i v_i} \prod_j e^{a_j h_j}$$

$$z(\theta) = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h};\theta))$$

玻尔兹曼分布

（即：整个模型是用联合概率分布来描述的）

RBM

- **描述**
  - **v** 为观测变量，**h** 为隐变量，其能量函数为：$E(\mathbf{v}, \mathbf{h}; \theta)$
  - 概率形式：$p(\mathbf{v},\mathbf{h})$, $p(\mathbf{v})$, $p(\mathbf{h})$, $p(\mathbf{v}|\mathbf{h})$, $p(\mathbf{h}|\mathbf{v})$：

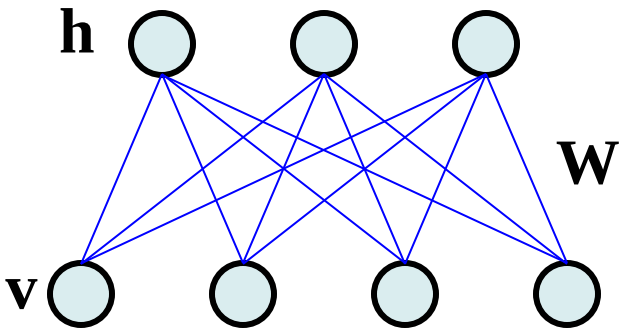
$$p(\mathbf{v},\mathbf{h}) = \frac{\exp^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})}}$$

$$p(\mathbf{v}) = \frac{\sum_{\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})}} \qquad p(\mathbf{h}) = \frac{\sum_{\mathbf{v}} \exp^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})}}$$

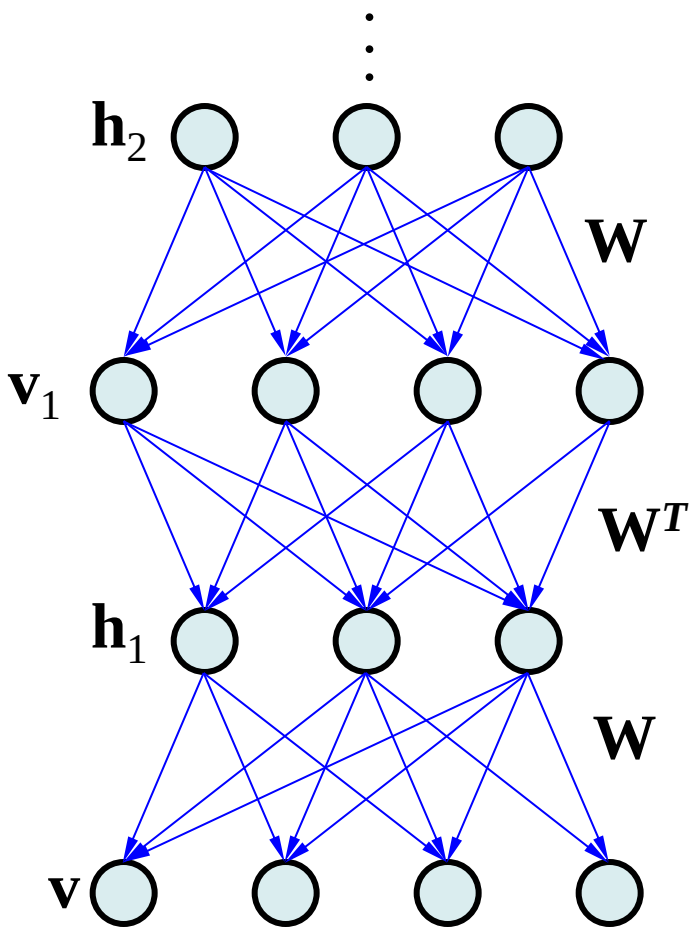
$$p(\mathbf{v}\,|\,\mathbf{h}) = \frac{\exp^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v}} \exp^{-E(\mathbf{v},\mathbf{h})}} \qquad p(\mathbf{h}\,|\,\mathbf{v}) = \frac{\exp^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})}}$$

RBM

- **对网络结构的等价解释**



(注：双方向有向边)



无穷层有向网络（但层间权值相同）

Input: visible dataset $\boldsymbol{v}$, (initialization: optional)
Get initialization or do random initialization of $\boldsymbol{v}$
while *until burn-in* **do**
  for j *from* 1 *to* p **do**
    $h_j^{(\nu)} \sim \mathbb{P}(h_j|\boldsymbol{v}^{(\nu)})$
  for i *from* 1 *to* d **do**
    $v_i^{(\nu+1)} \sim \mathbb{P}(v_i|\boldsymbol{h}^{(\nu)})$

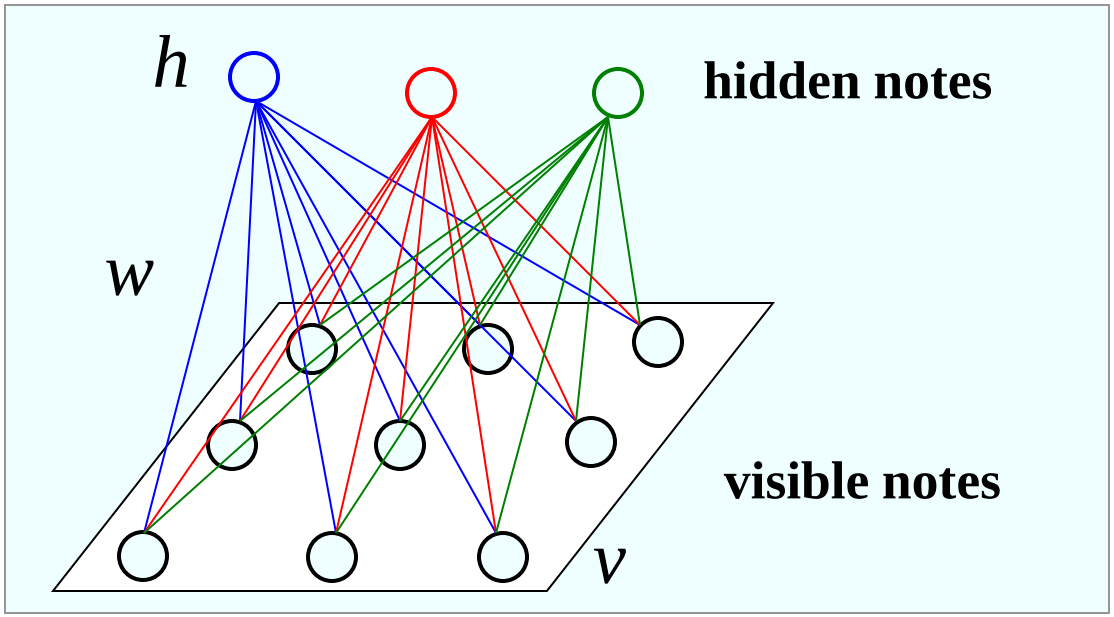
Algorithm 1: Gibbs sampling in RBM

- 目标：给定 N 个样本，对RBM进行训练，以概率最大化生成（观察到）这些样本。
- 最大似然：$\max \sum_{i=1}^{N} \log p(\mathbf{v}^i)$

$$p(\mathbf{v}) = \frac{\sum_{\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})}}$$

$(\mathrm{h} \in \{0,1\})$

$$\log(p_\theta(\mathbf{v})) = \log\left(\prod_i \exp(b_i v_i) \prod_j \left(1 + \exp(a_j + \sum_i w_{ij} v_i)\right)\right) - \log z(\theta)$$



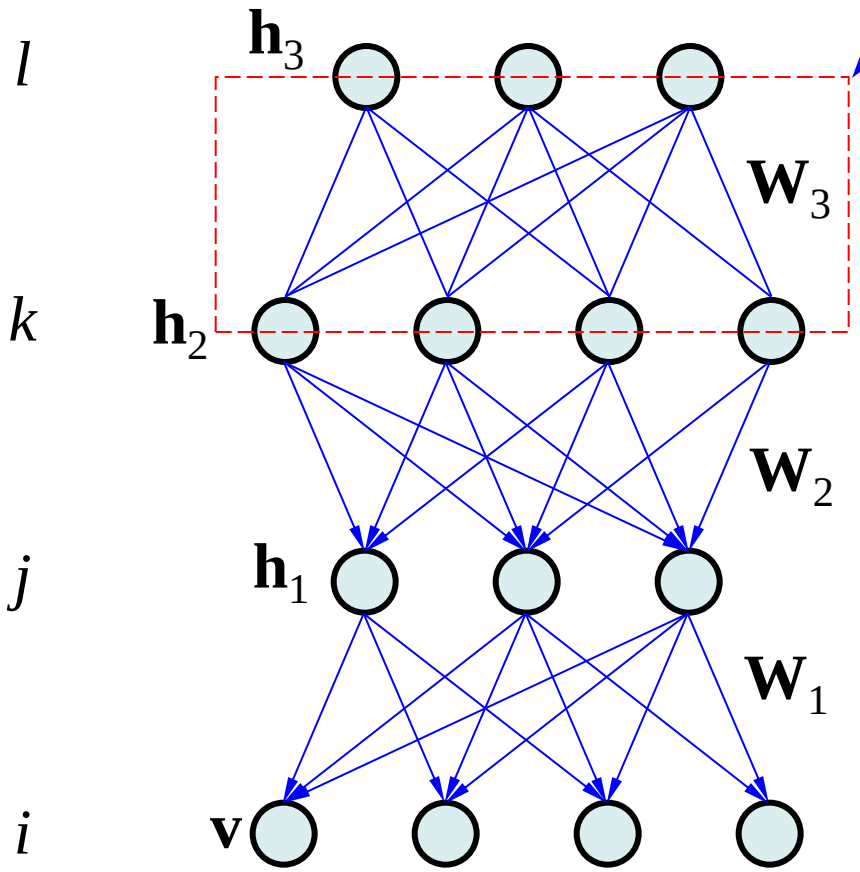
借助该模型

信息双方流动

生成式模型
generative model

深度信念网络(Deep Belief Networks, DBN)

具有联想存储功能



最高一层为双向(无向)连接（比如RBM）

联合概率分布

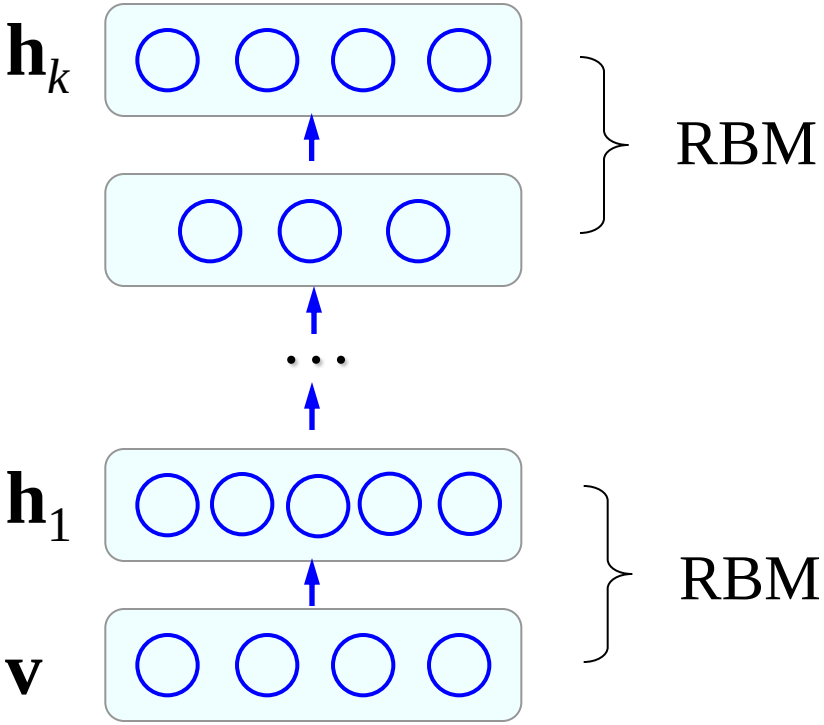
$$
\begin{aligned}
& p(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{v} \mid \boldsymbol{\theta}) \\
&= p(\mathbf{v} \mid \mathbf{h}_1)\, p(\mathbf{h}_1 \mid \mathbf{h}_2)\, p(\mathbf{h}_2, \mathbf{h}_3) \\
&= \prod_i Ber(v_i \mid sigm(\mathbf{h}_1^T \mathbf{w}_{1i})) \cdot \\
&\quad \prod_j Ber(h_{1j} \mid sigm(\mathbf{h}_2^T \mathbf{w}_{2j})) \cdot \\
&\quad \frac{1}{z(\boldsymbol{\theta})} \exp\left(\sum_{kl} h_{2k} h_{3l} \mathbf{w}_{3kl} \right)
\end{aligned}
$$

参考文献：

Kevin P. Murphy. Machine Learning: A Probabilistic Perspective, Chapter 28, Deep Learning, The MIT Press, 2012

DBN训练

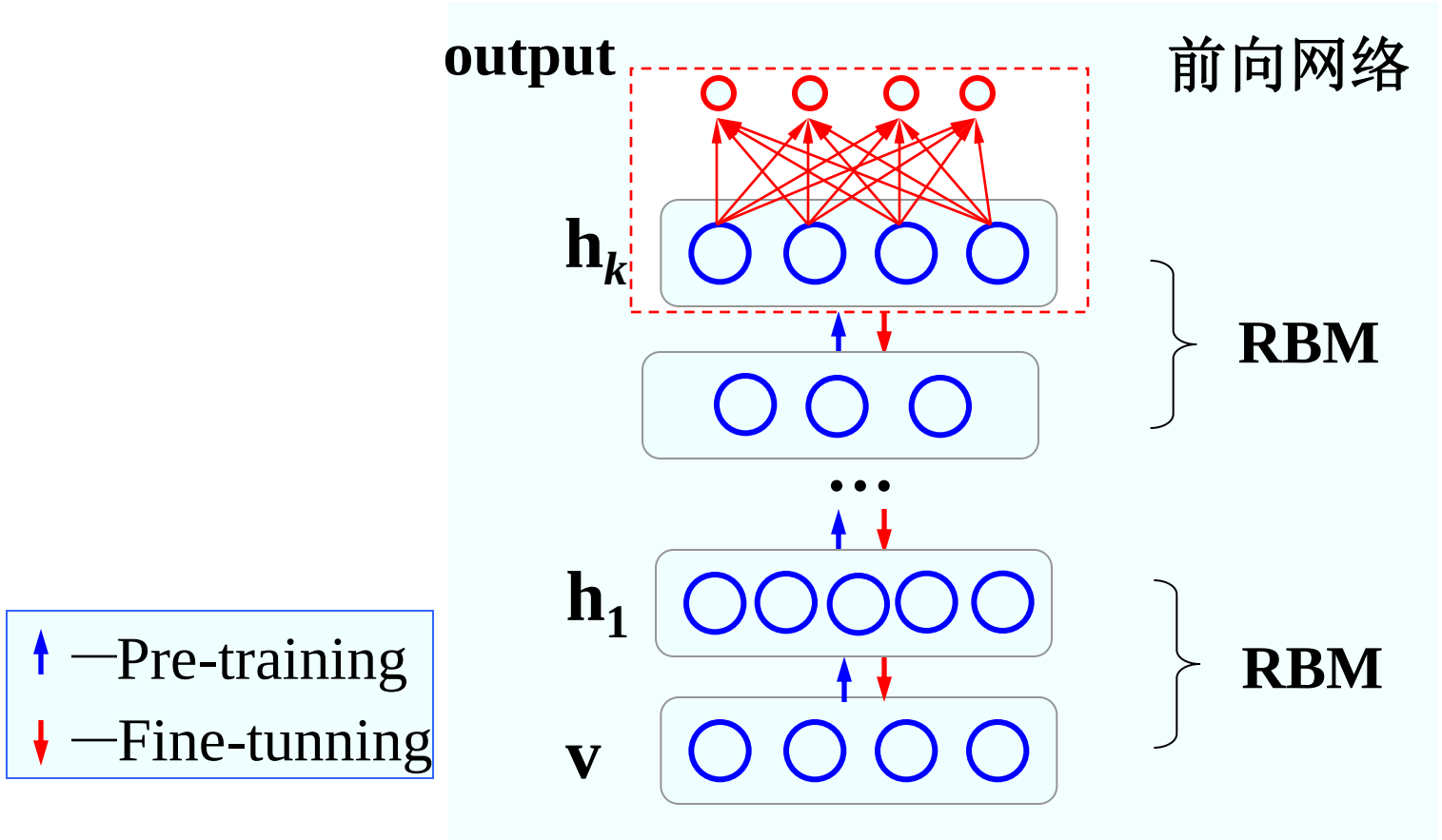
- **For feature learning**
  - **采用多个RBM进行贪婪训练 — stack by stack**



Kevin P. Murphy. Machine Learning: A Probabilistic Perspective, Chapter 28, Deep Learning, The MIT Press, 2012

DBN训练

- **For classification: 附加一个前向网络，采用有标签数据进行fine-tuning**



6.9.3 卷积神经网络

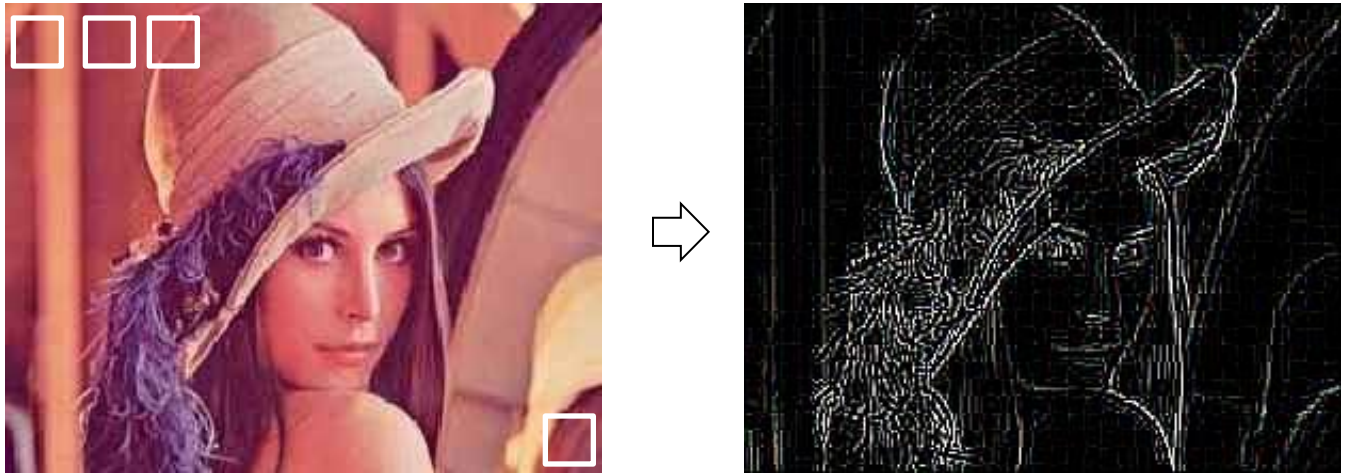
- **Convolutional Neural network (CNN)**
  - CNN为语音分析和图像识别领域的经典方法。
  - CNN的显著特点是局部连接和权值共享，其网络结构更加类似于生物神经网络。由于权值的数量的减少，网络模型的复杂度更低。
  - CNN本质上是一个多层感知器，这种网络结构对平移、比例缩放、倾斜或者其它形式的变形具有一定的不变性。

6.9.3 卷积神经网络

- **CNN**
  - **1962年**，Hubel 和 Wiesel 通过对猫视觉皮层细胞的研究（局部敏感和方向选择的神经元具有独特的网络结构），提出了感受野(receptive field) 的概念。
  - **1984年**，日本学者 Fukushima 基于感受野概念提出的神经认知机(neocognitron)，是感受野概念在人工神经网络领域的首次应用。
    - 它将一个视觉模式分解成若干个子模式（特征），对目标有位移或轻微变形时，仍然能完成识别。
  - **1989年**， LeCun等首先引入了权值共享和下采样策略来设计网络结构。
  - **1989年**，Hinton 等提出时延神经网络(Time-Delay Neural Network, TDNN)用于语音识别，性能高于HMM
  - **1998年**，Yann LeCun设计了一个处理图像的卷积神经网络，并用于文本识别，取得了不错的效果。

6.9.3 卷积神经网络

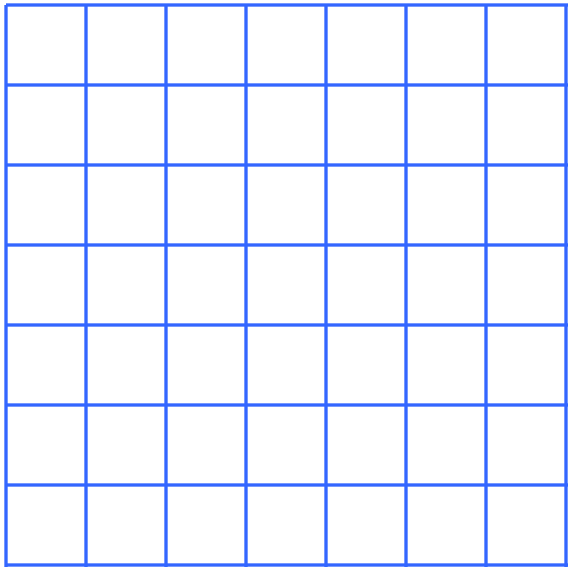
- **卷积：信号处理的基本操作**



卷积操作的要素

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

1	2	1
0	0	0
-1	-2	-1



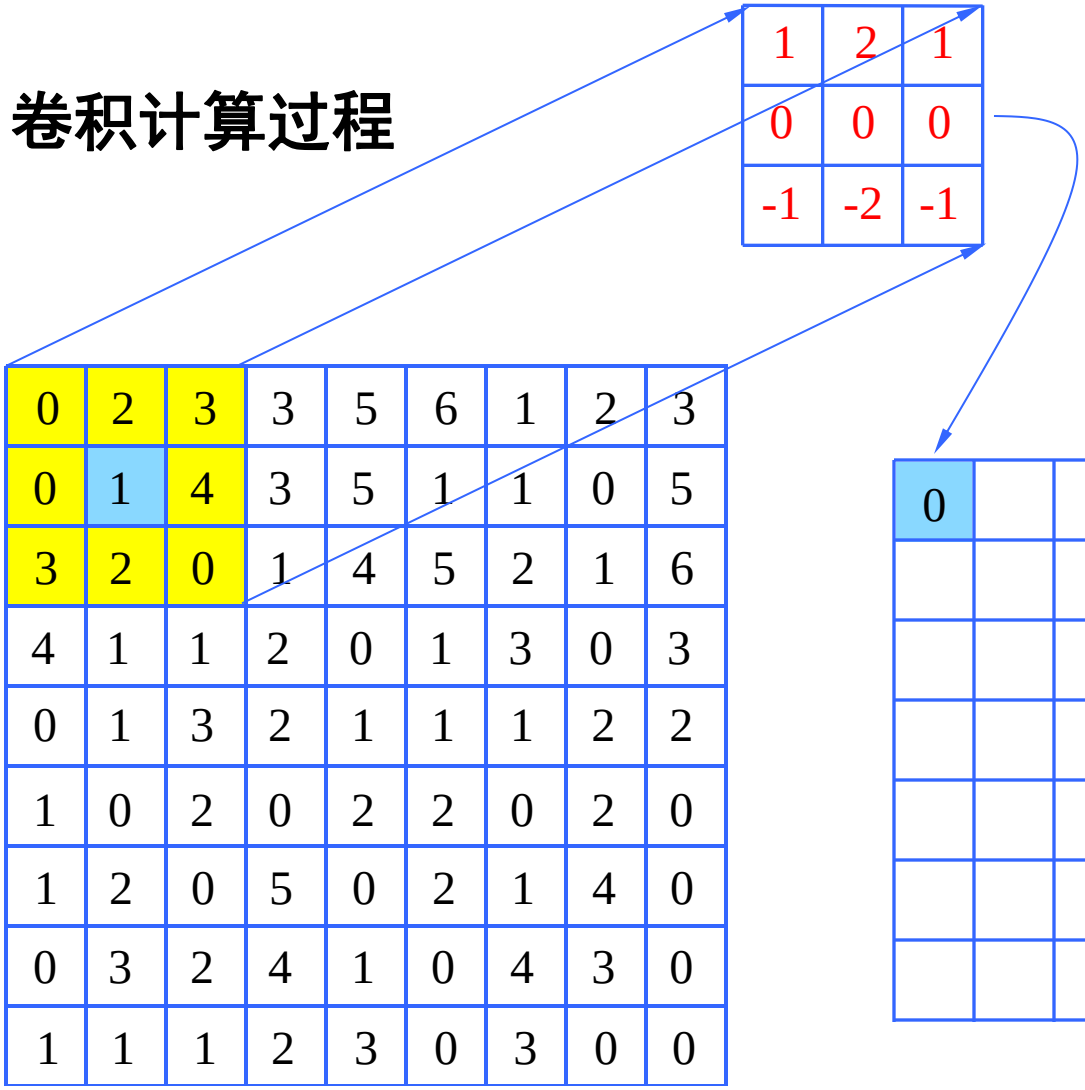
输入图像
（灰度图）

卷积核
（卷积核的大小：3×3）

输出特征图

一个包含可调参数的小窗口

卷积计算过程



（卷积核的大小：3×3）

$$0\times1 + 2\times2 + 3\times1 + 0\times0 + 1\times0 + 4\times0 + 3\times(-1) + 2\times(-2) + 0\times(-1) = 0$$

