

第十二届中国中文信息学会暑期学校
暨中国中文信息学会《前沿技术讲习班》

深度学习基础

邱锡鹏

复旦大学

2017年8月17日

<https://nndl.github.io/>



大纲

▶ 概述

- ▶ 机器学习概述
- ▶ 感知器
- ▶ 应用

▶ 基础模型

- ▶ 前馈神经网络
- ▶ 卷积神经网络
- ▶ 循环神经网络
- ▶ 网络正则化与优化
- ▶ 应用

▶ 进阶模型

- ▶ 注意力机制与外部记忆
- ▶ 无监督学习
- ▶ 概率图模型
- ▶ 深度生成模型
- ▶ 深度强化学习
- ▶ 模型独立的学习方式



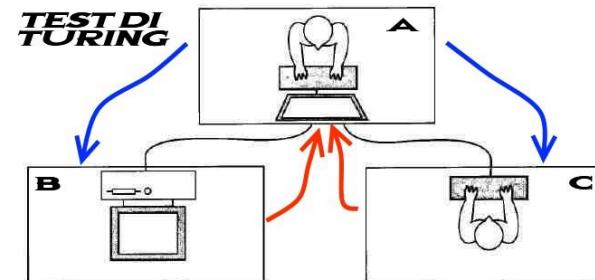
概述

从人工智能开始

- ▶ 让机器具有人类的智能
- ▶ 机器感知（计算机视觉、语音信息处理）
- ▶ 学习（模式识别、机器学习、强化学习）
- ▶ 语言（自然语言处理）
- ▶ 记忆（知识表示）
- ▶ 决策（规划、数据挖掘）

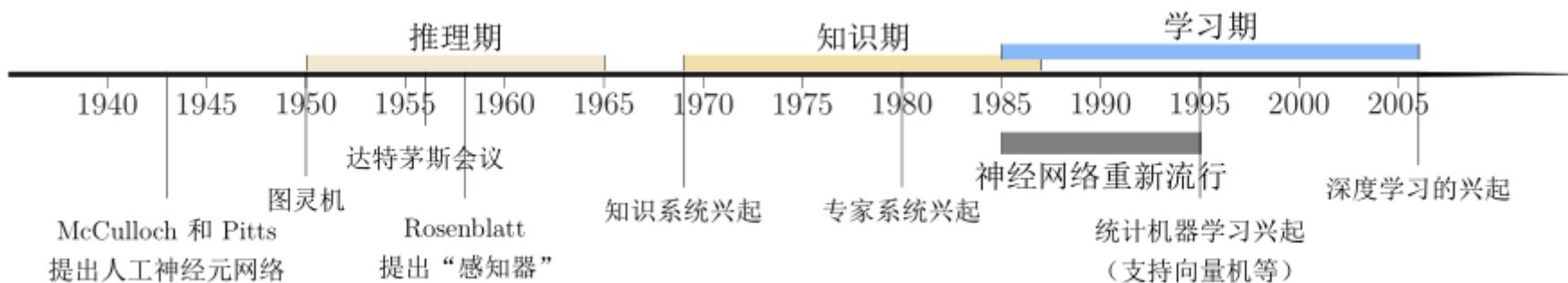


Alan Turing



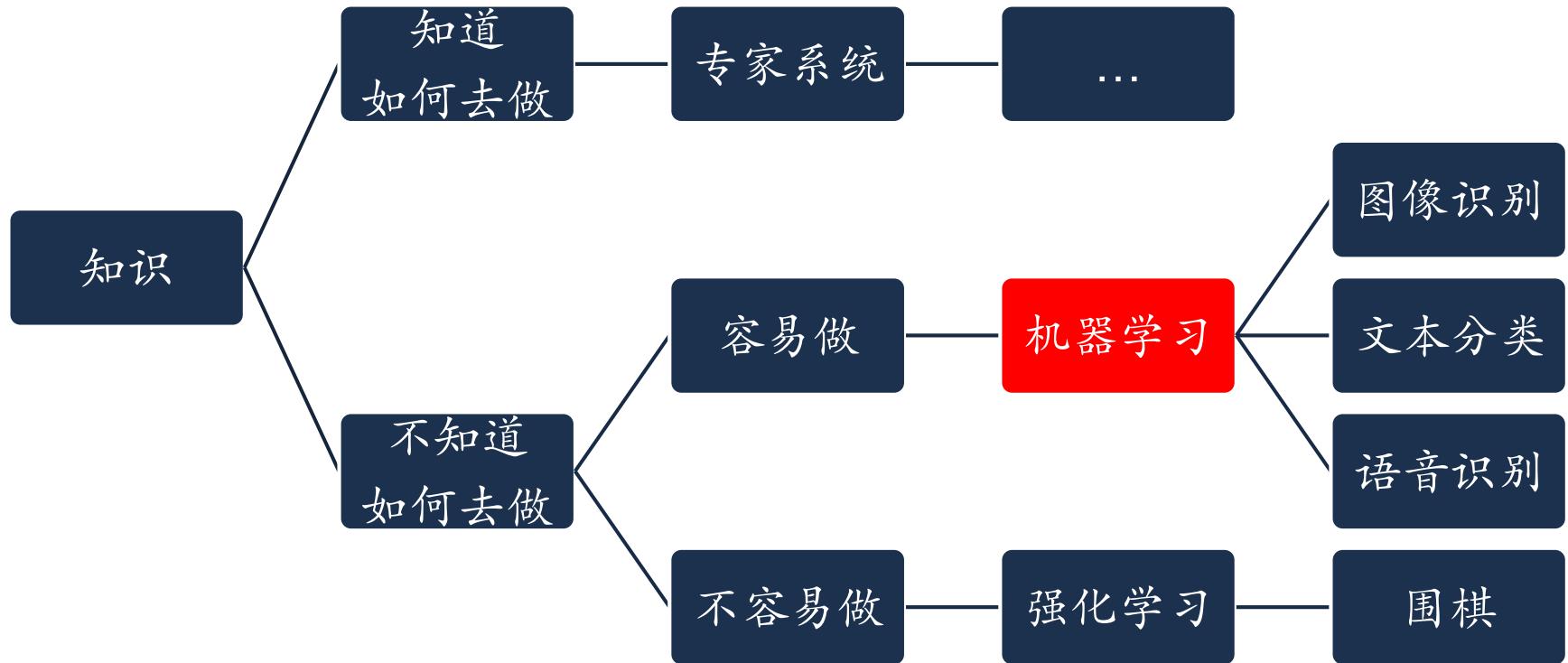
人工智能的发展历史

- ▶ 从诞生至今，人工智能这个领域经历了一次又一次的繁荣与低谷，其发展上大体上可以分为三段时期：
 - ▶ 推理期
 - ▶ 知识期
 - ▶ 学习期





如何开发一个人工智能系统？



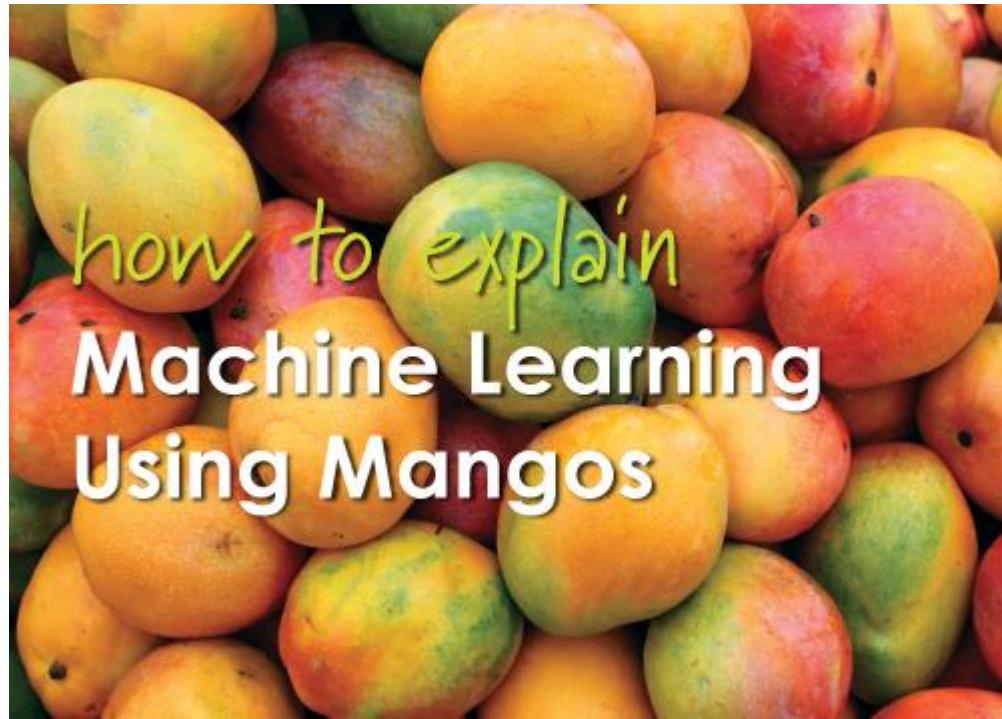


机器学习概述



芒果机器学习

如果判断芒果是否甜蜜？



<https://www.quora.com/How-do-you-explain-Machine-Learning-and-Data-Mining-to-non-Computer-Science-people>

