

第6章第3讲

人工神经网络

Artificial Neural Networks

向 世 明

smxiang@nlpr.ia.ac.cn

助教： 杨学行(xhyang@nlpr.ia.ac.cn); 吴一超(yichao.wu@nlpr.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、LSTM等

第九节 深度学习

6.9.1 深度学习浪潮！

Deep Learning Since 2006 (vol. 313, pp. 504-507)

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which

finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer “encoder” network

Neural networks are coming back!

6.9.1 深度学习浪潮！

- **深度神经网络是传统人工神经网络的扩展，是人工智能领域的热点研究方向**
 - **深度学习**概念由Hinton等人于2006年提出。
 - 随后，**面向深度信念网络的非监督贪心逐层训练算法**为深层神经网络的训练带来了希望。
 - LeCun等人提出的**卷积神经网络向深度化发展**。深度神经网络在语音识别、图像识别等诸多应用领域取得了空前的成功。
 - 为大数据领域的应用带来曝光—阿里巴巴、百度、谷歌、...

6.9.1 深度学习浪潮！

- Answer from G. Hinton, 2012.10



85%, 2012

74%, 2011

72%, 2010

6.9.1 深度学习浪潮！

- 时代背景-数据爆炸

图像数据



产品推荐



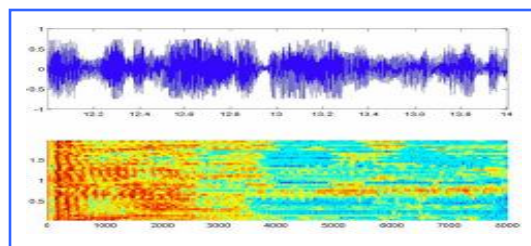
文本数据



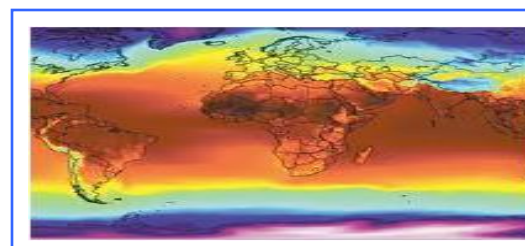
社交网络



语音数据



科学计算



6.9.1 深度学习浪潮！

- 时代背景-计算性能提升



6.9.1 深度学习浪潮！



Geoffrey Hinton



Yann LeCun



Yoshua Bengio



Stephane Mallat



Andrew Ng



Rob Fergus



Stephen Wright



Honglak Lee



Ruslan
Salakhutdinov



Gramham Taylor



Bruno A.
Olshausen



Marc'Aurelio
Ranzato

Most of them focus on Deep learning and Feature Learning

6.9.1 深度学习浪潮！

- **Academicals**

- Machine Learning @ UofT
 - Geoffrey E. Hinton, Rich Zemel, Ruslan Salakhutdinov, Brendan Frey, Radford Neal
- Université de Montréal: LISA Lab
 - Yoshua Bengio, Pascal Vincent, Aaron Courville, Roland Memisevic
- New York University: Yann Lecun's and Rob Fergus' group
- Stanford University: Andrew Ng's group
- UBC –Nando de Freitas's group
- UC Berkeley –Bruno Olshausen's group
- University of Washington –Pedro Domingos' group
- University of Michigan –Honglak Lee's group

6.9.1 深度学习浪潮！

- **Industrial**

- Google: Jeff Dean, Samy Bengio, Jason Weston, etc.
- MSR: Li Deng et al.
- Facebook: Yann Lecun et al.
- IBM Research – Brian Kingsbury et al.
- Kai Yu et al. (before 2015 @Baidu)
- Huawei: Xiaogang Wang et al.
- Alibaba: Xiansheng Hua
- Tencent:



- 深度学习

“Deep learning is a branch of machine learning based on a set of algorithms that attempt to **model high-level abstractions** in data by using **multiple processing layers with complex structures** or otherwise, **composed of multiple non-linear transformations.**” **(Nov. 2015)**

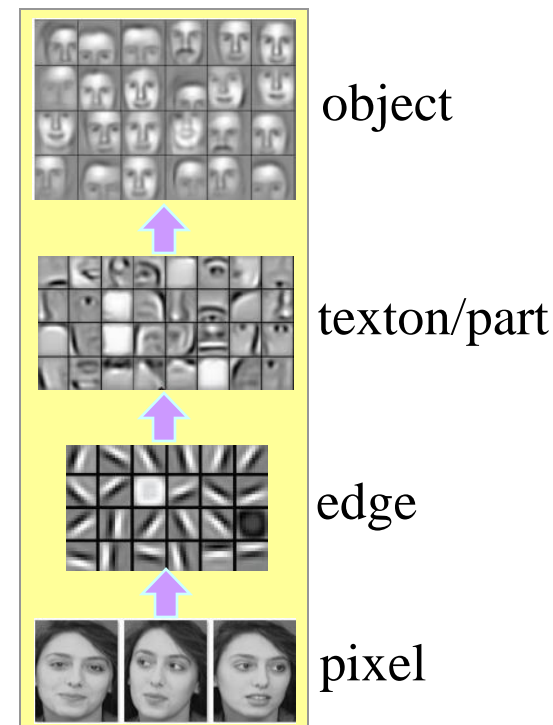
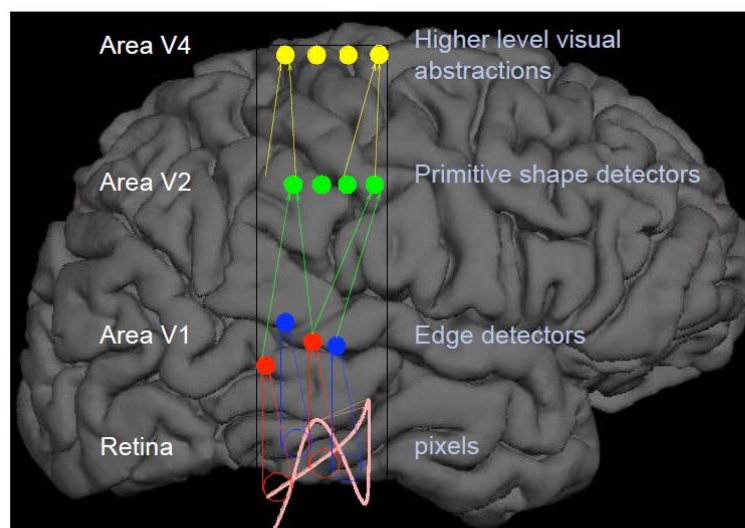
“Deep learning is a set of algorithms in machine learning that attempt to **model high-level abstractions in data** by using **model architectures composed of multiple non-linear transformations.**” **(Aug. 2014)**

“Deep learning is a set of algorithms in machine learning that attempt to **learn in multiple levels, corresponding to different levels of abstraction.** It typically uses artificial neural networks. The levels in these learned statistical models correspond to distinct levels of concepts, where higher-level concepts are defined from lower-level ones, and the same lower-level concepts can help to define many higher-level concepts.” **(Oct. 2013)**



6.9.1 深度学习浪潮！

- **Why** Hint from human learning
Deep Architecture in the Brain



Yoshua Bengio:

Learning Deep architectures for AI, Foundations and Trends in Machine Learning, 2009

6.9.1 深度学习浪潮！

- **Why**

- 浅层神经网络可以近似任意函数，为何多层？

- 深层网络结构中，**高层可以综合应用低层信息**
 - 低层关注“局部”，高层关注“全局”、更具有语义化信息
 - 为**自适应地学习非线性处理过程**提供了一种可能的简洁、普适的结构模型
 - **参数学习与分类器的训练可以一起学习**

6.9.1 深度学习浪潮！

- 发展历程

- Hopfield network
- Boltzman machine
- Restricted Boltzman machine
- CNN
- RNN
- LSTM
- Autoencoder
- DBN
- DBM
- Deep Learning

} before 2000

} Deep learning

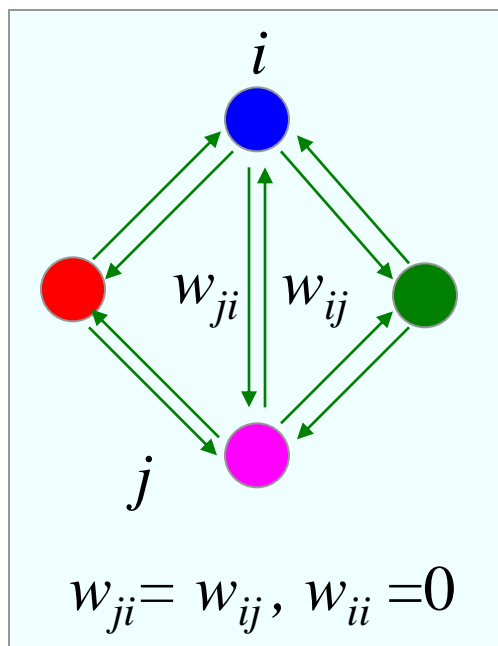
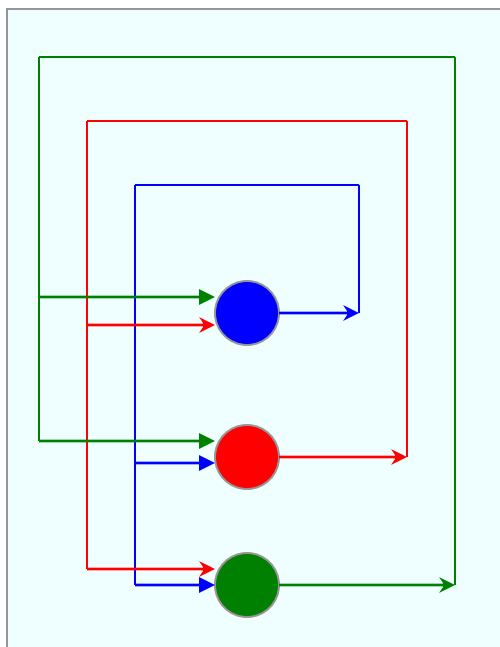
6.9.2 深度网络

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

Hopfield network

- 结构

- 单层全互连、对称权值的反馈网络
- 状态: $-1(0)$, $+1$



常见的两种形式

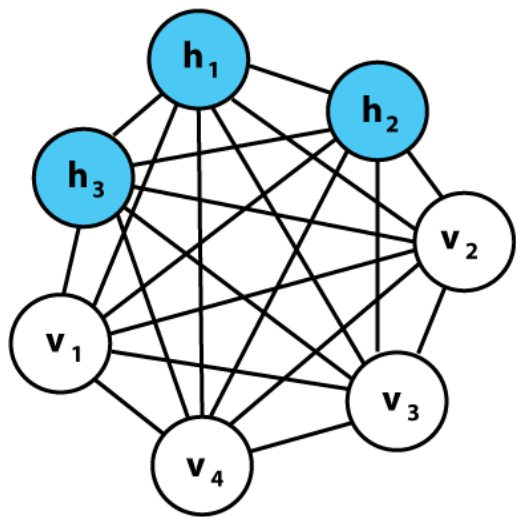
Hopfield网络按动力学方式运行，其工作过程为状态的演化过程，即从初始状态按能量减小的方向进行演化，直到达到稳定状态。稳定状态即为网络的输出

网络演化特点

Boltzman 机

- 结构

- 结构类似于Hopfield 网络，但它是具有隐单元的反馈互联网络



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

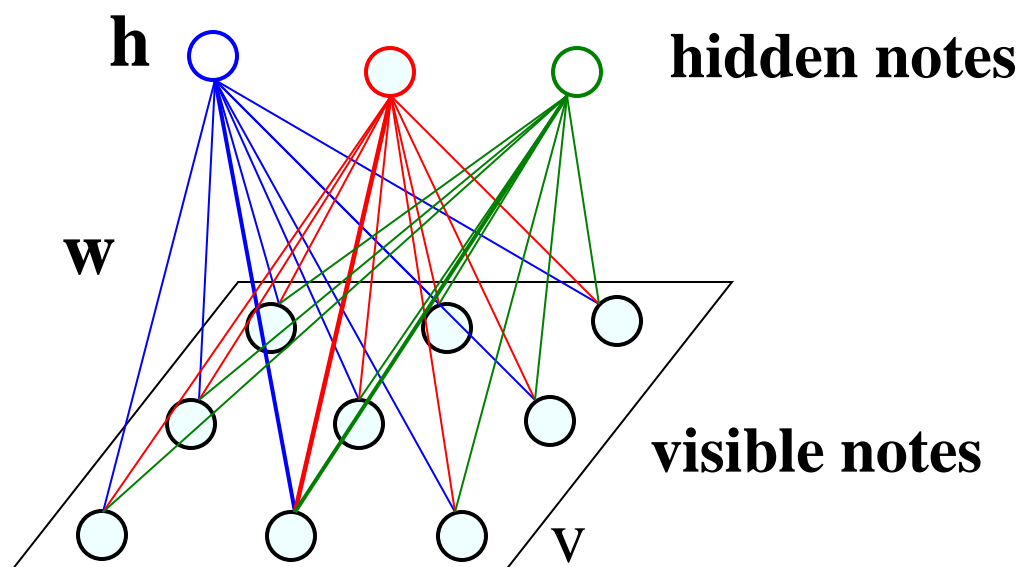
1. Hopfield网络的神经元的结构功能及其在网络中的地位是一样的。但BM中一部分神经元与外部相连，可以起到网络的输入、输出功能，或者严格地说可以受到外部条件的约束。另一部分神经元则不与外部相连，因而属于隐单元
2. 神经元的状态为0或1的概率取决于相应的输入。

网络结构复杂、训练代价大、局部极小

RBM

- 结构

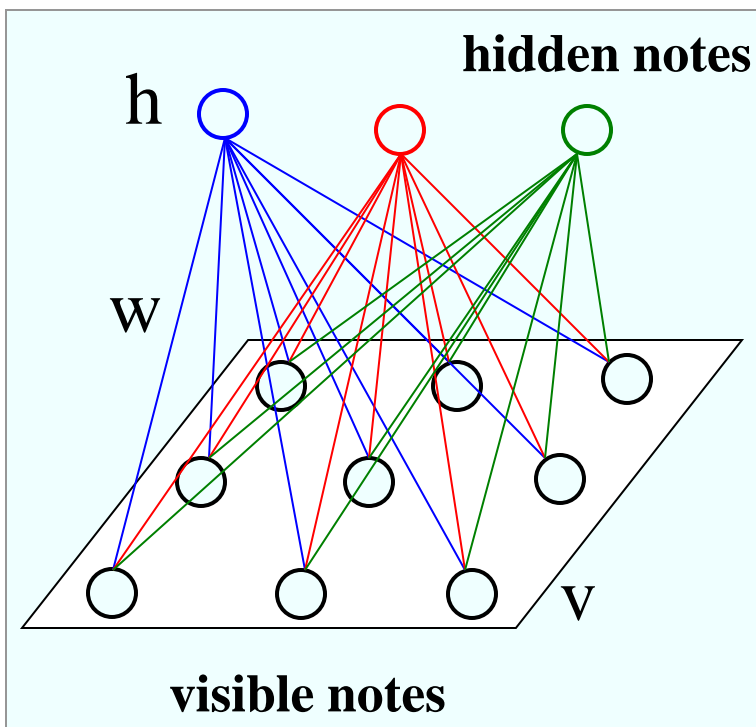
- In RBM, the hidden units are conditionally independent given the visible states (条件独立)



(信息是双向可流动的)

RBM

结构 (Bipartite Structure)



Classical structure

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

The energy of the **joint configuration**:

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

$\theta = \{w, a, b\}$ —model parameters

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

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

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

(即：整个模型是用联合概率分布来描述的)



RBM

- 描述

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

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

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

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

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

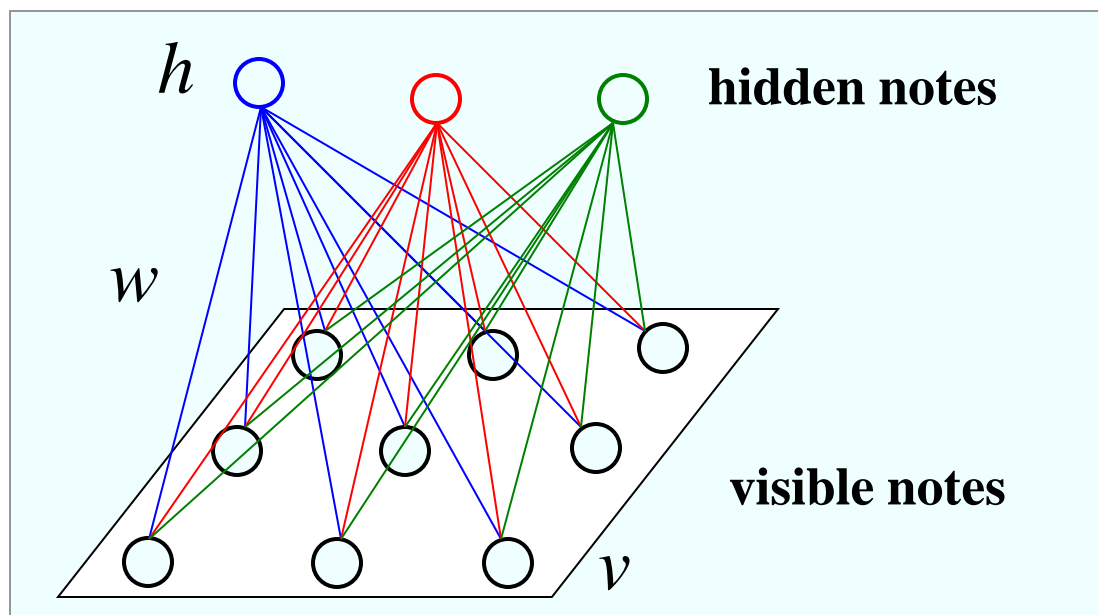
$$p(\mathbf{h} | \mathbf{v}) = \frac{\exp^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{h}} \exp^{-E(\mathbf{v}, \mathbf{h})}}$$

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

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

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

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



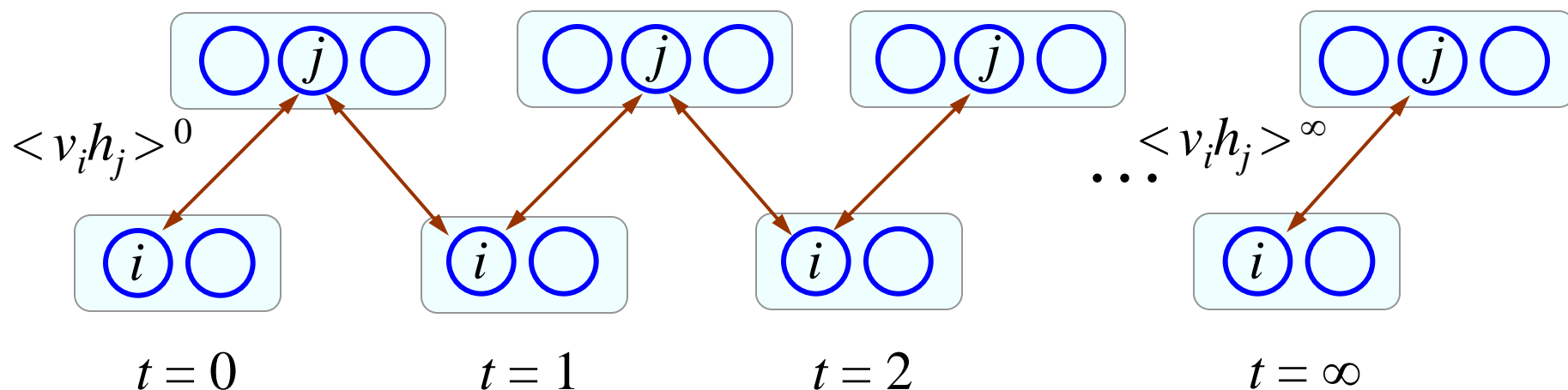
借助该模型

生成式模型
generative model

RBM

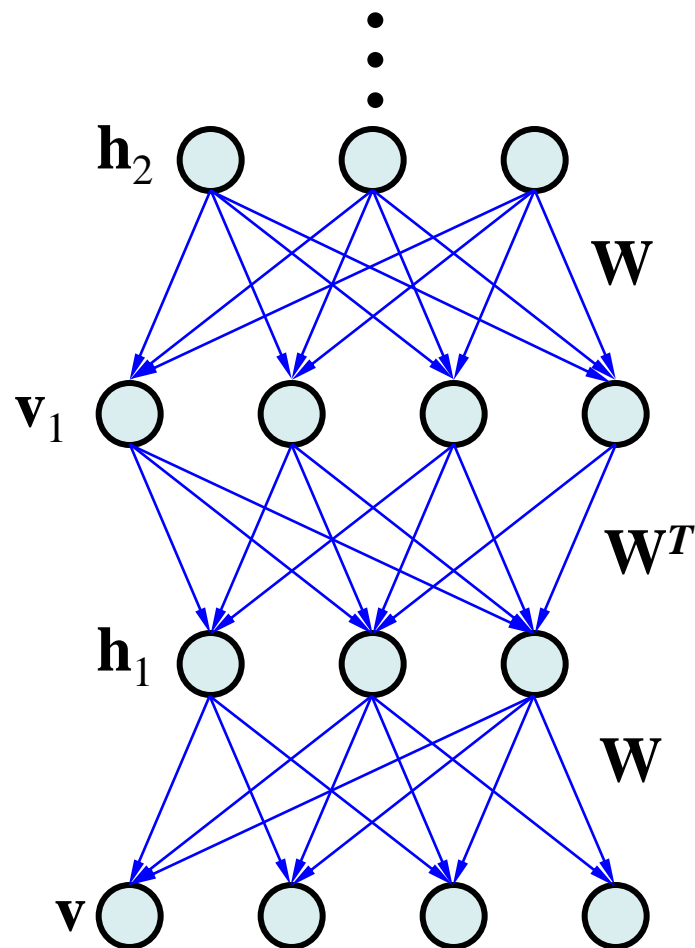
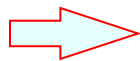
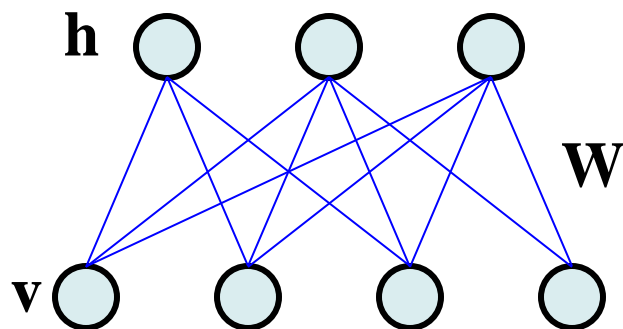
- **网络训练**