卷积计算过程

1	2	1
0	0	0
-1	-2	-1

（卷积核的大小：3×3）

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

0	8					

$2\times1 + 3\times2 + 3\times1 + 1\times0 + 4\times0 + 3\times0 + 2\times(-1) + 0\times(-2) + 1\times(-1) = 8$

卷积计算过程

1	2	1
0	0	0
-1	-2	-1

（卷积核的大小：3×3）

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

0	8	8				

$3\times1 + 3\times2 + 5\times1 + 4\times0 + 3\times0 + 5\times0 + 0\times(-1) + 1\times(-2) + 4\times(-1) = 8$

卷积计算过程

1	2	1
0	0	0
-1	-2	-1

（卷积核的大小：3×3）

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

0	8	8				
						6

$1\times1 + 4\times2 + 0\times1 + 4\times0 + 3\times0 + 0\times0 + 3\times(-1) + 0\times(-2) + 0\times(-1) = 6$

卷积计算过程

1	2	1
0	0	0
-1	-2	-1

（卷积核的大小：3×3）

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

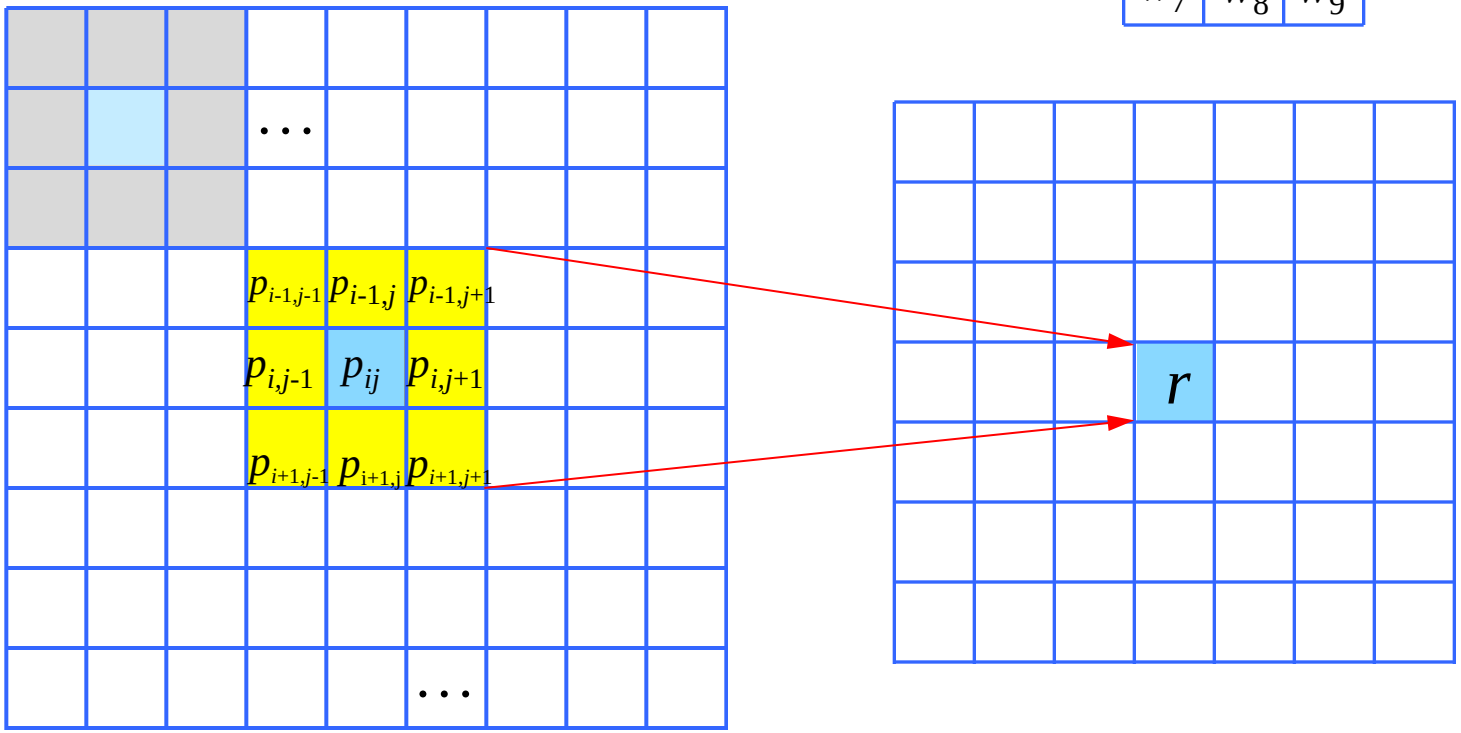
为了在每个输入位置上计算卷积，需要边缘补齐

Just Padding!

（卷积核的大小：3×3）

二维卷积的一般化描述

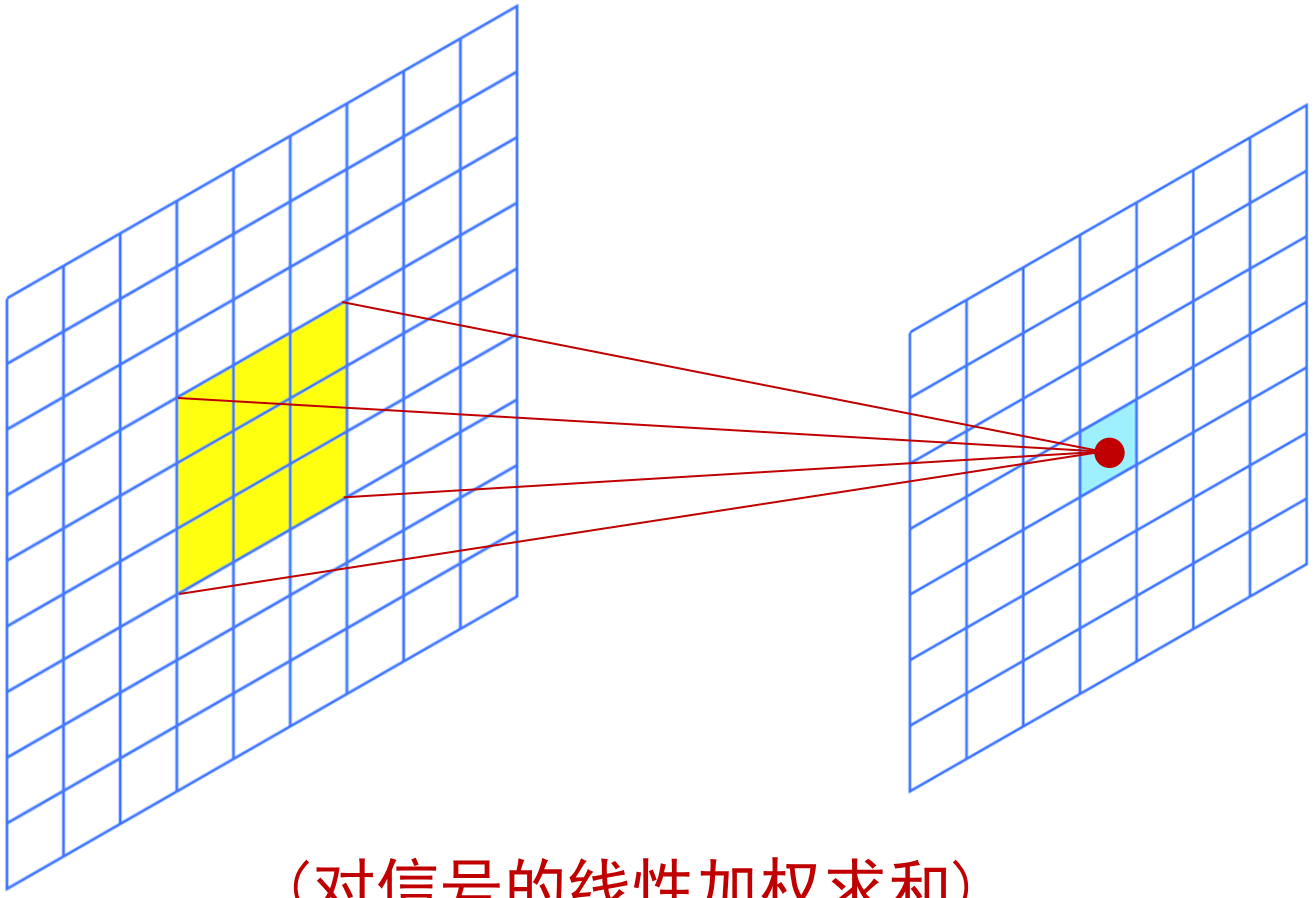
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



$$r = p_{i-1,j-1}w_1 + p_{i-1,j}w_2 + p_{i-1,j+1}w_3 + p_{i,j-1}w_4 + p_{i,j}w_5 + p_{i,j+1}w_6 + p_{i+1,j-1}w_7 + p_{i+1,j}w_8 + p_{i+1,j+1}w_9$$

（对信号的线性加权求和）

二维卷积的一般化描述

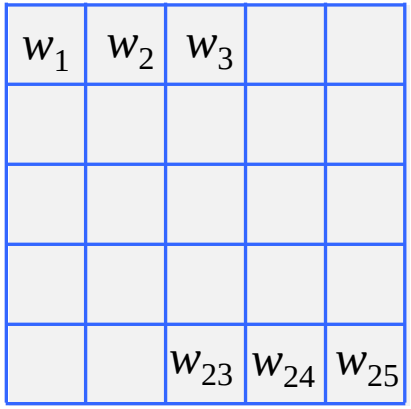


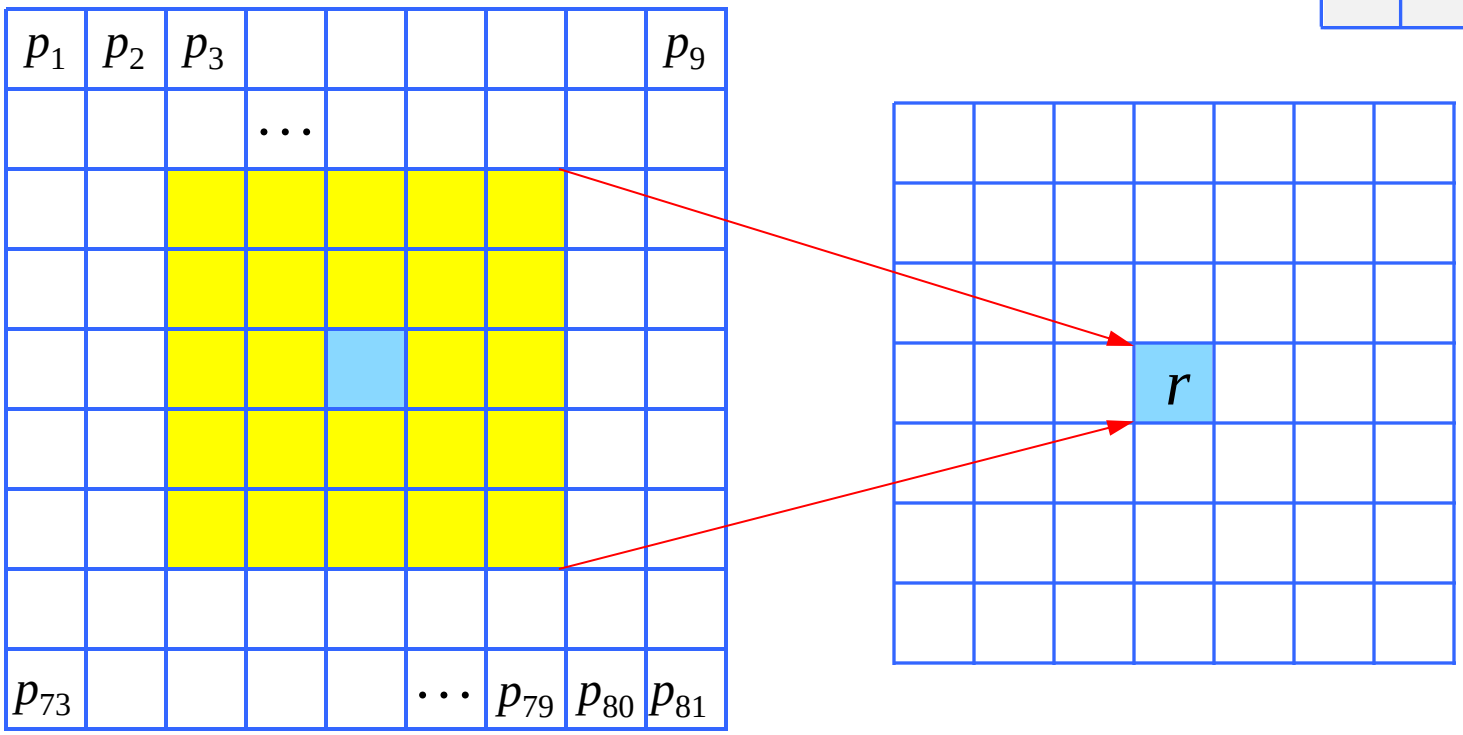
（对信号的线性加权求和）

扩大卷积核的大小

感受野(receptive field)

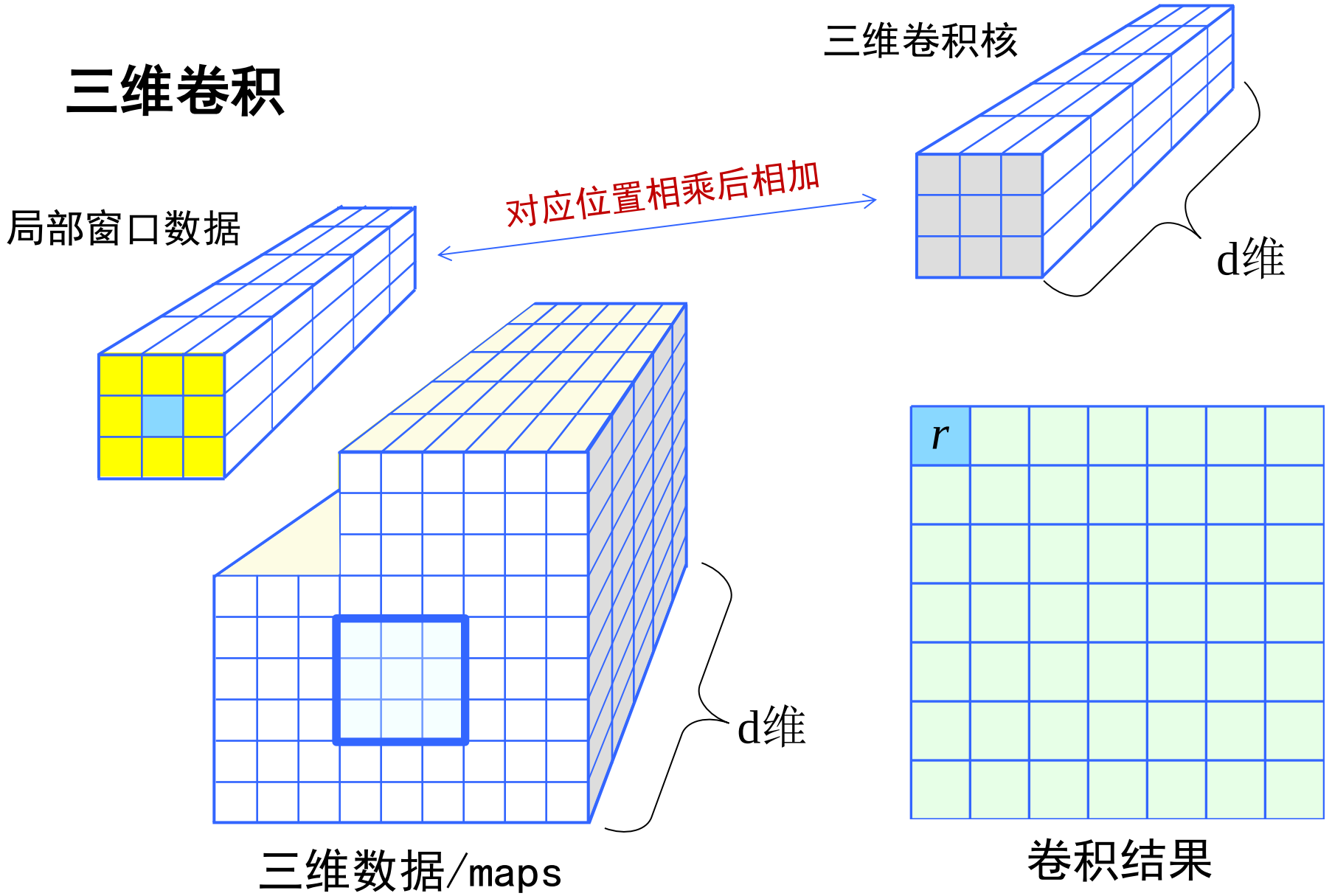
（卷积核的大小：5×5）





（对信号的线性加权求和）

三维卷积



图像卷积操作一举例

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

（平均平滑）

图像卷积操作一举例

1	1	1
1	-8	1
1	1	1

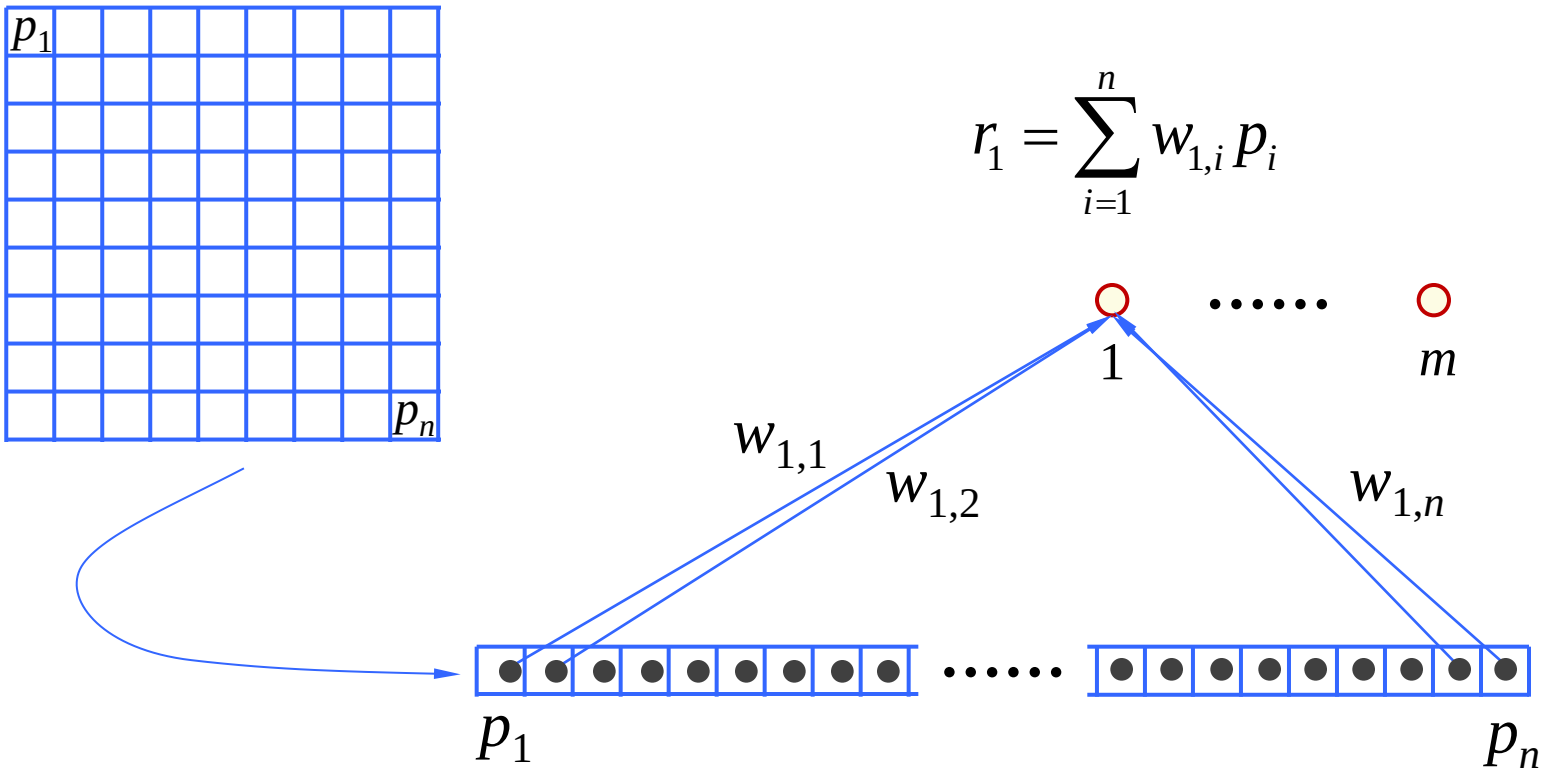
（拉普拉斯边缘检测）

6.9.3 卷积神经网络

- **图像卷积给我们的启示**
  - 在空间上，卷积是一种**局部窗口内**的运算，即将卷积核覆盖在该窗口上。
  - 卷积操作是线性的，是**线性加权求和**，结果被存放于窗口中心的像素上。
  - 卷积核记录了一组权值，在从图像左上角到右下角的滑动覆盖的过程中**保持不变**。
  - 每个卷积核相当于对某一种特征的提取器、检测器。