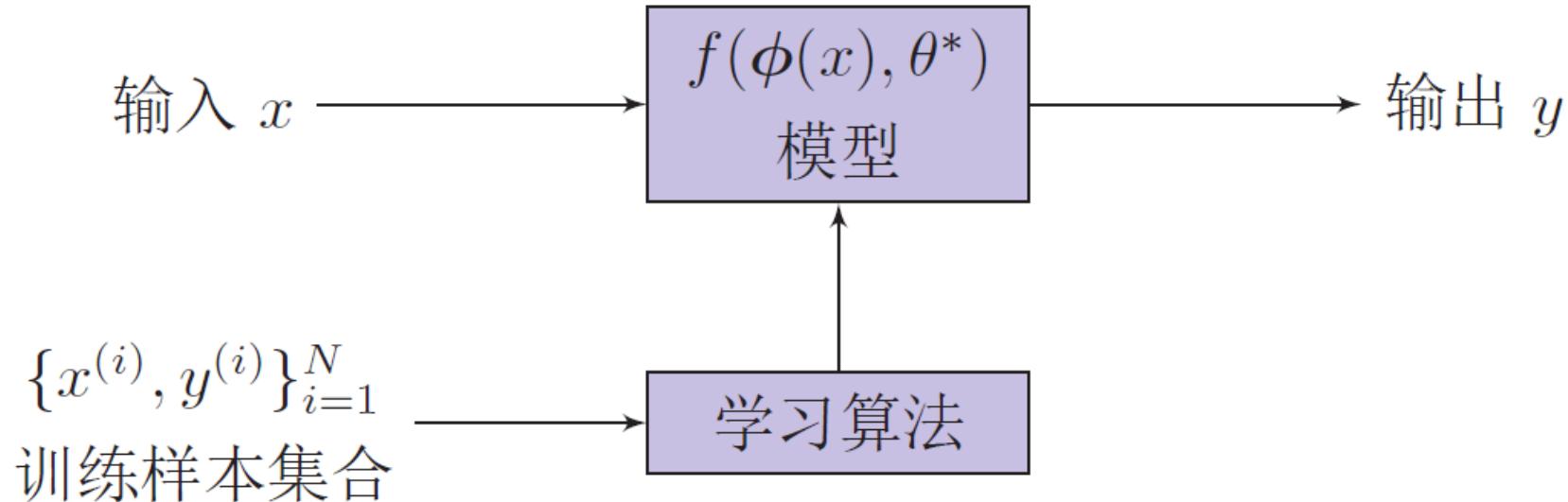


芒果机器学习

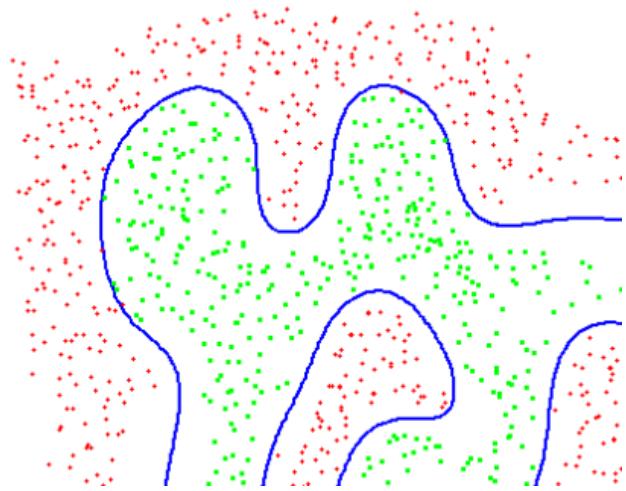
- ▶ 从市场上随机选取的芒果样本（训练数据），列出每个芒果的所有特征：
 - ▶ 如颜色，大小，形状，产地，品牌
- ▶ 以及芒果质量（输出变量）：
 - ▶ 甜蜜，多汁，成熟度。
- ▶ 设计一个学习算法来学习芒果的特征与输出变量之间的相关性模型。
- ▶ 下次从市场上买芒果时，可以根据芒果（测试数据）的特征，使用前面计算的模型来预测芒果的质量。

什么是机器学习？

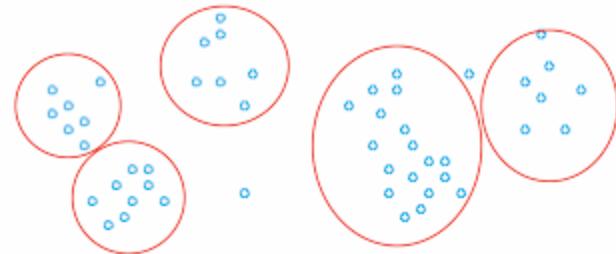
- ▶ 机器学习：从数据中获得决策（预测）函数
- ▶ 使得机器可以根据数据进行自动学习，通过算法使得机器能从大量历史数据中学习规律从而对新的样本做决策。



常见的机器学习问题



分类



聚类



机器学习概览

► 训练数据: $\{\mathbf{x}^{(i)}, y^{(i)}\}, i \in [1, N]$

► 模型:

► 线性方法: $y = f(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} + b$

► 广义线性方法: $y = f(\mathbf{x}) = \boldsymbol{\theta}^T \varphi(\mathbf{x}) + b$

► 非线性方法: 神经网络

► 优化

► 经验风险

$$\triangleright R(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N L(y^{(i)}, f(\mathbf{x}^{(i)}))$$

► 结构风险:

$$\triangleright R(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2$$



常见的机器学习类型

	监督学习	无监督学习	强化学习
输入	有标签数据	无标签数据	决策过程
反馈方式	直接反馈	无反馈	奖励
目标	分类、预测	发现隐藏结构	动作



机器学习的几个关键点



泛化错误

- ▶ 机器学习是从有限的观测数据中学习（或“猜测”）出具有**一般性**的规律，并可以将总结出来的规律推广应用到未观测样本上。
- ▶ 期望风险

$$\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} [\mathcal{L}(f(\mathbf{x}), y)],$$

- ▶ 经验风险

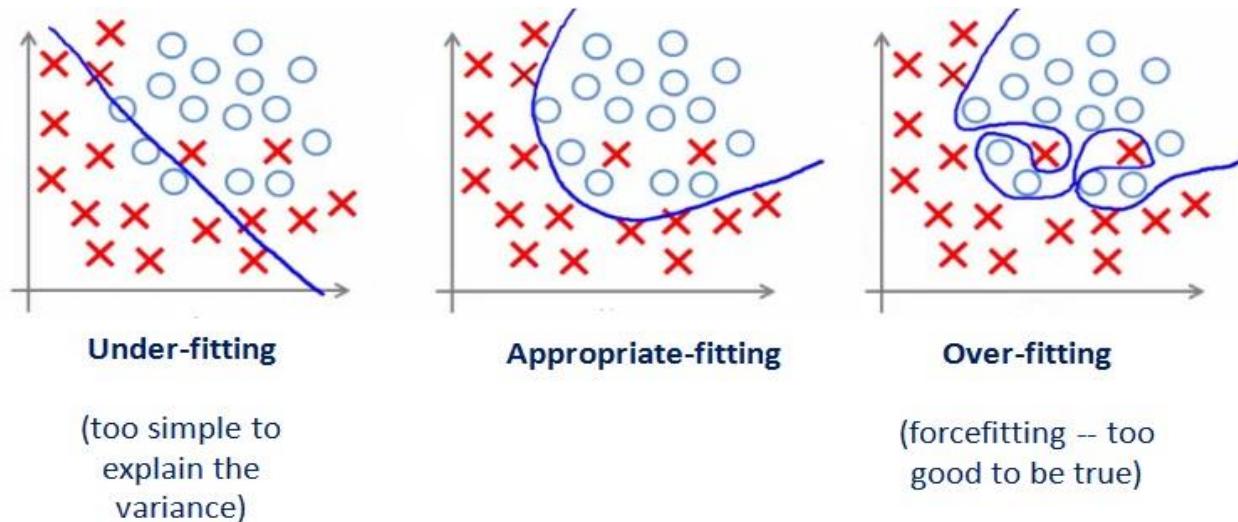
$$\hat{\mathcal{R}}_{\mathcal{D}}(f) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}^{(i)}), y^{(i)}),$$

- ▶ 泛化错误

$$G = \mathcal{R}(f) - \hat{\mathcal{R}}_{\mathcal{D}}(f)$$

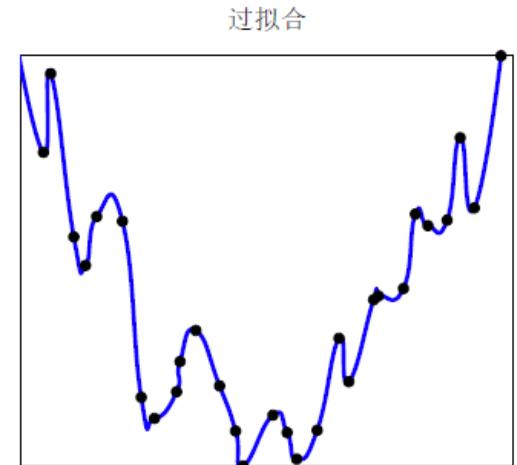
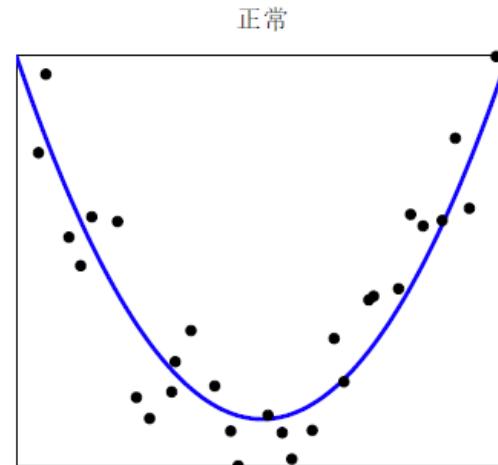
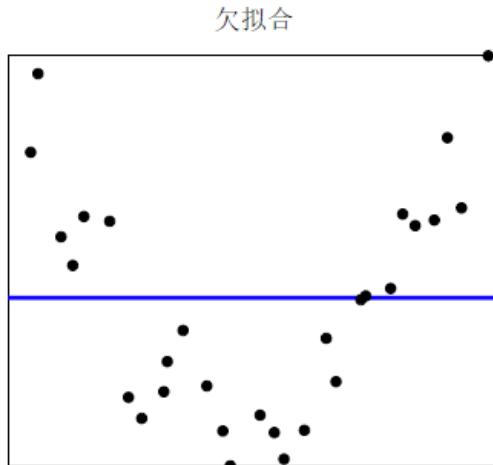
过拟合

- ▶ 过拟合：经验风险最小化原则很容易导致模型在训练集上错误率很低，但是在未知数据上错误率很高。
- ▶ 过拟合问题往往是由于训练数据少和噪声等原因造成的。



过拟合

▶ 回归问题





损失函数

► 0-1 损失函数

$$\mathcal{L}(y, f(x, \theta)) = \begin{cases} 0 & \text{if } y = f(x, \theta) \\ 1 & \text{if } y \neq f(x, \theta) \end{cases}$$