- 总体思路——采样梯度下降算法
- 主要的难度在于估计 $\text{negative}/\text{model}$ 。理论上需要全空间的数据 $\{(\mathbf{v}, \mathbf{h})\}$ 来估计，因此需要无穷多次生成 \mathbf{v} 和 \mathbf{h} ：



RBM

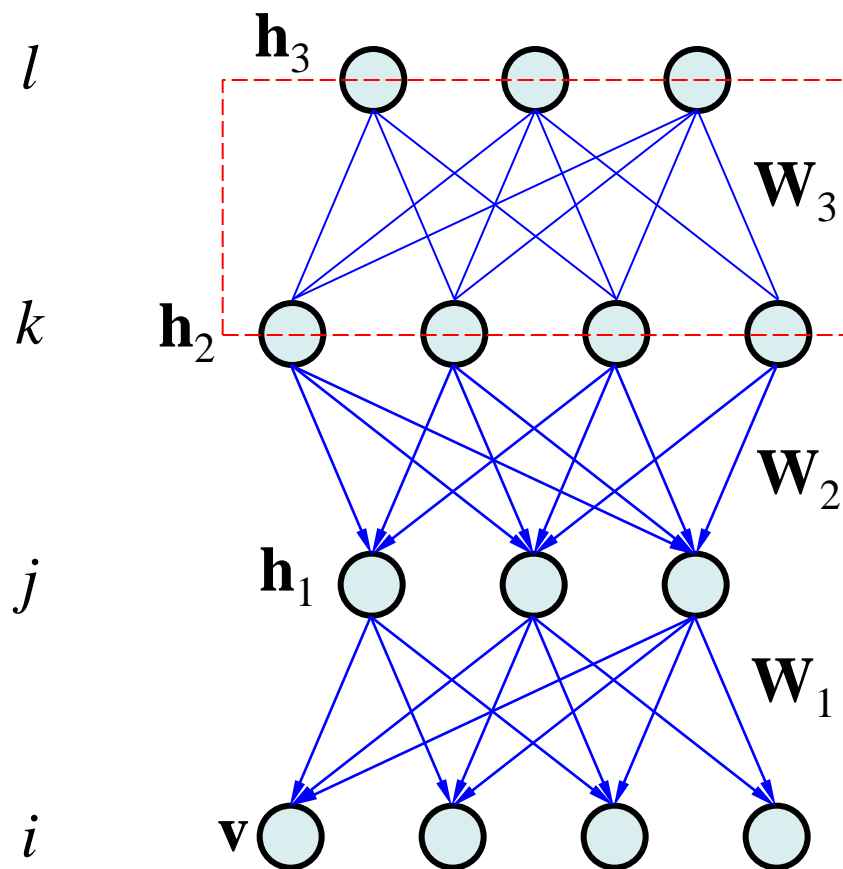
- 对网络结构的等价解释



无穷层有向网络（但层间权值相同）

深度信念网络(DBN)

具有联想存储功能



联合概率分布

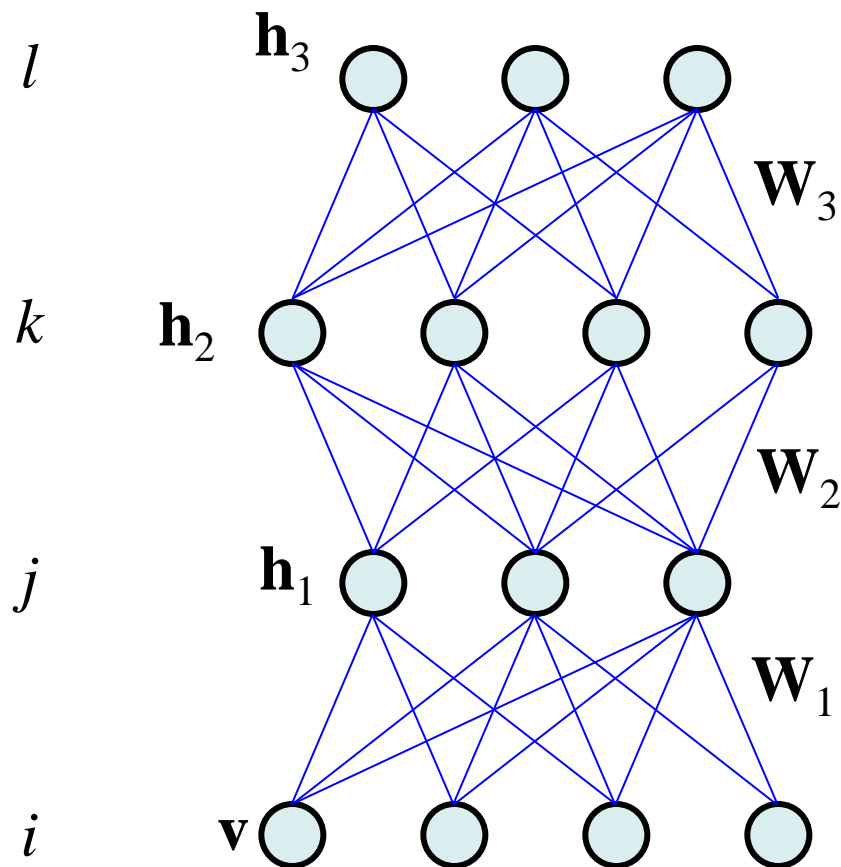
$$\begin{aligned} p(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{v} \mid \boldsymbol{\theta}) \\ = \prod_i \text{Ber}(v_i \mid \text{sigm}(\mathbf{h}_1^T \mathbf{w}_{1i})) \cdot \\ \prod_j \text{Ber}(h_{1j} \mid \text{sigm}(\mathbf{h}_2^T \mathbf{w}_{2j})) \cdot \\ \frac{1}{z(\boldsymbol{\theta})}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

参考文献:

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

最高一层为无向连接（比如RBM）

深度Boltzman机 (DBM)



联合概率分布

$$\begin{aligned} p(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{v} | \boldsymbol{\theta}) \\ = \frac{1}{z(\boldsymbol{\theta})} \cdot \exp \left(\sum_{ij} v_i h_{1j} \mathbf{w}_{1ij} \right) \cdot \\ \exp \left(\sum_{jk} h_{1j} h_{2k} \mathbf{w}_{2jk} \right) \cdot \\ \exp \left(\sum_{kl} h_{2k} h_{3l} \mathbf{w}_{3kl} \right) \end{aligned}$$

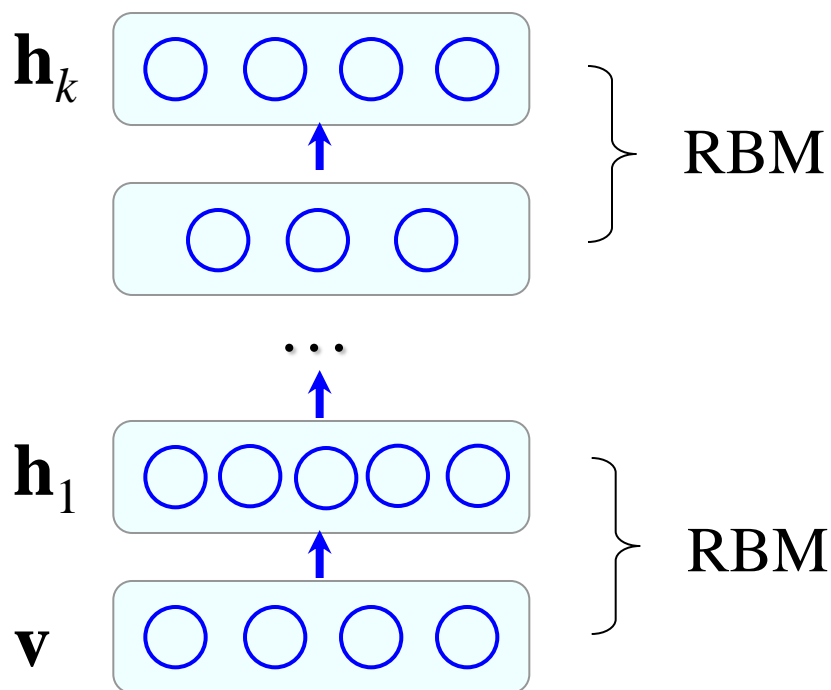
层与层均为无向连接 (比如RBM)

参考文献:

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

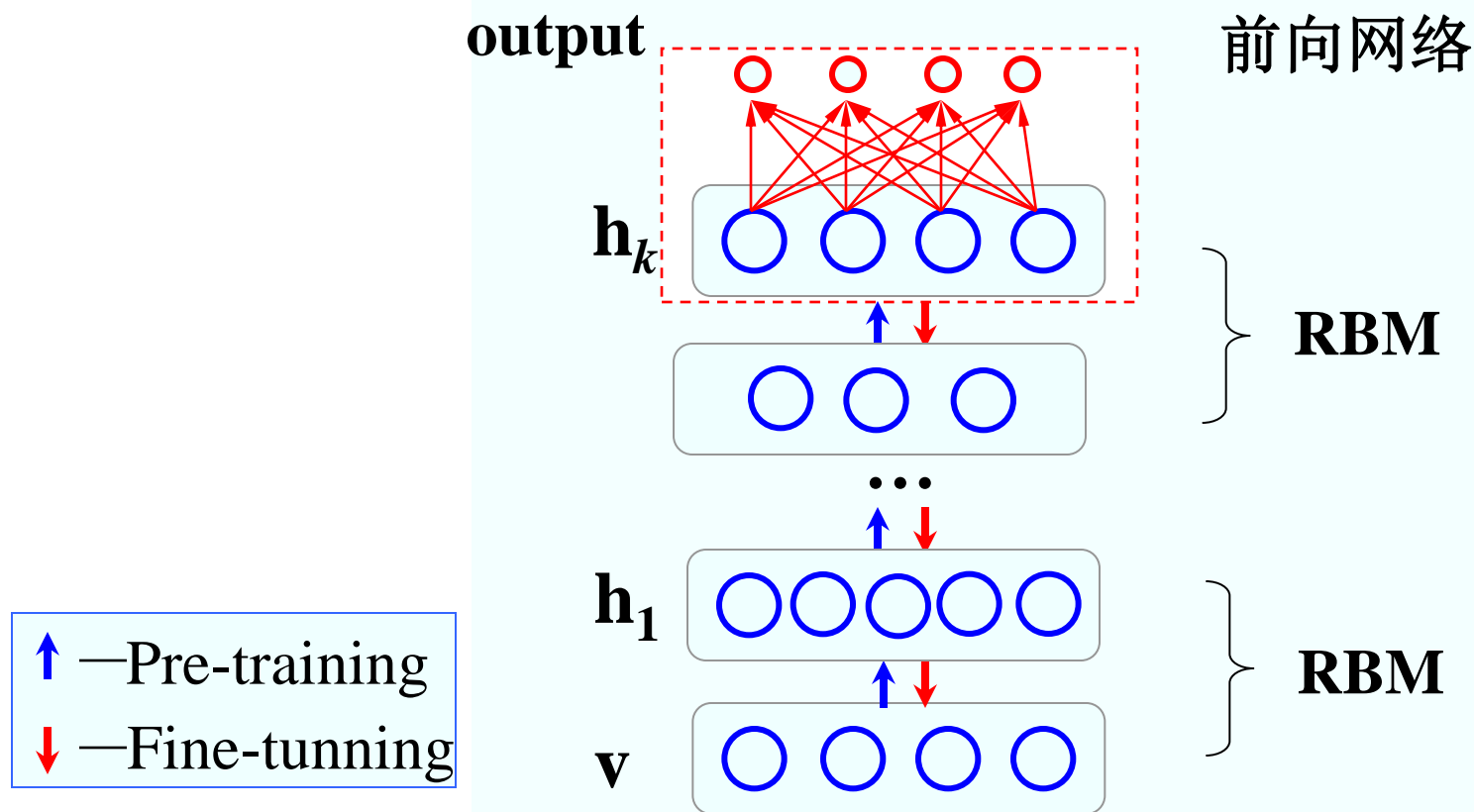
DBN训练

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



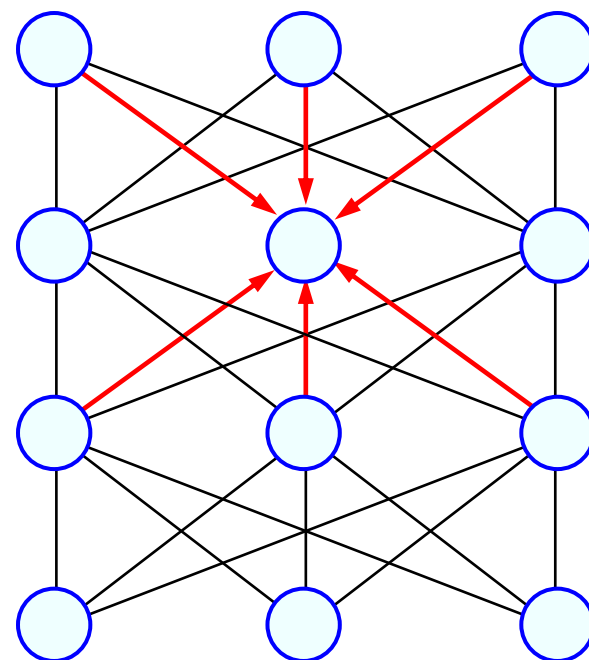
DBN训练

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



DBM训练

- 与DBN一样，但在训练时采用双方向（上下两层）
- 在训练单层时需同时考虑两个或者多个隐含层（整个网络相当于一个MRF）
 - 因为：能量模型与DBN不一样；
 - 训练更困难，更耗时；
 - 基本不用了。



考虑团(clique)

6.9.3 卷积神经网络

- 卷积神经网络

- 卷积神经网络已成为当前语音分析和图像识别领域的研究热点。
- 卷积神经网络的显著特点是**权值共享**。因此，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。
- **卷积神经网络本质上讲是一个多层感知器**，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。

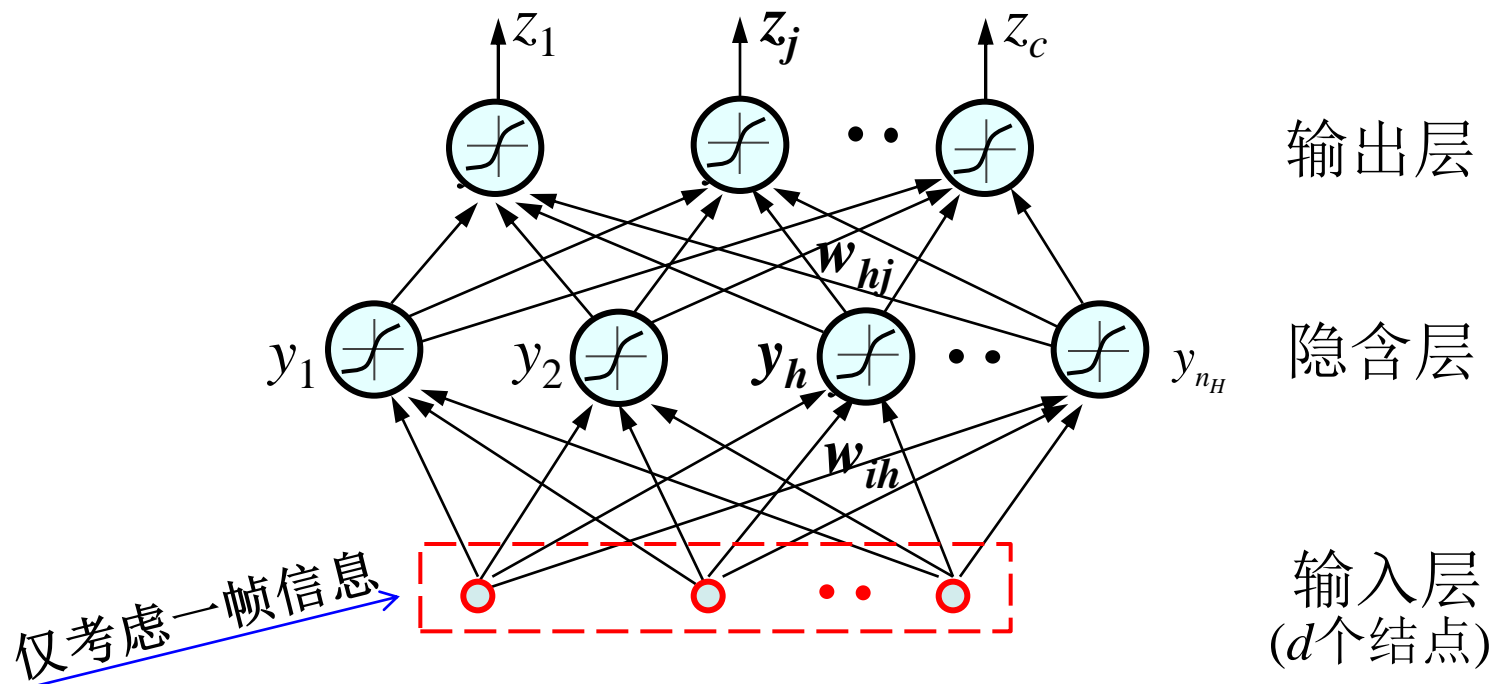
6.9.3 卷积神经网络

- 卷积神经网络
 - 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），对物体有位移或轻微变形时，仍然能完成识别。
 - 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

- 时延神经网络(Time-Delay Neural Network)

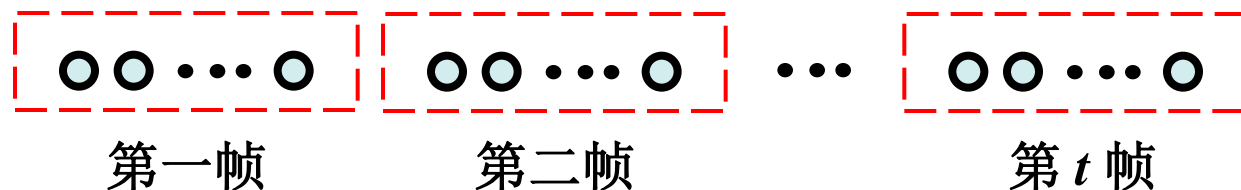
- 考察某个事物的时间序列信号，比如语音信号。假定每一时刻获得一个描述该信号的 d 维特征，记为帧信息。采用原始的多层感知器网络，则有如下结构：



6.9.3 卷积神经网络

- 时延神经网络

- 但是，数据是按序列接收到的。如果采用单帧建模方法，帧与帧之间的时间相关性则丢失了。但它是理解序列内容的一种重要信息。

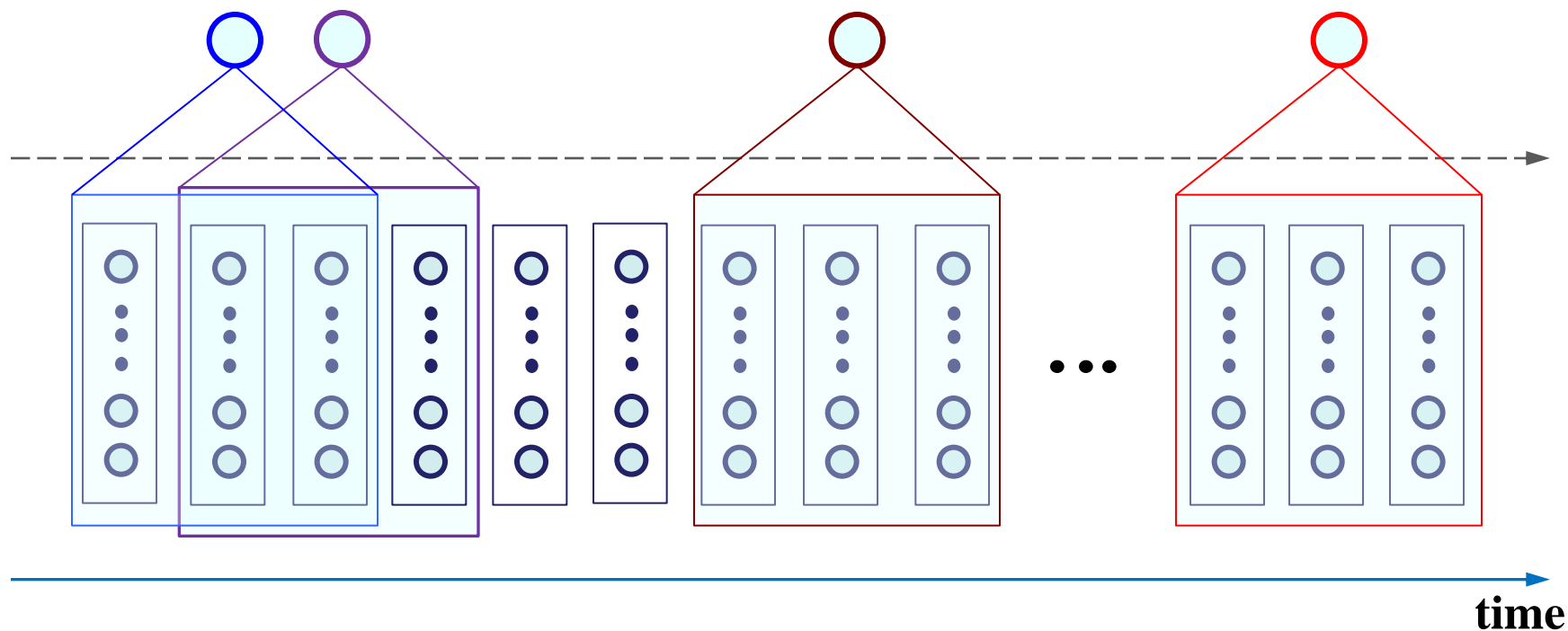


- 一种重要的方法是同时考虑多帧，即所谓的时间延迟
- 对于神经网络结构而言，对连续多帧信息构成一个组，组内结点向隐含层映射时，其连接权重不会因时间顺序的不同而不同，从而达到权重共享的目的。