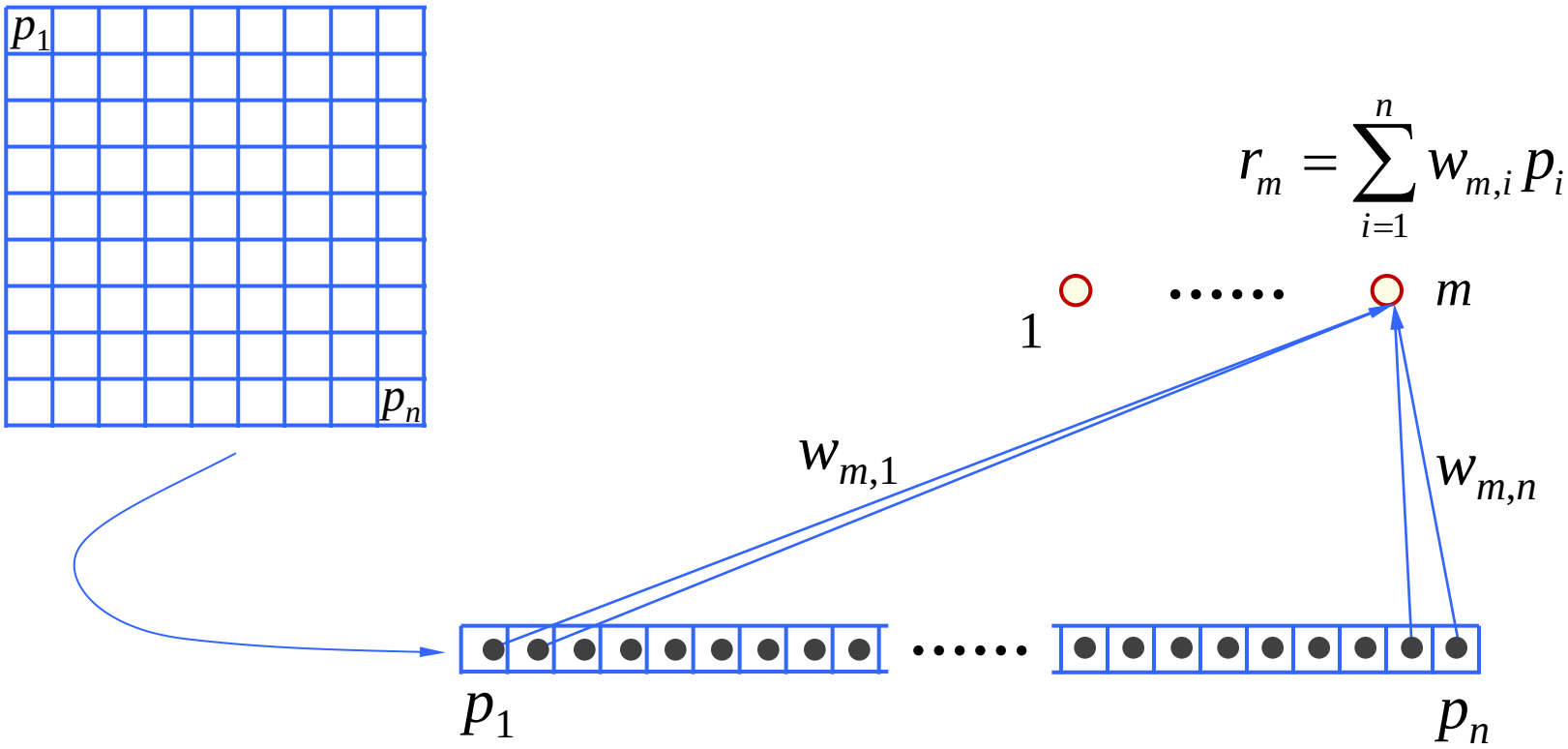
6.9.3 卷积神经网络

- **从加权求和的角度—按全连接**



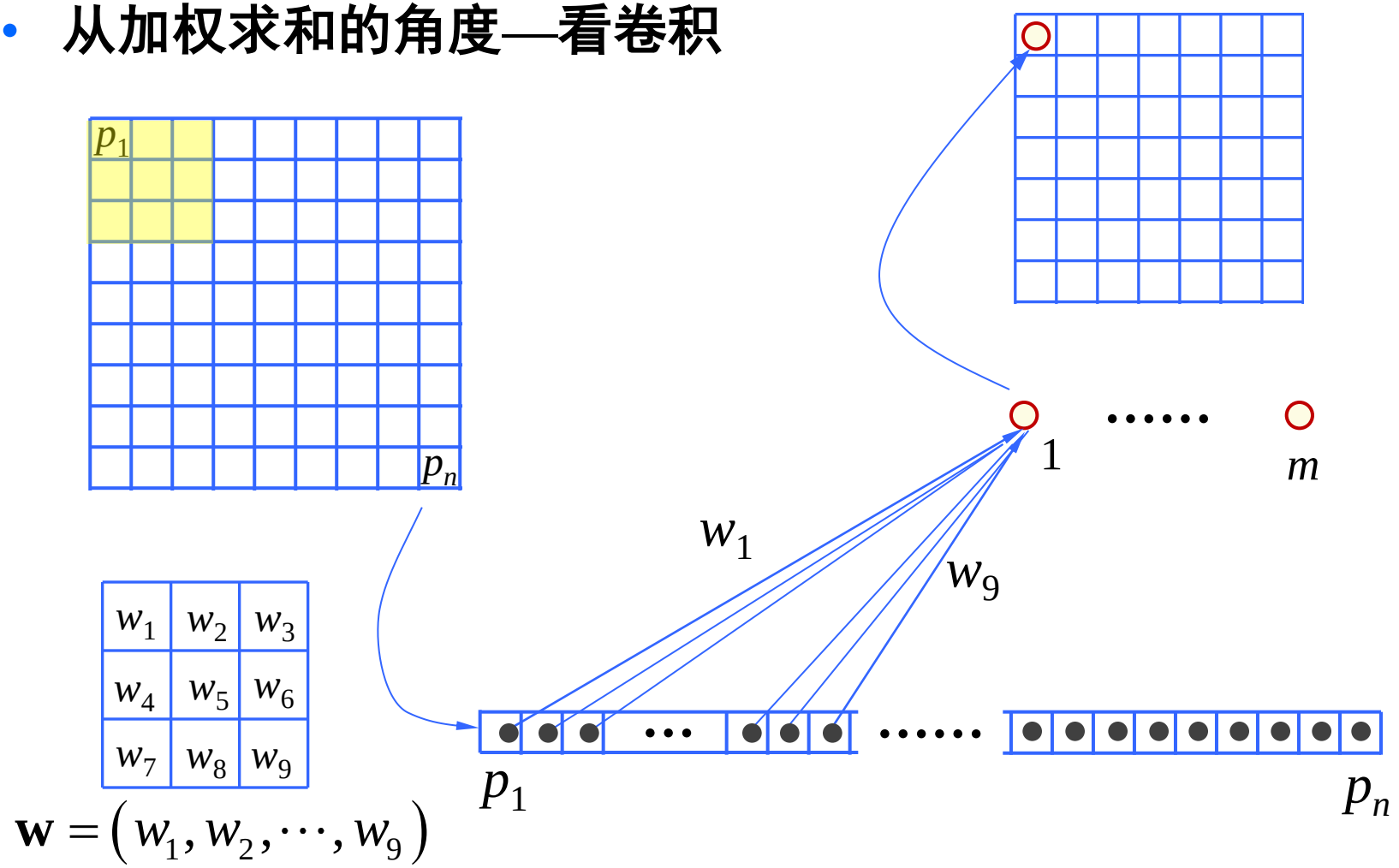
6.9.3 卷积神经网络

- **从加权求和的角度—按全连接**



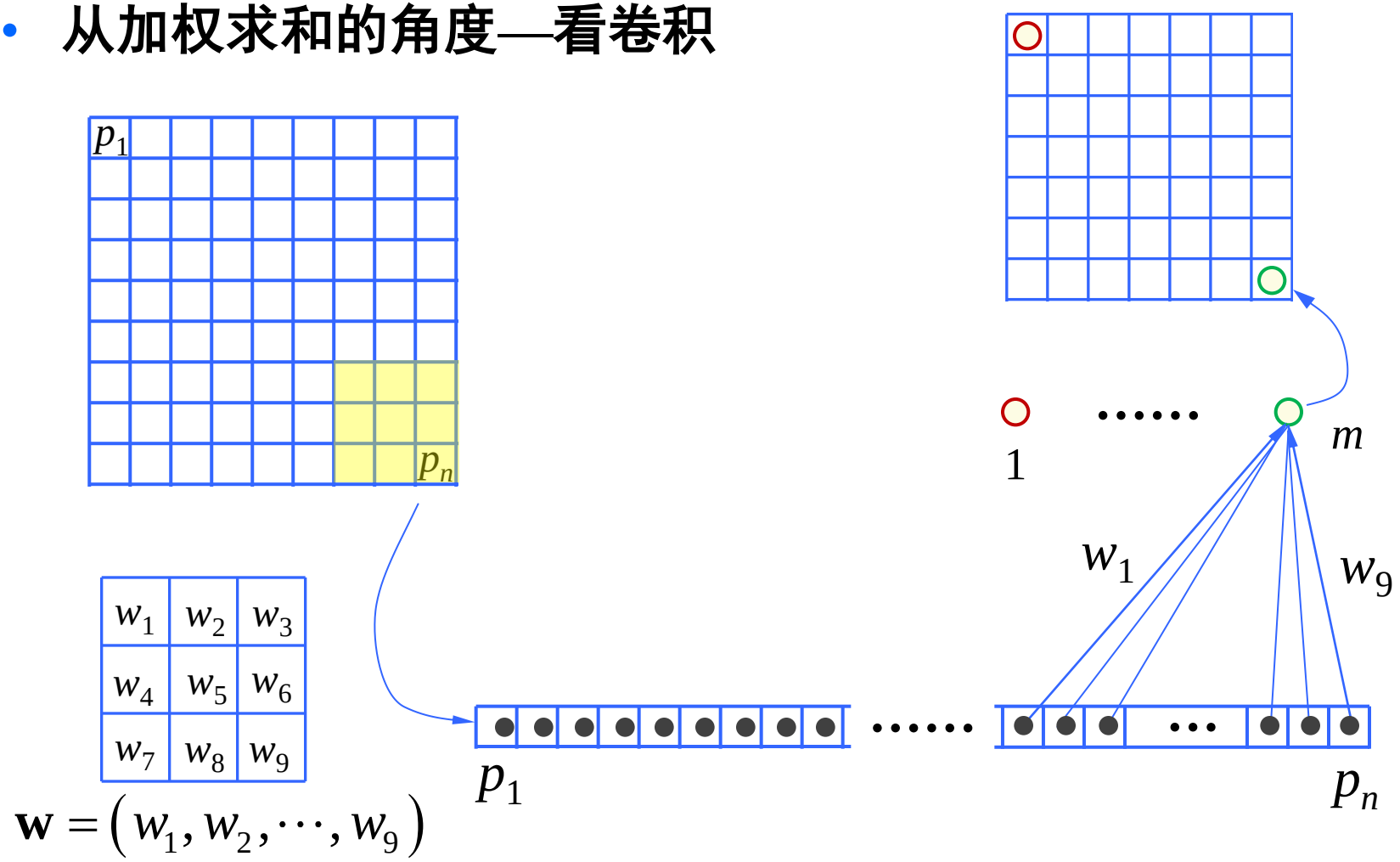
6.9.3 卷积神经网络

- **从加权求和的角度—看卷积**



6.9.3 卷积神经网络

- **从加权求和的角度—看卷积**



6.9.3 卷积神经网络

- 全连接的问题
  - 如果以图像直接作为输入，并将每个像素看成一个输入层结点，对于200×200大小的图像，则输入层就有4万个结点；
  - 如果第一隐含层仅仅包含1000个结点，则权重数量将达到：

$$4万 \times 1000 = 4千万$$

  - 这是一个巨大的计算负担和学习负担。

6.9.3 卷积神经网络

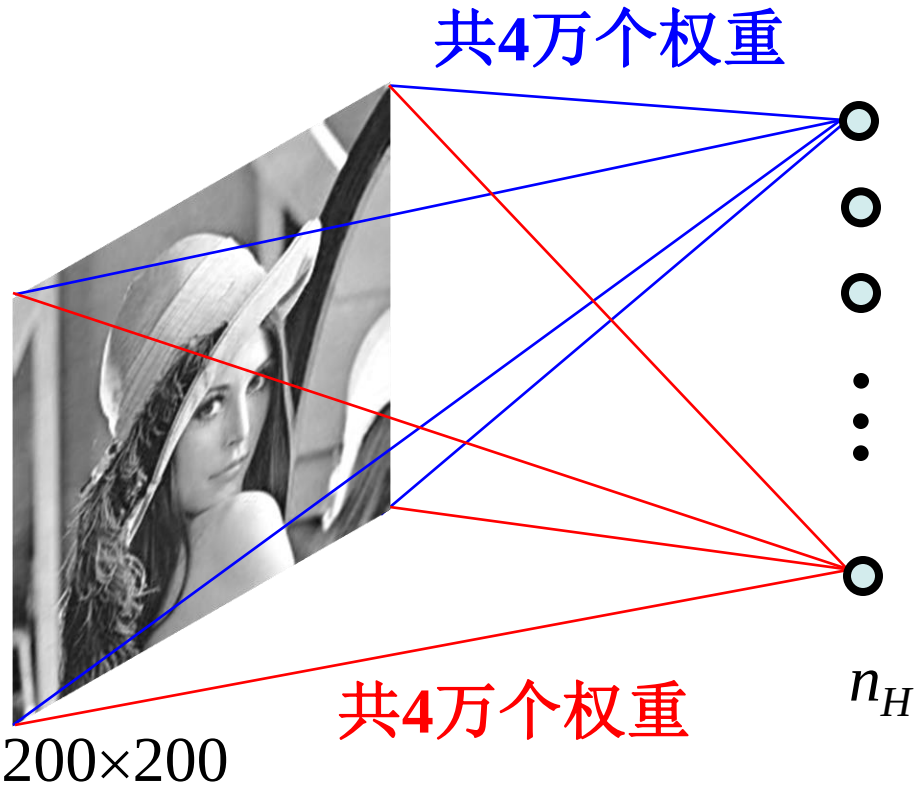
- **降低网络权重数量－局部连接（方法一）**
  - **视觉系统局部感受野**
    - 视觉生理学相关研究普遍认为：人对外界的认知是从局部到全局的。视觉皮层的神经元就是局部接受信息的（即只响应某些特定区域的刺激）
  - **图像空间相关性：**对图像而言，局部邻域内的像素联系较紧密，距离较远的像素相关性则较弱。
- **神经网络由全连接变为部分连接**
  - 每个神经元其实没有必要对全局图像进行感知，只需要对局部进行感知。（局部感受野）
  - 在更高层将局部的信息综合起来获得全局信息。

6.9.3 卷积神经网络

- **降低网络权重数量－权值共享（方法二）**
  - 局部连接可降低网络的权重数量，但仍不足够。
  - 引入权值共享机制。这一机制是：“从图像任何一个局部区域内连接到同一类型的隐含结点，其权重保持不变”。
  - 权值共享保证了CNN提取的特征具有平移不变性

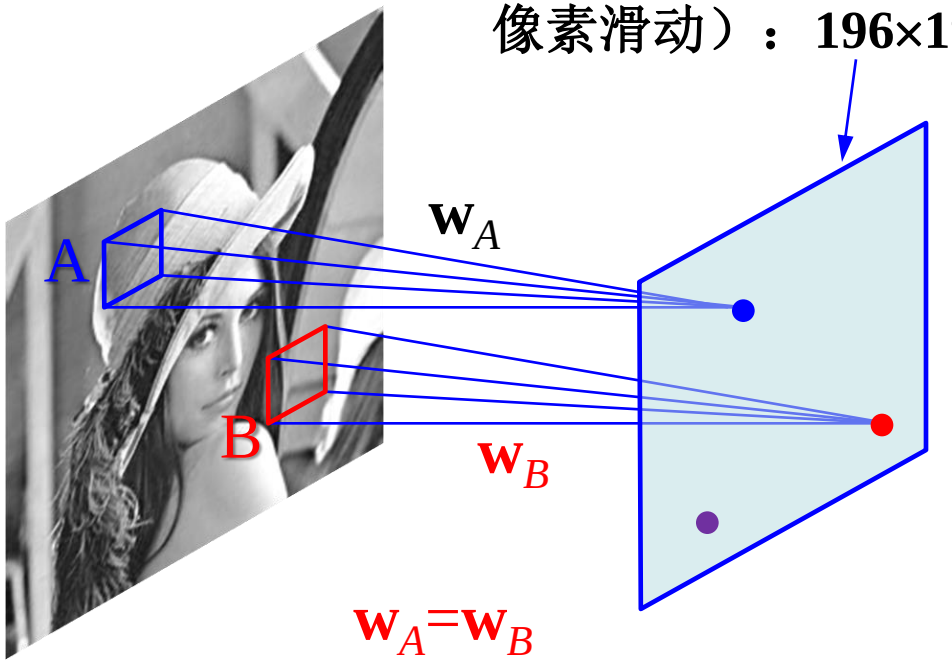
6.9.3 卷积神经网络

- **全连接**



（如果n_H=4000，则共有1亿6千万个权重需要学习）

- **局部连接**
  - 考虑5×5大小的卷积核且权值共享



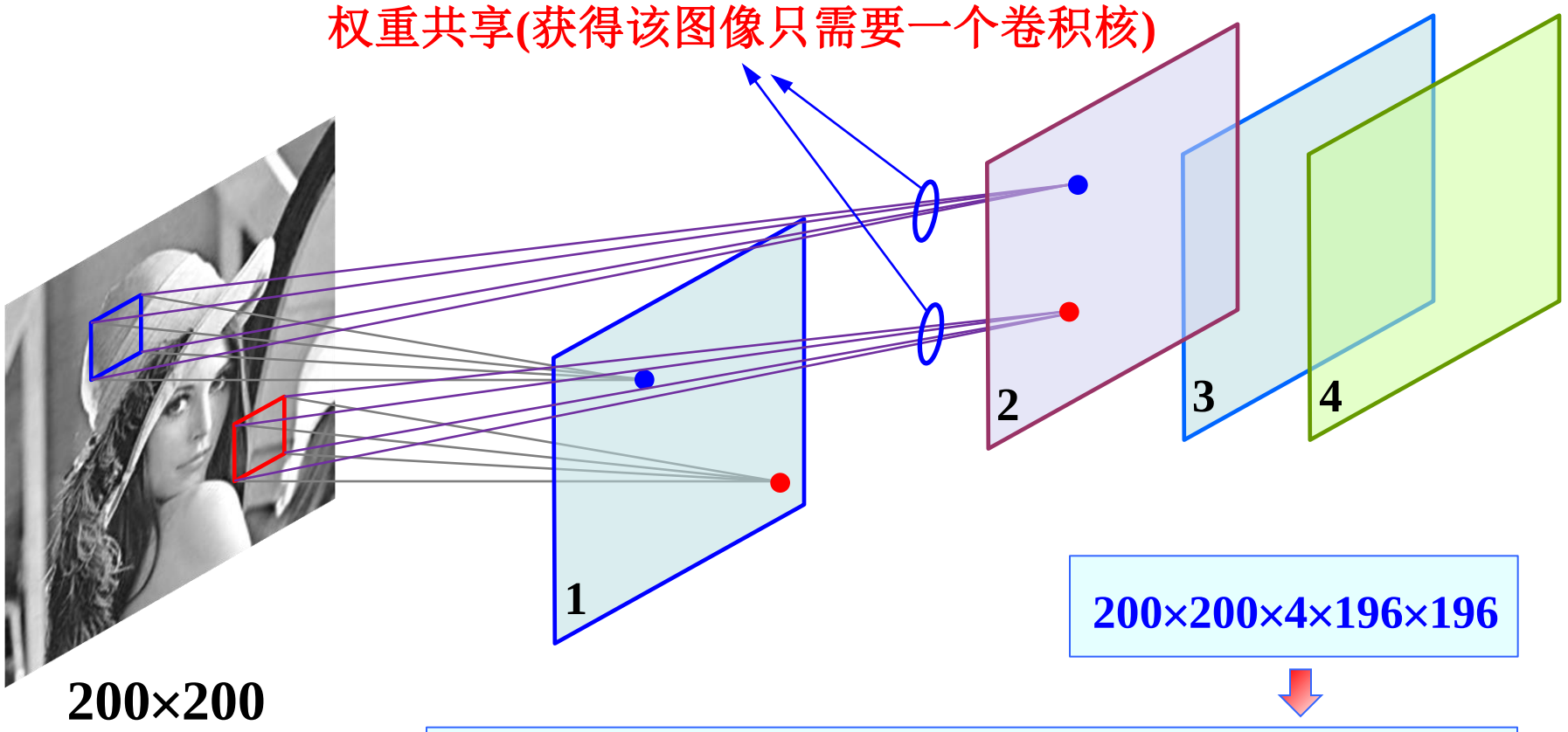
若采用全连接，权重数为：
200×200 × 196×196

若采用局部连接，权重数为：
25× 196×196

若采用局部连接+权值共享：一共25个权重！

6.9.3 卷积神经网络

- 可以使用多个卷积核：比如，4个

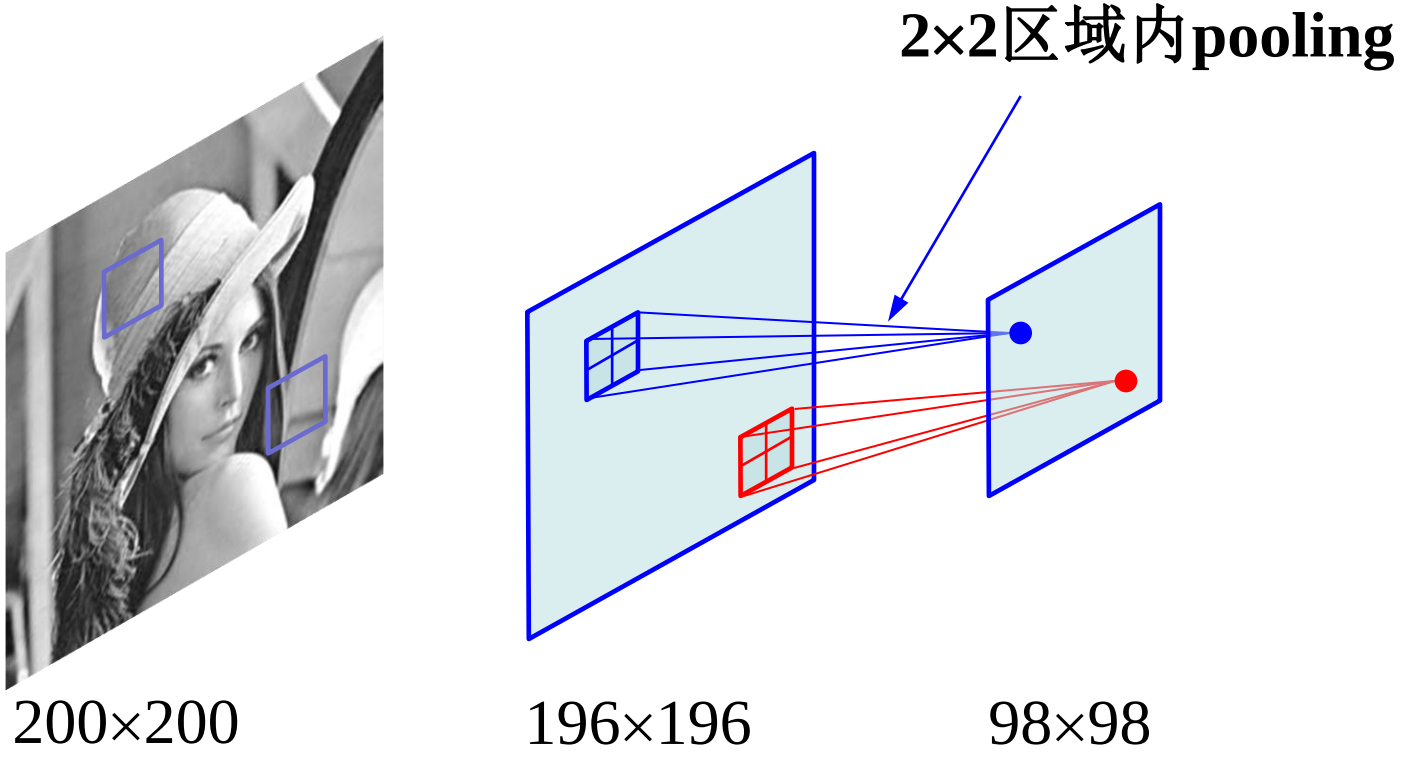


6.9.3 卷积神经网络

- **第一个隐含层**
  - **考虑4个卷积核的情形**
    - 4个卷积核通过卷积操作（局部加权求和）获得的结果，每个结果图像的大小为196×196
    - 每个卷积核对应一种特征提取。
    - 如果采用这些特征设计分类器，数据空间的维数将达到4×196×196。实际中卷积核更多，可能导致一个高维问题。采用高维数据设计分类器，容易产生过拟合（过学习）情形。
    - 一个办法就是对不同位置的特征进行聚合统计/池化(pooling)，比如，计算图像一个区域上的某个特定特征的平均值(或最大值)