► 平方损失函数

$$\mathcal{L}(y, \hat{y}) = (y - f(x, \theta))^2$$



交叉熵损失函数

▶ 负对数似然损失函数

$$\mathcal{L}(y, f(x, \theta)) = - \sum_{i=1}^C y_i \log f_i(x, \theta).$$

▶ 对于一个三类分类问题，类别为[0,0,1]，预测的类别概率为[0.3,0.3,0.4]，则

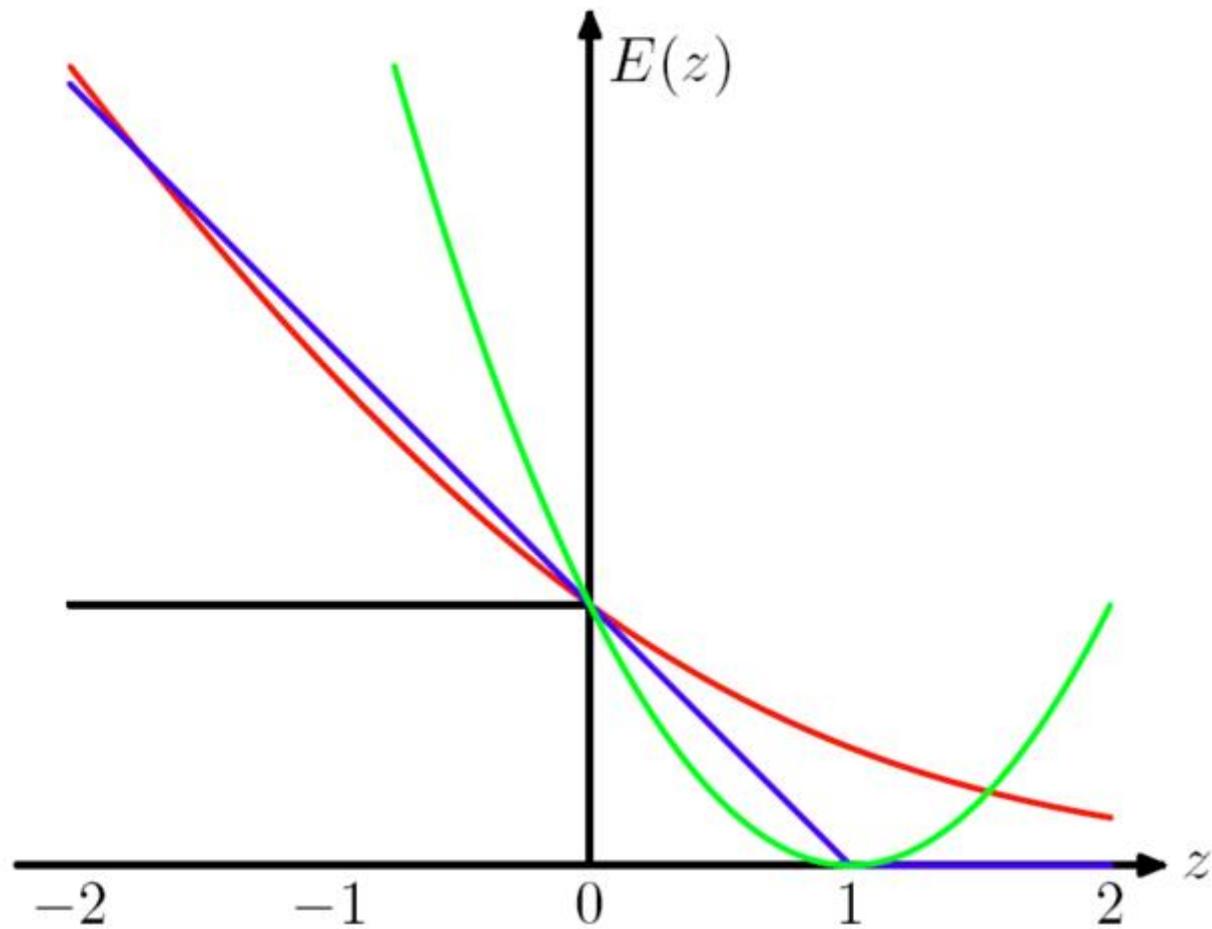
Ex:

Computed (\hat{y})	Targets (y)
[0.3, 0.3, 0.4]	[0, 0, 1]

$$\begin{aligned}\mathcal{L}(\theta) &= -(0 \times \log(0.3) + 0 \times \log(0.3) + 1 \times \log(0.4)) \\ &= -\log(0.4).\end{aligned}$$

损失函数

<http://www.cs.cmu.edu/~yandongl/loss.html>





参数学习

► 风险函数最小化

- 在选择合适的风险函数后，我们寻找一个参数 θ^* ，使得风险函数最小化。

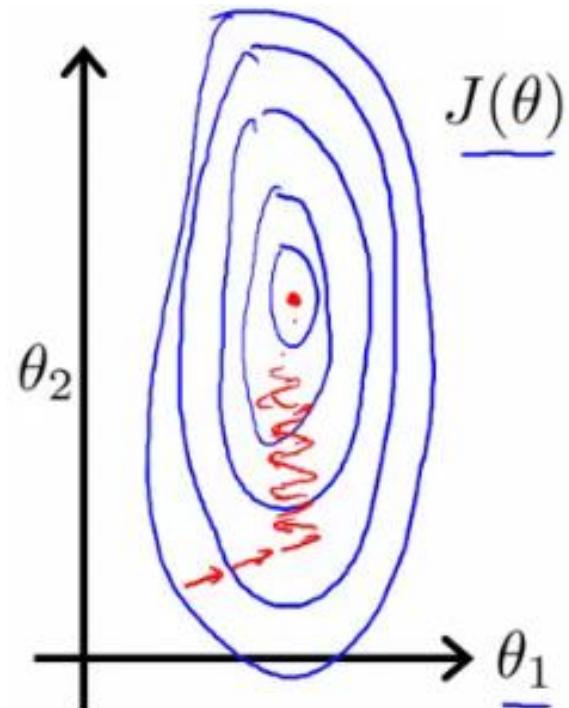
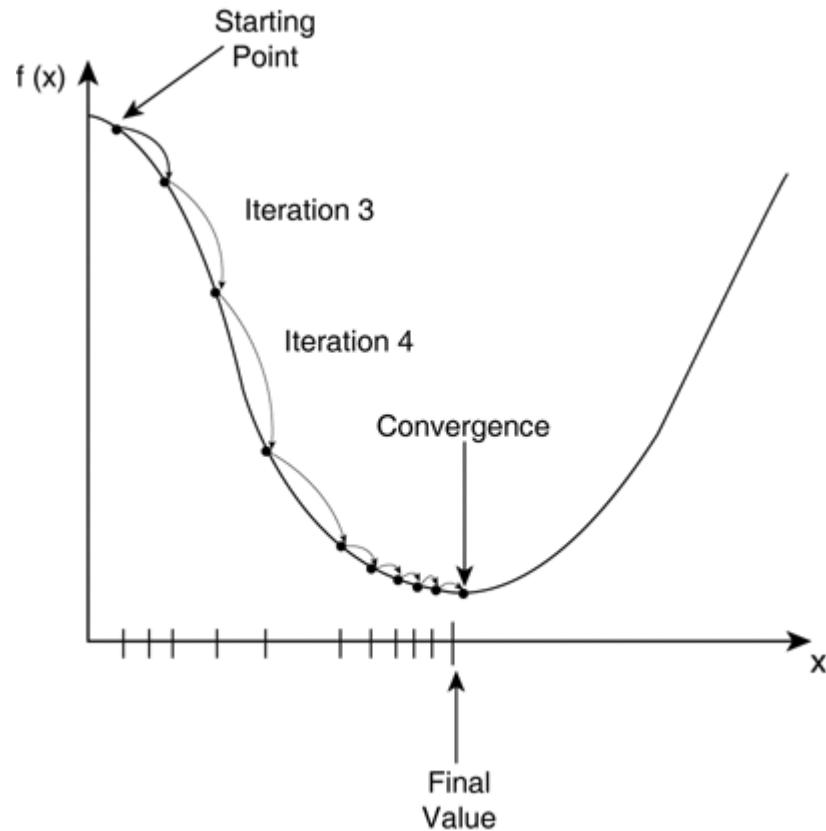
$$\theta^* = \arg \min_{\theta} \mathcal{R}(\theta)$$

$$= \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^{(i)}, f(x^{(i)}, \theta)).$$

► 机器学习问题转化成为一个最优化问题

优化

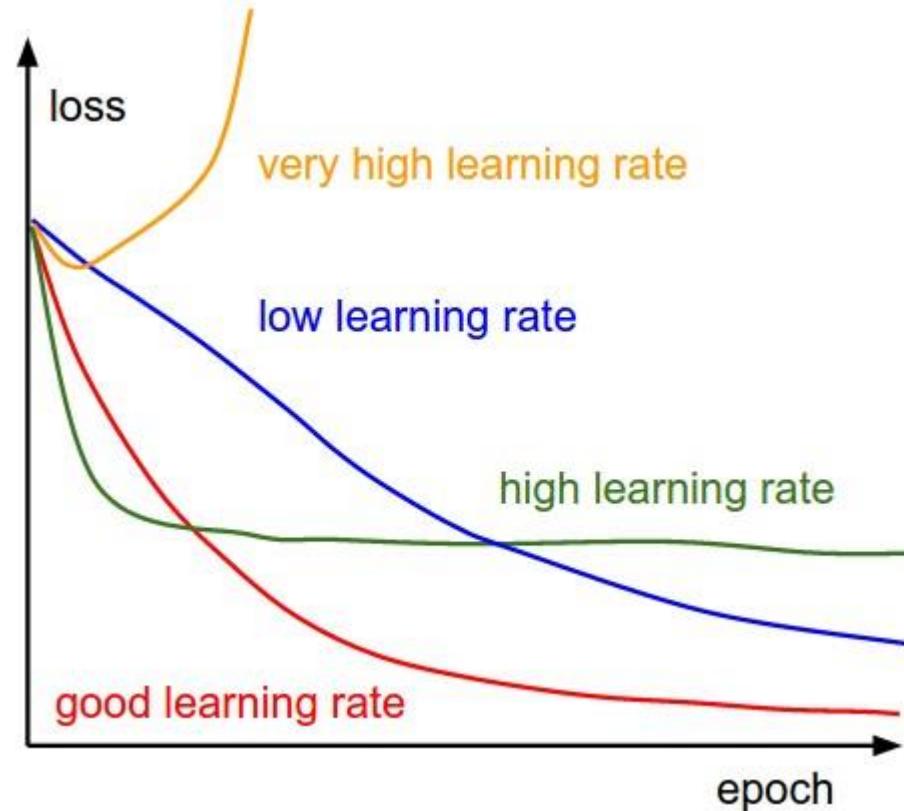
▶ 梯度下降法



批量梯度下降法

$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \\ &= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^N \frac{\partial \mathcal{L}(\theta_t; x^{(i)}, y^{(i)})}{\partial \theta}.\end{aligned}$$

搜索步长 α 中也叫作学习率
(Learning Rate)