6.9.3 卷积神经网络

- 时延神经网络

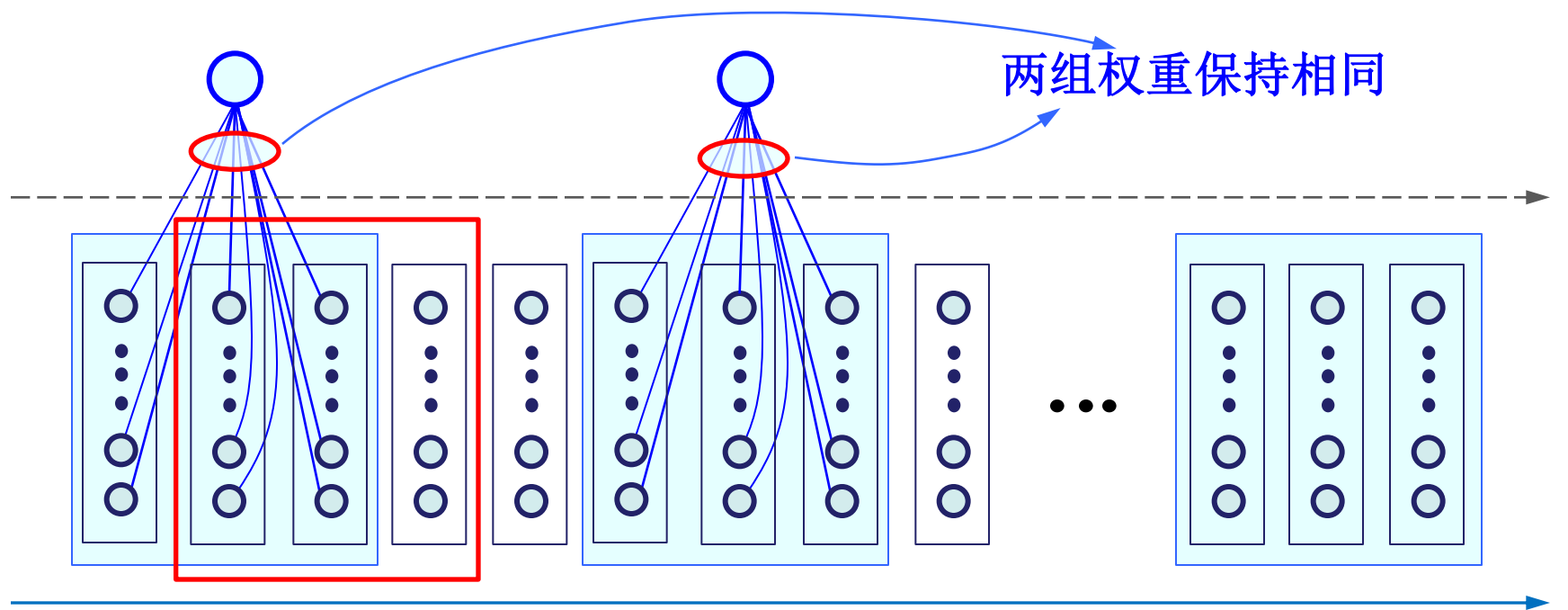
- 比如，将三帧信息组合起来，同时映射到下一层结点。为了方便起见，将一帧信息垂直排列，见下图：



6.9.3 卷积神经网络

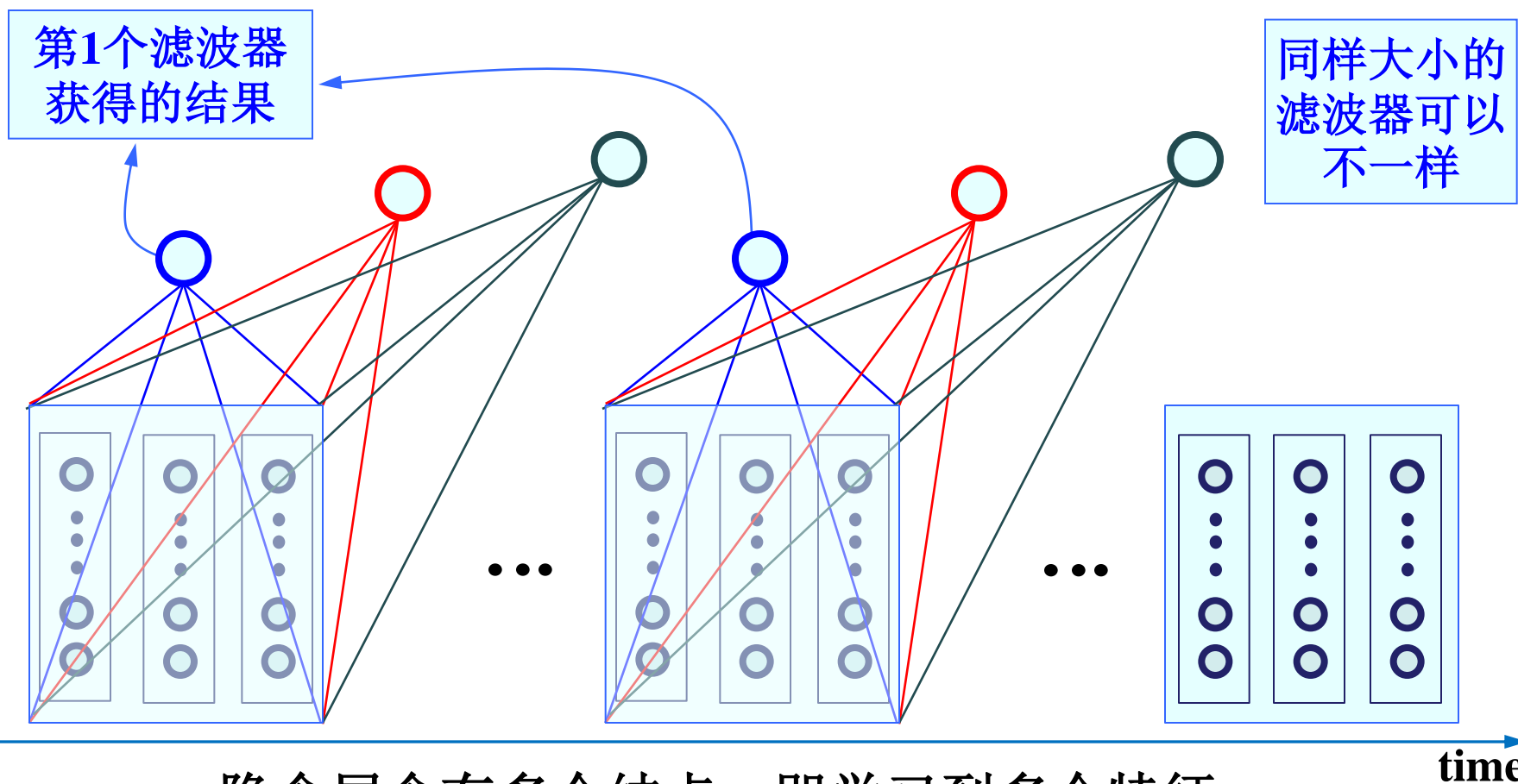
- 时延神经网络

- 比如，将三帧信息组合起来，同时映射到下一层结点。为了方便起见，将一帧信息垂直排列，见下图：



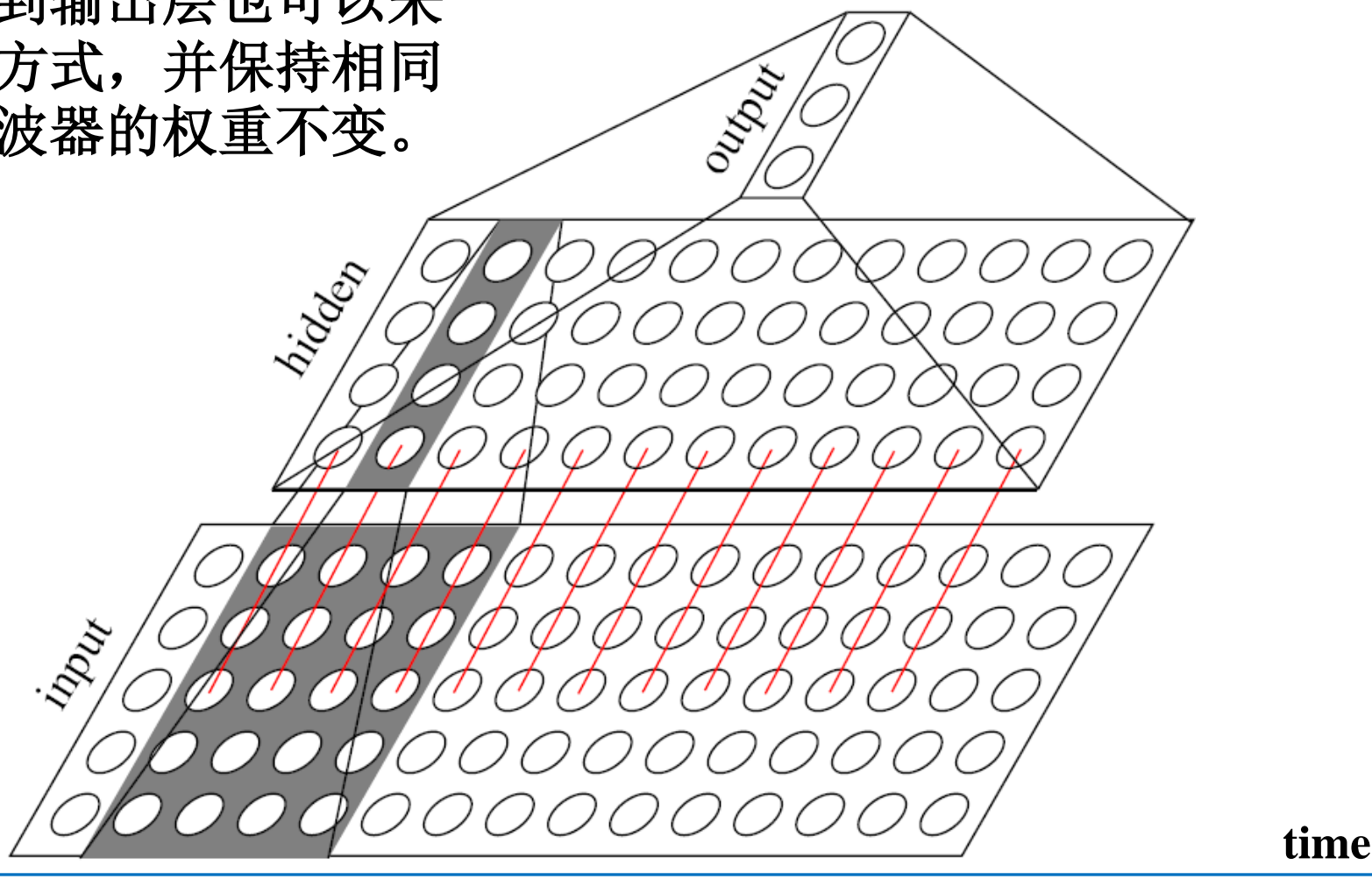
隐含层只含有一个结点的情形—即学习到一个特征 time

- 每组权重称为一个滤波器，包含 sd 个待求权重， s 为延时长度。
- 隐含层结点的个数对应于滤波器的个数
- 待学习滤波器在整个指定的时长范围内保持不变



隐含层含有多个结点—即学习到多个特征

隐含层到输出层也可以采用时延的方式，并保持相同位置的滤波器的权重不变。



信号特征为5维，时延4帧，滤波器个数为4，输出结点个数为3
(输入层至隐层待学权重数： $5 \times 4 \times 4$)

6.9.3 卷积神经网络

- 对于语音等信号，采用时间延迟卷积神经网络可以很好地对信号进行描述学习。
- 对于自然图像，希望直接从图像底层开始进行学习（非结构化特征学习）。即，对于图像识别任务，不必事先提取出人为设计的特征，比如Gabor纹理特征、多尺度小波特征、SIFT特征、HOG特征，等等。
 - 人为设计的特征的缺点：这些特征具有一些参数，如尺度、梯度方向、频域划分等，其泛化能力不强
- 但是，如果以图像直接作为输入，并将每个像素看成一个结点，对于 200×200 大小的图像，则仅输入层就有4万个结点；如果第一隐含层仅仅只包含1000个结点，则权重数量将达到4千万。这显然是一个巨大的计算负担。

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**

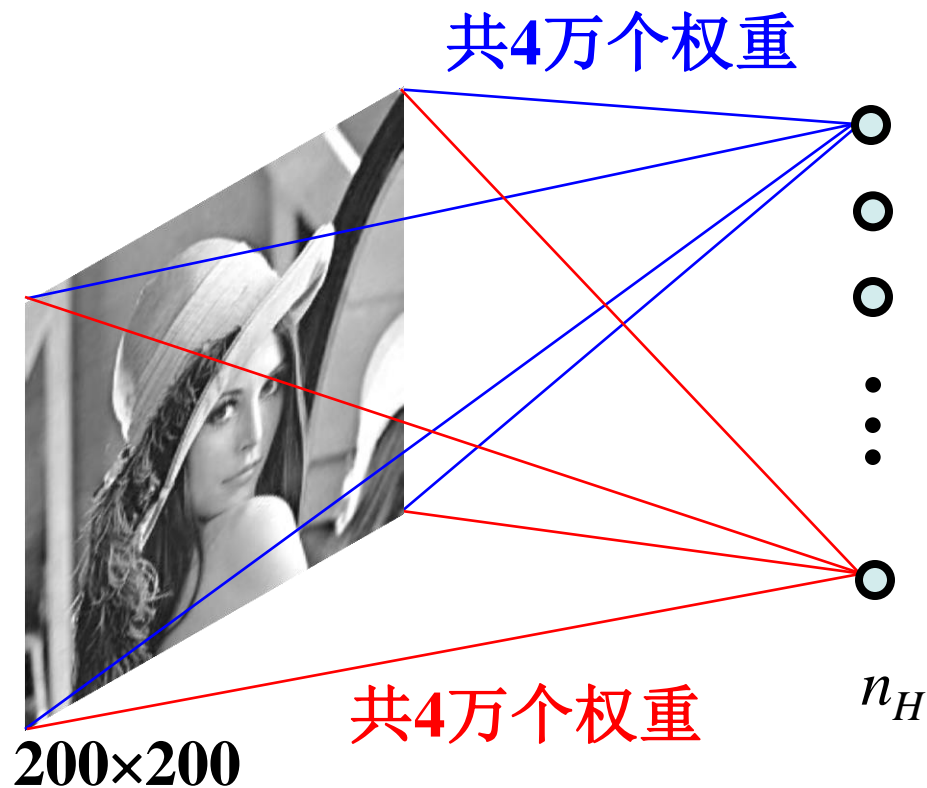
- 每个神经元其实没有必要对全局图像进行感知，只需要对局部进行感知。
 - 在更高层将局部的信息综合起来获得全局信息。

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”。
 - 采用滤波器（卷积核）的术语，则更能够清晰地描述这一机制。**滤波器是由一组待学习的权重所组成**，采用该滤波器对图像进行卷积滤波（局部区域内线性加权求和），并将结果输出至隐含层对应的结点。
 - **每个滤波器可以看成是学习一种特征**。采用多个滤波器，则可以理解为学习多个特征。

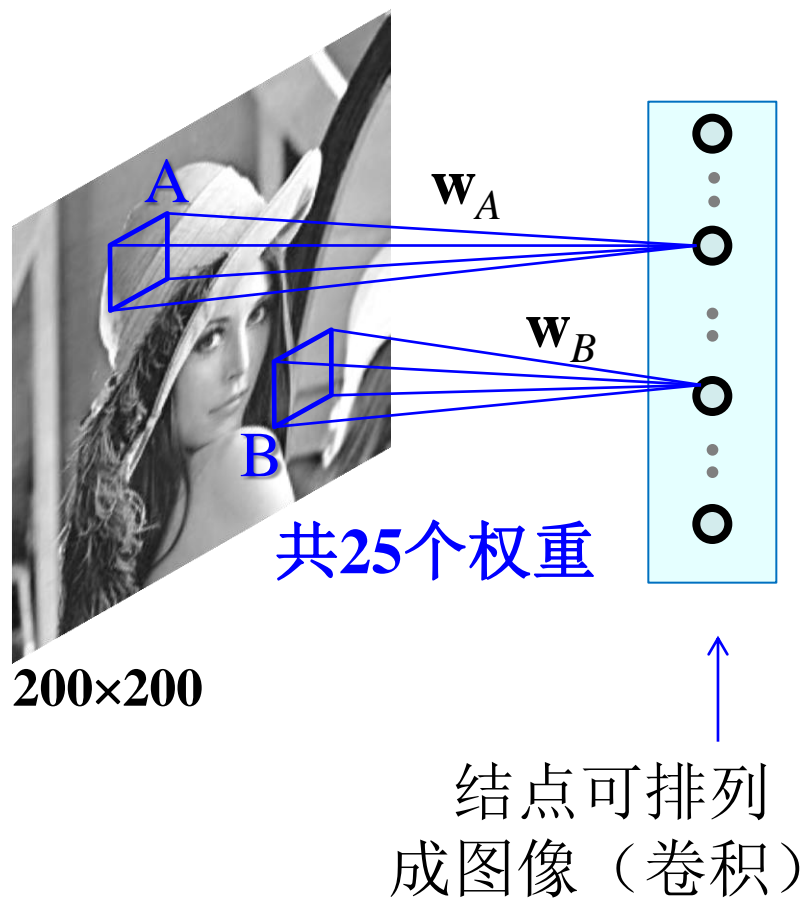
6.9.3 卷积神经网络

- 全连接



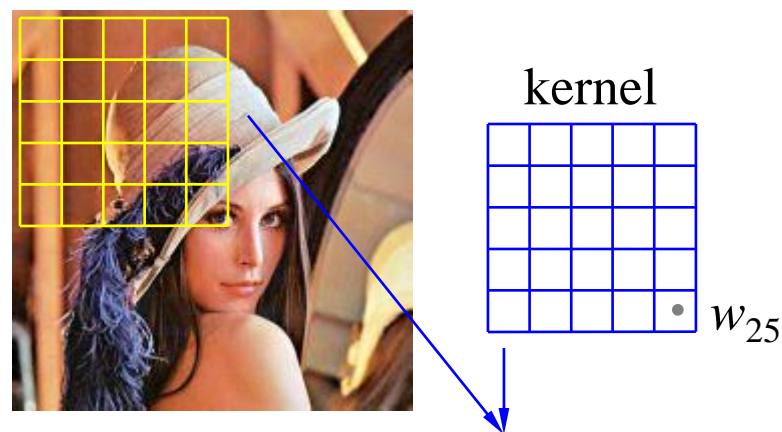
(如果 $n_H=40000$)

- 局部连接
 - 考虑5×5大小的窗口



回忆图像滤波操作:

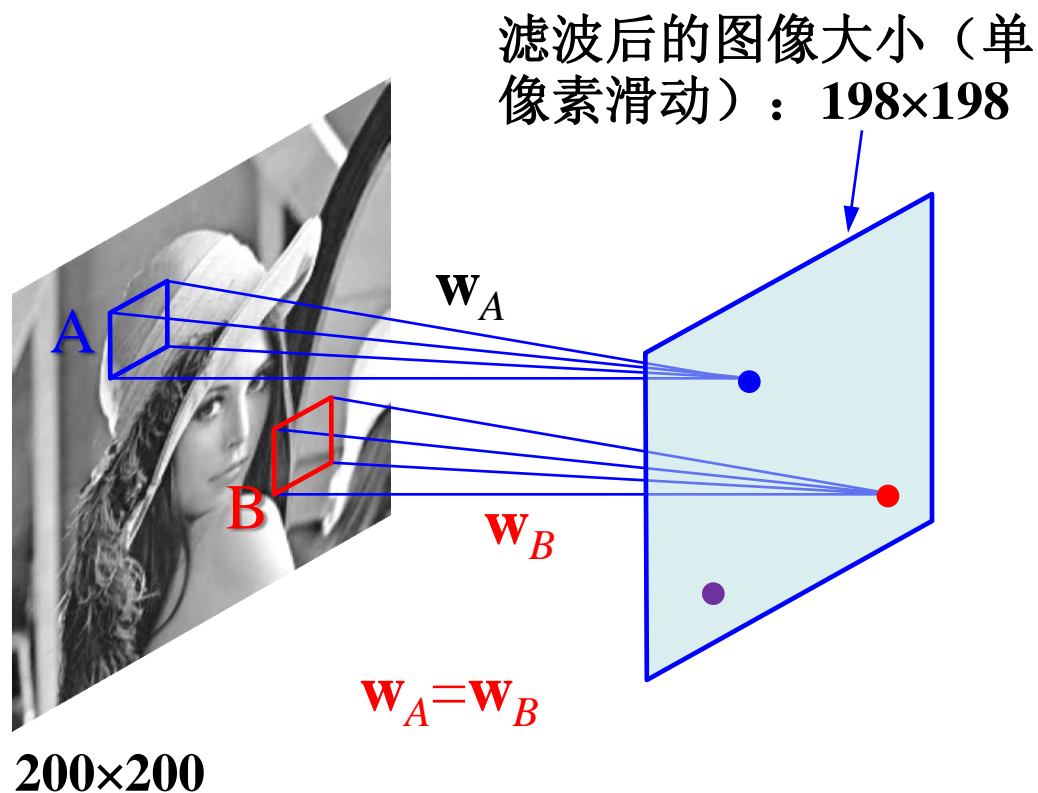
Image * filter



$$y = p_1 w_1 + p_2 w_2 + \dots + p_{25} w_{25}$$

(Just 加权求和!)