6.9.3 卷积神经网络

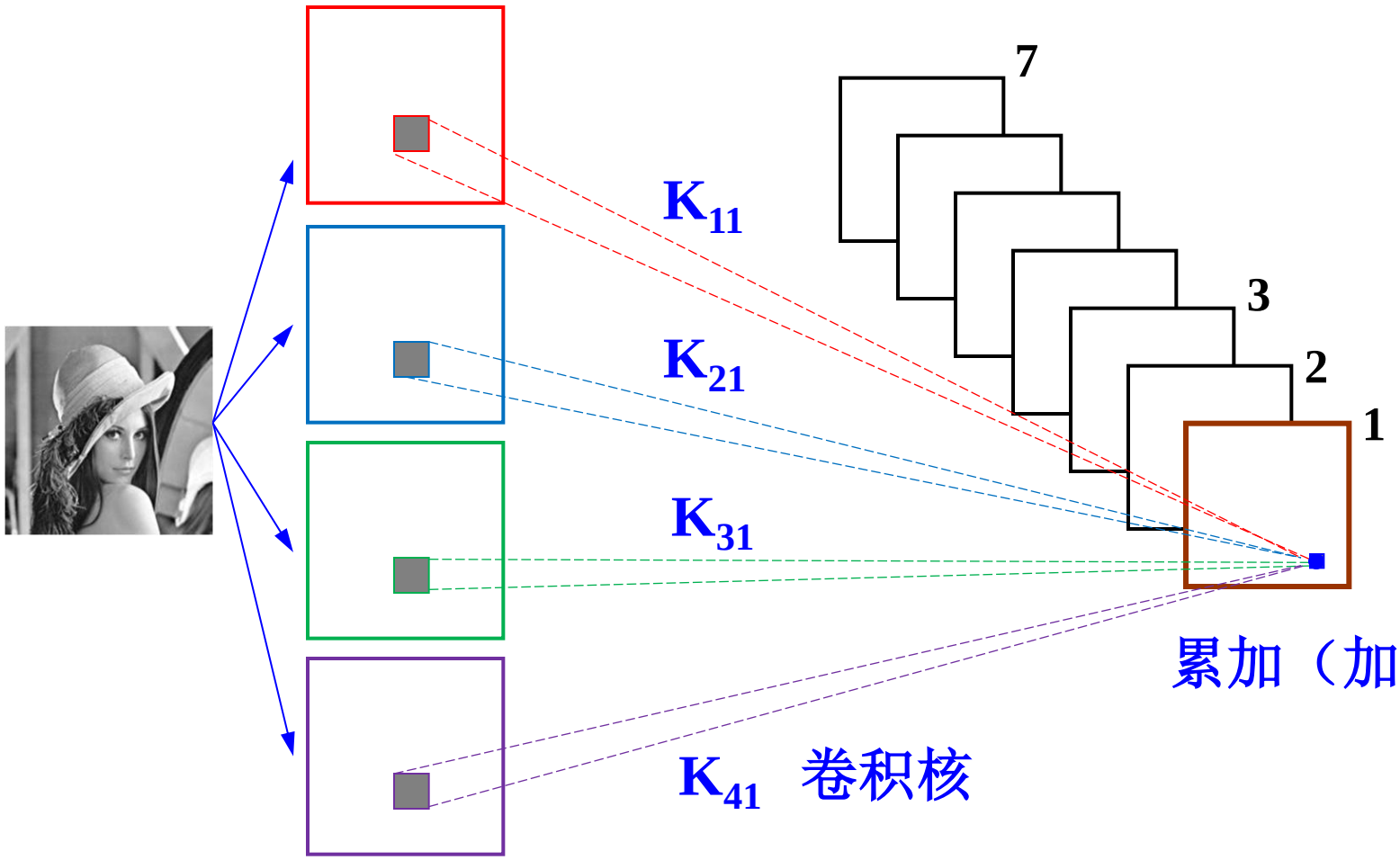
- 图像区域内关于某个特征的统计聚合
  - 两种方式：区域内取最大值或平均



6.9.3 卷积神经网络

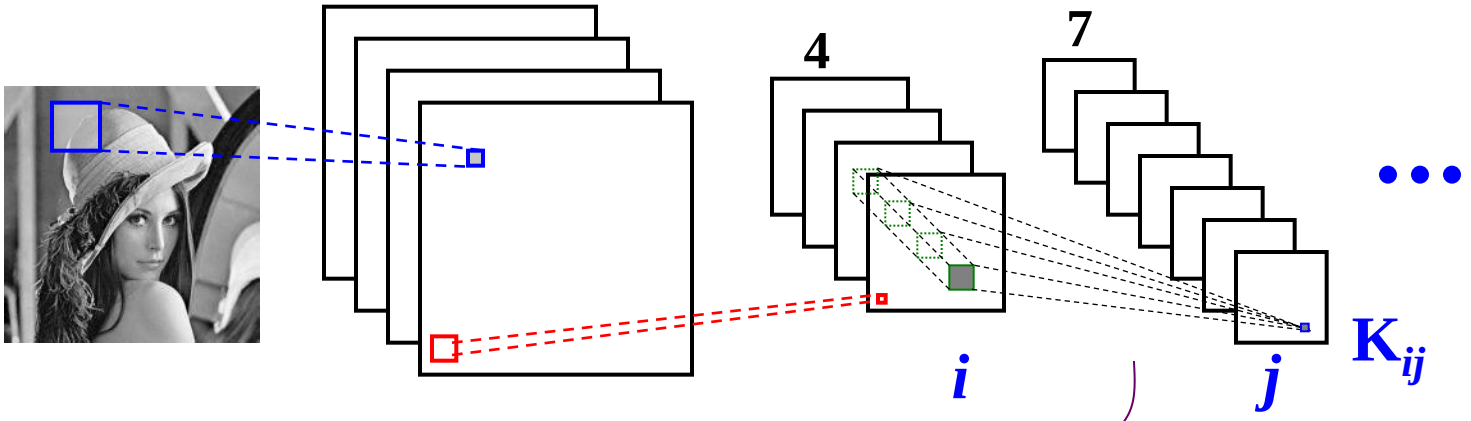
- **目前完成的步骤**
  - 对图像，采用多个卷积核进行卷积，获得多个滤波结果，每一个滤波结果对应一种图像特征
  - 对每一个局部滤波结果作一个非线性激励
  - 对每个滤波结果图像作pooling操作，图像大小得到降低，即通过pooling操作，完成了图像下采样 (downsample)
- **对200×200图像、4个5×5卷积核、 2×2 pooling：**
  - 第一步：得到4个196×196大小的滤波结果
  - 第二步：得到4个196×196大小的非线性激励结果
  - 第三步：得到4个98×98大小的pooling结果

- 第二个隐含层的设计：假定第二个隐层包含7个图像，采用3×3大小的卷积核



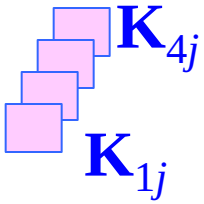
注：仍然只考虑局部连接

- 第一隐层至第二隐层：假定第二个隐层包含7个图像，采用3×3大小的卷积核



$$y_j^L = f\left(\sum_i \left(y_i^{L-1} * K_{ij}\right) + w_j\right)$$

可以将4个输入图像看成一个立方体，使用三维卷积核对立体数据进行卷积操作。此时只需要学习7个三维卷积核，每个包含4x3x3=36个权重参数。

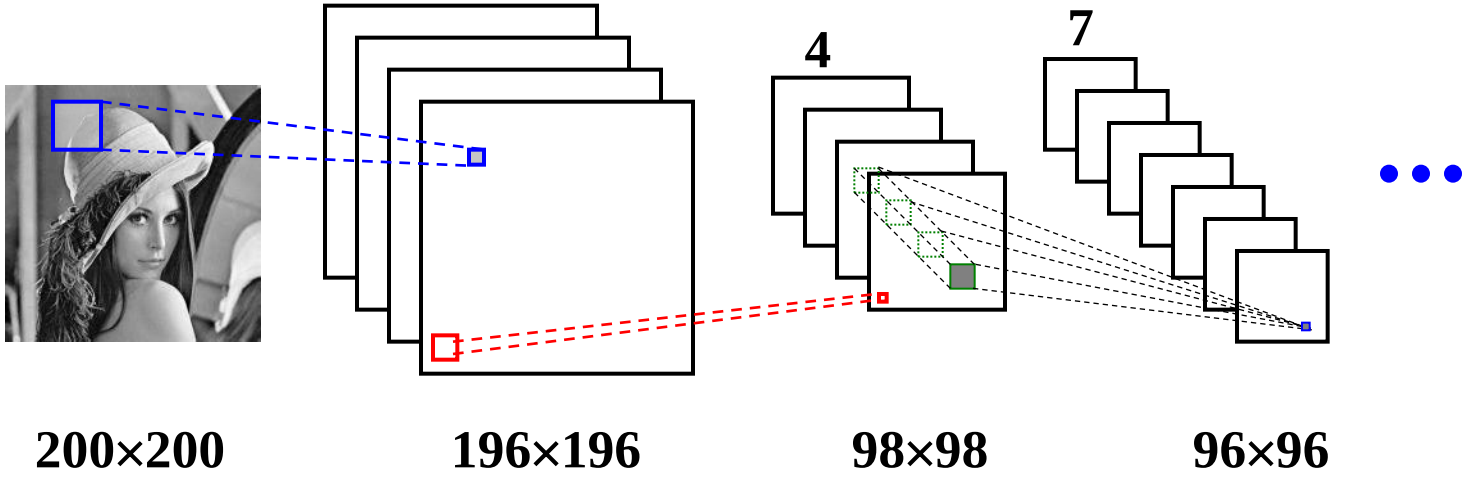


- **网络结构描述（举例）**
  - 图像大小：200×200
  - 第一隐含层卷积核大小：5×5
  - 第一隐含层图像个数：　4
  - 第一隐含层图像大小：　196 ×196
  - 第一隐含层pooling窗口大小：　2×2
  - 第一隐含层pooling之后图像大小：　98×98

  - 第二层卷积核大小：　3×3
  - 第二隐含层图像个数：　7
  - 第二隐含层图像大小：　96 ×96
  - 第二隐含层pooling窗口大小：　2×2
  - 第二隐含层pooling之后图像大小：　48×48
  - …

6.9.3 卷积神经网络

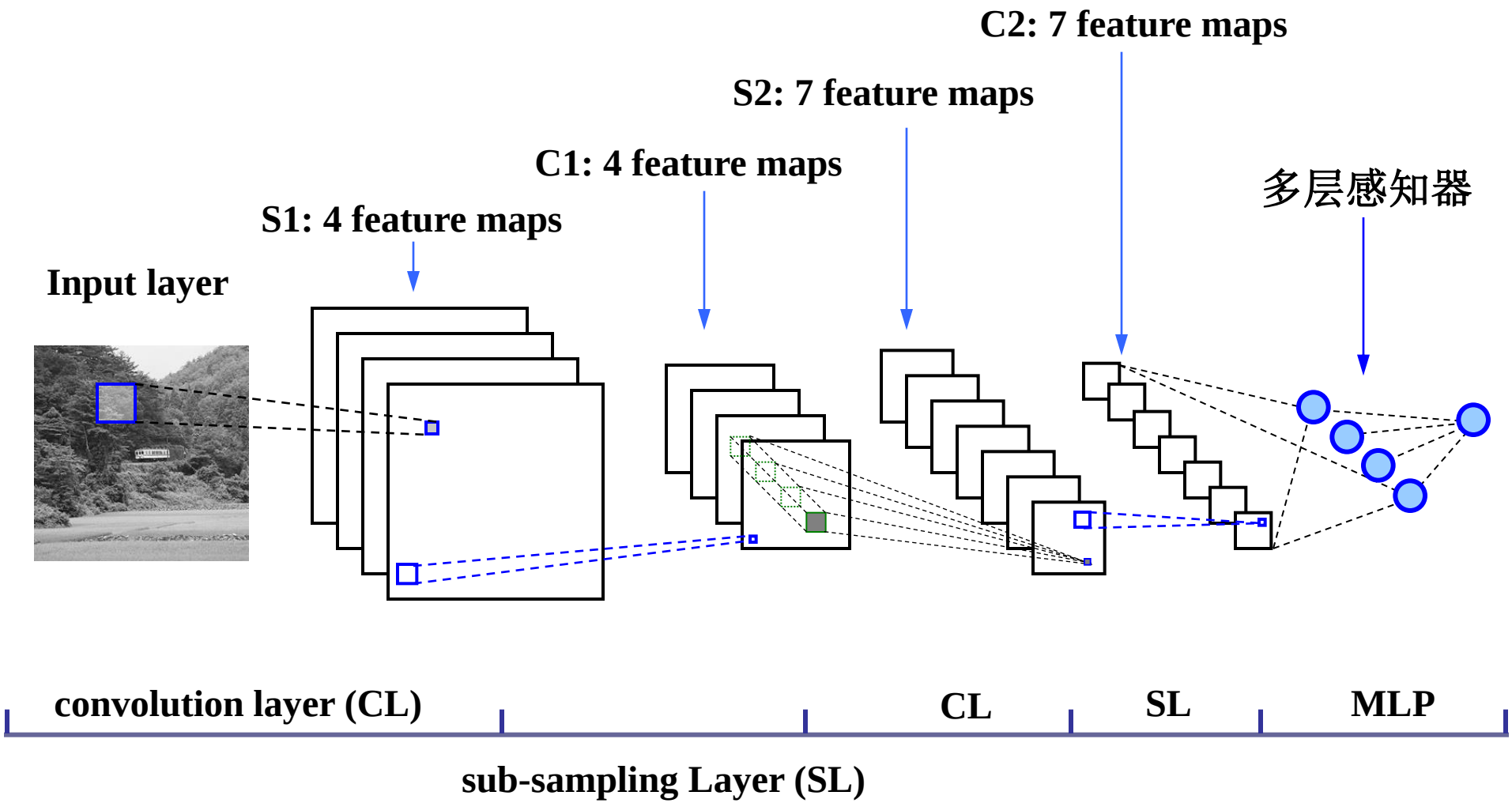
- **第一隐含层至第二隐含层权重数：**
  - 若采用全连接： 输入×输出 ＝ (4×98×98) × (4×96×96)
  - 若采用局部连接+权值共享（3×3卷积核）： 9 × 4 × 7



6.9.3 卷积神经网络

- **层与层之间的基本操作**
  - 卷积 (Convolution)
    - 将卷积之和加到下一层
    - 对卷积之和进行激励 （也可以在pooling之后求激励）
  - 聚合/池化 (Pooling)
    - 下采样，两种基本的运算：
      - 2×2窗口取平均
      - 2×2窗口取最大值

整体结构


C2: 7 feature maps
S2: 7 feature maps
C1: 4 feature maps
多层感知器
S1: 4 feature maps
Input layer
convolution layer (CL)
CL
SL
MLP
sub-sampling Layer (SL)

6.9.3 卷积神经网络

- **网络结构：**
  - 多少个隐含层？
  - 每个隐含层多少个结点？
  - 多层感知器里面多少层？

  - 均与实际问题相关

6.9.3 卷积神经网络

- **网络训练**
  - 采用反向传播算法！
  - 选择一个样本 **x**，信息从输入层经过逐级的变换，传送到输出层；
  - 计算该样本的实际输出 **o** 与相应的理想输出 **t** 的差；
  - 按极小化误差反向传播方法调整权矩阵；
  - 可以采取小的样本组，逐步进行训练。
- 思考题：
  - 权值共享机制下，对权值的梯度如何计算？
  - 卷积神经网络中有一个pooling操作，因为这一点需要对现有BP算法做一些修改，如何改？

6.9.3 卷积神经网络

- **核心思想（总结）**
  - 我们之所以决定使用卷积后的特征是因为图像具有一种“静态性”的属性，这也就意味着在一个图像区域有用的特征极有可能在另一个区域同样适用。
  - 局部感受野、权值共享以及时间或空间下采样这三种结构化思想结合起来获得了某种程度的位移、尺度、形变不变性。

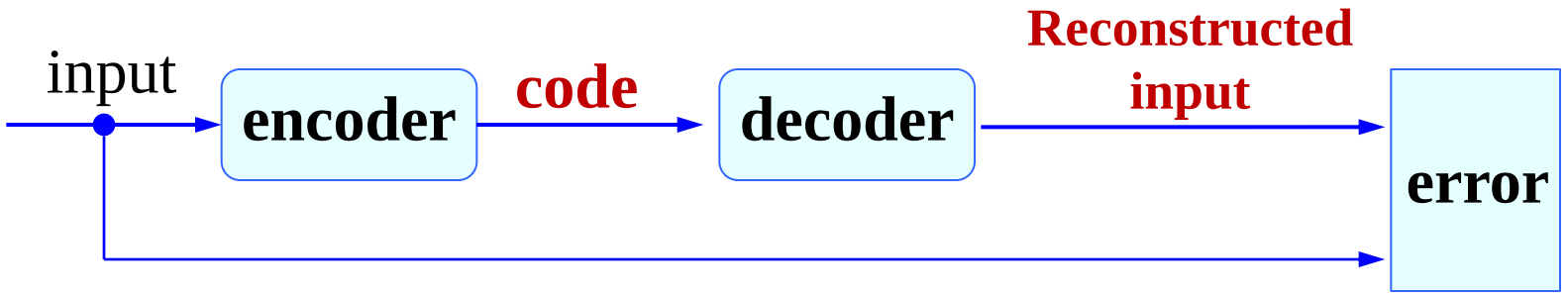
6.9.4 自编码器(Autoencoder)

- **背景**
  - MLP，CNN的训练样本是有标签的。在很多实际应用中，训练样本是没有标签的。
  - 自动编码器是一种以无监督方式学习的神经网络，其训练目标是让输出与输入相等，通常用于特征提取。
  - 实现这一宏观思想的一种著名神经网络：
    - 2006年G. E. Hinton和R. R. Salakhutdinov提出的自编码器。
    - 这一思想开创了**深度学习浪潮（神经网络的第三次高潮）**。

6.9.4 Autoencoder

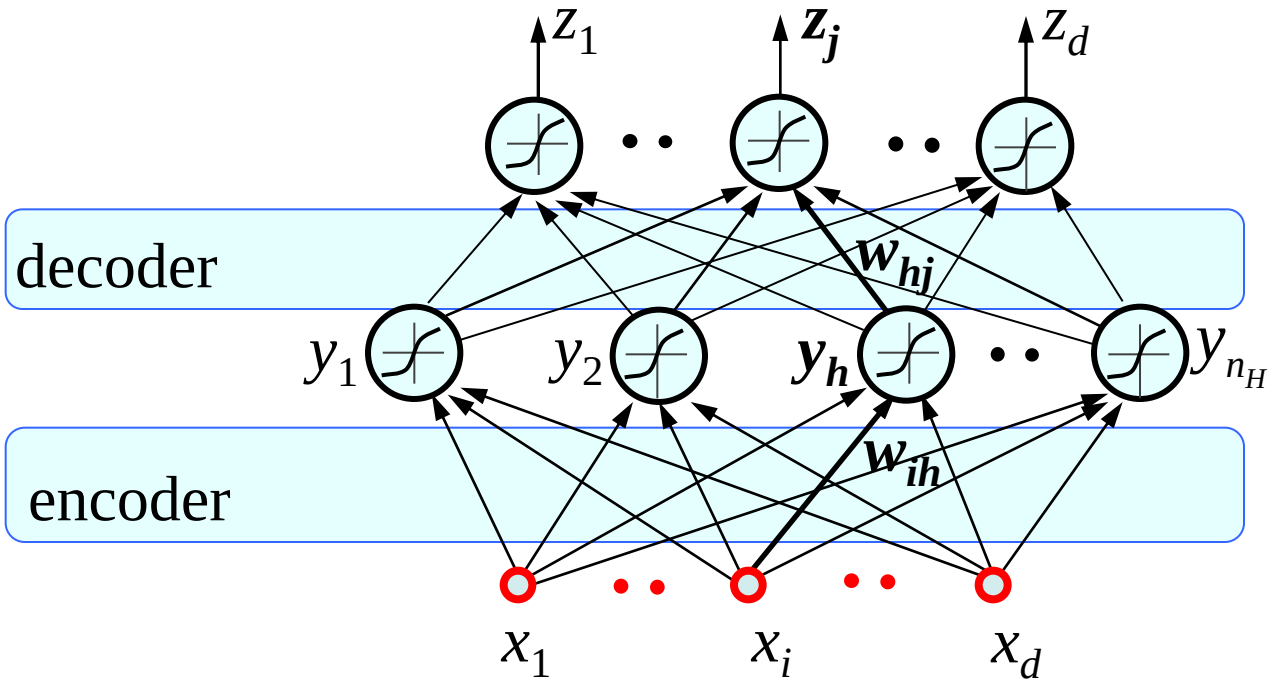
- **核心思想**
  - 自动编码器是一种重构输入信号的神经网络
    - 由一个编码器和一个解码器组成
    - 训练过程中，为了重构输入信号，网络必须提取到输入数据的最重要的内在因素，从而实现了对数据的特征学习。
      - 这也是主成分分析(PCA)的基本原理
    - 训练完成后，编码器输出则可以理解为提取到的数据特征，因此这是一种典型的**表示学习方法**。

- **网络结构**
  - 将数据输入编码器(encoder)，就会得到一个编码(code)。这个 code 也就是输入的一个表示。
  - 将code输入解码器(decoder)，并采用信号重构的方式来评价这个 code 的质量。
  - 理想情况下，希望 decoder 所输出的信息与输入信号input 是相同的。实际情况下，会存在误差，希望这个误差最小。

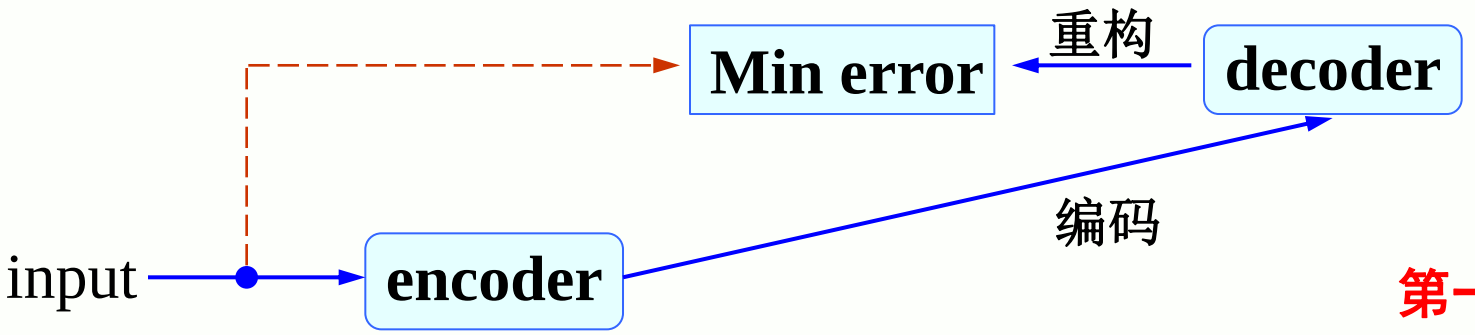


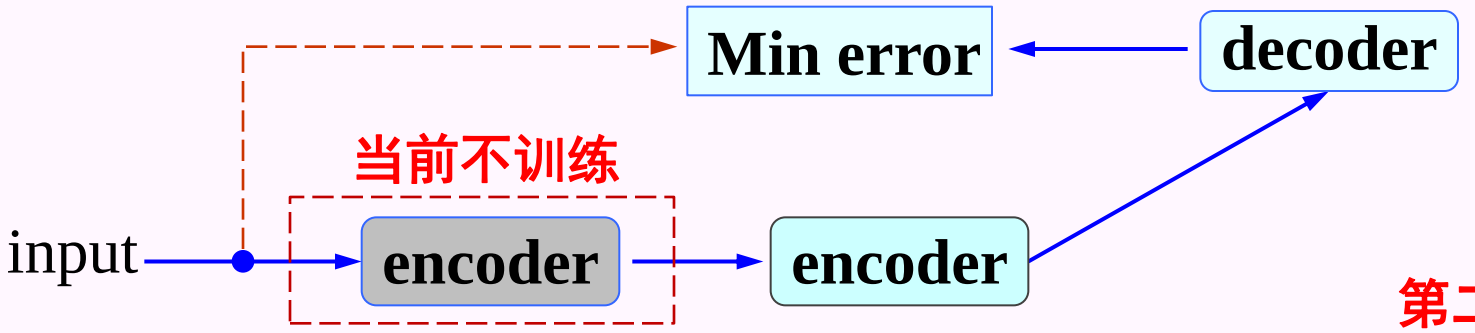
- **网络表达、第一次编解码训练：**
  - 编码：建立输入层至隐含层的权重；
  - Code: 隐含层的输出，也称为表达或特征
  - 解码：建立隐含层至输出层的权重