随机梯度下降法

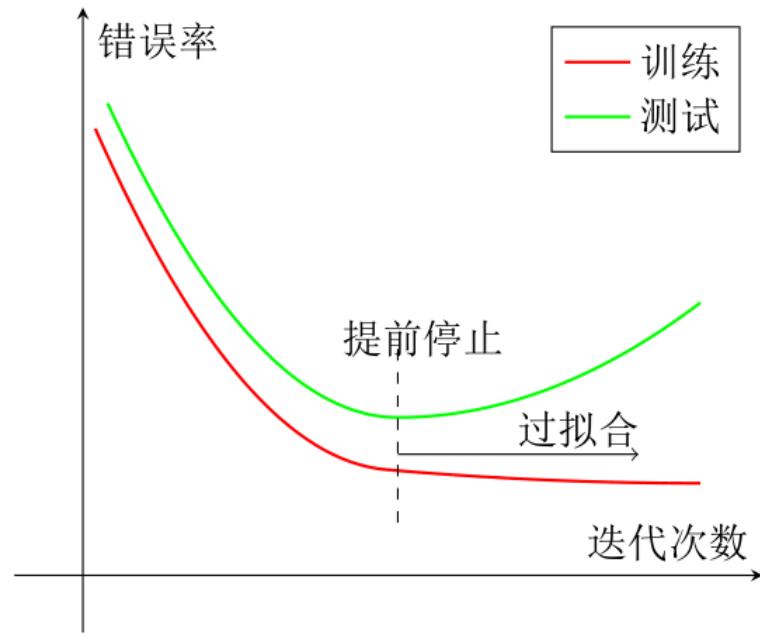
- ▶ 随机梯度下降法 (Stochastic Gradient Descent, SGD) 也叫增量梯度下降，每个样本都进行更新

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\theta_t; x^{(t)}, y^{(t)})}{\partial \theta},$$

- ▶ 小批量 (Mini-Batch) 随机梯度下降法

提前停止

- ▶ 我们使用一个验证集（Validation Dataset）来测试每一次迭代的参数在验证集上是否最优。如果在验证集上的错误率不再下降，就停止迭代。





学习率设置

► 动量法

$$\theta_t = \theta_{t-1} + (\rho \nabla \theta_{t-1} - \lambda g_t)$$

第t步的梯度

$$\nabla \theta_t = \theta_t - \theta_{t-1}$$

- AdaGrad
- AdaDelta
- AdaM

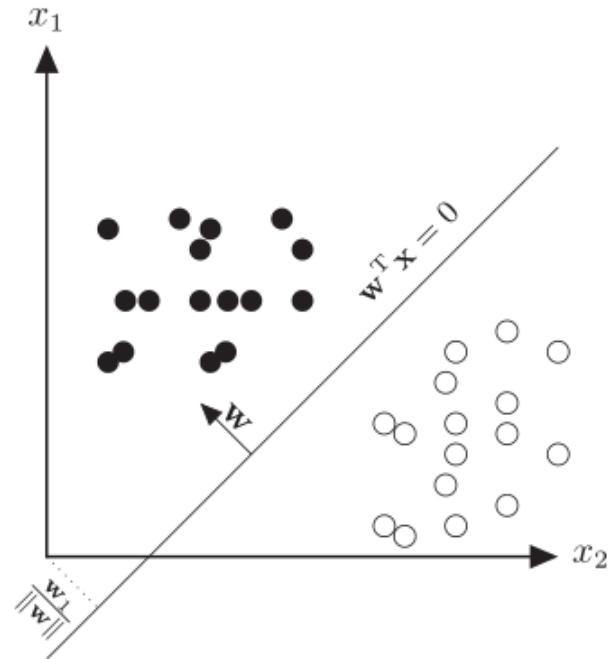


例子：Logistic回归和Softmax回归

Logistic回归

► 线性分类器

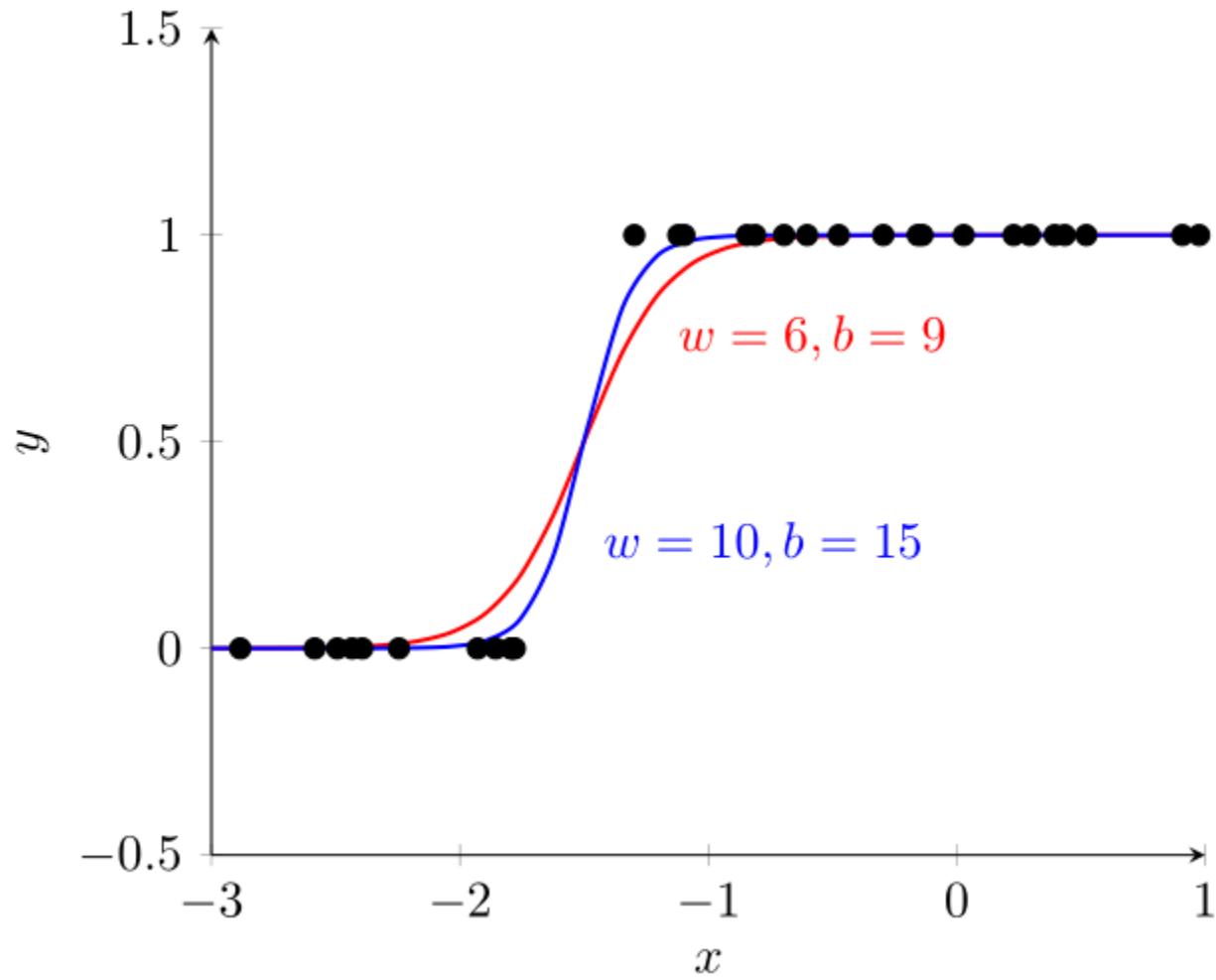
$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} > 0 \\ 0 & \text{if } \mathbf{w}^T \mathbf{x} \leq 0 \end{cases}$$



► 目标类别 $y = 1$ 的后验概率为

$$\begin{aligned} P(y = 1 | \mathbf{x}) &= \sigma(\mathbf{w}^T \mathbf{x}) \\ &\triangleq \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}, \end{aligned}$$

Logistic回归





梯度下降

- ▶ 交叉熵损失函数，模型在训练集的风险函数为

$$\mathcal{R}(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N \left(y^{(i)} \log \left(\sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) \right) + (1 - y^{(i)}) \log \left(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) \right) \right).$$

- ▶ 梯度为

$$\frac{\partial \mathcal{R}(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N} \sum_{i=1}^N \left(\mathbf{x}^{(i)} \cdot \left(\sigma(\mathbf{w}^\top \mathbf{x}^{(i)}) - y^{(i)} \right) \right)$$



Softmax回归

► Softmax回归是logistic回归的多类推广。

$$\hat{y} = \arg \max_{c=1}^C \mathbf{w}_c^\top \mathbf{x}$$

► 利用softmax函数，我们定义目标类别 $y = c$ 的后验概率为：

$$\begin{aligned} P(y = c | \mathbf{x}) &= \text{softmax}(\mathbf{w}_c^\top \mathbf{x}) \\ &= \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{\sum_{i=1}^C \exp(\mathbf{w}_i^\top \mathbf{x})}. \end{aligned}$$



梯度下降

- ▶ 交叉熵损失函数，模型在训练集的风险函数为

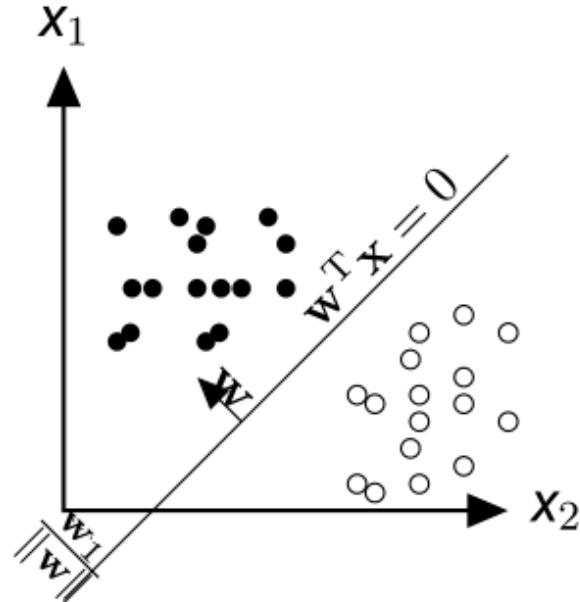
$$\mathcal{R}(W) = -\frac{1}{N} \sum_{i=1}^N (\mathbf{y}^{(i)})^\top \log \left(\text{softmax}(W^\top \mathbf{x}^{(i)}) \right).$$

- ▶ 梯度为

$$\frac{\partial \mathcal{R}(W)}{\partial W} = -\frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right)^\top.$$

线性分类器小结

	损失函数	优化方法
线性回归	平方误差	最小二乘、梯度下降
Logistic 回归	交叉熵	梯度下降
感知器	0-1 损失	$\mathbf{w}_{k+1} = \mathbf{w}_k + y_i \mathbf{x}_i$
支持向量机	Hinge 损失	SMO 等