- 局部连接
 - 考虑 5×5 大小的滤波器且权值共享



若采用全连接，
权重数为：
 $200\times 200 \times 198\times 198$



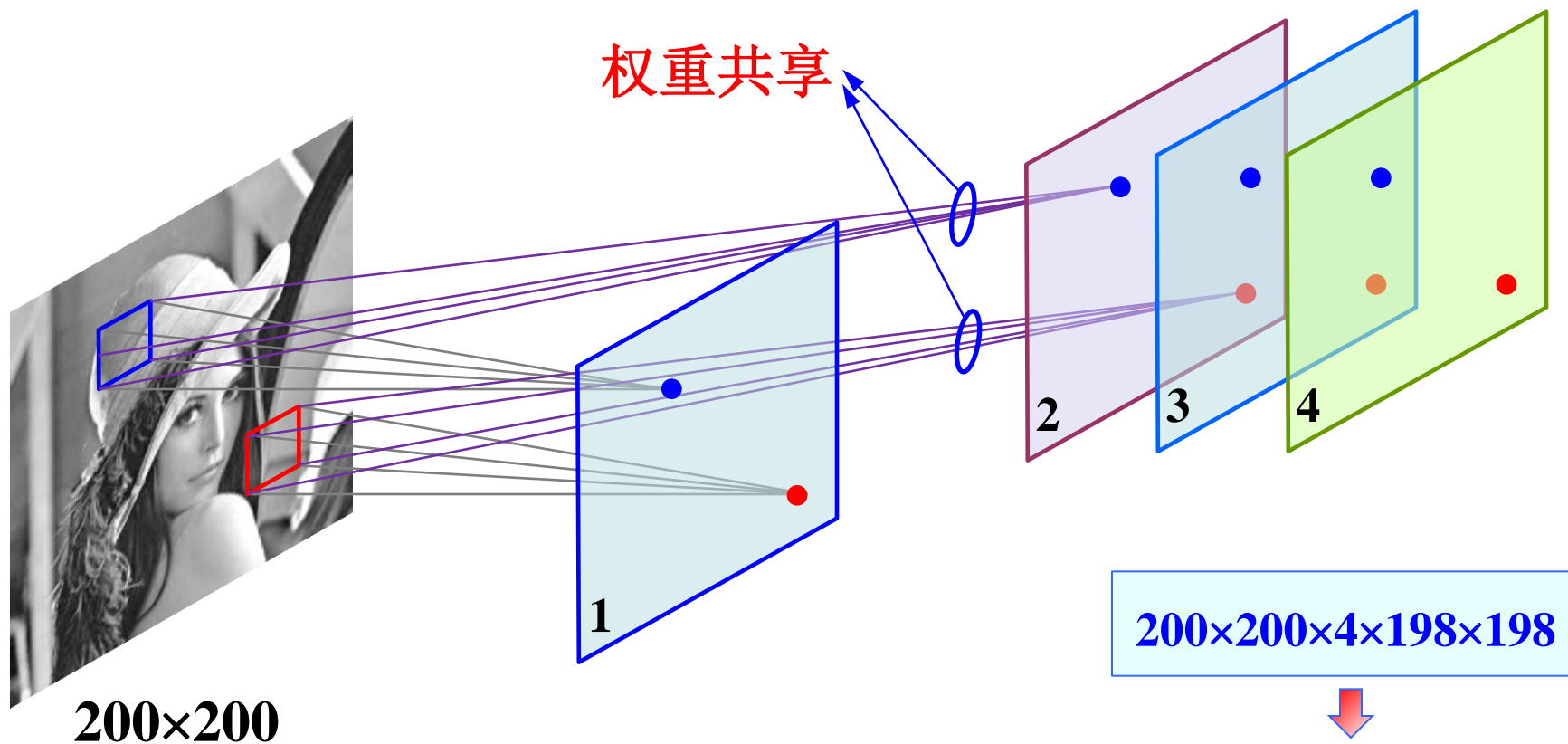
若采用局部连接，
权重数为：
 $25 \times 198\times 198$



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

- 可以有多个滤波器：比如，4个



从输入层至第一隐层权重仅有： 25×4 个！

6.9.3 卷积神经网络

- 第一个隐含层

- 考虑学习4个滤波器的情形

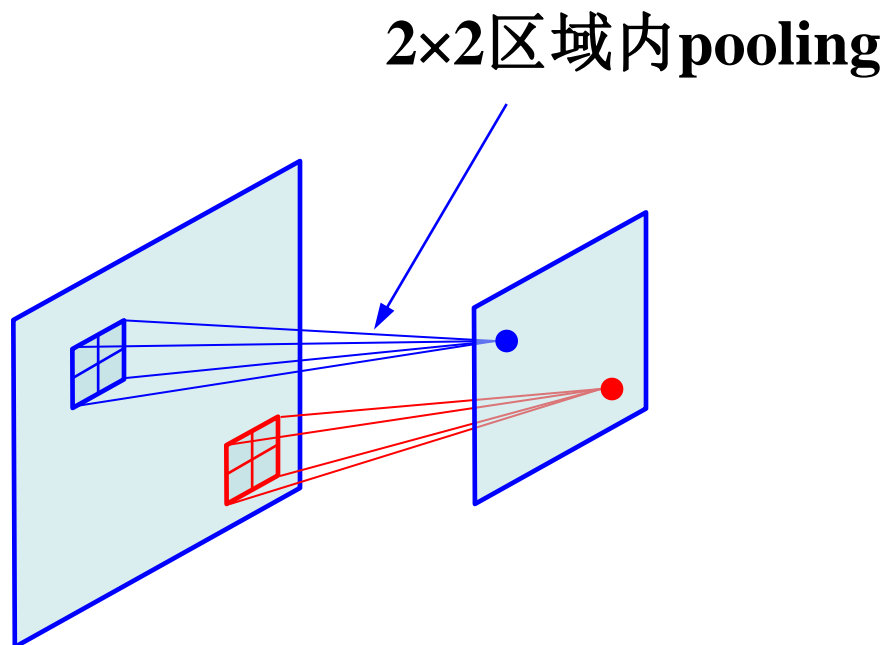
- 4个滤波器通过卷积滤波（就是局部加权求和）获得的结果，每个结果图像的大小为 198×198
 - 每个滤波器对应一种特征提取。
 - 如果采用这些特征设计分类器，原始数据空间的维数将达到 $4 \times 198 \times 198$ 。实际应用中滤波器更多，可能导致一个高维问题。采用高维数据设计分类器，容易产生过拟合（过学习）情形。
 - 一个自然的想法就是对不同位置的特征进行聚合统计，比如，计算图像一个区域上的某个特定特征的平均值(或最大值)

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—pooling
 - 两种方式: 区域内取最大值或平均



200×200



198×198

99×99

6.9.3 卷积神经网络

- 目前完成的步骤

- 对图像，采用多个滤波器（即待学习权重）进行滤波，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)

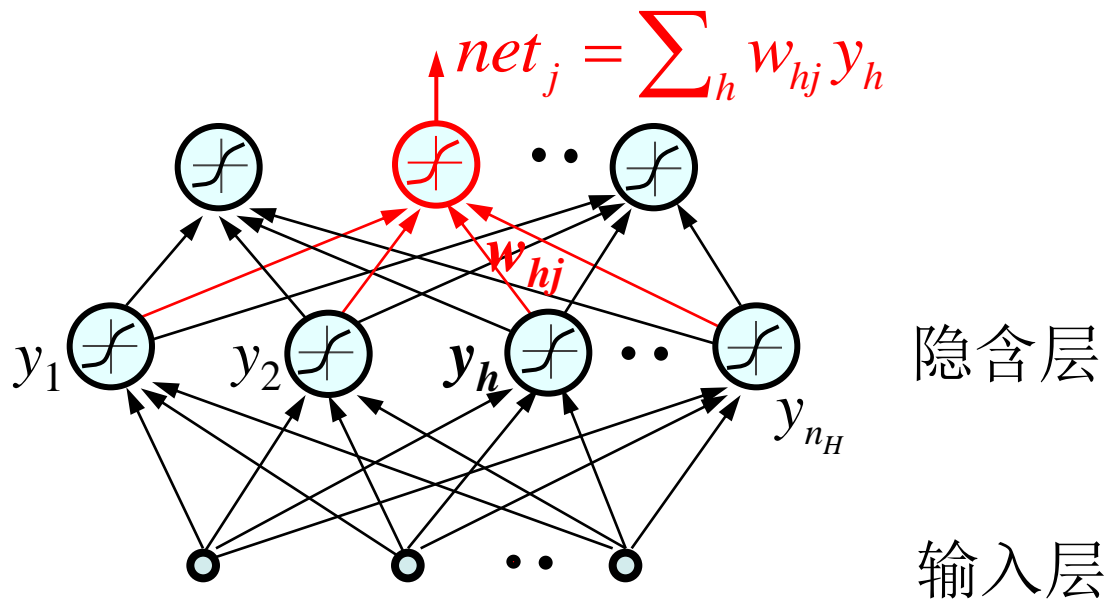
- 对 200×200 图像、4个 5×5 滤波器、 2×2 pooling:

- 第一步：得到4个 198×198 大小的滤波结果
- 第二步：得到4个 198×198 大小的非线性激励结果
- 第三步：得到4个 99×99 大小的pooling结果

6.9.3 卷积神经网络

- 第二个隐含层的设计

- 回忆前向神经网络第一个层到第二层的操作：将所有结点的输出**收集**起来，加进加权求和。

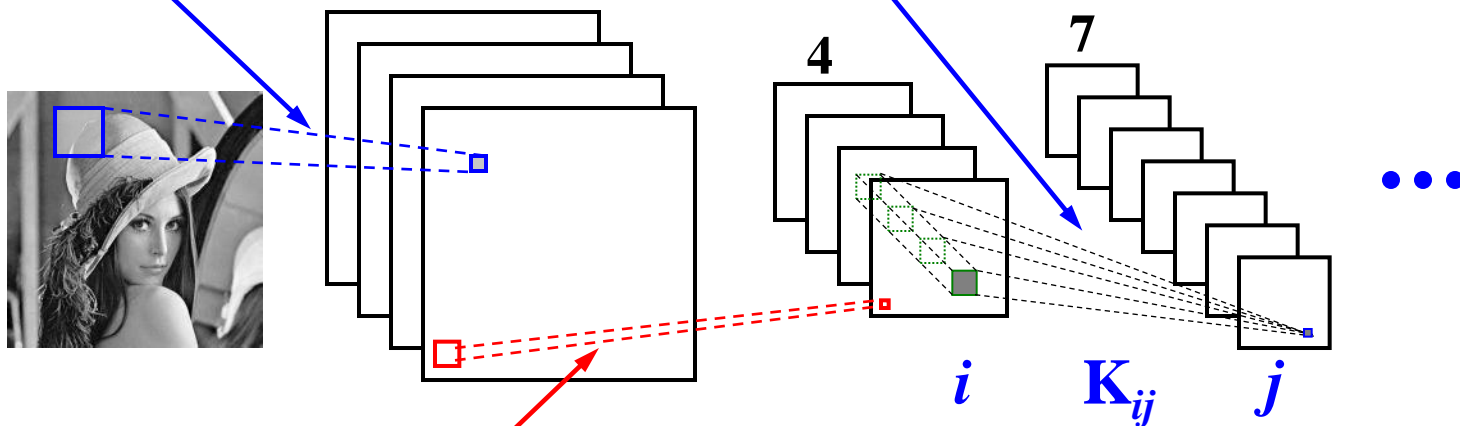


在CNN中，将“全连接”变成“局部连接+权值共享”

- 第二个隐含层的设计：假定第二个输出7个图像，采用 3×3 大小的滤波器

4个滤波器，均作用于输入图像

28个滤波器， K_{ij} 作用于上一层第*i*个图像，结果累加至下一层第*j*个图像

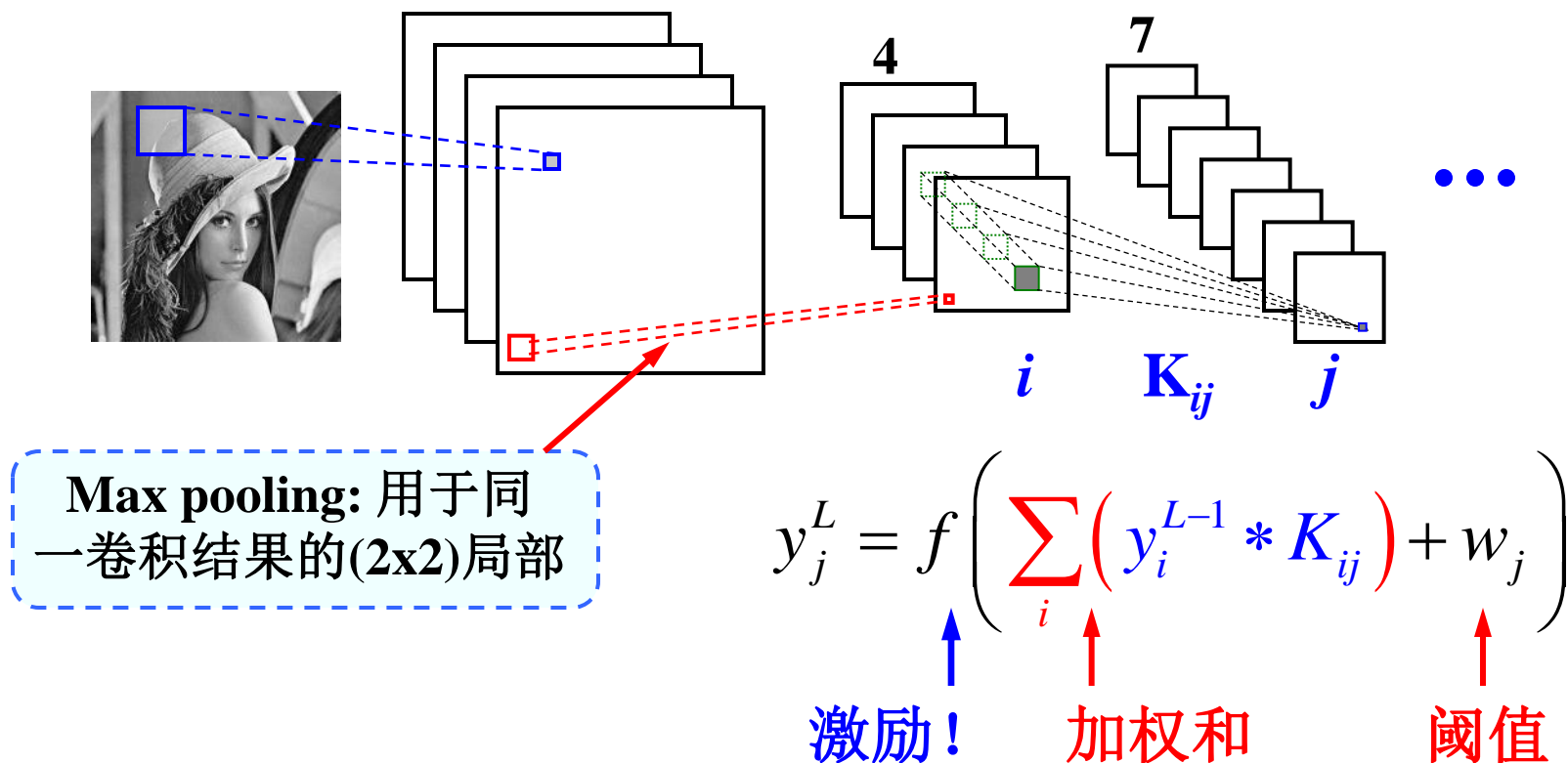


Max pooling: 用于同一卷积结果的(2x2)局部

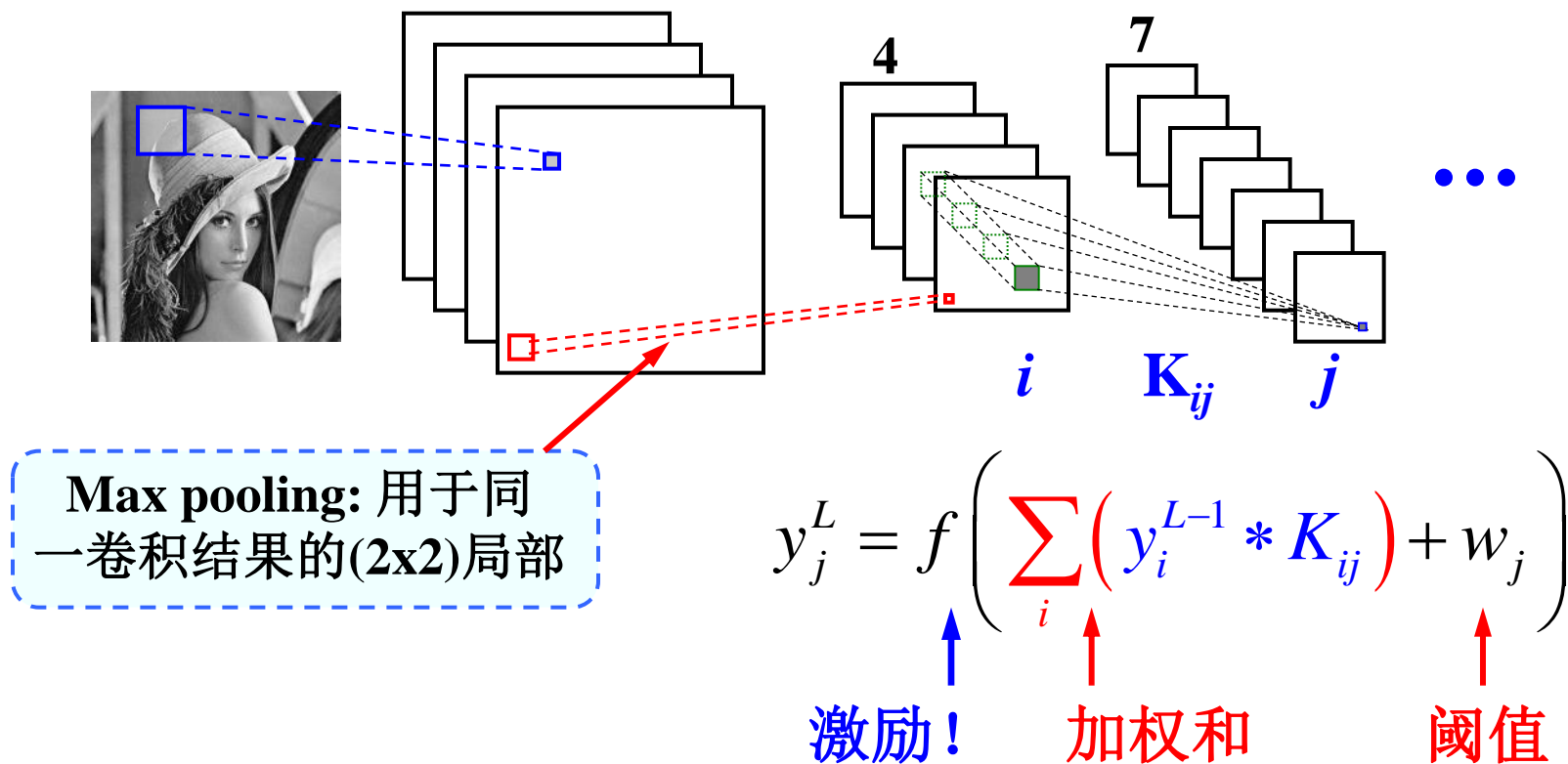
$$y_j^L = f \left(\sum_i \left(y_i^{L-1} * K_{ij} \right) + w_j \right)$$