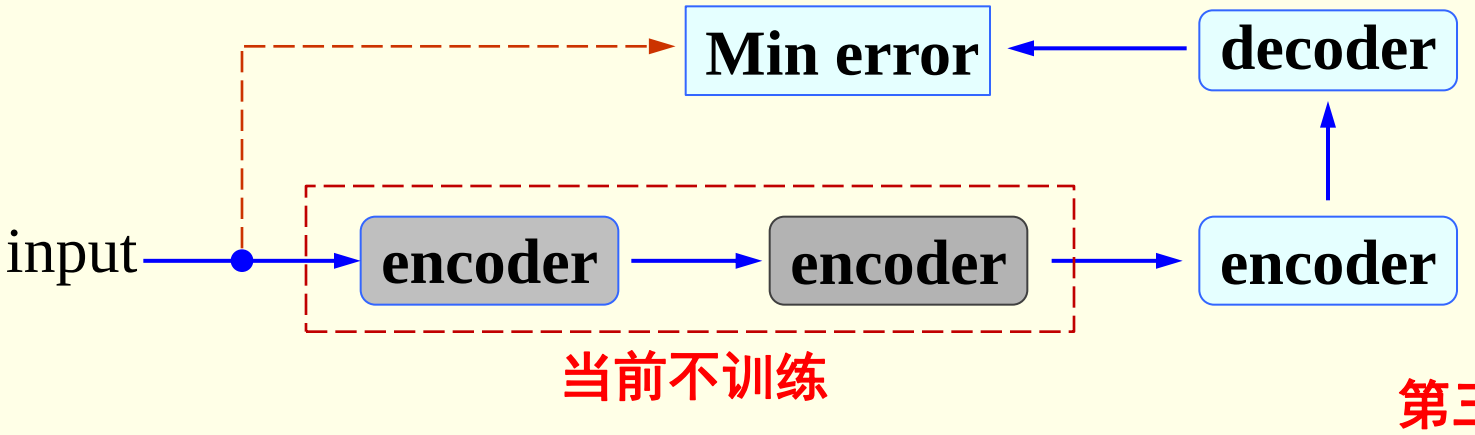
$$\min \sum_k \| \mathbf{x}_k - \mathbf{z}_k \|^2$$



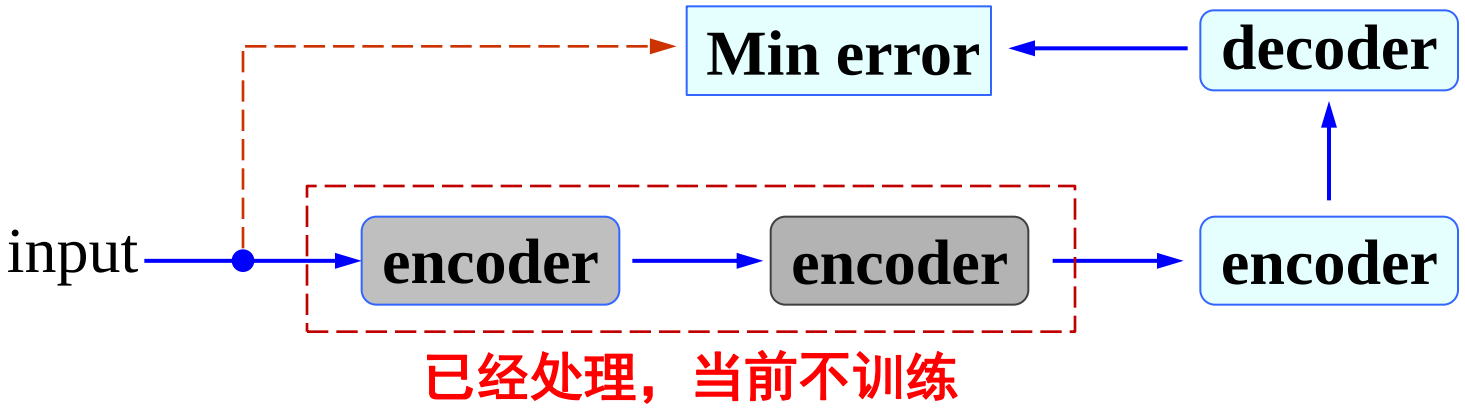
- 网络训练：（逐层静态进行）


Min error
重构
decoder
input
encoder
编码
第一次训练


Min error
decoder
当前不训练
input
encoder
encoder
第二次训练

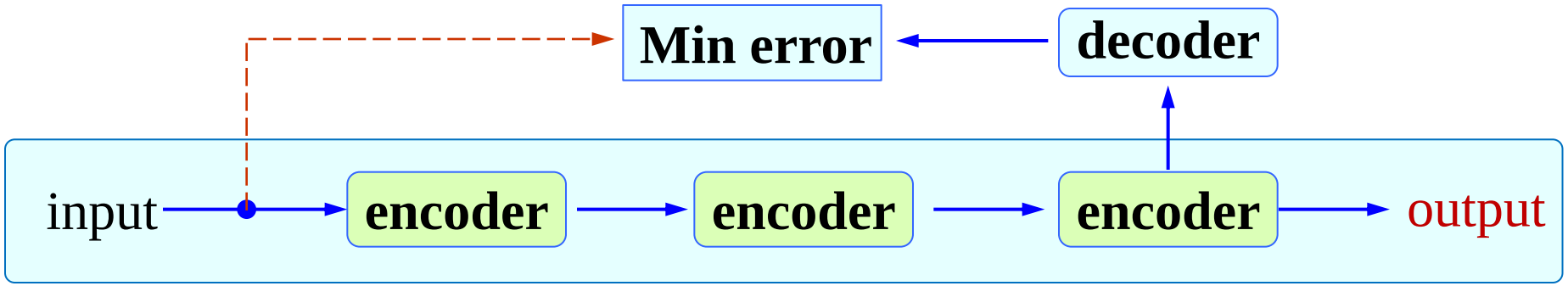

Min error
decoder
input
encoder
encoder
encoder
当前不训练
第三次训练

- **网络训练：**
  - 对每个样本，将第一层输出的 code 当成第二层的输入信号，再次利用一个新的三层前向神经网络来最小化重构误差，得到第二层的权重参数（获得第二个编码器）；同时得到样本在该层的code，即原始号的第二个表达。
  - 在训练当前层时，其它层固定不动。完成当前编码和解码任务。前一次“编码”和“解码”均不考虑。
  - 因此，这一过程实质上是一个静态的堆叠(stack)过程
  - 每次训练可以用BP算法对一个三层前向网络进行训练



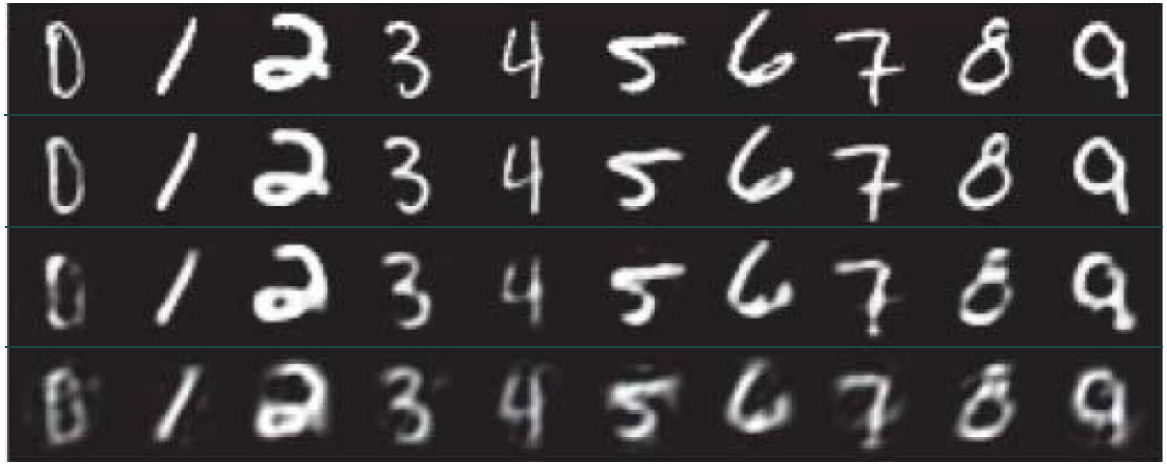
6.9.4 Autoencoder

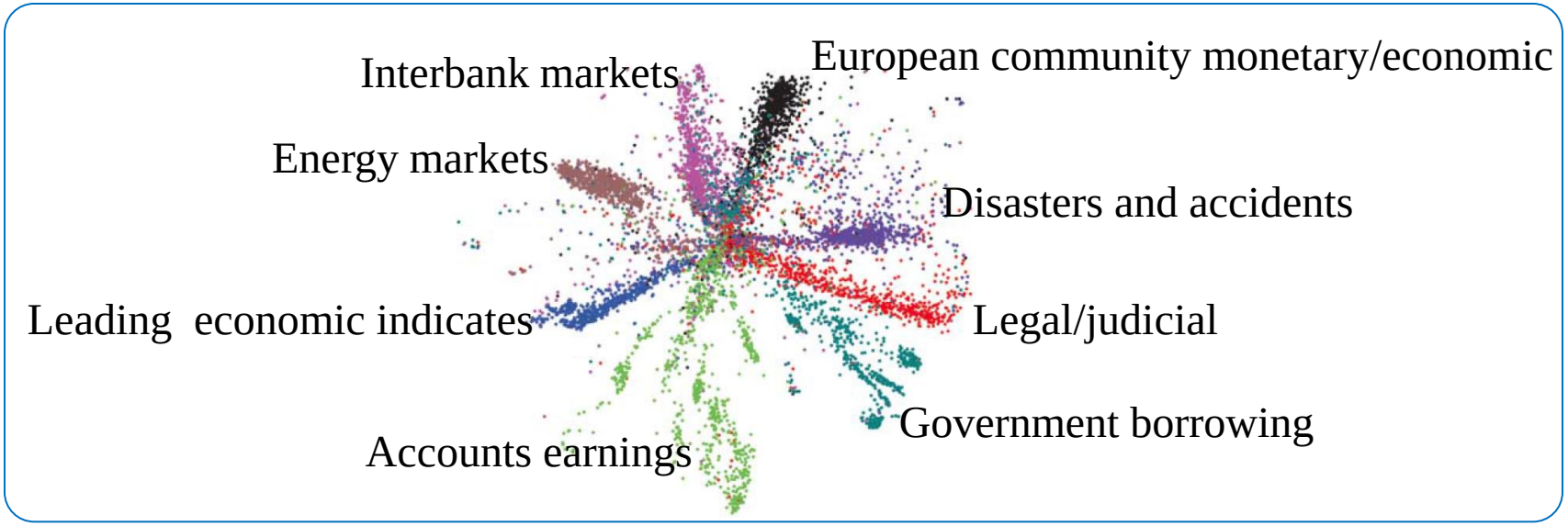
- 作为特征提取器来使用：
  - 只应用编码器：样本逐步通过多个编码器（层），然后以最后一层编码器的输出作为该样本的新特征。



在完成 Autoencoder 的学习任务之后，在实际应用中，在解码阶段学习得到的权重将不进行考虑。

- **应用举例—重构、降维**


Ground Truth
AE
Logistic PCA
PCA


Interbank markets
European community monetary/economic
Energy markets
Disasters and accidents
Leading economic indicates
Legal/judicial
Government borrowing
Accounts earnings

6.9.4 Autoencoder

- **扩展 1**
  - 编码阶段所获得的网络其实质是一种特征学习。层级越高，特征的语义性、抽象性、结构性越明显。
  - 对于分类任务，可以事先用 Autoencoder 对数据进行学习。然后以学习得到的权值作为始初权重，采用带有标签的数据对网络进行再次学习，即微调技术(fine-tuning)。
    - 可在编码阶段的最后一层加上一个分类器，比如一个多层感知器（MLP）。

6.9.4 Autoencoder

- **扩展 2**
  - Autoencoder的输入输出在同一空间，训练目标是重构输入数据
  - 如果使encoder的输入和decoder的输出对应于不同的数据空间（模态），并引入任务相关的训练目标，则Autoencoder扩展为一般的encoder-decoder框架，该框架在CV/NLP领域极为常用
    - 机器翻译：A语言→ B语言
    - 图像描述：图像→语言
    - 语义分割：图像→语义标签
    - 文本图像生成：语言→图像

6.9.4 Autoencoder

- **扩展 3**
  - Autoencoder的编码器和解码器都是确定性映射
  - 如果在decoder的输入层和输出层分别引入随机性，输入层对应隐变量**h**，输出层对应观测变量**x**；则decoder扩展为产生式模型 $p(\mathbf{x}|\ \mathbf{h})$，用于描述隐变量已知条件下，观测变量的分布
  - encoder的输出对应已知观测变量下隐变量的后验分布$p(\mathbf{h}|\ \mathbf{x})$，训练结束后，encoder不再使用
  - 这就是著名的变分自动编码器VAE

Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR, 2014.

6.9.5 Recurrent Neural Networks

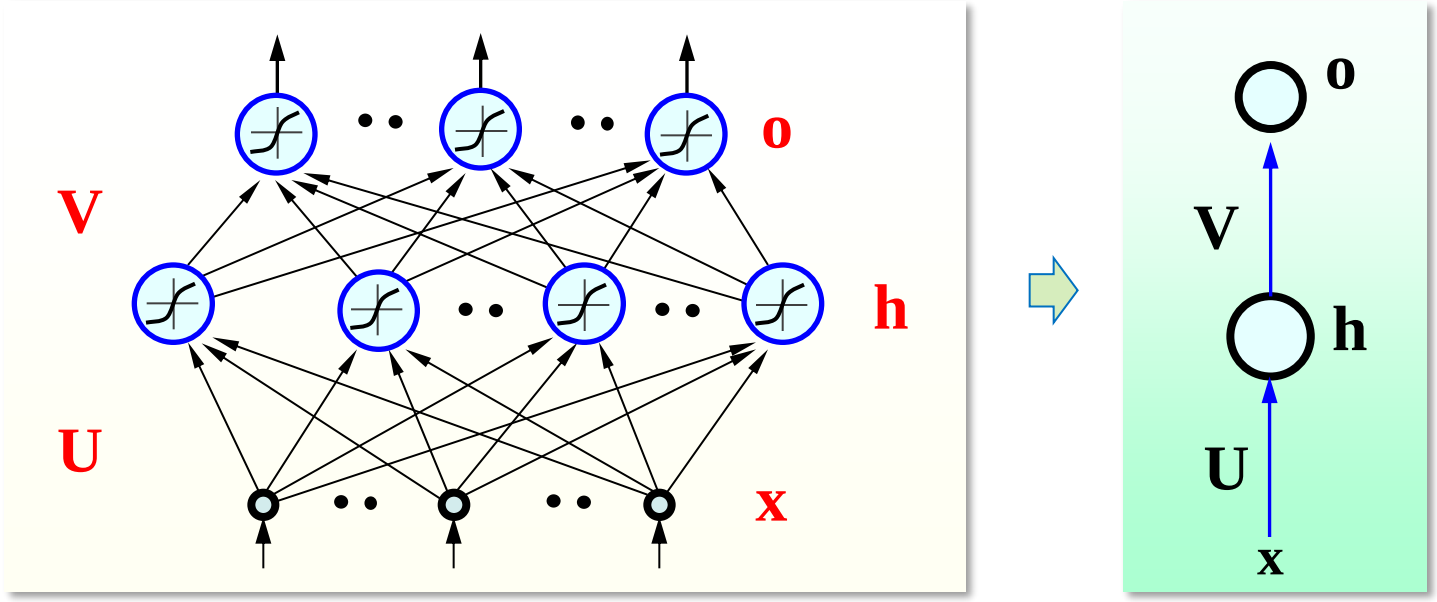
- **前馈神经网络**
  - 信息向前传递，层内结点之间并不连接，适合于处理静态、独立同分布的数据分析，如回归、分类等任务。
- **反馈神经网络**
  - 全部或者部分神经元可以接受来自其它神经元或自身的神经网络结构，其拓扑结构可以是网状的，也可以是具有一定层级的。
  - 人们通常将反馈神经网络视为一个动态系统，主要关心其随时间变化的动态过程。
    - Hopfield 网络

6.9.5 Recurrent NN

- **Recurrent NN（RNN）**
  - 至少包含一个反馈连接的神经网络结构。因此，网络的输出可以沿着一个loop 进行流动。这种网络结构特别适合于处理时序数据。
- **几种经典的RNN：**
  - Hopfield 网络 – 全连接
  - Elman 网络：包含输入层、隐含层和输出层，但隐含层和输入层之间存在一个反馈连接，这种连接通常称为recurrent 连接，即回归连接。
    - 这种回归连接使得Elman 网络具有检测和产生时变模式的能力
  - 对角自反馈神经网络：隐含层的神经元具有反馈连接，但隐含层神经元之间并不连接

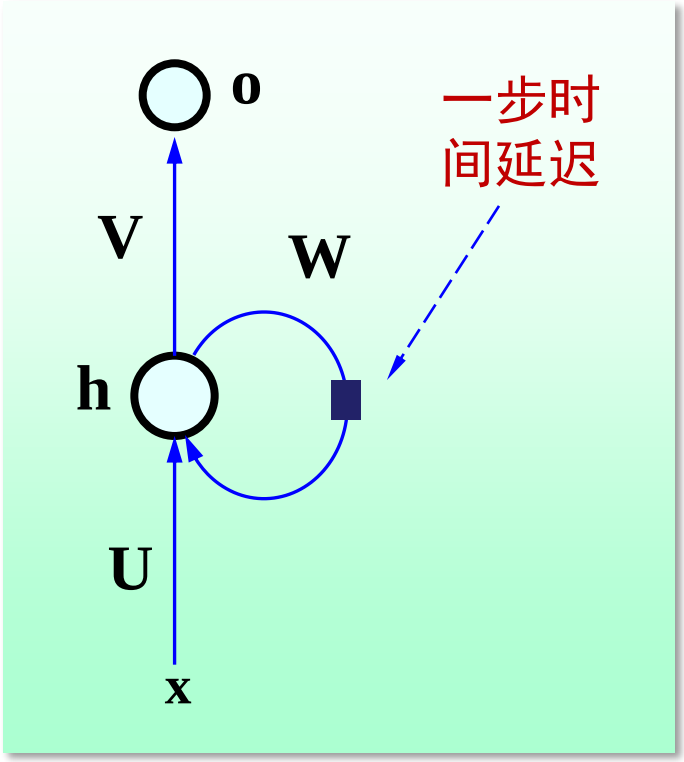
6.9.5 Recurrent NN

- **前向神经网络**
  - **x**：输入信号（向量）；　**o**：网络输出（向量）
  - **h**：隐含层的输出（向量）
  - **U**：输入至隐含层的**连接权重矩阵**　(input to hidden)
  - **V**：隐含层至输出层的**连接权重矩阵**　(hidden to output)



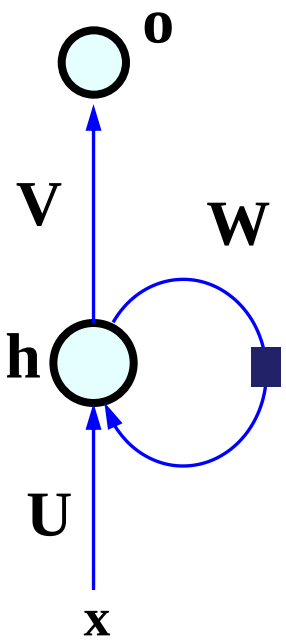
6.9.5 Recurrent NN

- **RNN的基本结构**
  - **x**：输入信号（向量）
  - **o**：网络输出（向量）
  - **h**：隐含层的输出（向量）
  - **U**：输入至隐含层的**连接权重矩阵** (input to hidden)
  - **V**：隐含层至输出层的**连接权重矩阵** (hidden to output)
  - **W**：隐含层神经元之间的连接权重矩阵 (hidden to hidden)



6.9.5 Recurrent NN

- **网络结构**



$$\mathbf{net}_t = \mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t$$
$$\mathbf{h}_t = f(\mathbf{net}_t)$$
$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{h}_t$$

b, c：　偏移向量
U, V, W：权重矩阵
$f(\,)$：　激励函数，比如tanh

网络的功能可以解释为：How the state $\mathbf{h}_t$ at time t captures and summarizes the information from the previous inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$.