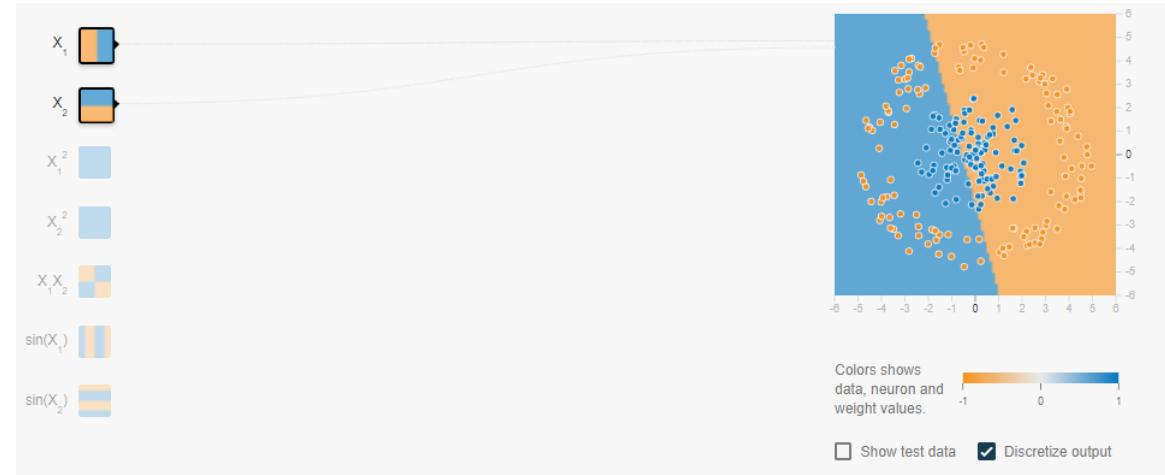


特征

<http://playground.tensorflow.org>



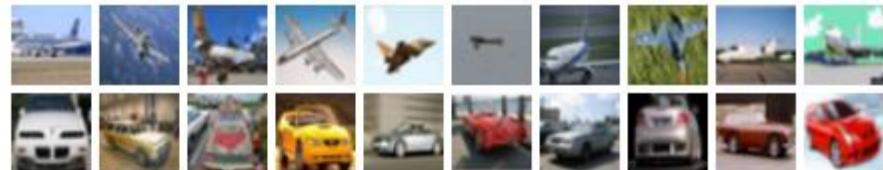
如何处理非线性可分问题?



数据集：CIFAR-10

- ▶ 60000张32x32色彩图像，共10类，每类6000张图像。

airplane



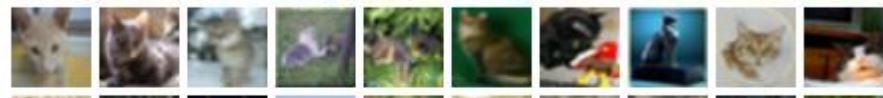
automobile



bird



cat



deer



dog



frog



horse



ship

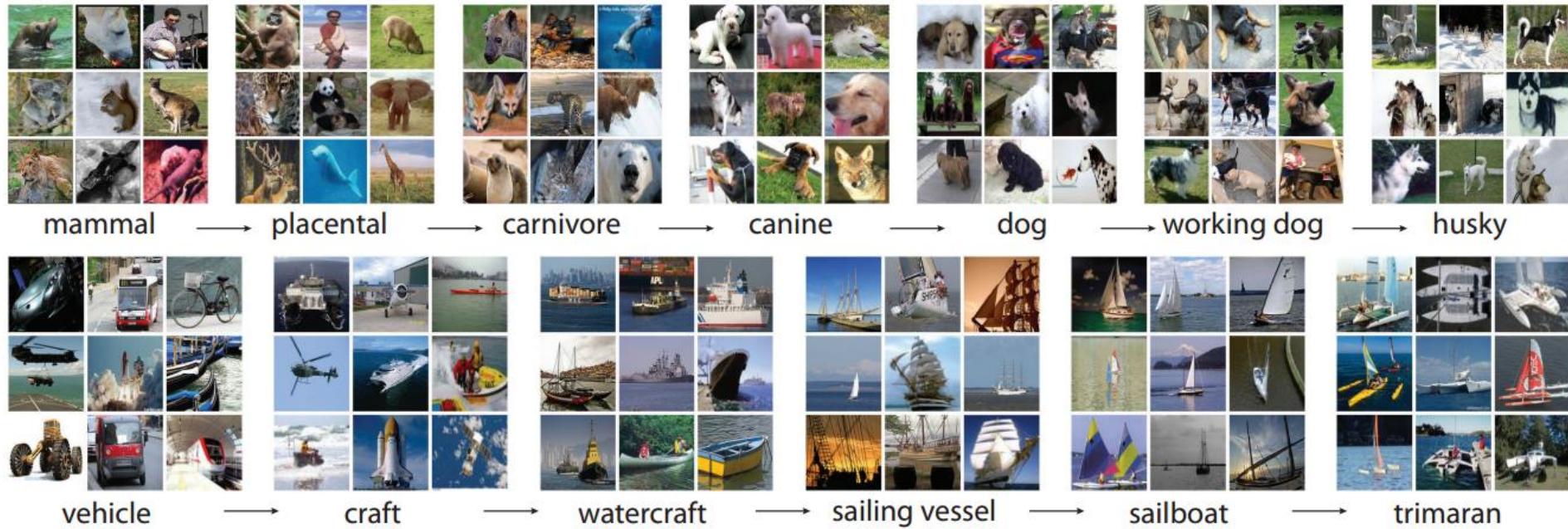


truck



数据集：ImageNet

► 14,197,122 images, 21841 synsets



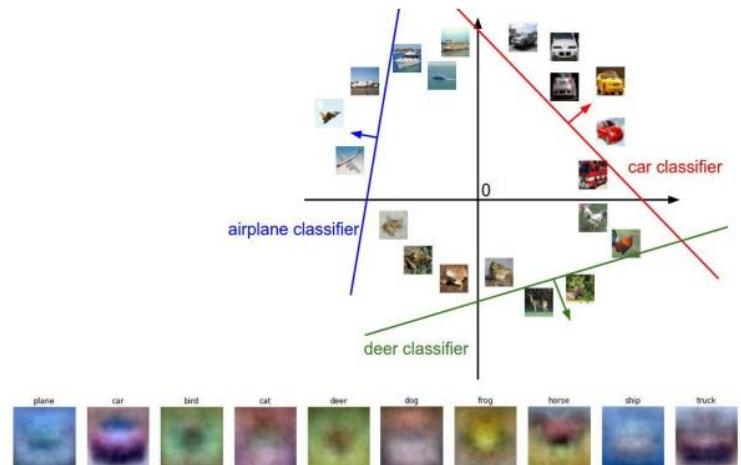
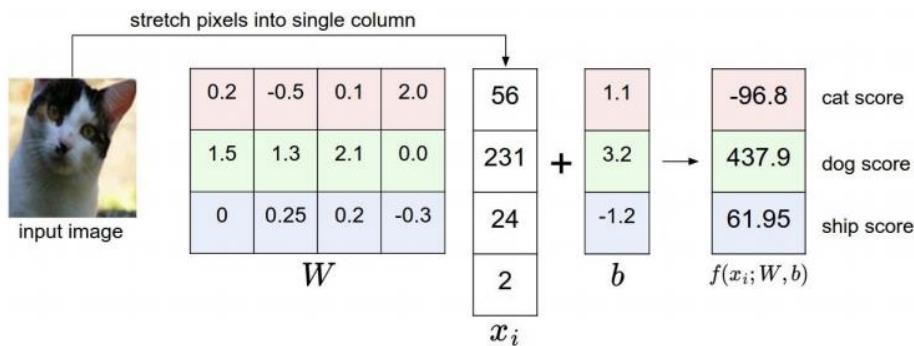
应用：图像分类



[32x32x3]
array of numbers 0...1
(3072 numbers total)

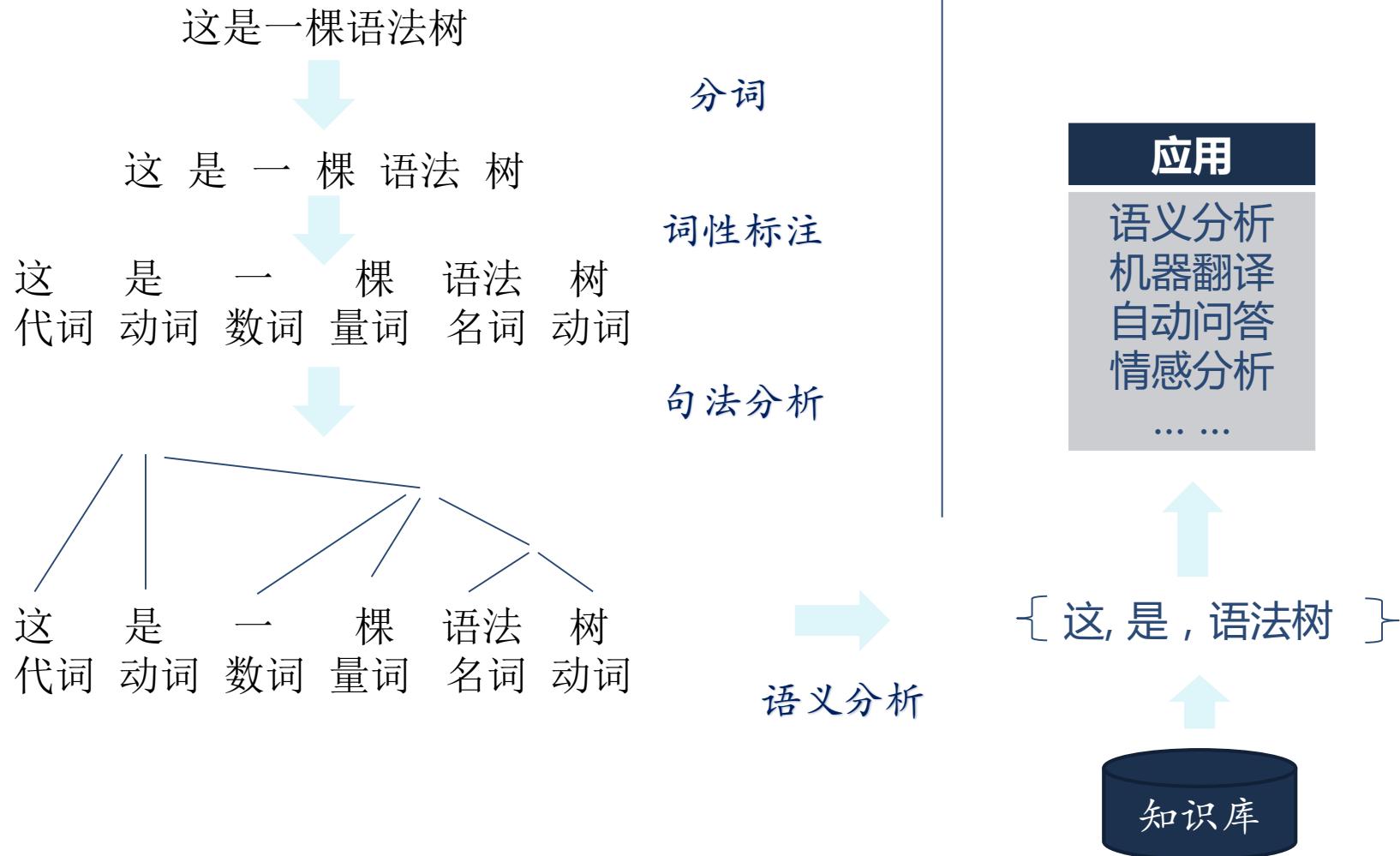
image parameters
 $f(\mathbf{x}, \mathbf{W})$