↑ 激励! ↑ 加权和 ↑ 阈值

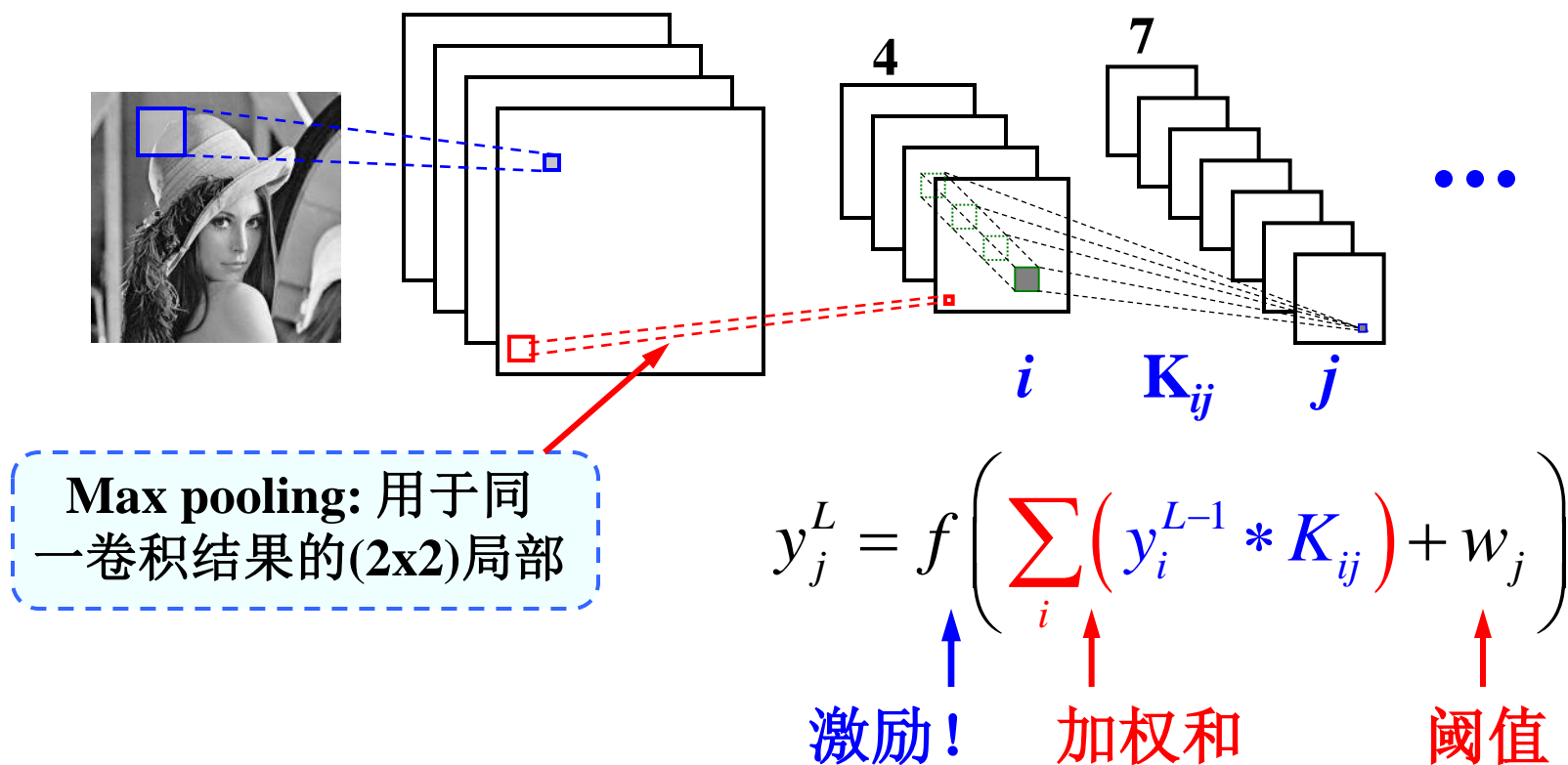
- 反过来考虑输入至第一隐层：由于只有一个图像，则不需要累加。但第一隐含层包含4个图像（每个图像可以视为一个大的结点集组），因此需要4个滤波器。



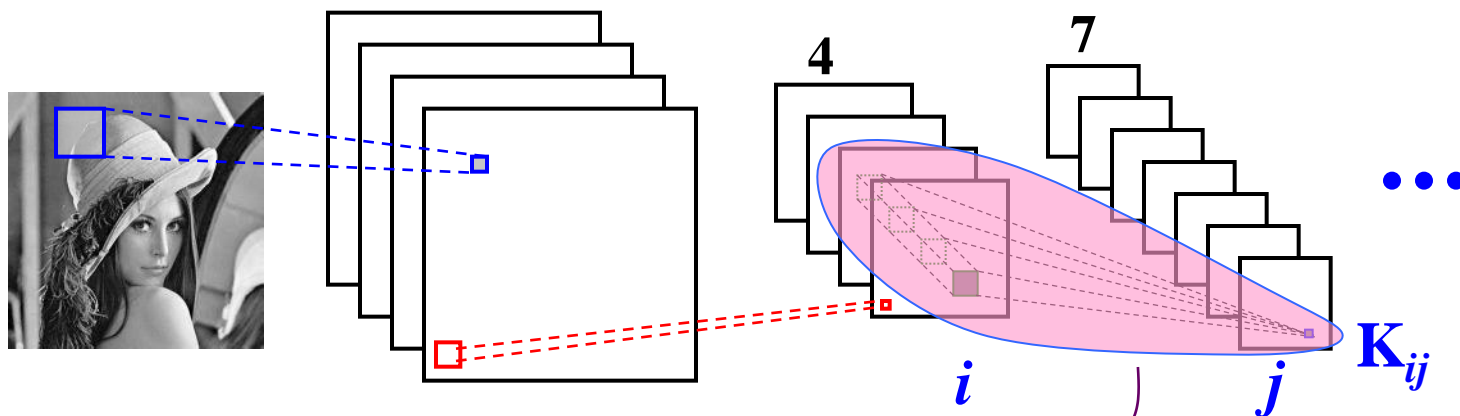
- 第一隐层至第二隐层：由于第一隐层包含4个图像，第二隐层包含7个图像，因此需要28个滤波器。如果第一隐层只包含1个图像，则只需要7个滤波器。



- 第一隐层至第二隐层：由于第一隐层包含4个图像，即4个大的结点集群（图像），需要将这些大的结点集群的输出值进行加权(即卷积)累加，累加到第二隐层的同一个集群结点（图像）。这正是前向神经网络的普遍操作特点。

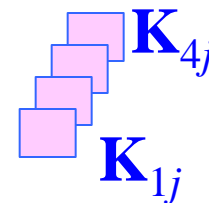


- 第一隐层至第二隐层：如果将4个卷积结果排列成一个产物，这种累加操作也可以直接在立体数据上进行。



也可以将4个图像看成一个立方体，称为volume，待学习的滤波器为一个三维滤波器。三维滤波器对立体数据进行卷积操作。此时只需要学习7个三维滤波器。

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



• 网络结构描述（举例）

- 图像大小：200×200
- 第一隐含层滤波器（卷积核）大小：5×5
- 第一隐含层结点集群（图像）个数：4
- 第一隐含层图像大小：198 × 198
- 第一隐含层pooling窗口大小：2×2
- 第一隐含层pooling之后图像大小：99×99

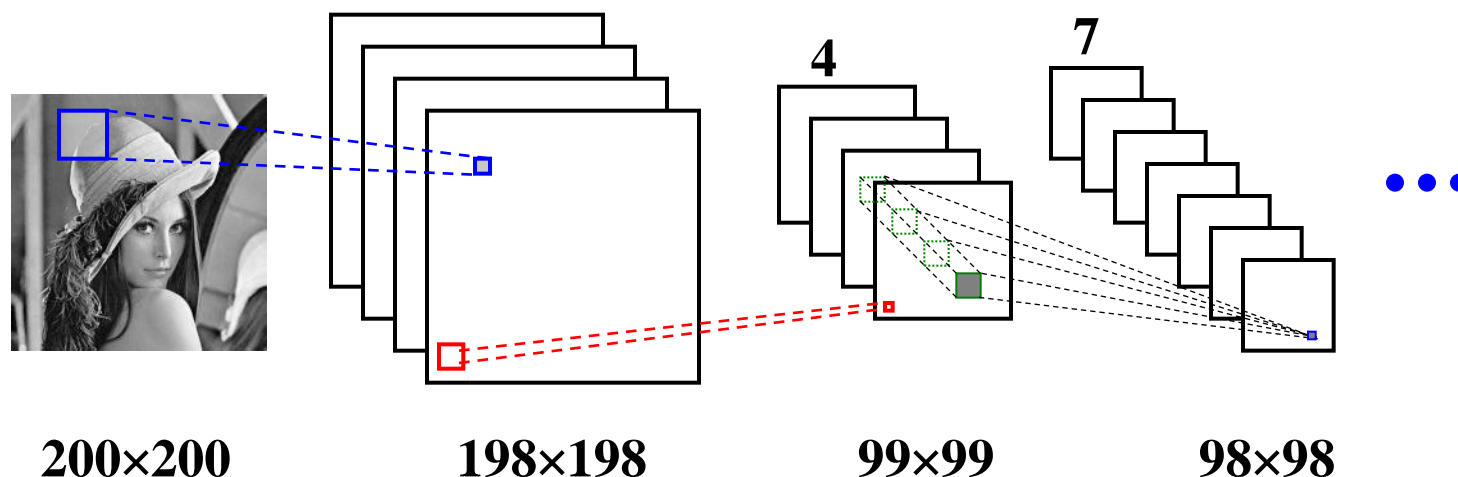
- 第二层滤波器（卷积核）大小：3×3
- 第二隐含层结点集群(图像) 个数：7
- 第二隐含层滤波器总数：28
- 第二隐含层图像大小：98 × 98
- 第二隐含层pooling窗口大小：2×2
- 第二隐含层pooling之后图像大小：49×49

(也可以理解为7个3×3 ×4的三维滤波器)

— ...

6.9.3 卷积神经网络

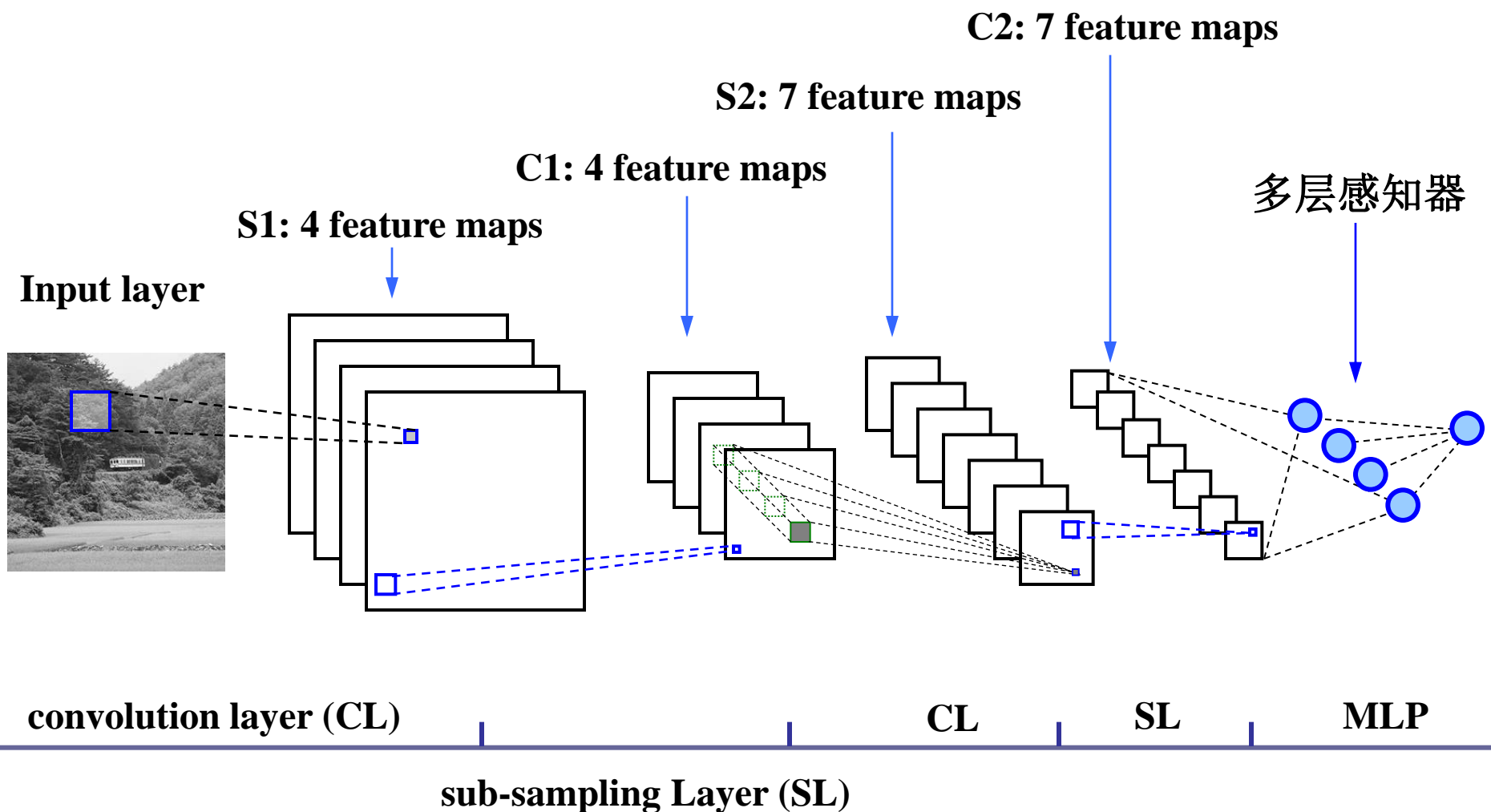
- 第一隐含层至第二隐含层权重数：
 - 全连接：输入 \times 输出 = $(4 \times 99 \times 99) \times (4 \times 98 \times 98)$
 - 局部连接+权值共享（ 3×3 滤波器）： $9 \times 4 \times 7$



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 **（特别指出：也可以在pooling之后求激励）**
 - 聚合 (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:



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 \mathbf{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 \mathbf{o} 与相应的理想输出 \mathbf{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：
 - 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，请问如何改？

6.9.3 卷积神经网络

- 思考题：
 - 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，请问如何改？

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 \mathbf{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 \mathbf{o} 与相应的理想输出 \mathbf{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。

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（或者权值复制）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

- 卷积神经网络的训练样本是带有标签的。在很多实际应用中，训练样本是没有标签的。此时要采用多层感知器的误差反向传播算法会遇到一些困难。
- 一种自然地扩展是在整个网络中让**输出与输入相等**。在此网络中，隐含层则可以理解为用于记录数据的特征，因此这是一种典型的表示学习方法。
- 实现这一宏观思想的一种著名的神经网络结构是就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

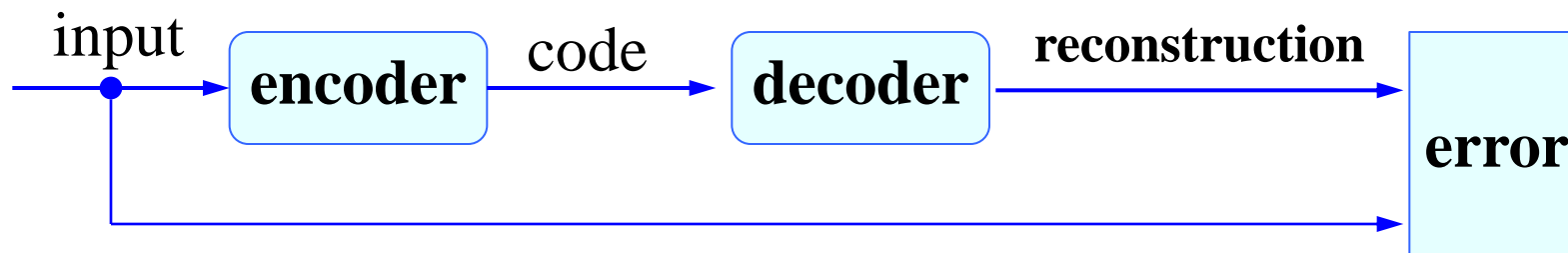
- 核心思想

- 给定一个神经网络，假设其输出与输入是相同的。通过训练该网络，可以得到各层中的权重。相应地，可得到关于输入样本 \mathbf{x} 的几种不同表示。即是说，每一层代表一种表示。这些表示就是特征。
- 自动编码器是一种尽可能重构输入信号的神经网络。
- 为了重构原始信号，就像主成分分析一样，它必须学习（捕捉）到可以代表输入数据的最重要的内在因素（比如，像主成分分析中获得的主成分那样）。

6.9.4 Autoencoder

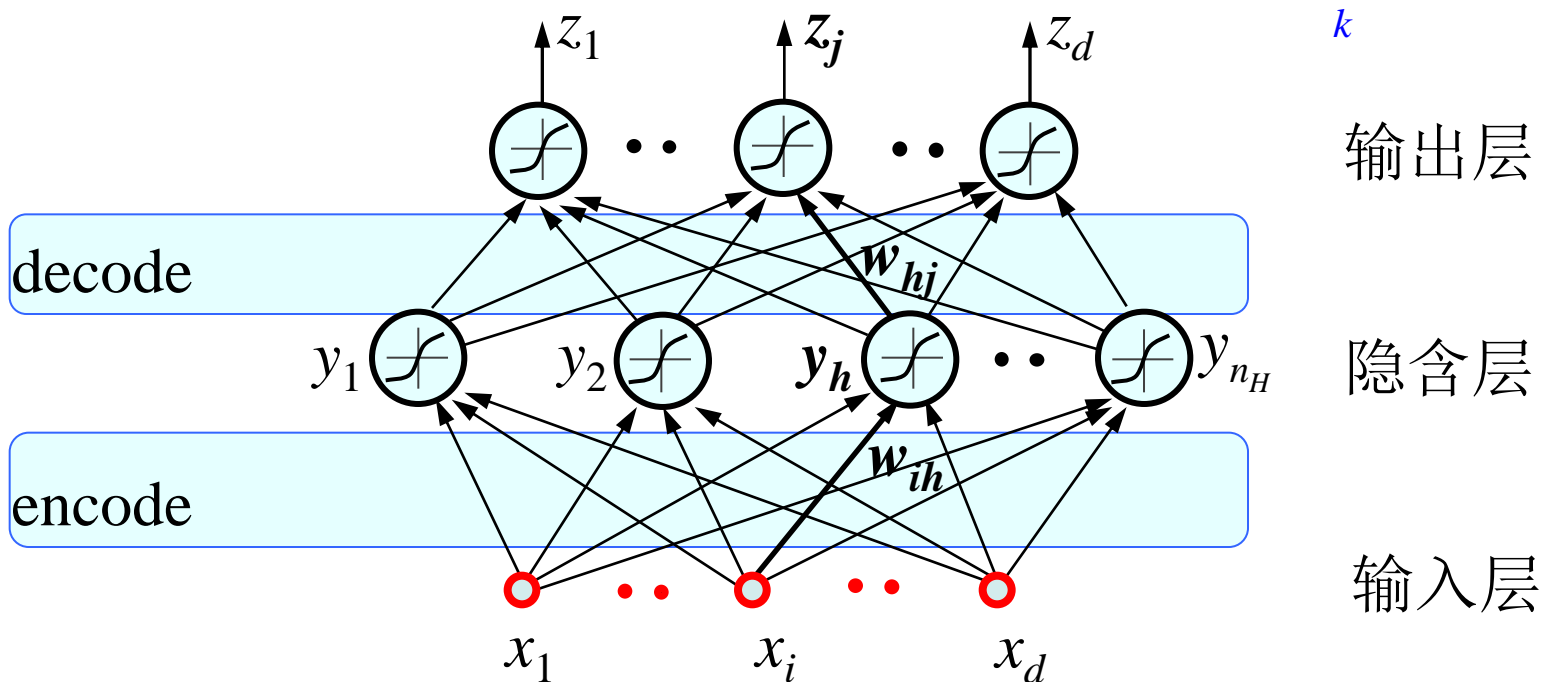
- 核心思想

- 将 input 输入一个 encoder 编码器，就会得到一个 code。这个 code 也就是输入的一个表示。
- 通过增加一个 decoder 解码器，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（表达可能不一样，但本质上反应的是同一个模式）与输入信号 input 是相同的。
- 此时会产生误差，我们期望这个误差最小。



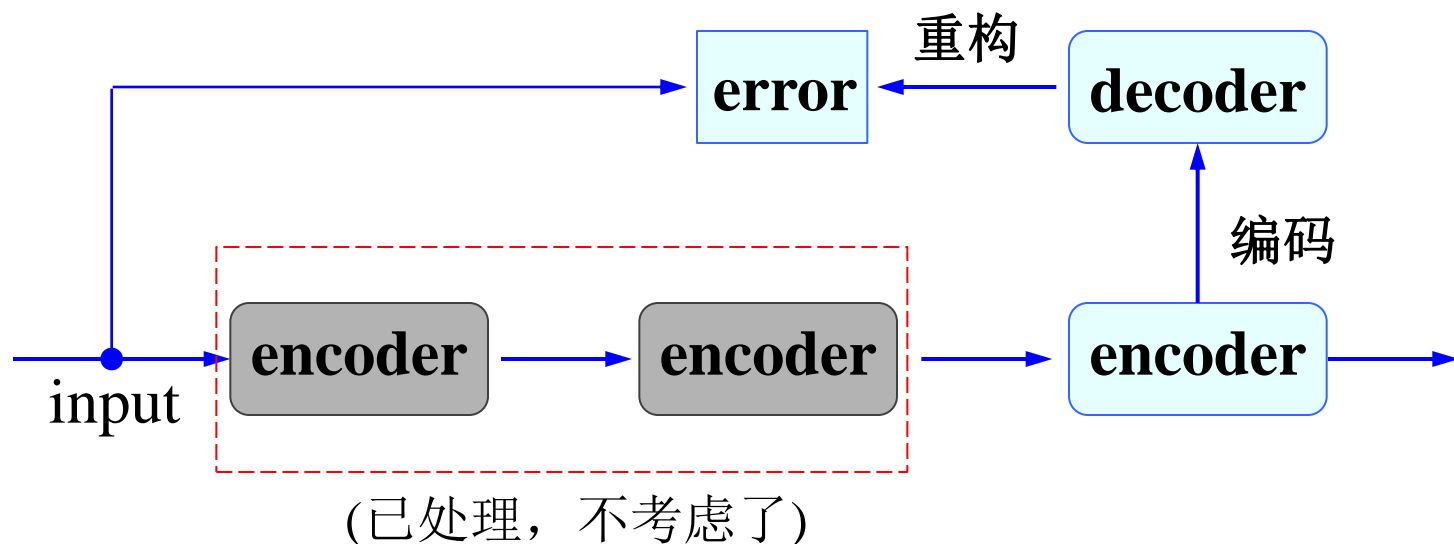
- 网络表达、第一次编解码训练：
 - 编码：建立输入层至隐含层的权重；
 - Code: 隐含层的输出，也称为表达或特征
 - 编码：建立隐含层至输出层的权重