6.9.5 Recurrent NN

- **A simple comparison with Hopfield**

RNN:

$$\mathbf{h}_t = f(\mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t)$$

$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{h}_t$$

Hopfield:

$$\mathbf{h}_0 = \mathbf{x}$$

$$\mathbf{h}_t = f(\mathbf{b} + \mathbf{W}\mathbf{h}_{t-1}), \qquad \mathbf{W} = \mathbf{W}^T$$

$$\mathbf{o} = \mathbf{h}_T$$

6.9.5 Recurrent NN

- **A simple comparison with MLP**

RNN:

$$\mathbf{h}_t = f(\mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t)$$

$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{h}_t$$

MLP:

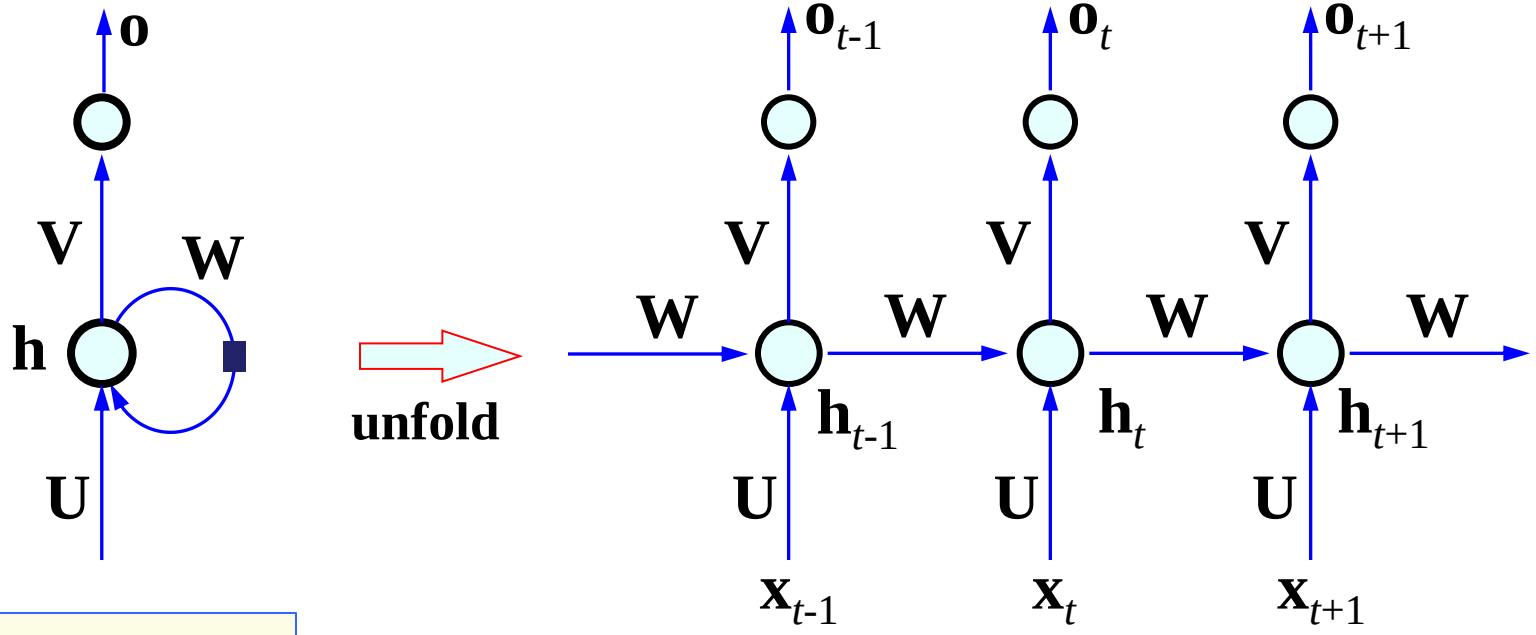
$$\mathbf{h} = f(\mathbf{b} + \mathbf{U}\mathbf{x})$$

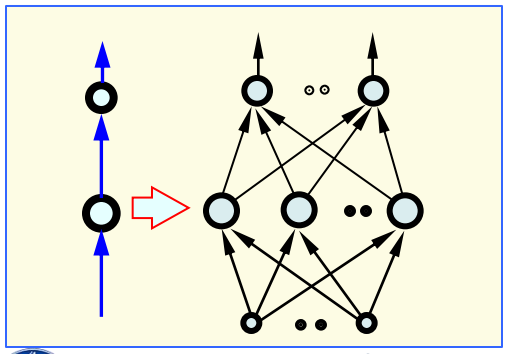
$$\mathbf{o} = \mathbf{c} + \mathbf{V}\mathbf{h}$$

6.9.5 Recurrent NN

- 按时间顺序展开

权值共享: U, V, W



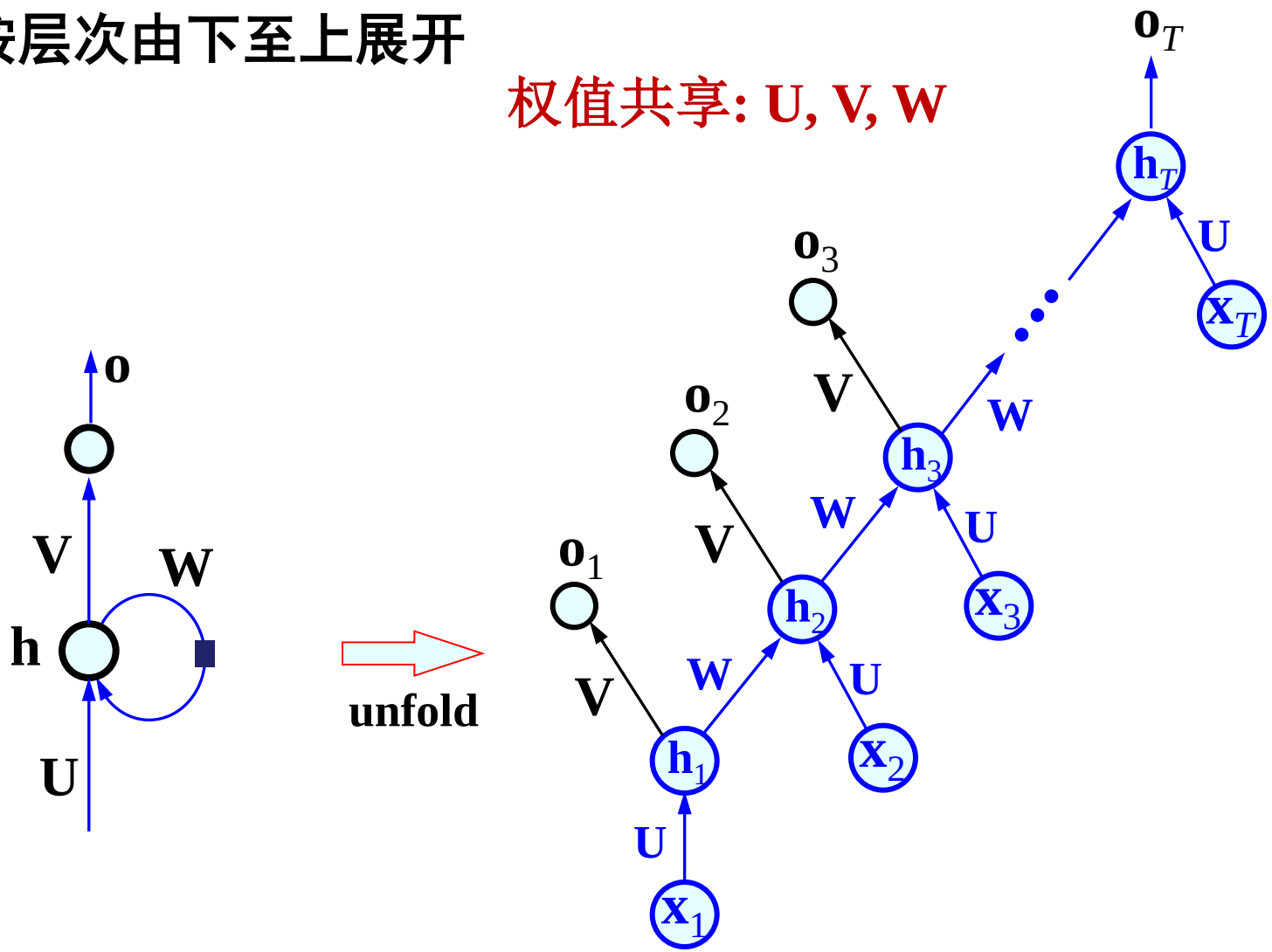


flow graph

$$\mathbf{h}_t = f(\mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t)$$

- 按层次由下至上展开

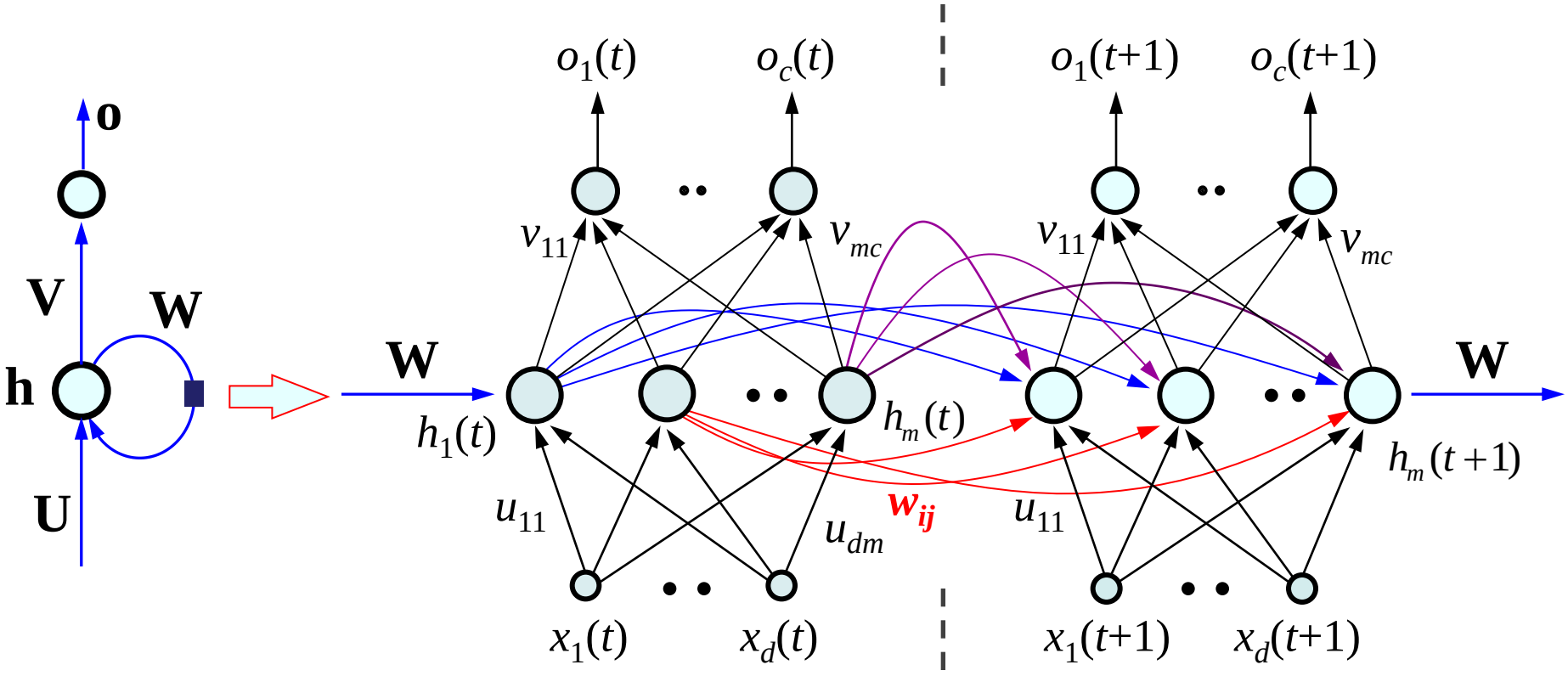
权值共享: U, V, W



hierarchical graph

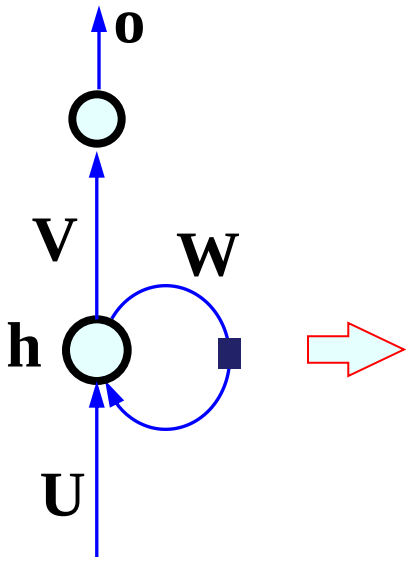
6.9.5 Recurrent NN

- **按时间顺序全结点展开**



6.9.5 Recurrent NN

- **网络目标函数（示例：序列标注问题）**



$$\mathbf{net}_t = \mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t$$

$$\mathbf{h}_t = f(\mathbf{net}_t)$$

$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{h}_t$$

$$\hat{\mathbf{y}}_t = \text{soft max}(\mathbf{o}_t)$$ (例如)

The total loss for a given input/target sequence pair (**x**, **y**) would then be just **the sum of the losses over all the time steps**:

$$L(\mathbf{x}, \mathbf{y}) = \sum_t L(\mathbf{x}, y_t),$$

对分类：$L(\mathbf{x}, \mathbf{y}) := -\sum_t \log \hat{y}_{y_t}$

6.9.5 Recurrent NN

- **Softmax：** 样本分别属于c个类别的概率

$$\text{soft max}(\mathbf{o}) = \hat{\mathbf{y}}(\mathbf{o}) = \left(\frac{\exp(o_1)}{\sum_{j=1}^{c}\exp(o_j)}, \frac{\exp(o_2)}{\sum_{j=1}^{c}\exp(o_j)}, \cdots, \frac{\exp(o_c)}{\sum_{j=1}^{c}\exp(o_j)} \right)$$

o_j为输出层第 j 个结点收集到的加权和

- **Softmax loss**

假设数据 **x** 对应的类别为 y，y取 1, 2, …, c 中的某个整数。我们的目标就是要最大化 $[\text{softmax}(o)]_y$ 的值 (即第 y 个分量）。

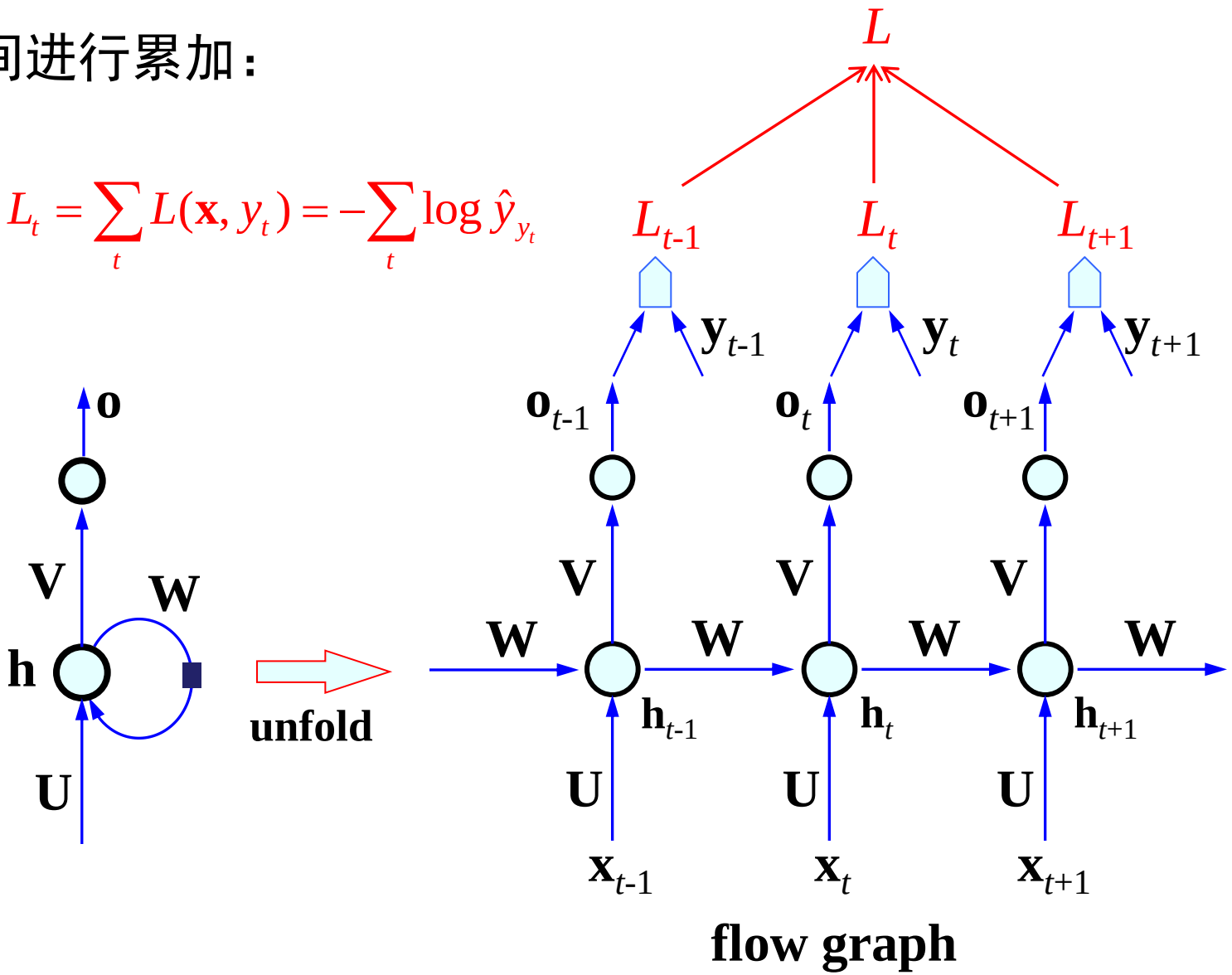
采用负对数似然，有：

$$l(\mathbf{x}, y) = -\log\left(\left[\hat{\mathbf{y}}(\mathbf{o})\right]_y\right) = -\log\left(\frac{e^{o_y}}{\sum_{j=1}^{c} e^{o_j}} \right) = \log\left(\sum_{j=1}^{c} e^{o_j}\right) - o_y$$

$[\ ]_y$ 表示向量的第y个分量， y取对应类别的整数

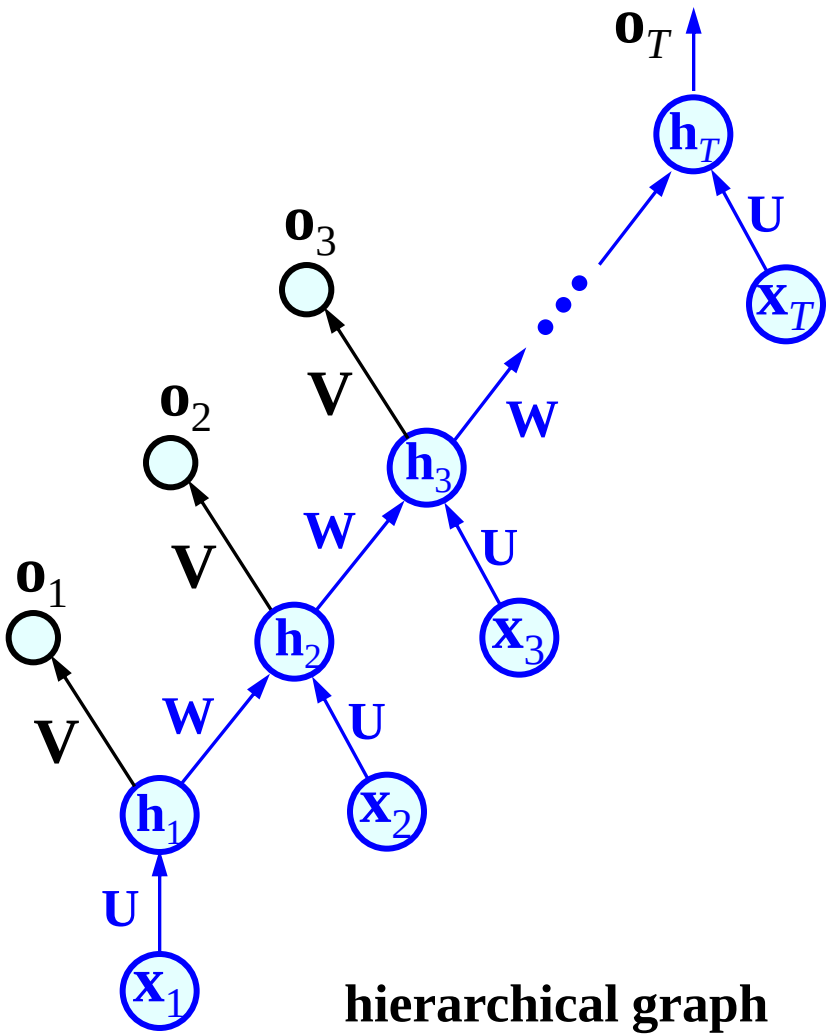
按时间进行累加：

$$L(\mathbf{x},\mathbf{y}) = \sum_t L_t = \sum_t L(\mathbf{x}, y_t) = -\sum_t \log \hat{y}_{y_t}$$



6.9.5 Recurrent NN

- RNN的训练
  - 从多层前向网络的角度来看，可采用类似的反向传播算法来训练网络。
  - 唯一不同之处在于：权重是共享的，是相同的。



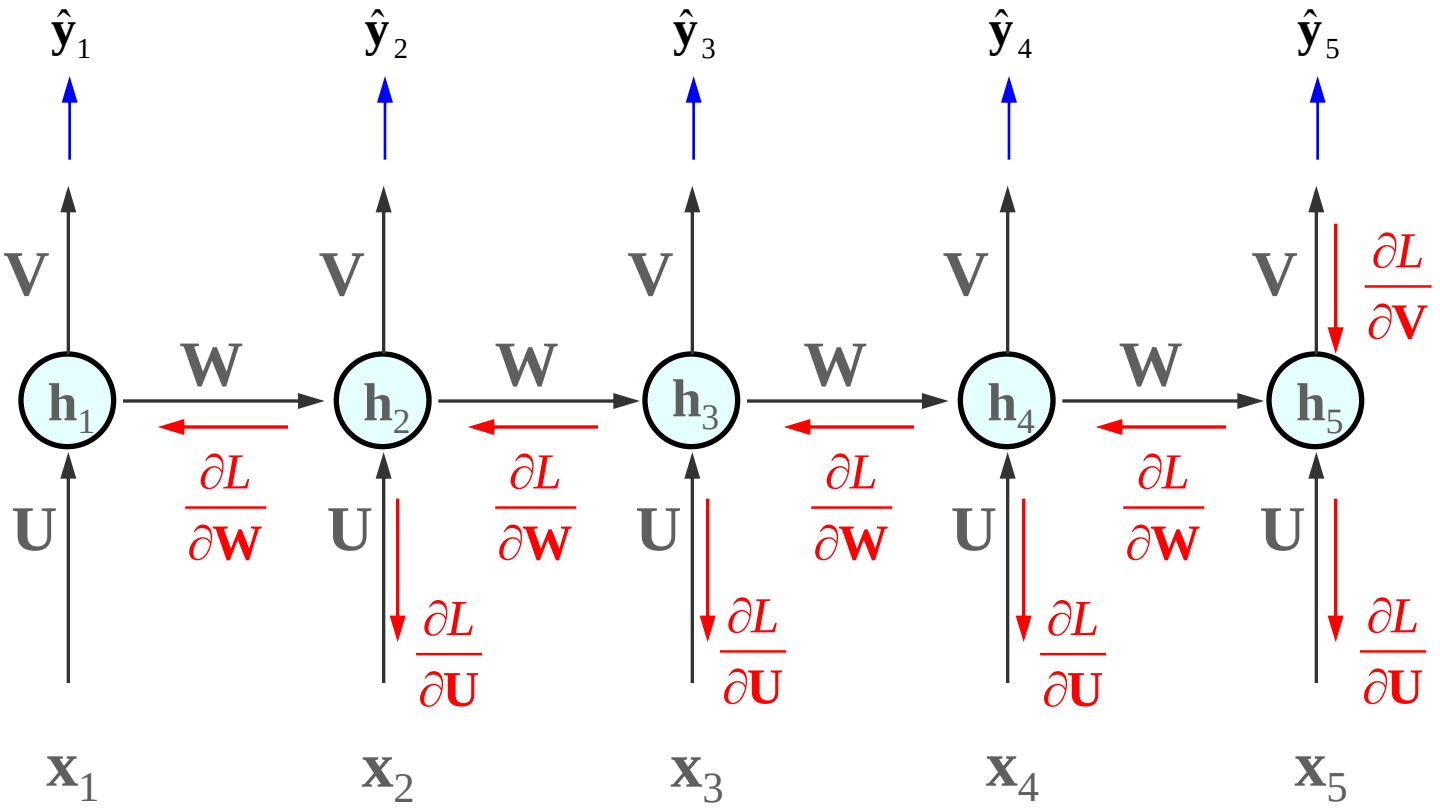
hierarchical graph

- RNN的训练
  - Back Propagation Through Time (BPTT)算法：
    - 前向计算每个神经元的输出值；
    - 反向计算每个神经元的误差项值，它是损失函数L对各个神经元的加权输入（加权和）的偏导数；
    - 计算每个权重的梯度。
    - 最后，利用随机梯度下降算法更新权重。
  - **以一步时间延迟为例：**
    - The nodes in our flow graph will be the sequence of $\mathbf{x}_t$'s, $\mathbf{h}_t$'s, $\mathbf{o}_t$'s, and L_t's. （输入、隐状态、输出）
    - The parameters are **U**, **V**, **W**, **b**, **c**
  - 核心任务是计算目标函数对各参数的导数：

$$\nabla_{\mathbf{U}} L, \quad \nabla_{\mathbf{V}} L, \quad \nabla_{\mathbf{W}} L, \quad \nabla_{\mathbf{b}} L, \quad \nabla_{\mathbf{c}} L$$

6.9.5 Recurrent NN

- RNN的训练：BPTT



6.9.6 Long Short-Term Memory (LSTM)

- **RNN的问题**

  传统RNN将激活函数作用于“当前时刻输入信号＋上一时刻的隐含结点输出”的仿射变换结果，作为当前隐含结点的输出：

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b})$$

  – 这种更新方式使 $\boldsymbol{h}_t$ 随时间 t 发生较大改变，造成历史信息的快速遗忘

  – 训练时，梯度容易出现消失或爆炸

$$\text{recall} \quad \frac{\partial E}{\partial \mathbf{net}_l} = f'(\mathbf{net}_l) \odot \mathbf{W}_{l,l+1} \frac{\partial E}{\partial \mathbf{net}_{l+1}}$$