10 numbers, indicating class scores





理想中的自然语言处理流程



实际流程 : End-to-End

我喜欢读书。

我讨厌读书。

分类模型



模型表示

特征抽取

参数学习

解码算法

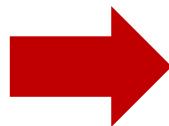
文本情感分类

文本情感分类

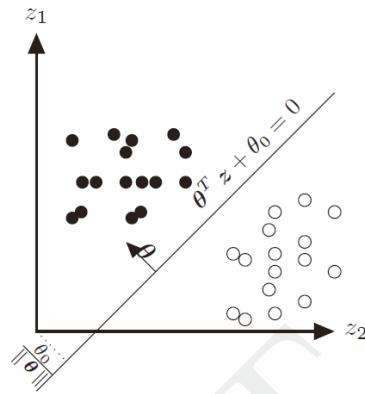
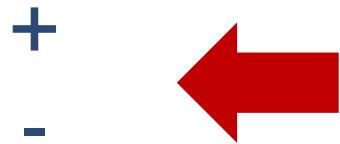
根据文本内容来判断文本的相应类别

D_1 : “我喜欢读书”

D_2 : “我讨厌读书”



	我	喜欢	讨厌	读书
D_1	1	1	0	1
D_2	1	0	1	1

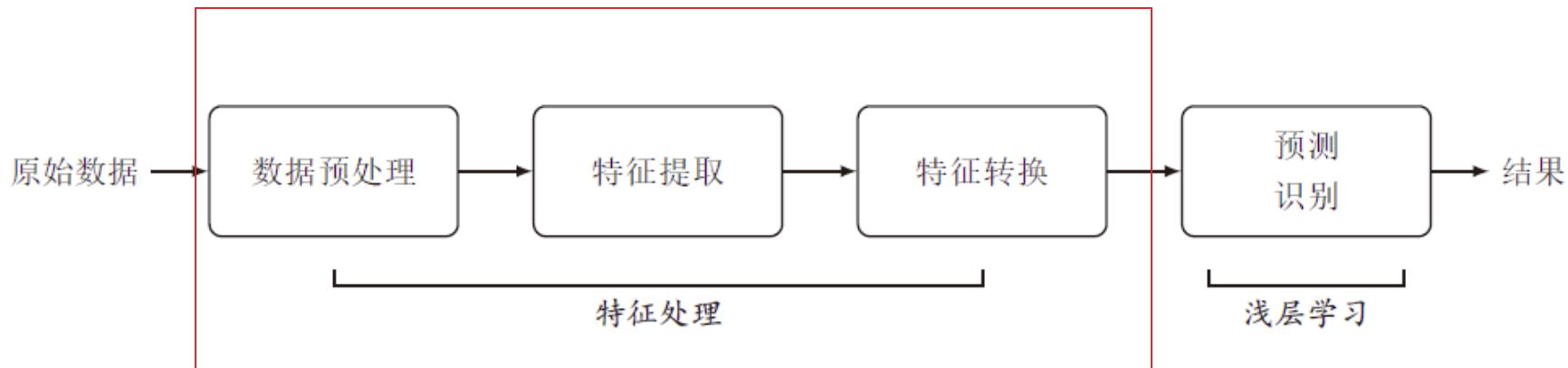




特征工程问题

► 在实际应用中，**特征往往比分类器更重要**

- 预处理：经过数据的预处理，如去除噪声等。比如在文本分类中，去除停用词等。
- 特征提取：从原始数据中提取一些有效的特征。比如在图像分类中，提取边缘、尺度不变特征变换特征等。
- 特征转换：对特征进行一定的加工，比如降维和升维。降维包括
 - 特征抽取（Feature Extraction）：PCA、LDA
 - 特征选择（Feature Selection）：互信息、TF-IDF



背后的难点之一：语义鸿沟

- ▶ 底层特征 VS 高层语义
- ▶ 人们对文本、图像的理解无法从字符串或者图像的底层特征直接获得

表示学习



床前明月光，
疑是地上霜。
举头望明月，
低头思故乡。

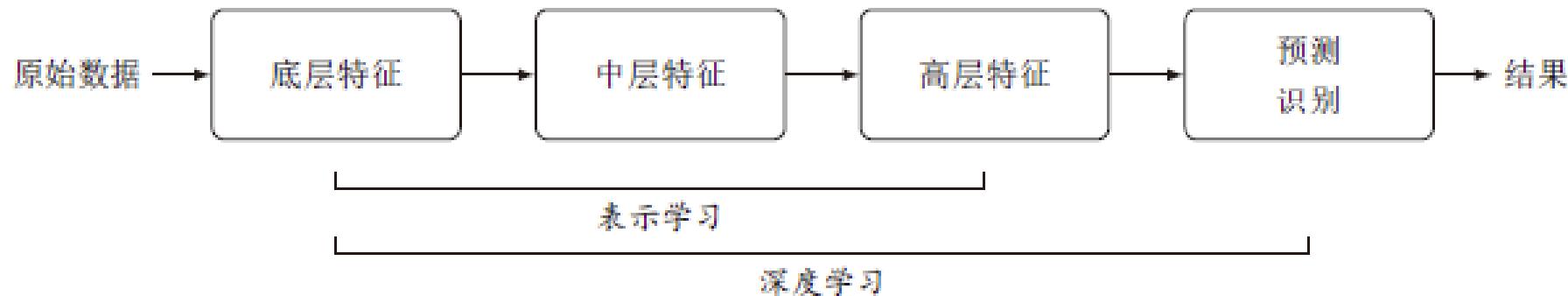


深度学习



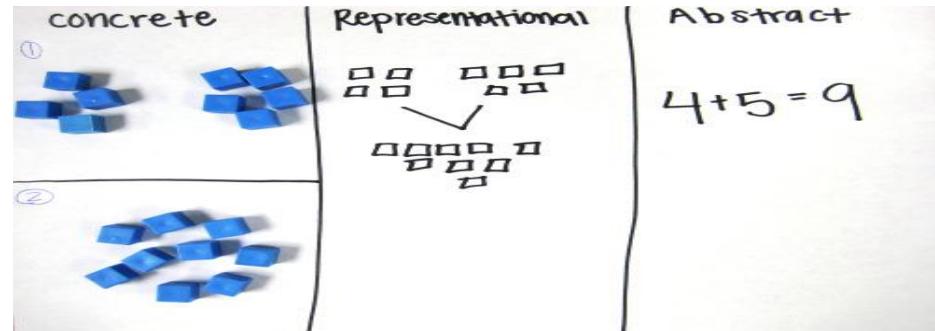
深度学习

- ▶ 深度学习=表示学习+浅层学习
- ▶ 难点： 贡献度分配问题



表示学习与深度学习

- 一个好的表示学习策略必须具备一定的深度
 - 特征重用
 - 指数级的表示能力
 - 抽象表示与不变性
 - 抽象表示需要多步的构造

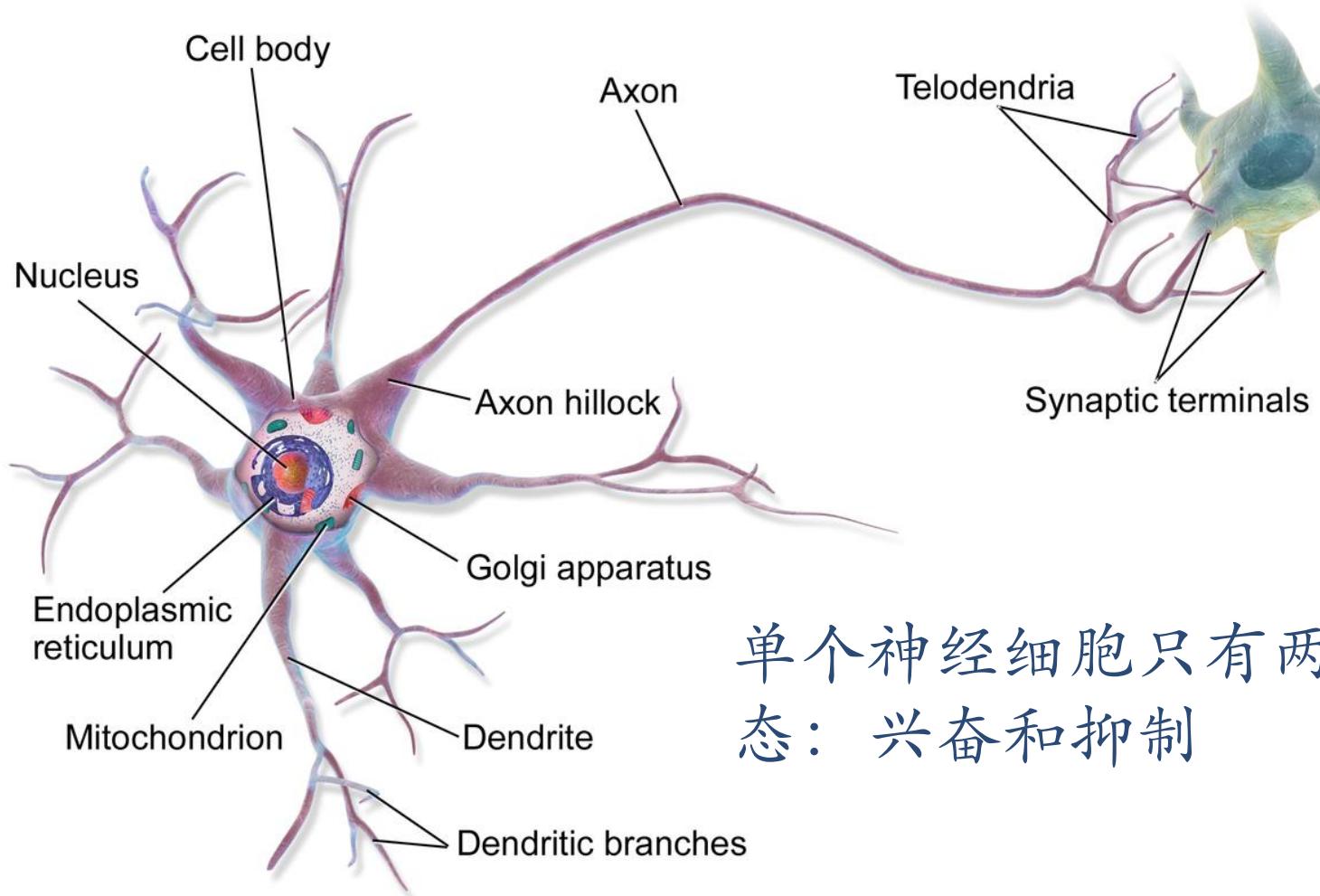


<https://mathteachingstrategies.wordpress.com/2008/11/24/concrete-and-abstract-representations-using-mathematical-tools/>