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



- 网络训练：

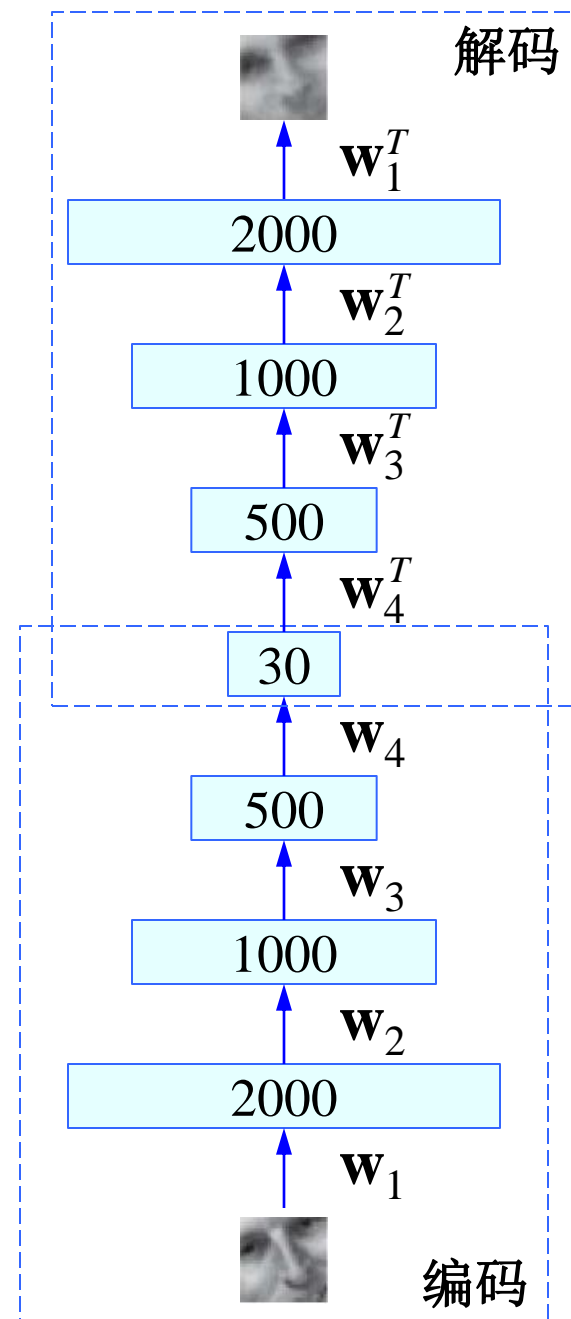
- 将第一层输出的 code 当成第二层的输入信号，**再次最小化重构误差**，得到第二层的权重参数，同时获得第二层的code，即原始号的第二个表达。
- 在训练当前层时，其它层固定不动。完成当前编码和解码任务。前一次“编码”和“解码”均不考虑。
- 因此，这一过程实质上是一个静态的堆叠(stack)过程
- 每次训练可以用BP算法对一个**三层前向网络**进行训练



• 对早期工作的说明*

- 在 Hinton 和 Salakhutdinov 的 2006年发表在《Science》的论文中，采用 **RBM** 来预训练网络，同时采用对称的网络结构（即解码器中的权重矩阵与编码器中的权重矩阵呈转置关系）。
 - 一个原因：当时训练数据少，且计算能力有限
- 现有的Auto-encoder更加灵活。

* G. E. Hinton and R. R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science, vol. 313, pp. 504-507



6.9.4 Autoencoder

- 扩展

- 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。因此，编码阶段获得的网络系统其实质是一种特征学习。层级越高，结构特征越大越明显。
- 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初始权重，采用带有标签的数据对网络进行再次学习，即所谓的 fine-tuning。为了实现分类任务，需要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）。
- 其它扩展：引用稀疏表示，对code 进行Hash快速查询，对数据增加噪声，等等。

6.9.4 Recurrent NN

- 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

6.9.4 Recurrent NN

- **Recurrent NN (RNN)**

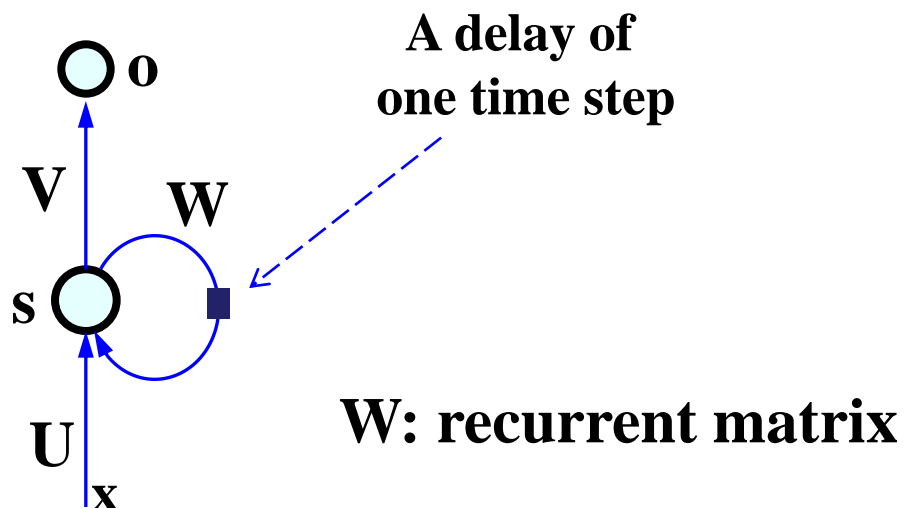
- 至少包含一个反馈连接的神经网络结构。因此，**网络的激励可以沿着一个loop 进行流动**。这种网络结构特别适合于处理时序数据。

- **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

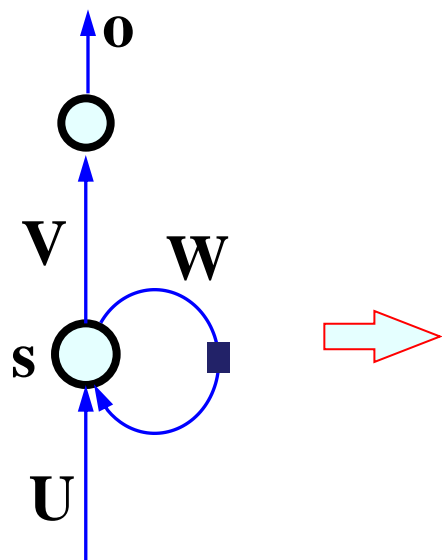
6.9.4 Recurrent NN

- 层次型Recurrent NN的基本结构(普通RNN)
 - \mathbf{x} : 输入信号; \mathbf{o} : 输出信号
 - \mathbf{s} : 隐含层结点的输出
 - \mathbf{U} : 输入至隐含层的连接权重矩阵 (input to hidden)
 - \mathbf{V} : 隐含层至输出层的连接权重矩阵 (hidden to output)
 - \mathbf{W} : 隐含层神经元之间的连接权重矩阵 (hidden to hidden)



6.9.4 Recurrent NN

- 网络结构



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

$$\mathbf{s}_t = \tanh(\mathbf{a}_t)$$

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

$$\mathbf{p}_t = \text{soft max}(\mathbf{o}_t)$$

$$(p(y_t = i) = p_t(i)) \quad \text{比如分类}$$

\mathbf{b}, \mathbf{c} : 偏移向量

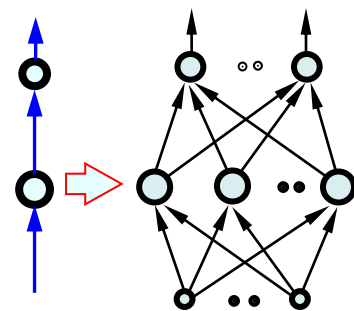
$\mathbf{U}, \mathbf{V}, \mathbf{W}$: 权重矩阵

\mathbf{p}_t : 属于 c 类的概率向量

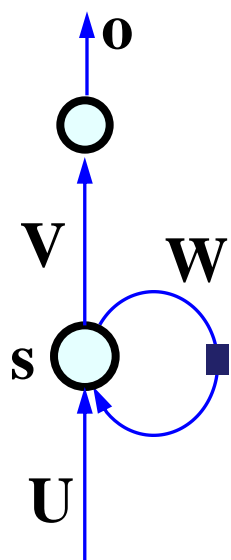
网络的功能可以解释为: How the state \mathbf{s}_t at time t captures and summarizes the information from the previous inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t$.

6.9.4 Recurrent NN

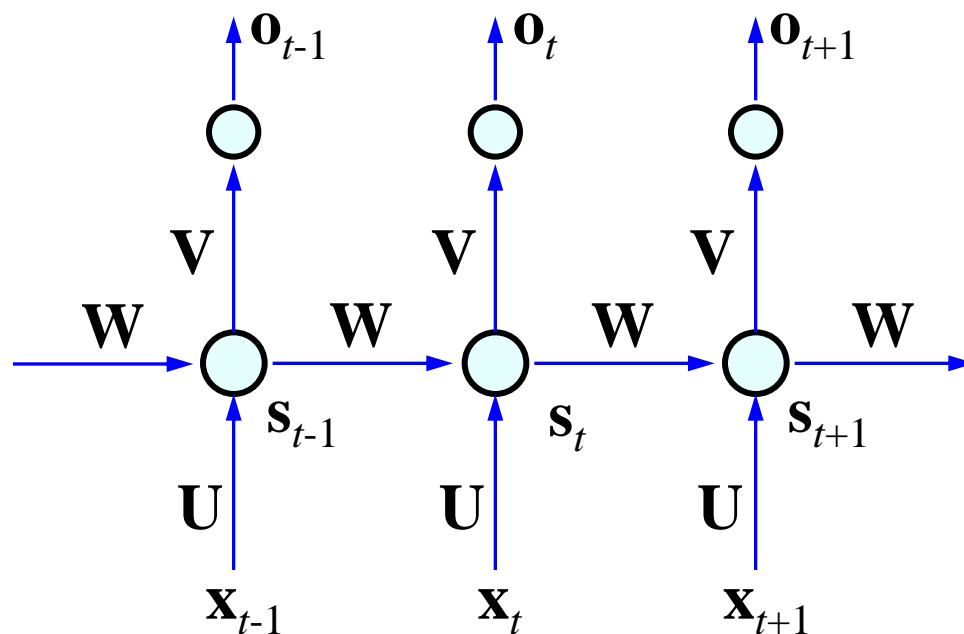
- 按时间顺序展开



权值共享: U, V, W

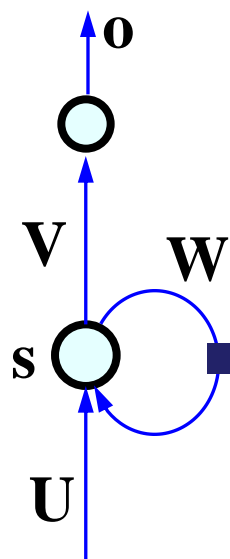


unfold

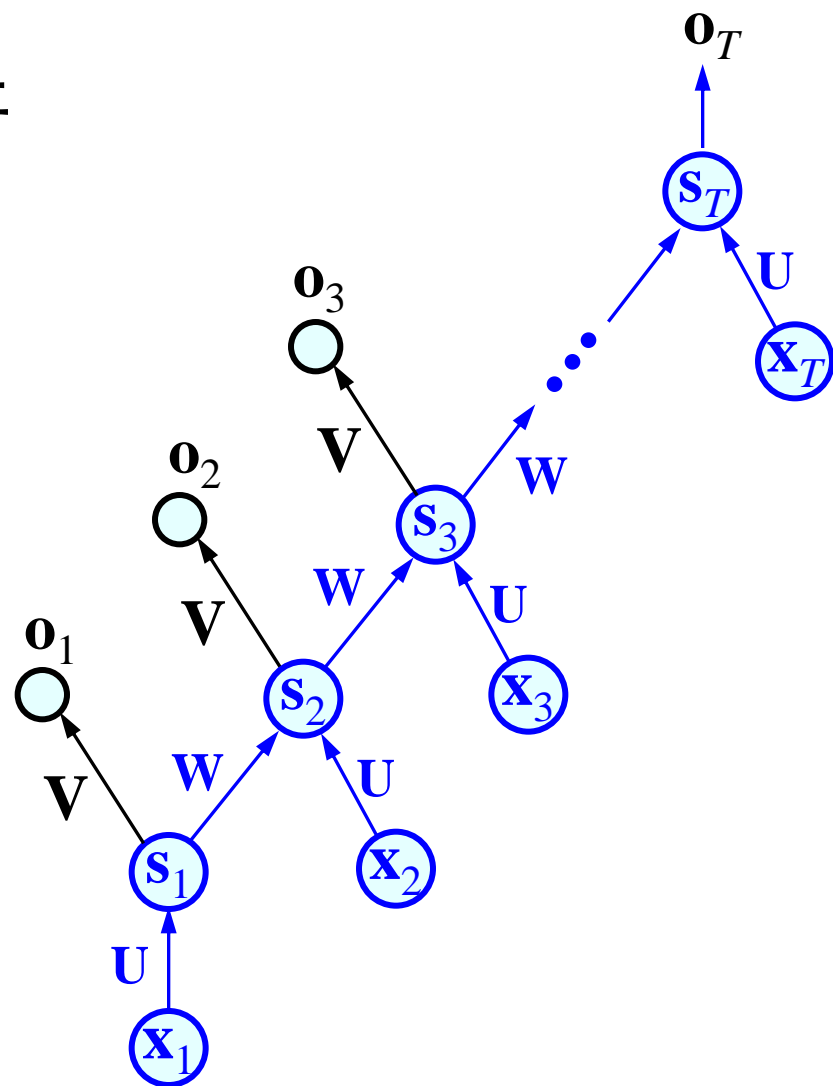


flow graph

- 按层次由下至上展开



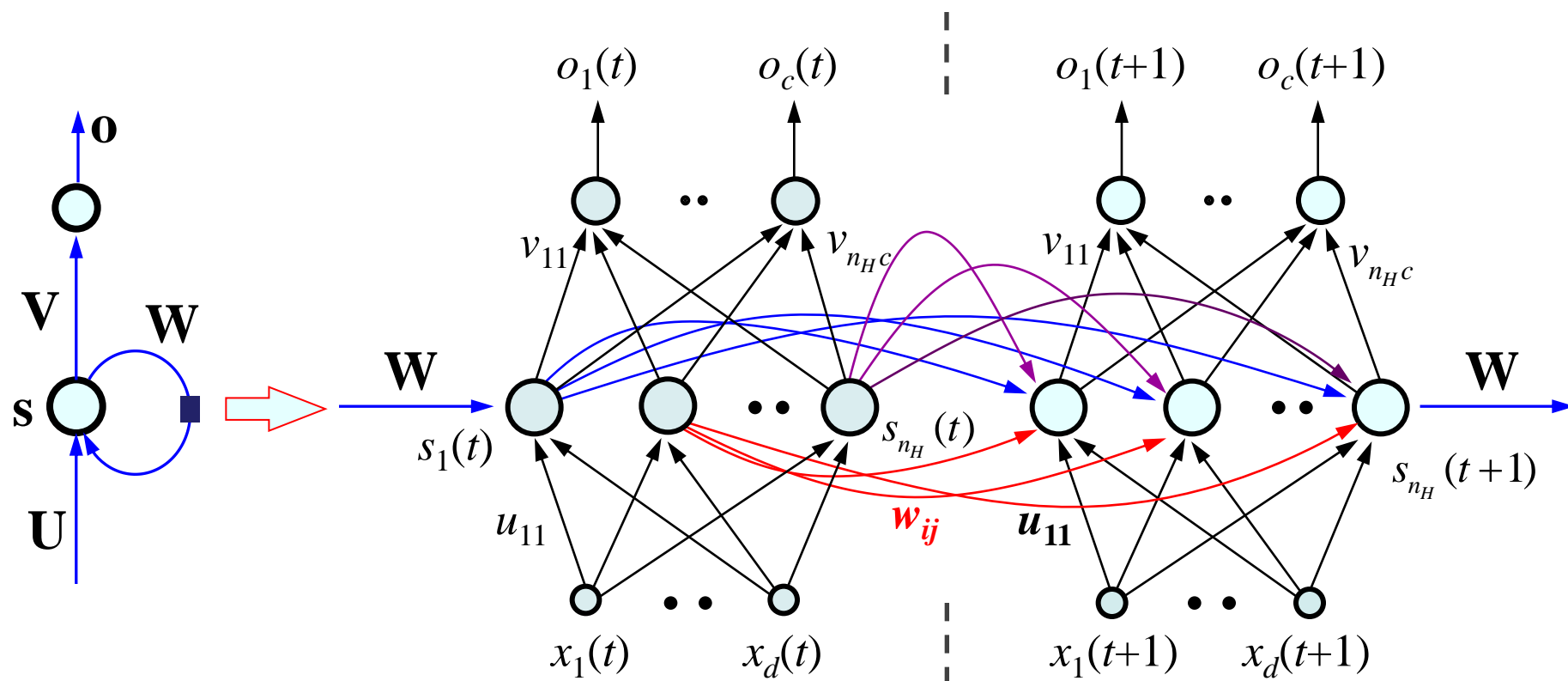
unfold



hierarchical graph

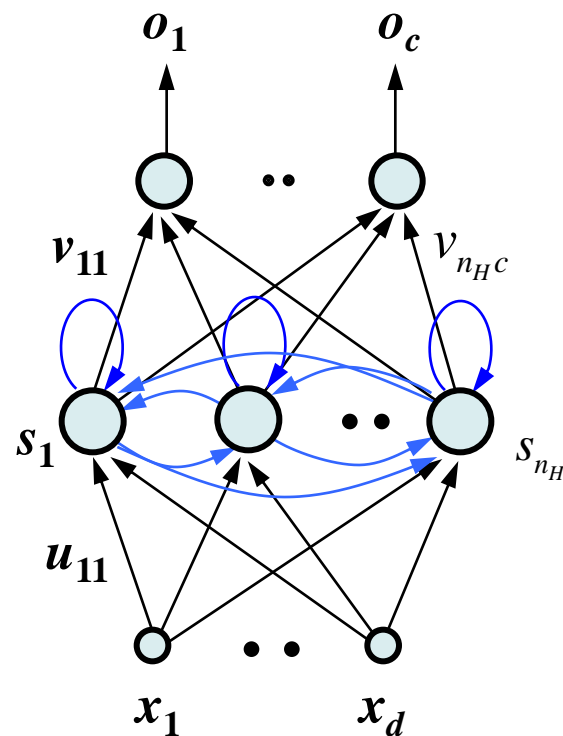
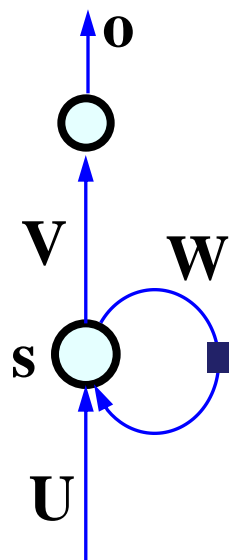
6.9.4 Recurrent NN

- 按时间顺序全结点展开



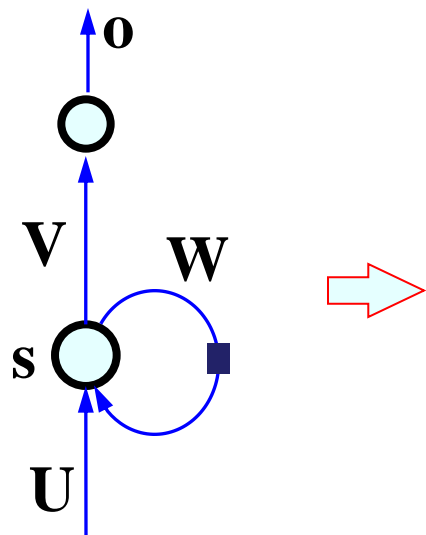
6.9.4 Recurrent NN

- 注意，与该结构不同：



6.9.4 Recurrent NN

- 网络训练



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

$$\mathbf{s}_t = \tanh(\mathbf{a}_t) \text{ (hyperbolic tangent func.)}$$

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

$$\mathbf{p}_t = \text{soft max}(\mathbf{o}_t) \quad (\text{对分类})$$

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

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

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

6.9.4 Recurrent NN

- softmax vs softmax loss

o_i 即为第 i 个结点
收集到的加权和

$$\text{soft max}(\mathbf{o}) = [\sigma_1(\mathbf{o}), \sigma_2(\mathbf{o}), \dots, \sigma_c(\mathbf{o})]^T \quad \leftarrow \text{样本}\mathbf{x}\text{属于}c\text{类的概率}$$

$$\sigma_i(\mathbf{o}) = \frac{\exp(o_i)}{\sum_{j=1}^c \exp(o_j)}, \quad i = 1, 2, \dots, c,$$