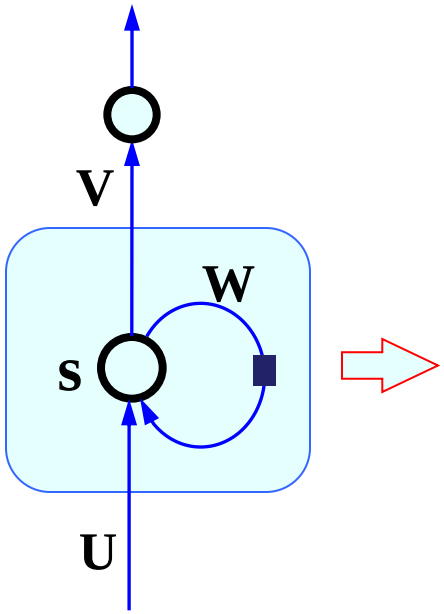
$\odot$: element wise multiplication

6.9.6 LSTM

- LSTM的核心思想：
  - 通过精细控制对“历史信息（记忆）”的操作，缓解了RNN的遗忘问题和训练问题。
- LSTM的结构
  - LSTM网络将隐含层的细胞单元（隐含单元）设计为所谓的LSTM细胞单元
  - 每一个LSTM细胞含有与传统的RNN细胞相同的输入和输出，但它额外包含一个控制信息流动的“门结点系统”。
  - 门系统包含三个部分，除了对LSTM细胞的输入、输出进行加权控制之外，还对记忆（遗忘）进行加权控制

6.9.6 LSTM

- **结构**
  - 由传统的神经元结构变为带有三个权重门的结构



由简单的输入、输出、隐含层自循环增加了三个门单元：

—遗忘门：控制对细胞内部状态的遗忘程度

—输入门：控制对细胞输入的接收程度

—输出门：控制对细胞输出的认可程度

三个门均由当前输入信号和隐含层前一时刻的输出共同决定

- **输入门“input gate”的作用是提供一个是否接收当前输入(包含信号和隐结点状态)信息的权重(0~1)。**
  - 其值取决于当前输入信号和前一时刻隐含层的输出:

$$\mathbf{in}_t = \text{sigmoid}\left(\mathbf{b}_{in} + \mathbf{U}_{in}\mathbf{x}_t + \mathbf{W}_{in}\mathbf{h}_{t-1}\right)$$

$\mathbf{x}_t$ ：当前输入向量；

$\mathbf{h}_{t-1}$ ：当前隐含层向量，所有结点在t-1时刻的输出；

$\mathbf{b}_{in}$ ：输入门的偏移向量；

$\mathbf{U}_{in}$ ：输入门的输入权重矩阵；

$\mathbf{W}_{in}$ ：输入门的回归权重矩阵；

$\mathbf{in}_t$ ： $=[in_{t,1}, in_{t,2}, in_{t,3}, \ldots.]$ (用向量记录所有权重）

- **遗忘门“forget gate”的作用是提供一个遗忘当前状态的权重(0~1)。**
  - 其值取决于当前输入信号和前一时刻隐含层的输出:

$$\mathbf{f}_t = \text{sigmoid}\left(\mathbf{b}_f + \mathbf{U}_f \mathbf{x}_t + \mathbf{W}_f \mathbf{h}_{t-1}\right)$$

$\mathbf{x}_t$ ：当前输入向量；

$\mathbf{h}_{t-1}$：当前隐含层向量，包含所有结点的t-1时刻输出；

$\mathbf{b}_f$ ：遗忘门的偏移向量；

$\mathbf{U}_f$ ：遗忘门的输入权重矩阵；

$\mathbf{W}_f$：遗忘门的回归权重矩阵；

$\mathbf{f}_t$ ： $=[f_{t,1}, f_{t,2}, f_{t,3}, \ldots.]$ (用向量记录所有权重）

- **LSTM 细胞产生的新记忆，由输入与遗忘两部分组成：**

读出旧记忆　　候选新记忆

产生新记忆：　$s_{t,i} = f_{t,i} \cdot s_{t-1,i} + in_{t,i} \cdot c_{t,i}$

$$\mathbf{c}_t = \tanh\left(\mathbf{b} + \mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1}\right)$$

$\mathbf{x}_t$ ： 当前输入向量；

$\mathbf{h}_{t-1}$ ： 当前隐含层向量，包含所有结点的t-1时刻的输出；

$\mathbf{b}$ ： 正常的输入层至隐层的偏移向量；

$\mathbf{U}$ ： 正常的输入层至隐层的输入权重矩阵；

$\mathbf{W}$ ： 正常的输入层至隐层的回归权重矩阵；

贡献候选新记忆

$\mathbf{s}_{t+1}$ ： $=[s_{t+1,1}, s_{t+1,2}, s_{t+1,3}, \ldots.]$　(用向量记录所有新记忆)

$\mathbf{c}_t$ ： $=[c_{t,1}, c_{t,2}, c_{t,3}, \ldots.]$　(用向量记录所有候选记忆)

- **输出门“output gate”的作用是对当前输出提供一个 0~1 之间的权重（对输出的认可程度）。**
  - 其值取决于当前输入信号和前一时刻隐含层的输出:

$$\mathbf{o}_t = \text{sigmoid}\left(\mathbf{b}_o + \mathbf{U}_o\mathbf{x}_t + \mathbf{W}_o\mathbf{h}_{t-1}\right)$$

$\mathbf{x}_t$ ： 当前输入向量；

$\mathbf{h}_{t\text{-}1}$ ： 当前隐含层向量，包含所有结点的t-1时刻的输出；

$\mathbf{b}_o$ ： 输出门的偏移向量；

$\mathbf{U}_o$ ： 输出门的输入权重矩阵；

$\mathbf{W}_o$ ： 输出门的回归权重矩阵；

$\mathbf{o}_t$ ： $=[o_{t,1}, o_{t,2}, o_{t,3}, \ldots.]$ (用向量记录所有权重)

6.9.6 LSTM

- **输出**
  - 隐含层结点的输出由激励后的内部状态（此时称为记忆）与输出门权重共同决定：

$$h_{t,i} = o_{t,i} \tanh\left(s_{t,i}\right)$$

输出门提供的权重

6.9.6 LSTM

- **结构描述——采用矩阵形式**

遗忘门、输入门、输出门：

$$\mathbf{f}_t = \text{sigmoid}\left(\mathbf{b}_f + \mathbf{U}_f \mathbf{x}_t + \mathbf{W}_f \mathbf{h}_{t-1}\right)$$

$$\mathbf{in}_t = \text{sigmoid}\left(\mathbf{b}_{in} + \mathbf{U}_{in} \mathbf{x}_t + \mathbf{W}_{in} \mathbf{h}_{t-1}\right)$$

$$\mathbf{o}_t = \text{sigmoid}\left(\mathbf{b}_o + \mathbf{U}_o \mathbf{x}_t + \mathbf{W}_o \mathbf{h}_{t-1}\right)$$

Cell产生的新记忆：

$$\mathbf{s}_t = \mathbf{f}_t \otimes \mathbf{s}_{t-1} + \mathbf{in}_t \otimes \mathbf{c}_t$$

Cell的输出：

$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh\left(\mathbf{s}_t\right)$$

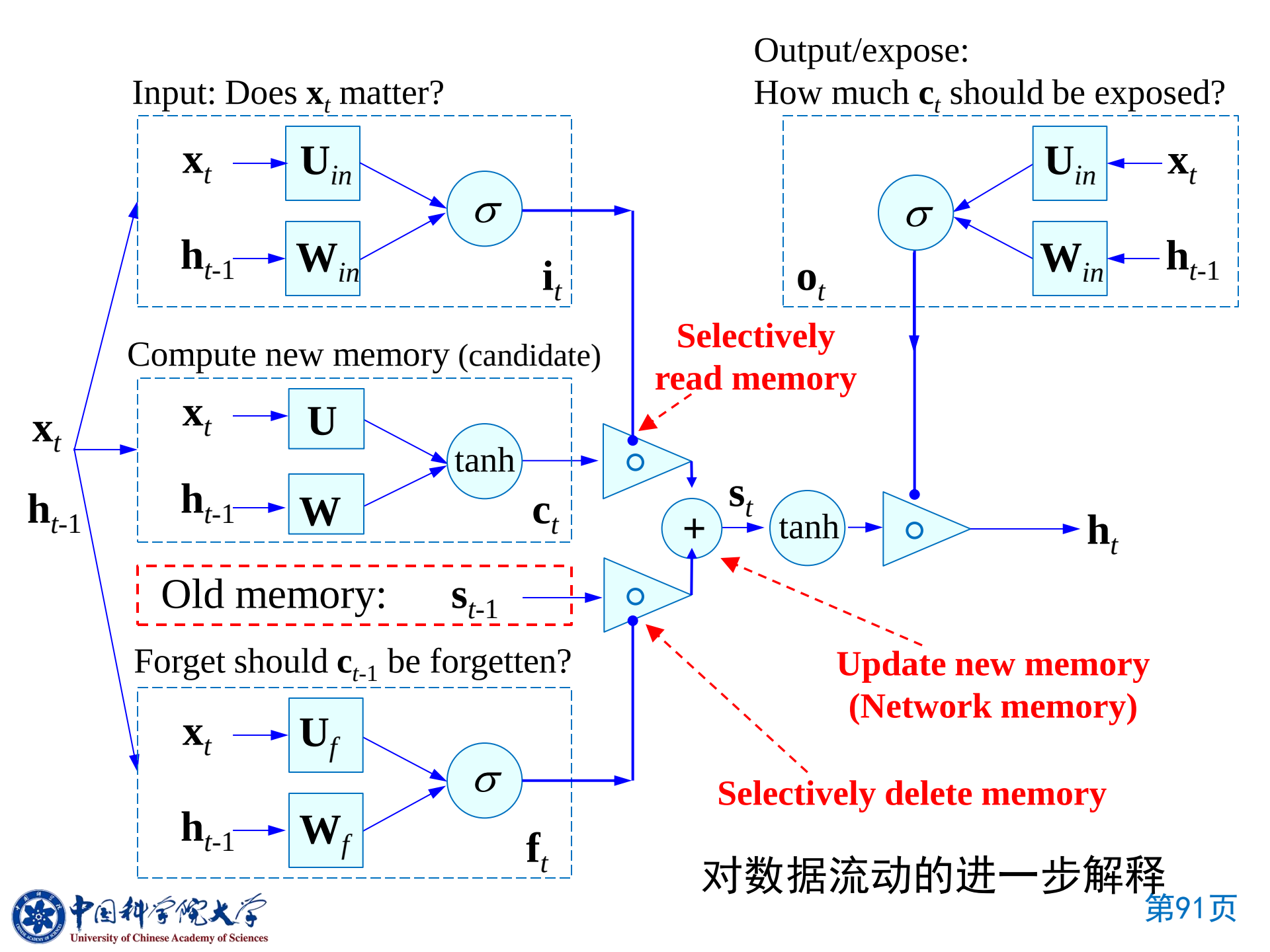
候选记忆（新贡献部分）：

$$\mathbf{c}_t = \tanh\left(\mathbf{b} + \mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1}\right)$$

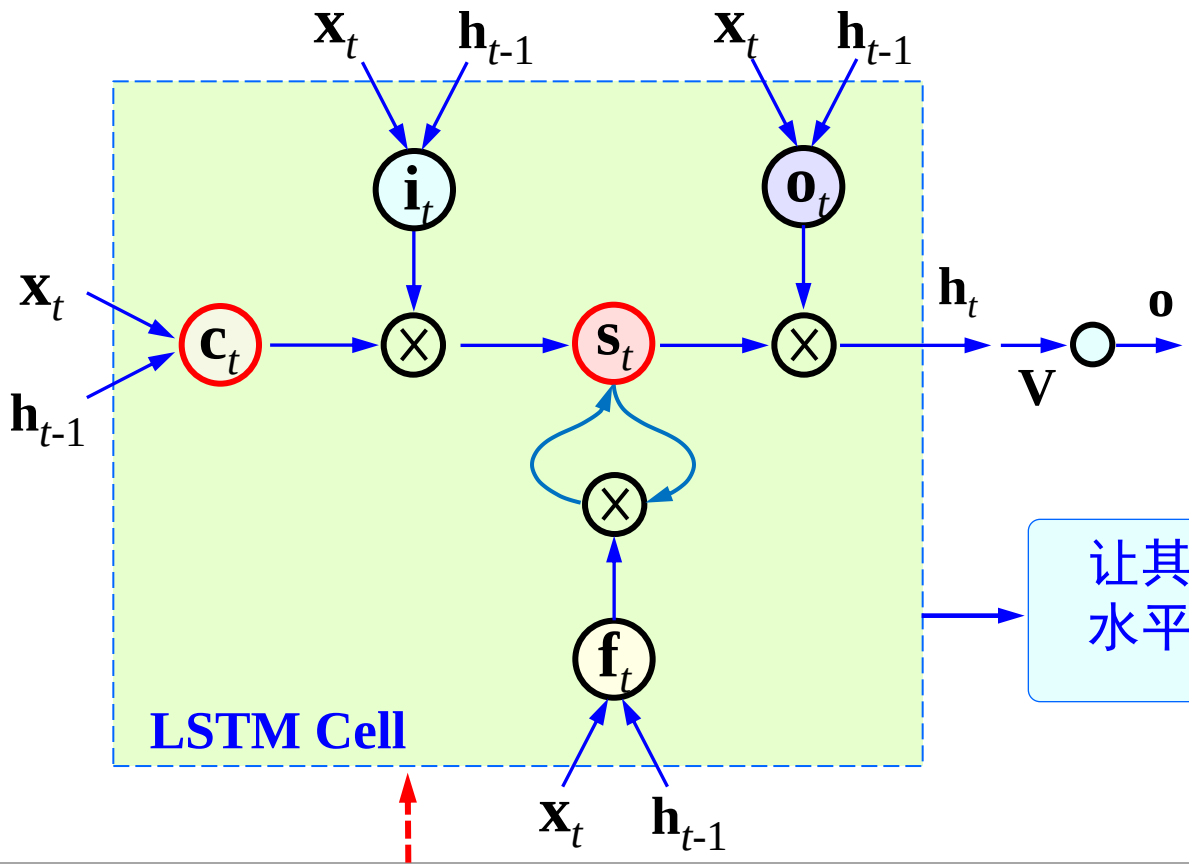
网络的输出：

$$\mathbf{z}_t = \text{softmax}\left(\mathbf{V}\mathbf{h}_t + \mathbf{c}\right)$$

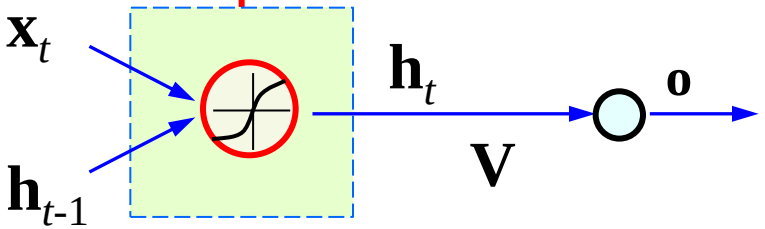
$\otimes$: element wise multiplication



对数据流动的进一步解释



让其始终维持在一个稳定的水平（即1附近）



传统RNN

$$\frac{\partial L_t}{\partial \mathbf{s}_k} \approx \left(f'(\mathbf{s})\mathbf{W} \right)^t \frac{\partial L_t}{\partial \mathbf{s}_{k+t}}$$

- **网络训练**
  - 估计如下矩阵或向量

    $\mathbf{U}_{in}$，$\mathbf{W}_{in}$，$\mathbf{b}_{in}$，$\mathbf{U}_{o}$，$\mathbf{W}_{o}$，$\mathbf{b}_{o}$，$\mathbf{U}_{f}$，$\mathbf{W}_{f}$，$\mathbf{b}_{f}$，$\mathbf{U}$，$\mathbf{W}$，$\mathbf{V}$，$\mathbf{c}$
  - 基于两点：
    - 基于信息流动的方式，采用反向传播算法
    - 计算目标函数、隐状态(t)对各变量的偏导数
  - 并不是说LSTM完全解决了梯度消失或扩散的问题，只是有所缓和。仍可考虑如下技术：
    - 充分利用二阶梯度的信息（但通常并不鼓励）
    - 更好的初始化
    - 动量机制
    - 对梯度的模或者元素作一些裁剪
    - 正则化一鼓励信息流动

6.9.7 Transformer

- RNN的问题
  - 由于RNN、LSTM模型的序列属性，在更新过程中必然逐渐丢失历史信息

$$h_t = f(h_{t-1}, x_t)$$

例如：h_{100}中只残留很少的x_1信息

  - 但在一些复杂的NLP任务中，长距依赖非常重要且普遍，这成为制约RNN、LSTM性能的关键
  - 如何设计有效的机制对长距依赖关系建模？
    - 注意力机制 (attention mechanism)

D. Bahdanau, etc. Neural machine translation by jointly learning to align and translate. ICLR, 2015.
A. Vaswani, etc. Attention is all you need. NIPS, 2017.

6.9.7 Transformer

- 注意力机制的基本原理

对于输入序列{ $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ }，计算 $\mathbf{x}_t$ 的特征表示

– $\mathbf{x}_t$ 与 $\mathbf{x}_i$ 的相似度

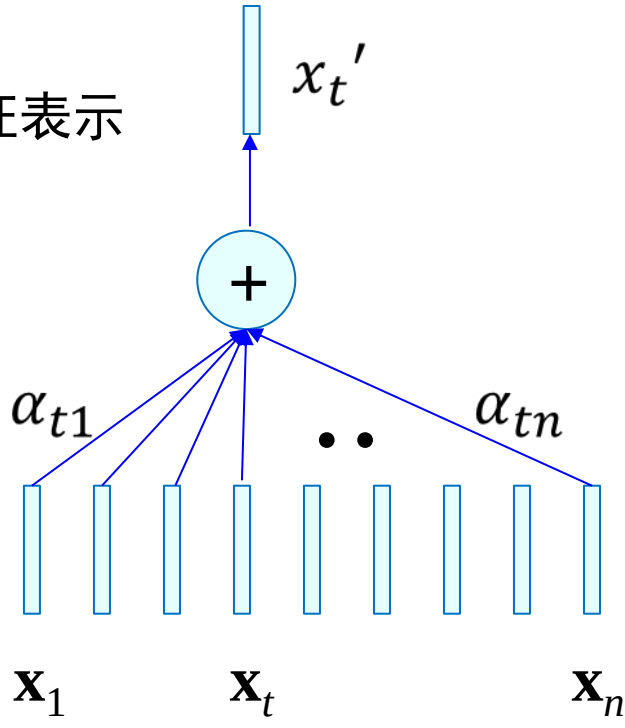
$$s_{ti} = \mathbf{x}_t^T \mathbf{x}_i \quad i = 1, \ldots, n$$

– 注意力系数

$$\alpha_{ti} = \frac{\exp(s_{ti})}{\sum_{j=1}^{n} \exp(s_{tj})}$$

– 加权和

$$\mathbf{x}_t' = \sum_{i=1}^{n} \alpha_{ti} \mathbf{x}_i$$



✓ 按注意力系数对所有数据求和（凸组合）

✓ 实现了全局感受野，是Transformer的核心

6.9.7 Transformer

- 单头自注意力(single head self attention)

  实际中，引入三个线性变换$\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$

  Query: $\mathbf{W}^Q\mathbf{x}_t$　　Keys: $\{\mathbf{W}^K\mathbf{x}_i\}$　　Values: $\{\mathbf{W}^V\mathbf{x}_i\}$

  – x_t与x_i的相似度

$$s_{ti} = (\mathbf{W}^Q\mathbf{x}_t)^T(\mathbf{W}^K\mathbf{x}_i)$$

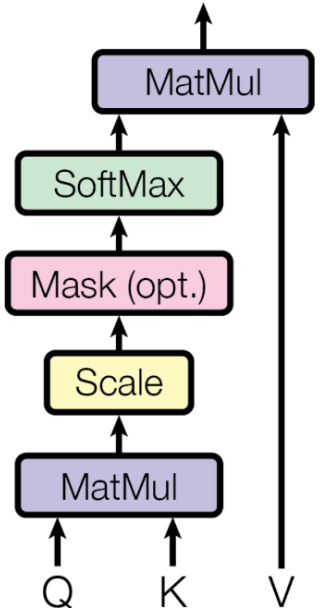
  – 注意力系数

$$\alpha_{ti} = \frac{\exp(s_{ti})}{\sum_{j=1}^{n}\exp(s_{tj})}$$

  – 加权和

$$\mathbf{x}_t' = \sum_{i=1}^{n}\alpha_{ti}(\mathbf{W}^V\mathbf{x}_i)$$

Scaled Dot-Product Attention



6.9.7 Transformer

- 多头自注意力(multi-head self attention)

  为进一步提升模型能力，引入多组注意力参数$\{\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V\}$，$h=1,...,H$

  - 注意力系数

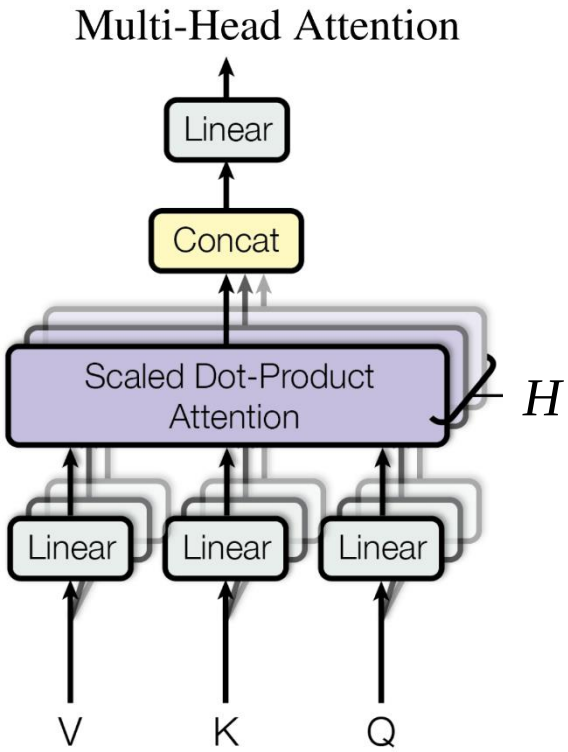
$$\alpha_{ti}^h = \frac{\exp((\mathbf{W}_h^Q \mathbf{x}_t)^T (\mathbf{W}_h^K \mathbf{x}_i))}{\sum_{j=1}^{n} \exp((\mathbf{W}_h^Q \mathbf{x}_t)^T (\mathbf{W}_h^K \mathbf{x}_j))}$$

  - 加权和

$$\mathbf{x}_t^h = \sum_{i=1}^{n} \alpha_{ti}^h (\mathbf{W}_h^V \mathbf{x}_i)$$

  - 多头特征融合

$$\mathbf{x}_t' = \mathbf{W}^O \text{Concat}(\mathbf{x}_t^1, \mathbf{x}_t^2, ..., \mathbf{x}_t^H)$$



6.9.7 Transformer

- 自注意力的矩阵表示

对于输入序列$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$，计算所有数据点的特征表示

记 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in R^{dxn}$

– 注意力系数

$$\Lambda^h = \text{softmax}((\mathbf{W}_h^Q \mathbf{X})^T (\mathbf{W}_h^K \mathbf{X})) \in R^{nxn}$$

非负、列和为1

– 加权和

$$\mathbf{X}^h = (\mathbf{W}_h^V \mathbf{X})\Lambda^h \quad h = 1, ..., H$$

– 多头融合

$$\mathbf{X}' = \mathbf{W}^O \text{Concat}(\mathbf{X}^1, \mathbf{X}^2, ..., \mathbf{X}^H)$$