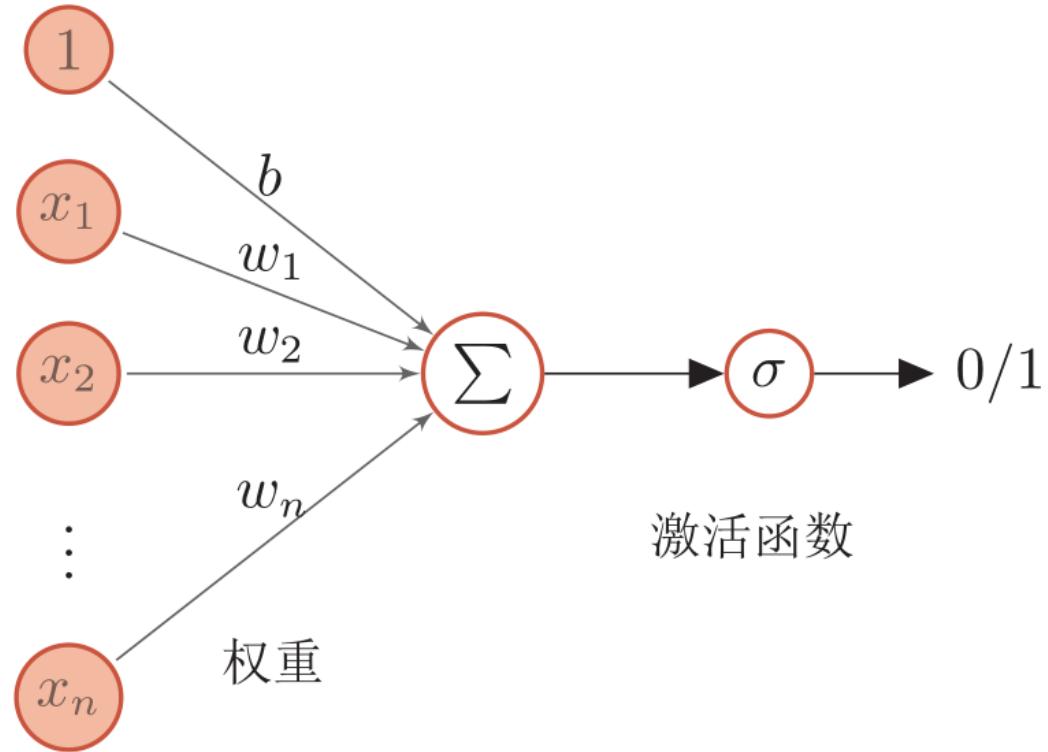
神经网络

生物神经元



单个神经细胞只有两种状态：兴奋和抑制

人工神经元



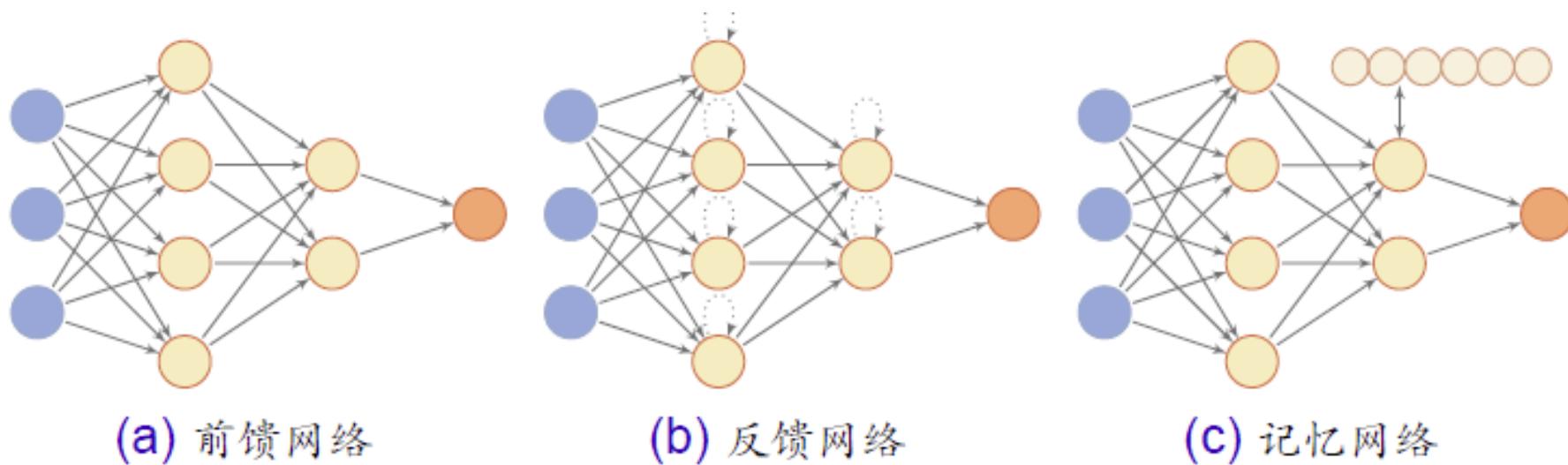


人工神经网络

- ▶ 人工神经网络主要由大量的神经元以及它们之间的有向连接构成。因此考虑三方面：
- ▶ 神经元的激活规则
 - ▶ 主要是指神经元输入到输出之间的映射关系，一般为非线性函数。
- ▶ 网络的拓扑结构
 - ▶ 不同神经元之间的连接关系。
- ▶ 学习算法
 - ▶ 通过训练数据来学习神经网络的参数。

人工神经网络

- ▶ 人工神经网络由神经元模型构成，这种由许多神经元组成的信息处理网络具有并行分布结构。





深度学习与神经网络

- ▶ 如果解决贡献度分配问题?
- ▶ 神经网络!

- ▶ 深度学习天然不是神经网络，但神经网络天然是深度学习！



深层神经网络

▶ 难点

- ▶ 参数过多，影响训练
- ▶ 非凸优化问题：即存在局部最优而非全局最优解，影响迭代
- ▶ 下层参数比较难调
- ▶ 参数解释起来比较困难

▶ 需求

- ▶ 计算资源要大
- ▶ 数据要多
- ▶ 算法效率要好：即收敛快



神经网络历史

► 第一阶段：模型提出 1943年～1969年

- 1943年，心理学家Warren McCulloch和数学家Walter Pitts最早描述了一种理想化的人工神经网络，并构建了一种基于简单逻辑运算的计算机制，称为MP模型。
- 阿兰·图灵在1948年的论文中描述了一种“B型图灵机”。
- Rosenblatt [1958]最早提出可以模拟人类感知能力的神经网络模型，并称之为感知器（Perceptron）。



神经网络历史

- ▶ 第二阶段：冰河期 1969年～1983年
 - ▶ 神经网络发展的第一个低谷期。
 - ▶ 1969年，Marvin Minsky出版《感知机》一书
 - ▶ 1974年，哈佛大学的Paul Webos发明反向传播算法，但当时未受到应有的重视。



神经网络历史

- ▶ 第三阶段：1983年～1995年
 - ▶ 反向传播算法引起的复兴
 - ▶ 1984年，Geoffrey Hinton提出 Boltzman机模型。
 - ▶ 1986年，David Rumelhart和James McClelland对于联结主义在计算机模拟神经活动中的应用提供了全面的论述，并重新发明了反向传播算法。
 - ▶ 1986年，Geoffrey Hinton等人将引入到多层感知器
 - ▶ 1989年，Yann LeCun在将反向传播算法引入了卷积神经网络，并在手写体数字识别上取得了很大的成功



神经网络历史

- ▶ 第四阶段：流行度降低 1995年～2006年
 - ▶ 支持向量机和其他更简单的方法（例如线性分类器）在机器学习领域的流行度逐渐超过了神经网络。
- ▶ 第五阶段：深度学习的崛起 2006年以后
 - ▶ Algorithm
 - ▶ Business
 - ▶ Computing
 - ▶ Data



深度学习革命

► AI领域

- 语音识别：可以使得词错误率从 $1/4$ 下降到 $1/8$
- 计算机视觉：目标识别、图像分类等
- 自然语言处理：分布式表示、机器翻译、问题回答等
- 信息检索、社会化网络

► 三个Deep：

- Deep Blue
- Deep QA
- Deep Learning



学术机构

- ▶ Toronto 大学
 - ▶ Hinton 75 年 Edinburgh 大学博士
- ▶ NYU
 - ▶ Lecun (Now Facebook) 87 年 Hinton 博士后
- ▶ Montreal 大学
 - ▶ Bengio 91 年 M. Jordan 博士后
- ▶ Stanford 大学
 - ▶ Ng (Now Baidu) 03 年 UC Berkeley 大学 M. Jordan 博士
- ▶ IDSIA
 - ▶ Jürgen Schmidhuber



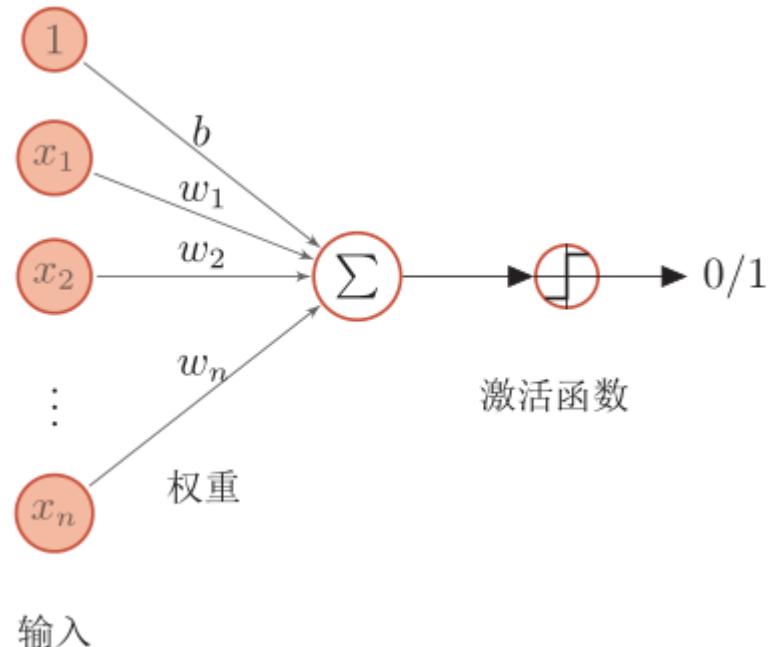


感知器

感知器

- ▶ 模拟生物神经元行为的机器，有与生物神经元相对应的部件，如权重（突触）、偏置（阈值）及激活函数（细胞体），输出为0或1。

$$\hat{y} = \begin{cases} +1 & \text{当 } \mathbf{w}^T \mathbf{x} > 0 \\ -1 & \text{当 } \mathbf{w}^T \mathbf{x} \leq 0 \end{cases},$$