比如，多类线性判别：

$$\mathbf{o} = \mathbf{W}^T \mathbf{X} + w_0$$

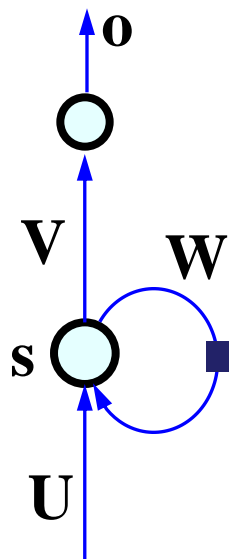
假设数据 \mathbf{x} 对应的类别为 y , y 取 $1, 2, \dots, c$ 中的某个整数。
计算最大似然就是要最大化 $\sigma_y(\mathbf{o})$ 的值 (即第 y 个分量)。

采用负对数似然，有：

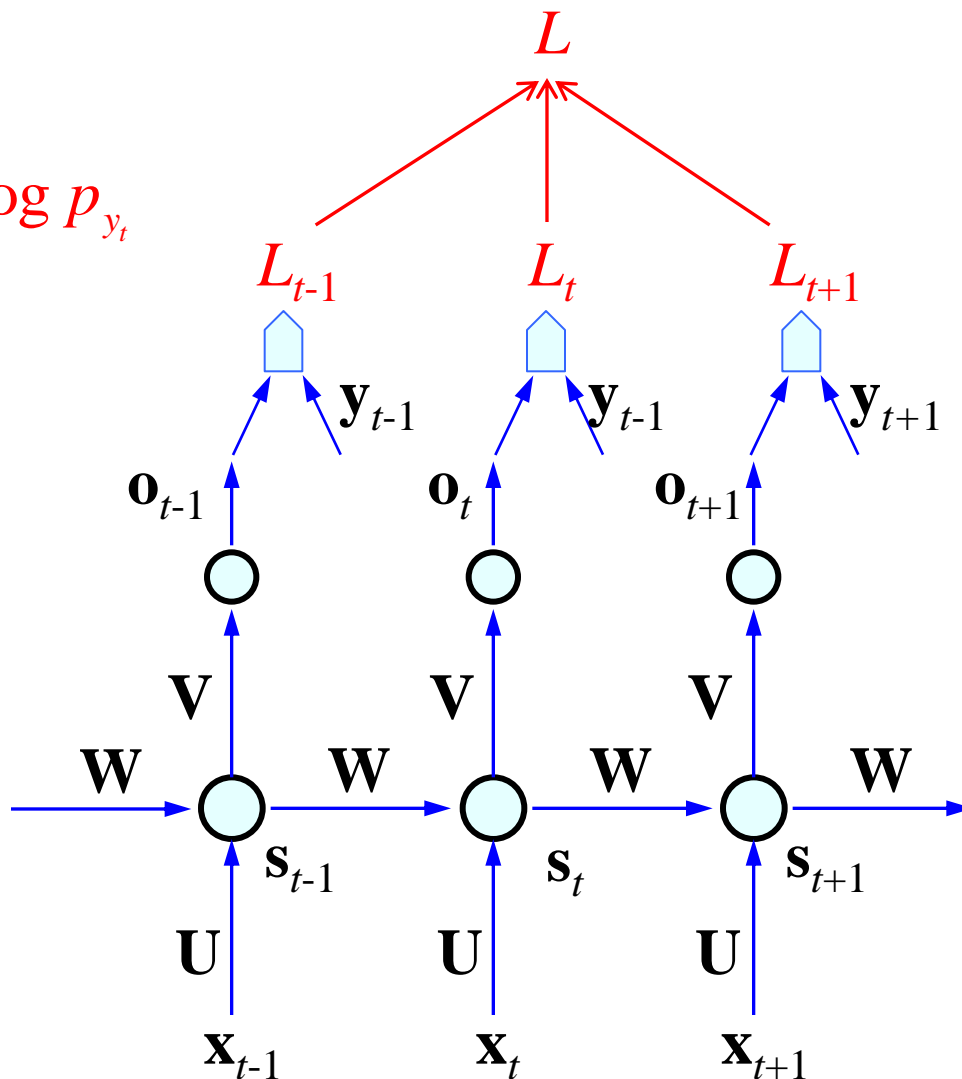
$$l(\mathbf{x}, y) = -\log(\sigma_y(\mathbf{o})) = -\log\left(\frac{e^{o_y}}{\sum_{j=1}^c e^{o_j}}\right) = \log\left(\sum_{j=1}^c e^{o_j}\right) - o_y$$

按时间进行累加：

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



unfold

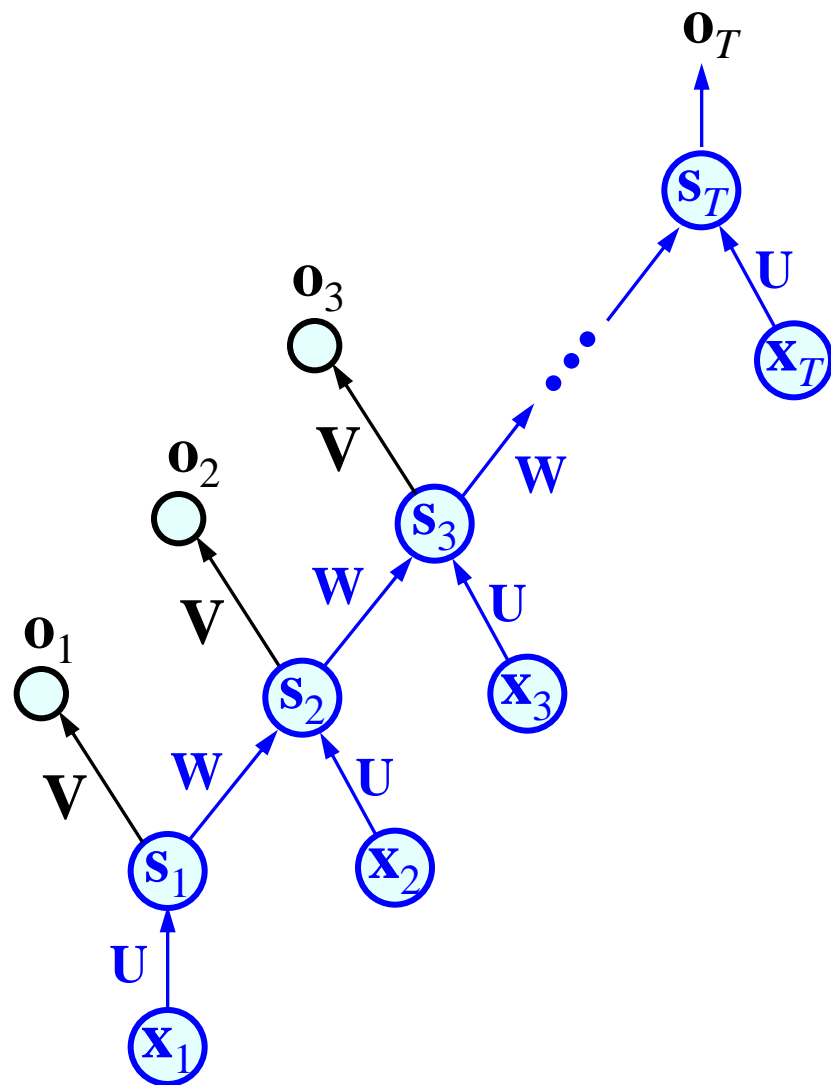


flow graph

6.9.4 Recurrent NN

- 网络训练

- 从多层前向网络的角度来看，其实上已经清楚怎样来训练这个网络，即采用类似的反向传播（BP）算法完成此事。通过准备误差来启动BP算法。
- 唯一不同之处在于：**权重是共享的，是相同的**。因此简单地采用现有BP方法显得效率不高。



hierarchical graph

6.9.4 Recurrent NN

- 网络训练

- 核心任务是计算目标函数对各参数的导数。其基本思路仍然是采用后向传播算法，即所谓的 **Back Propagation Through Time (BPTT)** 算法。
- 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{s}_t 's, \mathbf{o}_t 's, and L_t 's. （输入、隐状态、输出）
 - The parameters are \mathbf{U} , \mathbf{V} , \mathbf{W} , \mathbf{b} , \mathbf{c}

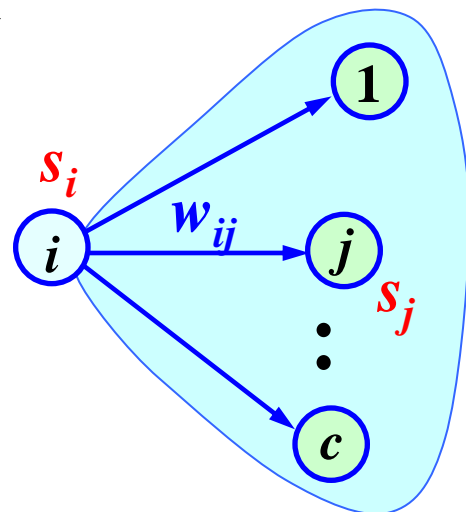
• 回忆BP算法

- 对于任意的一个准则函数 L , 在计算梯度下降时, 主要考察两方面的梯度:

- 其一: 对任一结点输出的梯度, 设 s_i 为当前层第 i 个结点的输出, 有:

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

$N(i)$ 为 i 的指向结点, 即反向传播中靠近输出层 (更深一层) 的与之有连接的结点



- 其二: 对权重的梯度:

$$\nabla_{w_{ij}} L = \frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial s_j} \frac{\partial s_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ij}}$$

其中, i 为靠近输入层的结点, j 为下一层(靠近输出层)结点。

求梯度任务之一：

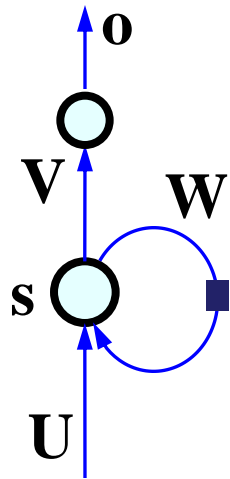
在 t 时刻，对输出结点 i 的导数：

$$\frac{\partial L}{\partial L_t} = 1, \quad \frac{\partial L}{\partial o_{ti}} = \frac{\partial L}{\partial L_t} \frac{\partial L_t}{\partial o_{ti}} = \frac{\partial L_t}{\partial o_{ti}} \quad \left(= p_{t,i} - \delta_{i,y_t}, \text{ for softmax} \right)$$

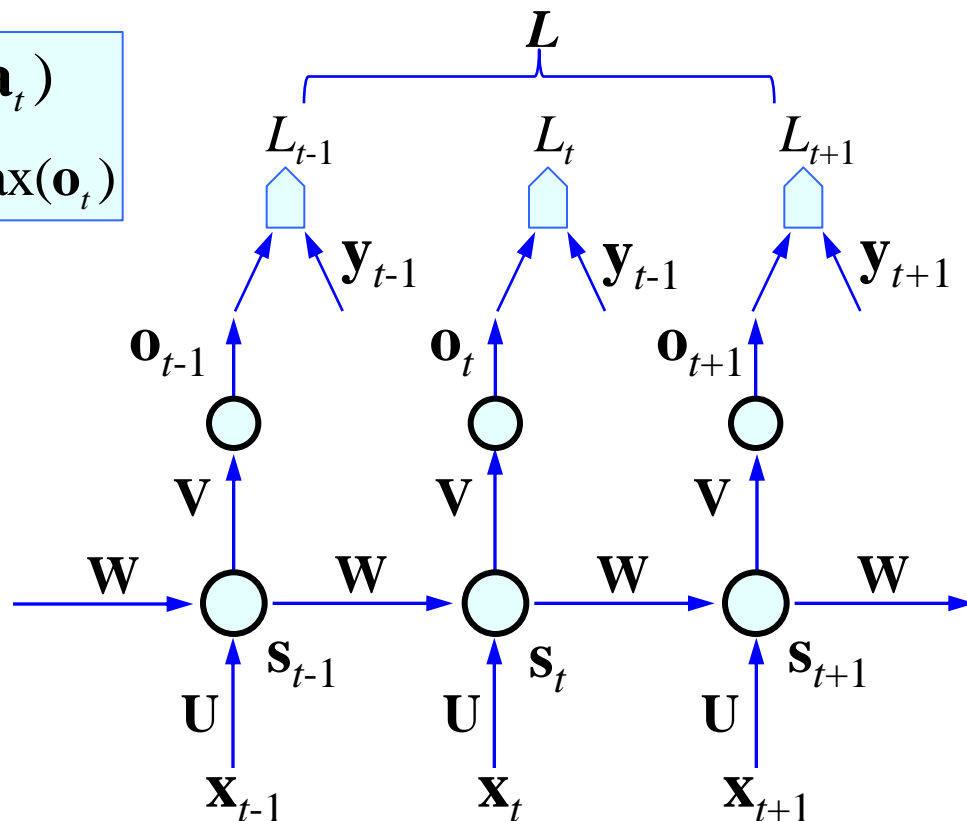
δ function

$$\mathbf{a}_t = \mathbf{b} + \mathbf{W}\mathbf{s}_{t-1} + \mathbf{U}\mathbf{x}_t, \quad \mathbf{s}_t = \tanh(\mathbf{a}_t)$$

$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{s}_t, \quad \mathbf{p}_t = \text{soft max}(\mathbf{o}_t)$$



unfold



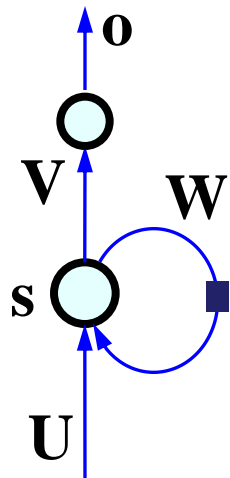
求梯度任务之二：

采用反向传播，从序列的末端点开始，比如时刻 T ，对所有 T 时刻的状态 \mathbf{s}_T ：

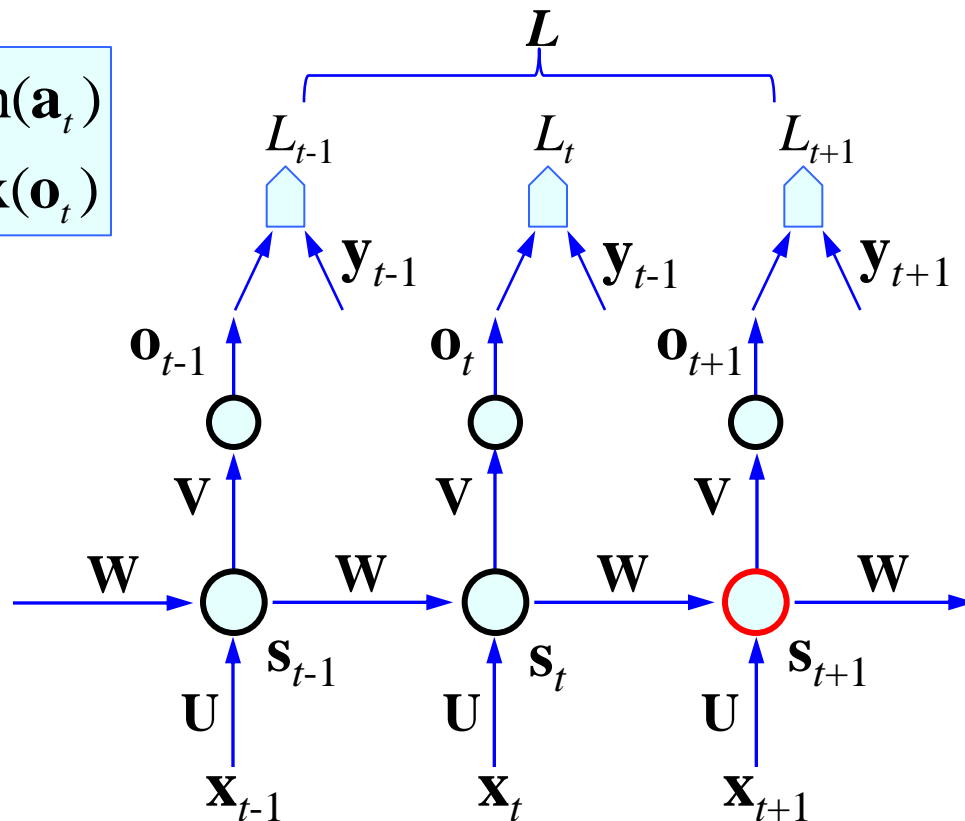
$$\boxed{\nabla \mathbf{s}_T L} = \nabla \mathbf{o}_T L \cdot \frac{\partial \mathbf{o}_T}{\partial \mathbf{s}_T} = \nabla \mathbf{o}_T L \mathbf{V}, \quad \text{即} \quad \frac{\partial L}{\partial \mathbf{s}_{T,j}} = \sum_i \frac{\partial L}{\partial o_{T,i}} V_{ji}, \quad j = 1, 2, \dots$$

(行向量)

$$\begin{aligned} \mathbf{a}_t &= \mathbf{b} + \mathbf{W}\mathbf{s}_{t-1} + \mathbf{U}\mathbf{x}_t, & \mathbf{s}_t &= \tanh(\mathbf{a}_t) \\ \mathbf{o}_t &= \mathbf{c} + \mathbf{V}\mathbf{s}_t, & \mathbf{p}_t &= \text{soft max}(\mathbf{o}_t) \end{aligned}$$



unfold



对所有隐结点

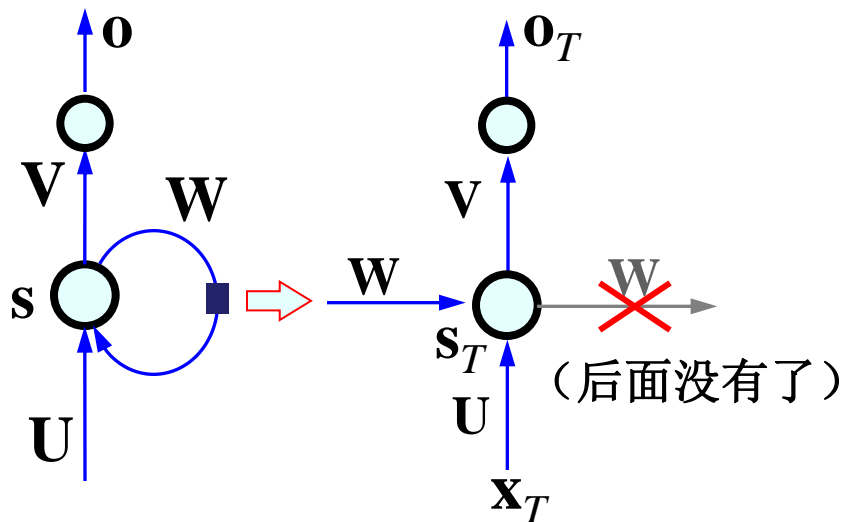
$$\nabla_{o_i} L = \frac{\partial L}{\partial o_i} = \sum_{j \in N(i)} \frac{\partial L}{\partial o_j} \frac{\partial o_j}{\partial o_i}$$

对单个隐结点

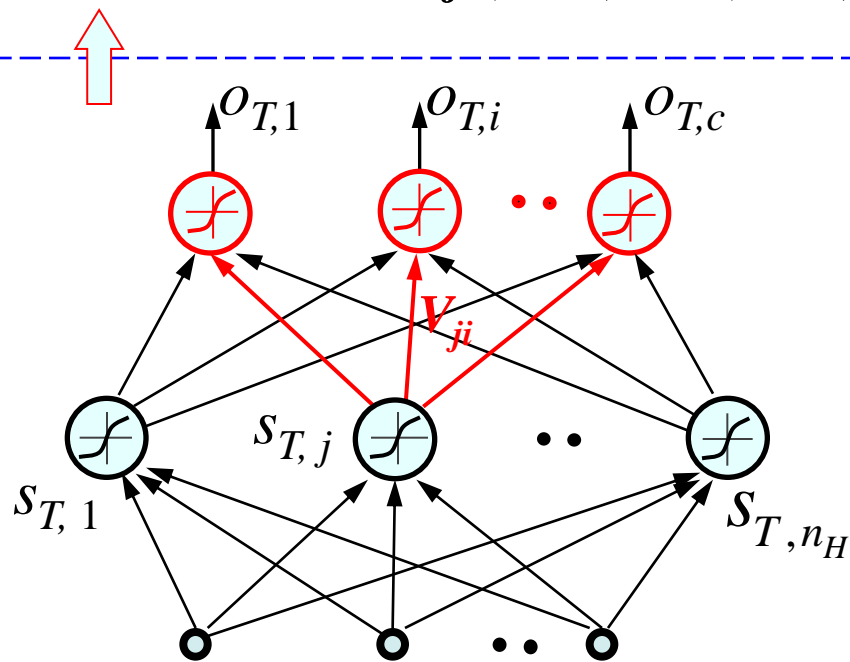
$$\nabla_{\mathbf{s}_T} L = \nabla_{\mathbf{o}_T} L \cdot \frac{\partial \mathbf{o}_T}{\partial \mathbf{s}_T} = \nabla_{\mathbf{o}_T} L \mathbf{V},$$

$$\text{即 } \frac{\partial L}{\partial \mathbf{s}_{T,j}} = \sum_i \frac{\partial L}{\partial o_{T,i}} \mathbf{V}_{ji}, \quad j = 1, 2, \dots$$

$$\begin{aligned} \mathbf{a}_t &= \mathbf{b} + \mathbf{W}\mathbf{s}_{t-1} + \mathbf{U}\mathbf{x}_t, & \mathbf{s}_t &= \tanh(\mathbf{a}_t) \\ \mathbf{o}_t &= \mathbf{c} + \mathbf{V}\mathbf{s}_t, & \mathbf{p}_t &= \text{soft max}(\mathbf{o}_t) \end{aligned}$$