6.9.7 Transformer

- 自注意力模块：两个子模块
  - 注意力子模块

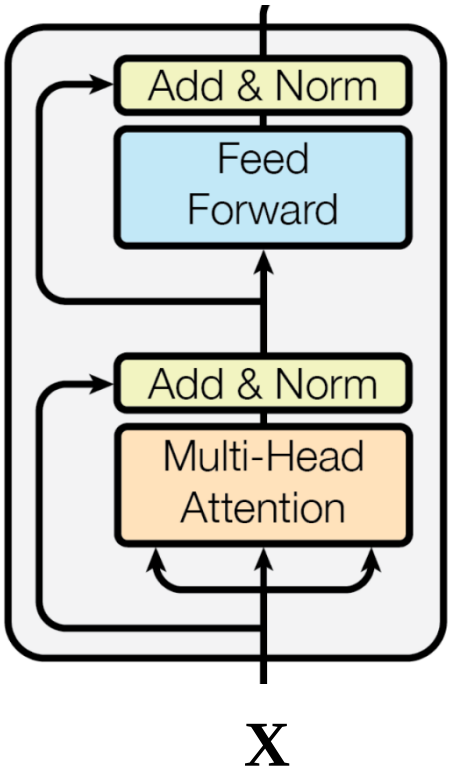
$$\mathbf{X}' = \mathbf{W}^{O}\mathrm{Concat}(\mathbf{X}^{1}, \mathbf{X}^{2}, ..., \mathbf{X}^{H})$$

残差连接+层规范化 $\mathbf{X}'' = \mathrm{LayerNorm}(\mathbf{X} + \mathbf{X}')$

  - 逐点特征变换

$$\mathbf{X}' = \mathbf{W}_2\,\mathrm{ReLU}(\mathbf{W}_1\mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2$$

$$\mathbf{X}'' = \mathrm{LayerNorm}(\mathbf{X} + \mathbf{X}')$$



✓ 输入序列与输出序列的长度相同
✓ 将多个自注意力模块串联起来，构成Transformer encoder

6.9.7 Transformer

- 很多任务中，存在输入序列与输出序列不等长（如机器翻译）甚至不同模态（如图像描述）的问题
- 可采用基于encoder-decoder结构的模型处理此类问题
  - encoder负责对源数据提取特征
  - decoder负责输出目标数据
- 如何让decoder有效利用encoder的输出？

✓ 交叉注意力(cross attention)

6.9.7 Transformer

- 交叉注意力
  - 假设编码器的输出为 $\mathbf{Z}=[\mathbf{z}_1,\mathbf{z}_2,...,\mathbf{z}_n]\in R^{d_1 xn}$
  - 假设解码器第 t 时刻输入为 $\mathbf{y}_t\in R^{d_2}$

相似度：
$$s_{ti}=(\mathbf{W}^Q\mathbf{y}_t)^T(\mathbf{W}^K\mathbf{z}_i)$$

注意力系数：
$$\alpha_{ti}=\frac{\exp(s_{ti})}{\sum_{j=1}^{n}\exp(s_{tj})},\quad i=1,...,n$$

加权和：
$$\mathbf{y}_t'=\sum_{i=1}^{n}\alpha_{ti}(\mathbf{W}^V\mathbf{z}_i)$$

✓ 通过注意力访问全部编码器输出数据
✓ 可类似地使用多头注意力

6.9.7 Transformer

- 交叉注意力模块：三个子模块

设编码器的输出为 $\mathbf{Z}=[\mathbf{z}_1,\mathbf{z}_2,...,\mathbf{z}_n]\in R^{d_1 x n}$

解码器当前的输入为 $\mathbf{Y}=[\mathbf{y}_1,\mathbf{y}_2,...,\mathbf{y}_{t-1}]\in R^{d_2 x (t-1)}$

- 自注意力子模块

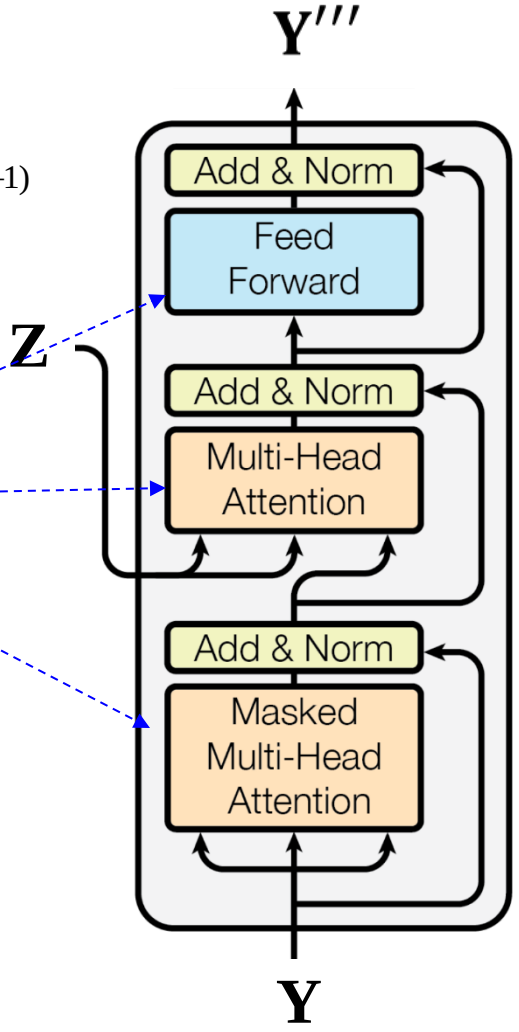
$\mathbf{Y}'=\text{Self\_Attention}(\mathbf{Y})$

- 交叉注意力子模块

$\mathbf{Y}''=\text{Cross\_Attention}(\mathbf{Y}',\mathbf{Z})$

- 逐点特征变换

$\mathbf{Y}'''=\text{FFN}(\mathbf{Y}'')$



6.9.7 Transformer

- 基于Transformer的机器翻译

X={‘hello’, ‘world’, ‘!’} → Y={‘世’, ‘界’, ‘你’, ‘好’, ‘!’}

– 编码器

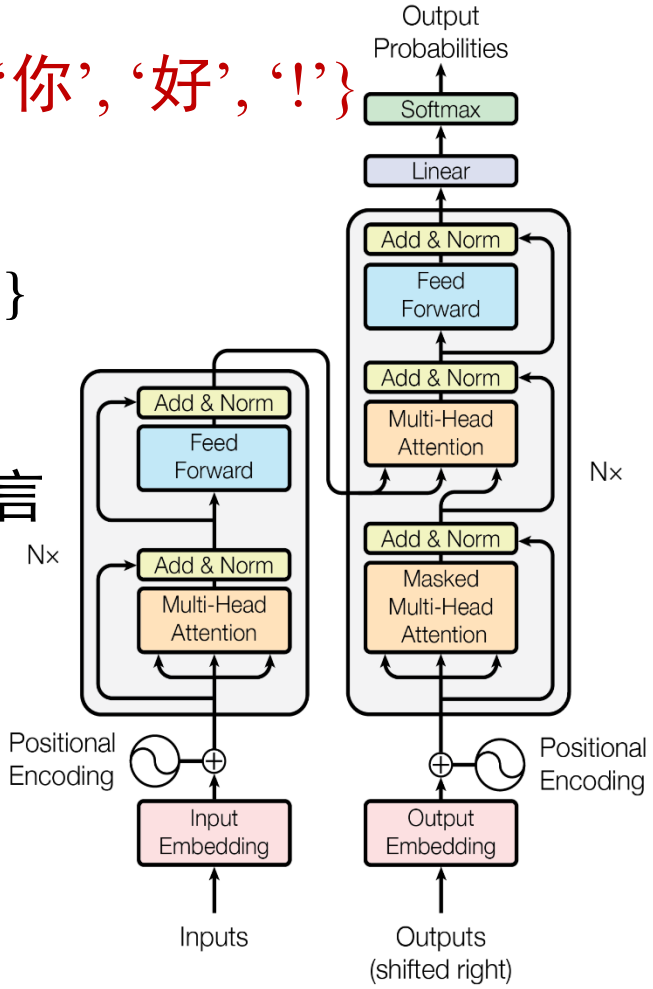
对源语言序列X提特征，得到$\mathbf{Z}=\{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$

– 解码器

以自回归(autoregressive)方式输出目标语言

第 t 时刻，根据已翻译出的结果$\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}\}$和 $\mathbf{Z}$预测 $\mathbf{y}_t$

$$P(y_t \mid y_1, \ldots, y_{t-1}, \mathbf{Z})$$



6.9.7 Transformer

- Transformer和Attention机制本质上是顺序无关的，如何表示位置？
  - 位置编码

$$\mathrm{PE}(pos, 2i)=\sin(pos/10000^{2i/d_{\text{model}}})$$

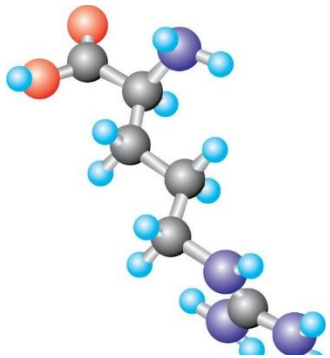
$$\mathrm{PE}(pos, 2i)=\cos(pos/10000^{2i/d_{\text{model}}})$$

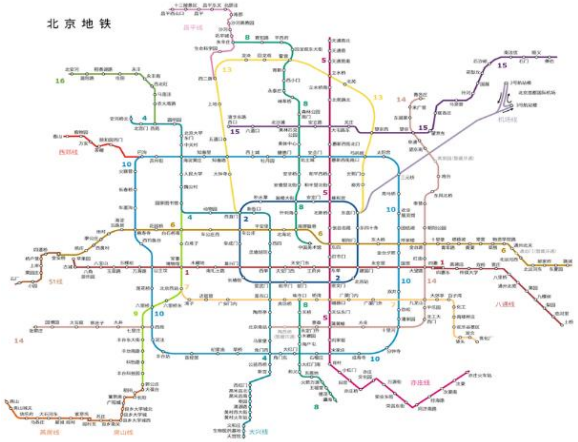
- 如何表示字/词?
  - 分布式词表示(distributed word representation)

下次课的内容

6.9.8 图神经网络 (Graph Neural Networks)

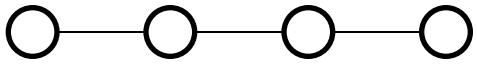
- 图(graph)是一类广泛存在的数据形式
  - 自然界中的图
    - 分子、蛋白质、生物网络
  - 人类社会中的图
    - 传感器网络、交通网络、能源网络
  - 虚拟世界中的图
    - 社交网络、引用网络、知识图谱

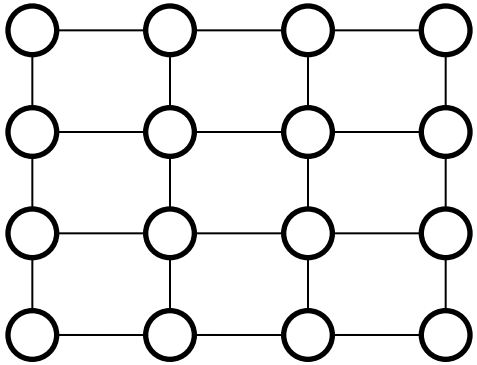


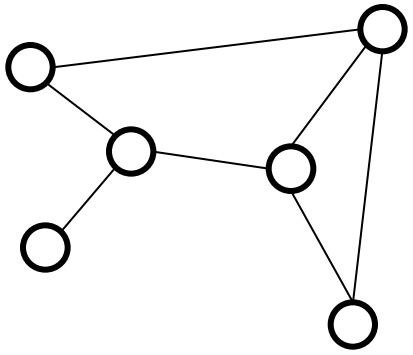


6.9.8 图神经网络 (Graph Neural Networks)

- 数学上，图是一种抽象的数据结构
  - G = (V, E): 图是节点和边的集合
  - V = {v_i}: 节点描述实体，如：人、地点、原子等
  - E = {e_{ij}}: 边描述实体间关系，如：交往、购买、距离、作用力等







sequence：如文本、语音

grid：如图像、视频

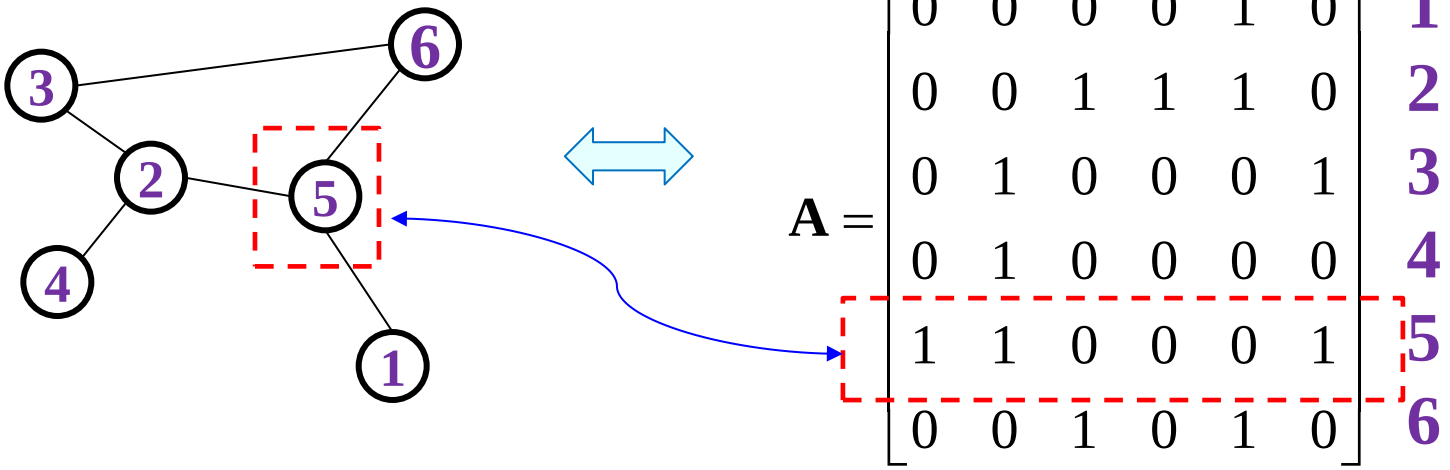
general graph：任意大小和拓扑结构

图的研究由来已久：1736年，欧拉《哥尼斯堡七桥》

6.9.8 图神经网络 (Graph Neural Networks)

- 数学上，图是一种抽象的数据结构
  - G = (V, E): 图是节点和边的集合
  - V = {v_i}: 节点描述实体，如：人、地点、原子等
  - E = {e_{ij}}: 边描述实体间关系，如：交往、距离、作用力等

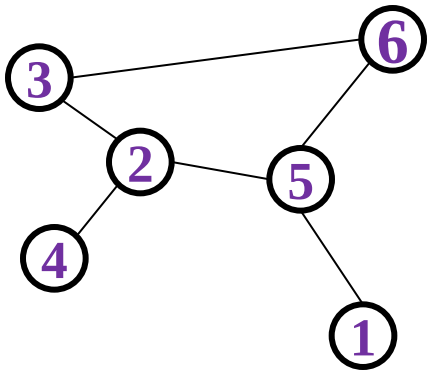
邻接矩阵表示



如果v_i, v_j相连，A_{ij}=1；否则，A_{ij}=0

6.9.8 图神经网络 (Graph Neural Networks)

- 模式识别/机器学习中的图
  - 除了图结构G=(V, E)，节点和边通常还具有特征，一般由向量表示



节点	特征1	特征2	…
v_1	-1	0	
v_2	0	0	
v_3	……		
v_4	……		
v_5	……		
v_6	……		

边	特征1	特征2	…
e_{15}	8	-0.1	
e_{23}	2	0	
e_{24}	……		
e_{25}	……		
e_{36}	……		
e_{56}	……		

$$\mathbf{A} \in \{0,1\}^{n \times n}$$

$$\mathbf{X} = \{\mathbf{x}_i \in \mathbf{R}^{d_1}, v_i \in V\}$$

$$\mathbf{Y} = \{\mathbf{y}_{ij} \in \mathbf{R}^{d_2}, e_{ij} \in E\}$$

6.9.8 图神经网络 (Graph Neural Networks)

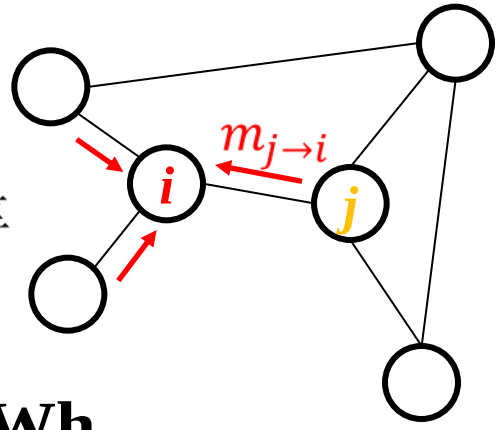
- 图的分类
  - 有向图vs无向图、静态图vs动态图、同质图vs异质图、同配图vs异配图
- 图上的学习任务
  - Graph level
    - 图分类：该分子具有某种性质吗？
    - 图生成：生成具有某种性质的分子结构
    - 图匹配：这两个蛋白质有多相似？
  - Node level
    - 节点分类/节点属性预测：该地铁站在 t 时刻的流量?
    - 节点聚类/社区发现：哪些人会被该病毒感染？
  - Edge level
    - 边分类/边属性预测：用户a会关注用户b吗？

6.9.8 图神经网络 (Graph Neural Networks)

- 如何设计适合图数据的模型？
  - 同时利用图的拓扑结构、节点特征、边特征学习节点表示和边表示
- 图神经网络的基本思想
  - 以“边”为基础进行近邻节点间的信息传递
  - 节点利用从近邻收到的信息更新自己的表示

6.9.8 图神经网络 (Graph Neural Networks)

- 一个简单的例子
  - 消息传递 $\mathbf{m}_{j\to i} = \mathbf{W}\mathbf{h}_j$



$\boldsymbol{h}_j \in \mathrm{R}^{\boldsymbol{d}}$：节点 j 的特征表示

$\mathbf{W} \in R^{d\times d'}$：可学习参数矩阵，所有节点共享

$\boldsymbol{m}_{j\to i} \in \mathrm{R}^{\boldsymbol{d'}}$：节点 j 传递给 i 的信息

  - 消息汇聚 $\mathbf{m}_i = \frac{1}{d_i}\sum_{j\in N(i)} \mathbf{m}_{j\to i} = \frac{1}{d_i}\sum_{j\in N(i)} \mathbf{W}\mathbf{h}_j$

d_i：节点 i 直接相连的节点个数

$N(i)$：所有与节点 i 直接相连的节点集合

$\boldsymbol{m}_i \in \mathrm{R}^{\boldsymbol{d'}}$：节点 i 接收到的全部消息

  - 节点表示更新 $\mathbf{h}_i' = \sigma(\mathbf{W}\mathbf{h}_i + \mathbf{m}_i)$

$\boldsymbol{h}_i' \in \mathrm{R}^{\boldsymbol{d'}}$：更新后节点 i 的特征表示

6.9.8 图神经网络 (Graph Neural Networks)

- 上述过程的矩阵表示

$$\mathbf{H}' = \sigma((\mathbf{I} + \mathbf{D}^{-1}\mathbf{A})\mathbf{H}\mathbf{W})$$

$\mathbf{A} \in \{0,1\}^{n \times n}$：邻接矩阵

$\mathbf{D} \in R^{n \times n}$：对角矩阵，$\mathbf{D}_{ii}=d_i$，其他元素为零

$\mathbf{H} \in R^{n \times d}$：节点的初始特征表示

$\mathbf{W} \in R^{d \times d'}$：可学习参数矩阵

$\mathbf{H}' \in R^{n \times d'}$：更新后的节点特征表示

- 两层图网络

$$\mathbf{H}' = \sigma((\mathbf{I} + \mathbf{D}^{-1}\mathbf{A})\mathbf{H}\mathbf{W}_1)$$

$$\mathbf{H}'' = \sigma((\mathbf{I} + \mathbf{D}^{-1}\mathbf{A})\mathbf{H}'\mathbf{W}_2)$$

6.9.8 图神经网络 (Graph Neural Networks)

- 基于消息传递机制的图神经网络（一般形式）
  - 消息传递 $m_{j\to i} = f_m(h_i, h_j, h_{j\to i})$
  - 消息汇聚 $m_i = \mathrm{Aggr}(\{m_{j\to i}, j \in N(i)\})$
  - 节点表示更新 $h_i' = f_{node}(h_i, m_i)$
  - 边表示更新 $h_{j\to i}' = f_{edge}(h_i, m_i, h_j, m_j, h_{j\to i})$

h_i：节点 i 的特征表示
$h_{j\to i}$：边 e_{ij} 的特征表示
$f_m, \mathrm{Aggr}, f_{node}, f_{edge}$：被所有节点共享的函数
Aggr：与顺序无关(order invariant)，如求平均
f_m, f_{node}, f_{edge}：包含可学习参数，如MLP

6.9.8 图神经网络 (Graph Neural Networks)

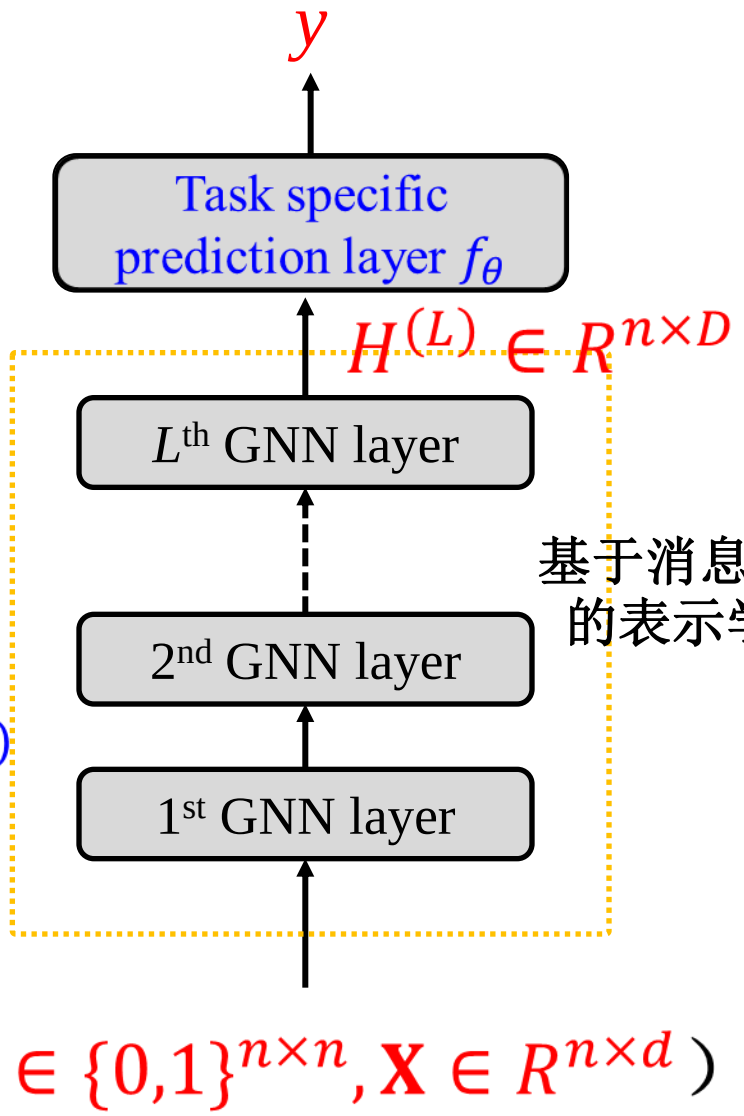
- 图神经网络对比卷积神经网络
  - 如果将节点看作像素，图上的消息传递操作和卷积操作非常相似：局部连接、权值共享
  - 因此，图神经网络经常又被称为图卷积网络(Graph Convolution Networks, GCN)

6.9.8 图神经网络 (Graph Neural Networks)

- 图神经网络的缺点
  - 过平滑(over-smoothing)
  - 过挤压(over-squashing)

6.9.8 图神经网络 (Graph Neural Networks)

- **GNN上的预测**
  - Node level
    - $y_i = f_\theta(h_i^{(L)})$
  - Edge level
    - $y_{ij} = f_\theta(h_i^{(L)}, h_j^{(L)})$
  - Graph level
    - $y_G = f_\theta(\text{pooling}(\{h_i^{(L)}, i = 1, \dots n\}))$



n: 节点数，d, D: 特征维度

致谢

- PPT由向世明老师提供

Thank All of You!

(Questions?)

张燕明

ymzhang@nlpr.ia.ac.cn

people.ucas.ac.cn/~ymzhang

模式分析与学习课题组(PAL)

中科院自动化研究所· 模式识别国家重点实验室