权连接: j (隐含) $\rightarrow i$ (输出)

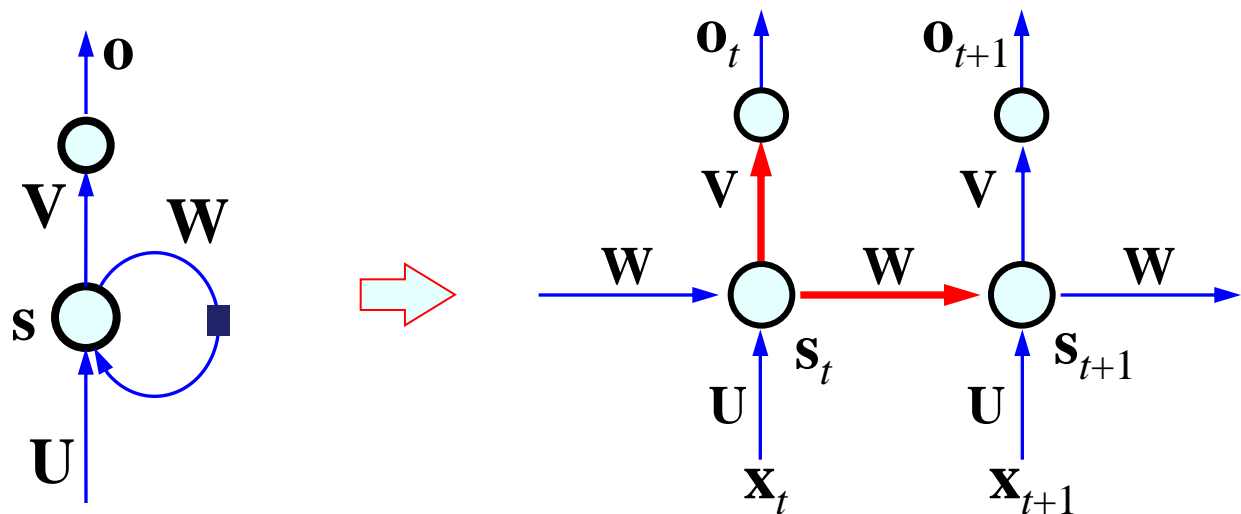


在时刻 T 将折叠的网络展开

求梯度任务之三：

对所有 $t < T$ 时刻的状态 \mathbf{s}_t , 由于 \mathbf{s}_t 同时向 \mathbf{o}_t 和 \mathbf{s}_{t+1} 传, 有:

$$\nabla_{\mathbf{s}_t} L = \nabla_{\mathbf{o}_t} L \frac{\partial \mathbf{o}_t}{\partial \mathbf{s}_t} + \nabla_{\mathbf{s}_{t+1}} L \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} = \nabla_{\mathbf{s}_t} L \mathbf{V} + \nabla_{\mathbf{s}_{t+1}} L \text{diag}(1 - \mathbf{s}_{t+1}^2) \mathbf{W}$$



$$\begin{aligned} \mathbf{a}_t &= \mathbf{b} + \mathbf{W}\mathbf{s}_{t-1} + \mathbf{U}\mathbf{x}_t, & \mathbf{s}_t &= \tanh(\mathbf{a}_t) \\ \mathbf{o}_t &= \mathbf{c} + \mathbf{V}\mathbf{s}_t, & \mathbf{p}_t &= \text{soft max}(\mathbf{o}_t) \end{aligned}$$

$$t < T$$

求梯度任务之四：

Once the gradients on the internal nodes of the flow graph are obtained, we can obtain the gradients on the parameter nodes, which have descendent at all the time steps:

$$\nabla_{\mathbf{c}} L = \sum_t \nabla_{\mathbf{o}_t} L \frac{\partial \mathbf{o}_t}{\partial \mathbf{c}} = \sum_t \nabla_{\mathbf{o}_t} L$$

$$\nabla_{\mathbf{b}} L = \sum_t \nabla_{\mathbf{s}_t} L \frac{\partial \mathbf{s}_t}{\partial \mathbf{b}} = \sum_t \nabla_{\mathbf{s}_t} L \text{diag}(1 - \mathbf{s}_t^2)$$

$$\nabla_{\mathbf{V}} L = \sum_t \nabla_{\mathbf{o}_t} L \frac{\partial \mathbf{o}_t}{\partial \mathbf{V}} = \sum_t \nabla_{\mathbf{o}_t} L \mathbf{s}_t^T$$

$$\nabla_{\mathbf{W}} L = \sum_t \nabla_{\mathbf{s}_t} L \frac{\partial \mathbf{s}_t}{\partial \mathbf{W}} = \sum_t \nabla_{\mathbf{s}_t} L \text{diag}(1 - \mathbf{s}_t^2) \mathbf{s}_{t-1}^T$$

$$\mathbf{a}_t = \mathbf{b} + \mathbf{W}\mathbf{s}_{t-1} + \mathbf{U}\mathbf{x}_t, \quad \mathbf{s}_t = \tanh(\mathbf{a}_t)$$

$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{s}_t, \quad \mathbf{p}_t = \text{soft max}(\mathbf{o}_t)$$

6.9.5 Long Short Term Memory

- 采用一些控制门来减少梯度累积的长度

- RNN

- L1 / L2 regularization on the recurrent weights
 - Smart initialization (no one knows..)
 - Self-loop, leak units with linear self-connections
 - Echo State Networks
 - Hessian-Free optimization
 - Gradient Clip
 - ...

Not content-dependent

- LSTM

一旦信息得到利用，我们希望该结点能释放(遗忘)这种累积效应，从而使网络更具有灵活性和自主学习能力(依赖于学习内容来自我决定)。Just LSTM!

6.9.5 LSTM

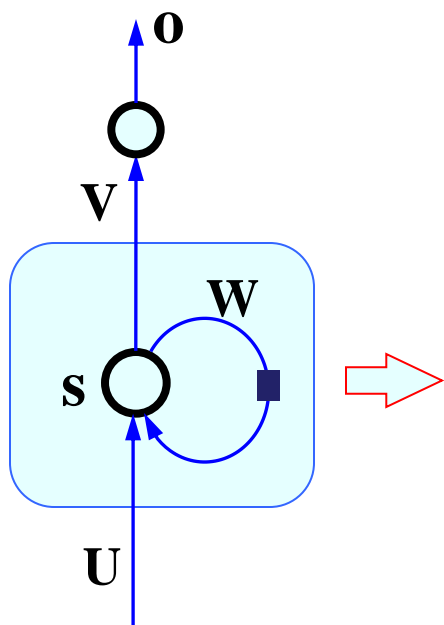
- 结构

- 传统的RNN仅仅应用激励函数作用于“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。
- LSTM网络则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

6.9.5 LSTM

- 结构

- 由传统的细胞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**当前正常输入(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:

$$in_{t,i} = \text{sigmoid}\left(b_{in}(i) + \sum_j U_{in}(i, j)x_{t,j} + \sum_j W_{in}(i, j)h_{t-1,j}\right)$$

time t , node i

-
- \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{i}_t : $= [in_{t,1}, in_{t,2}, in_{t,3}, \dots]$ (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:

$$f_{t,i} = \text{sigmoid}\left(b_f(i) + \sum_j U_f(i, j)x_{t,j} + \sum_j W_f(i, j)h_{t-1,j}\right)$$

time t , cell i

\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 : $= [f_{t,1}, f_{t,2}, f_{t,3}, \dots]$ (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：

产生新记忆

读出旧记忆

贡献部分新忘记


$$s_{t+1,i} = f_{t,i} \cdot s_{t,i} + in_{t,i} \cdot c_{t,i}$$

$$c_{t,i} = \tanh\left(b(i) + \sum_j U(i, j)x_{t,j} + \sum_j W(i, j)h_{t-1,j}\right)$$

- \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;
- \mathbf{s}_{t+1} : $= [s_{t+1,1}, s_{t+1,2}, s_{t+1,3}, \dots]$ (用向量记录所有新记忆)
- \mathbf{c}_t : $= [c_{t,1}, c_{t,2}, c_{t,3}, \dots]$ (用向量记录所有候选记忆)
- 贡献部分新忘记

- 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：

$$o_{t,i} = \text{sigmoid}\left(b_o(i) + \sum_j U_o(i, j)x_{t,j} + \sum_j W_o(i, j)h_{t-1,j}\right)$$

time t , cell i

-
- \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}_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 : $= [o_{t,1}, o_{t,2}, o_{t,3}, \dots]$ (用向量记录所有权重)

6.9.5 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：

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

输出门提供的权重

6.9.5 LSTM

- 结构描述——采用矩阵形式

遗忘门、输入门、输出门：

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

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

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

候选记忆（新贡献部分）：

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

Cell产生的新记忆：

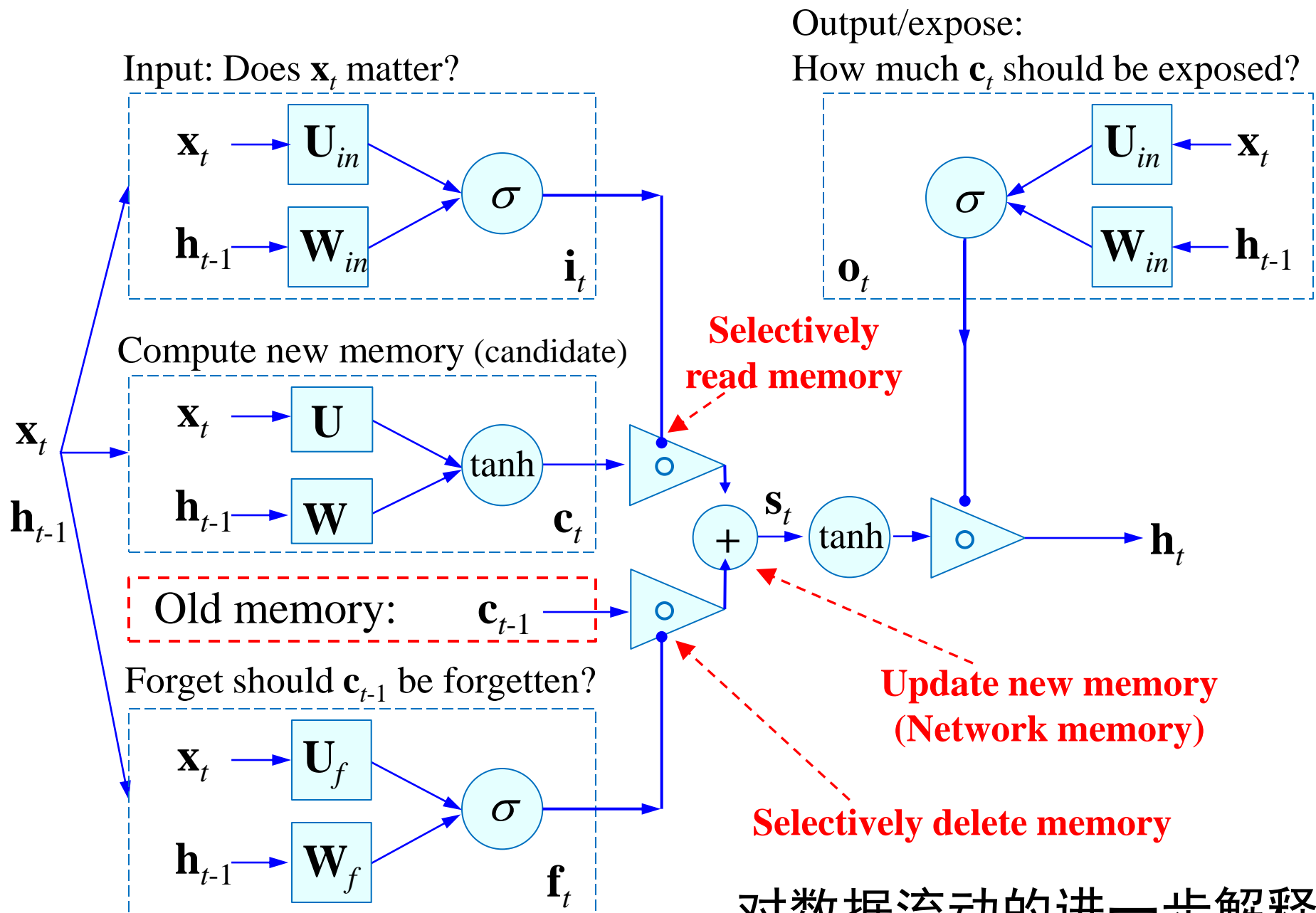
$$\mathbf{s}_t = \mathbf{f}_t \otimes \mathbf{s}_{t-1} + \mathbf{i}_t \otimes \mathbf{c}_t$$

Cell的输出：

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

网络的输出：

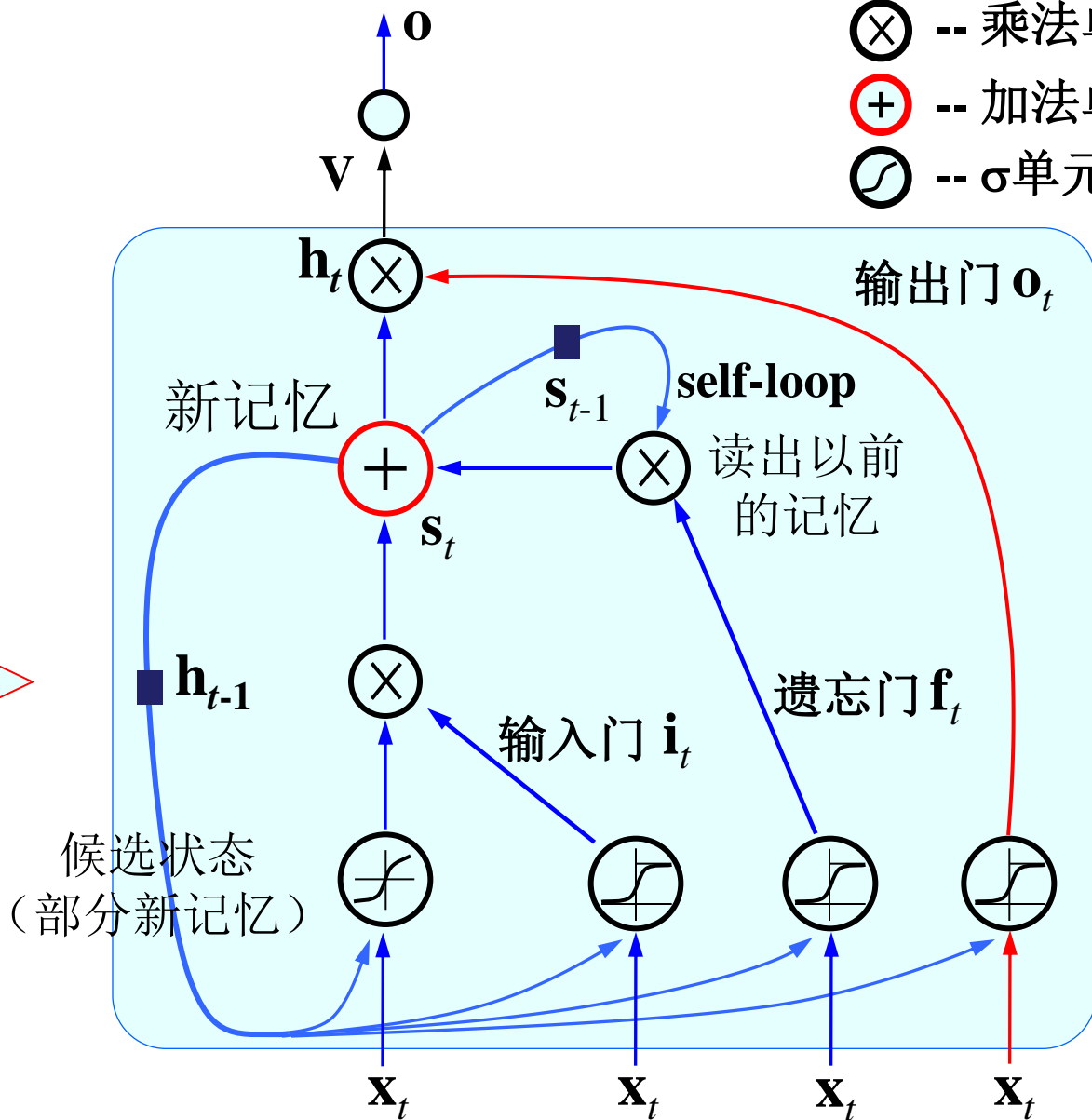
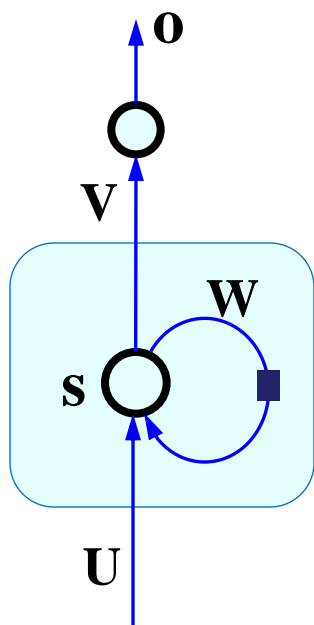
$$\mathbf{z}_t = \text{softmax}(\mathbf{V} \mathbf{h}_t + \mathbf{c})$$



对数据流动的进一步解释

隐含层细胞结构 (矩形内)

- \otimes -- 乘法单元
- \oplus -- 加法单元
- σ -- σ 单元



- 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—鼓励信息流动

6.9.6 深度学习小结

- 未来研究问题与实践
 - 从理论上研究深度学习为什么有效
 - 多感知信息的融合
 - 时间依赖性的更有效的构架
 - 利用深度学习思想增强传统学习方法的性能
 - 深度神经网络的训练与计算并行化

RNN的相关资源

- **Lectures**

- Stanford NLP (CS224d), by Richard Socher
 - Lecture Note 3 : neural network basics
 - Lecture Note 4 : RNN language models, bi-directional RNN, GRU, LSTM
- Stanford vision (CS231n), by Andrej Karpathy
 - About NN basic, and CNN
- Oxford Machine Learning, by Nando Freitas & Alex Graves
 - Lecture 12 : Recurrent neural networks and LSTMs
 - Lecture 13 : Hallucination with RNNs

RNN的相关资源 (open source)

- **Codes in python**

- **Theano**

- RNN for semantic parsing of speech
 - LSTM network for sentiment analysis
 - Theano-rnn, by Graham Taylor

- **Lasagne**

- Lightweight library to build and train neural networks in Theano

- **Neuraltalk & Gist, by Andrej Karpathy**

- Numpy-based RNN/LSTM implementation
 - Numpy code that implements batched LSTM

RNN的相关资源

- **Codes in Lua**

- **Torch**

- RNN/LSTM/GRU for training/sampling from character-level language models
 - LSTM for training a language model on word level Penn Tree Bank dataset

RNN的相关资源

- **Codes in C++**
 - Caffe, with MATLAB/Python wrappers
 - Long-term Recurrent Convolutional Networks, CVPR15
 - RNNLIB, by Alex Graves
 - LSTM library for speech and handwriting recognition
 - RNNLM, by Tomas Mikolov
 - Simple code for neural network based language models
 - Faster-RNNLM, by Yandex
 - RNNLM implementation aimed to handle huge datasets

相关阅读材料（建议读）

- 精读

- Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. Gradient-Based Learning Applied to Document Recognition, Proceedings of IEEE, vol. 86, no. 11, pp. 2278-2324, 1998. (主要内容：第1至第17页)
- G. E. Hinton and R. R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science, vol. 313, pp. 504-507
- Yoshua Bengio, Ian J. Goodfellow and Aaron Courville等的书：Deep Learning, <http://www.iro.umontreal.ca/~bengioy/DLbook/> (第10章的第1至第7节：有关RNN的大约30页的内容)

References（为更全面深入学习）

- C.M. Bishop, Chapter 6, Pattern Recognition and Machine Learning, Springer, 2006
- C. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995.
- G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines, Tech Report, No. UTML TR 2010-003, Department of Computer Science, University of Toronto, Canada
- Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. Gradient-Based Learning Applied to Document Recognition, Proceedings of IEEE, Vol. 86, No. 11, pp. 2278-2324, 1998.
- G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006

References（为更全面深入学习）

- Yoshua Bengio: Learning Deep architectures for AI, Foundations and Trends in Machine Learning, 2(1), 2009
- G. Hinton, S. Osindero and Y. W. Teh. A fast learning algorithm for deep belief nets. Neural Computation, 2006
- Kevin P. Murphy. Machine Learning: A Probabilistic Perspective. Chapter 28 : Deep Learning. The MIT Press, 2012
- Yoshua Bengio, Ian J. Goodfellow and Aaron Courville等的书： Deep Learning
<http://www.iro.umontreal.ca/~bengioy/DLbook/>
- Jürgen Schmidhuber. Deep Learning in Neural Networks: An Overview, Neural Network, vol. 61, 85-117, 2015.
- Yann LeCun, Yoshua Bengio, Geoffrey Hinton. Deep learning, Nature, Vol. 521, pp. 436-444, 2015.
- <https://github.com/robertsdionne/neural-network-papers>（资料较全）
- <http://deeplearning.net/tutorial/>

致谢

- Courtesy for some slides
 - Ying Wang
 - Kai Yu
 - Geoffrey Hinton
 - Zhongzhi Shi
 - Qingming Huang
 - Xiaogang Wang
 - Yan Huang
 - ...

Thank All of You!