感知器的学习过程

输入: 训练集: $(\mathbf{x}_i, y_i), i = 1, \dots, N$, 迭代次数: T

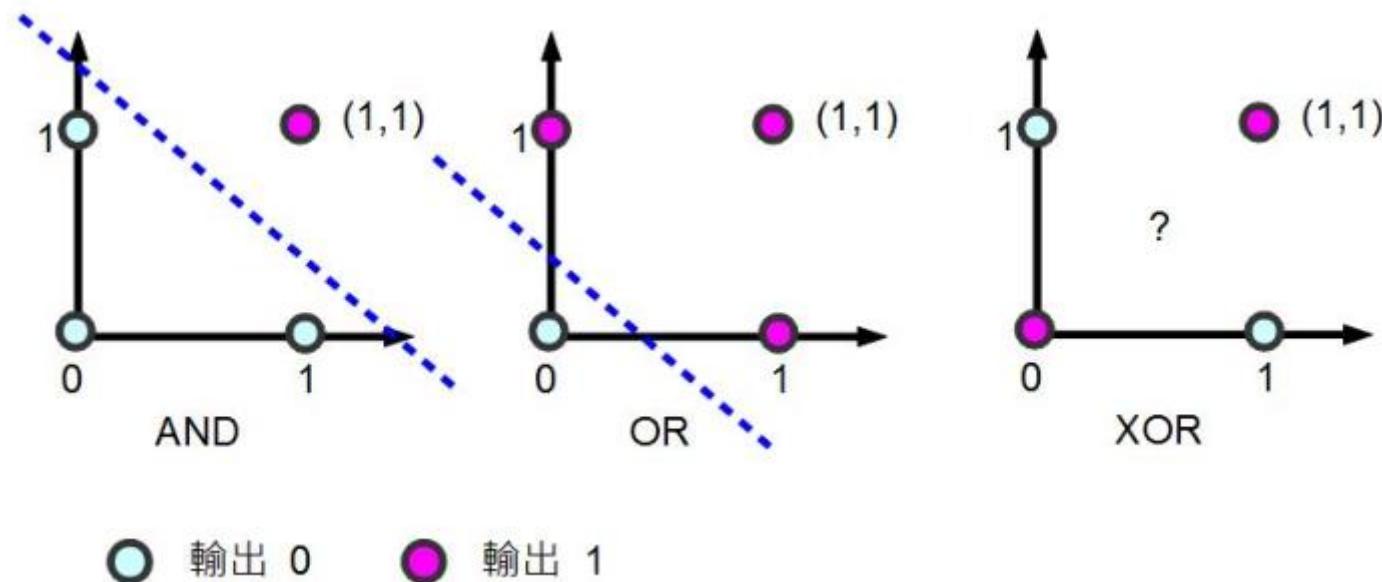
```
1 初始化:  $\mathbf{w}_0 = 0$  ;
2  $k = 0$  ;
3 for  $t = 1 \dots T$  do
4   for  $i = 1 \dots N$  do
5     选取一个样本  $(\mathbf{x}_i, y_i)$ , if  $\mathbf{w}^T(\mathbf{y}_i \mathbf{x}_i) < 0$  then
6        $\mathbf{w}_{k+1} = \mathbf{w}_k + y_i \mathbf{x}_i$ , ;
7        $k = k + 1$ ;
8   end
9 end
10 end
输出:  $\mathbf{w}_k$ 
```

表示分错

对比Logistic回归的更新方式:

$$\frac{\partial \mathcal{R}(W)}{\partial W} = -\frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right)^T.$$

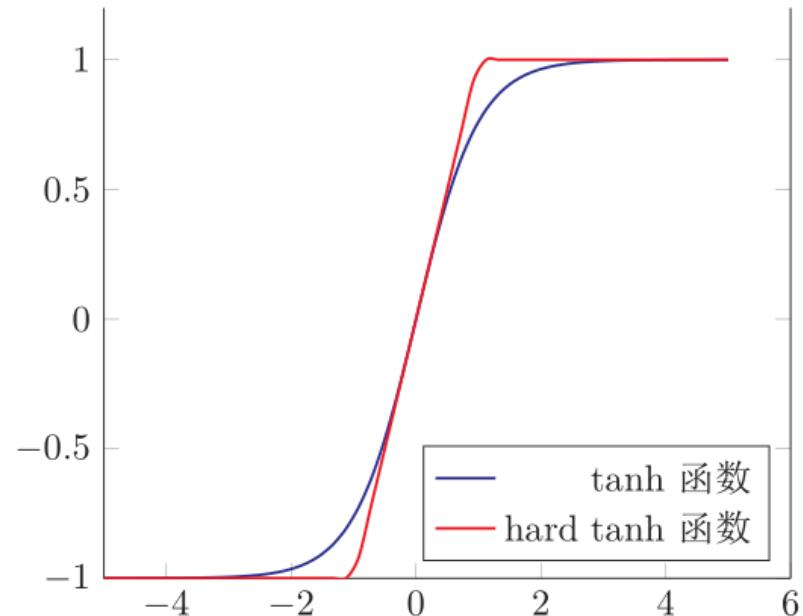
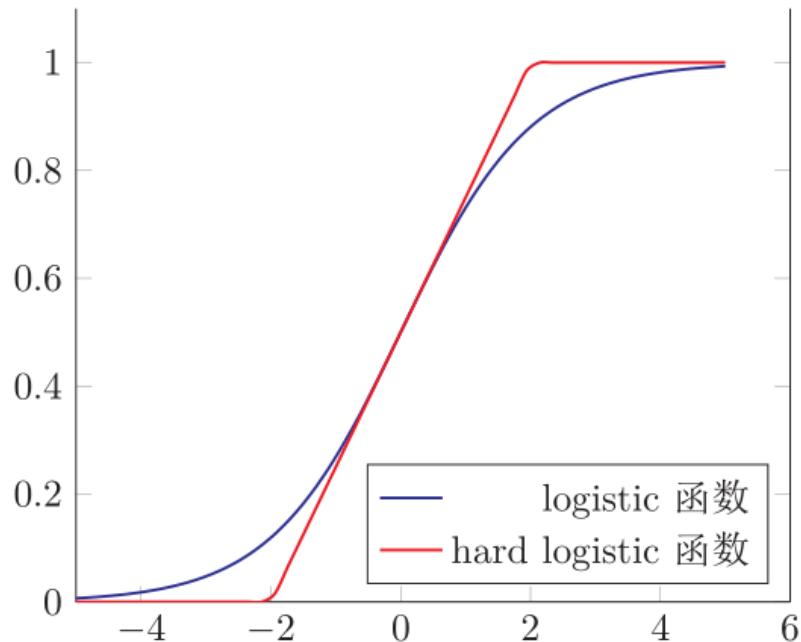
XOR問題





前馈神经网络

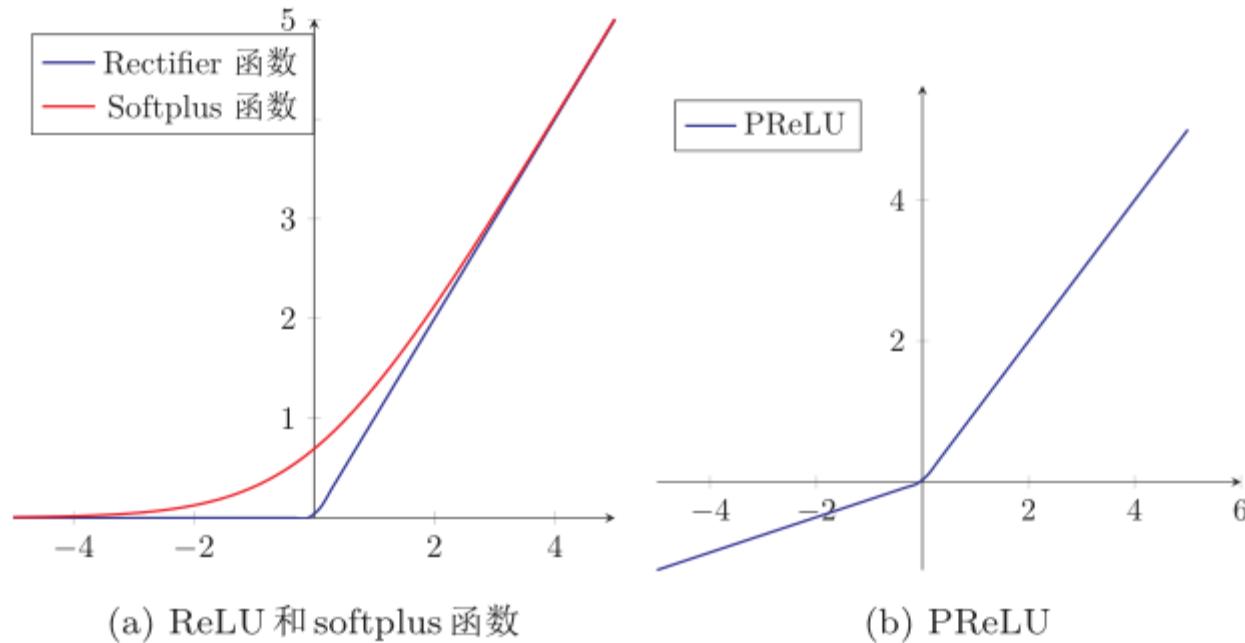
激活函数



$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

激活函数



$$\text{rectifier}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} = \max(0, x).$$

$$\begin{aligned} \text{PReLU}_i(x) &= \begin{cases} x & \text{if } x > 0 \\ a_i x & \text{if } x \leq 0 \end{cases} \\ &= \max(0, x) + a_i \min(0, x), \end{aligned}$$



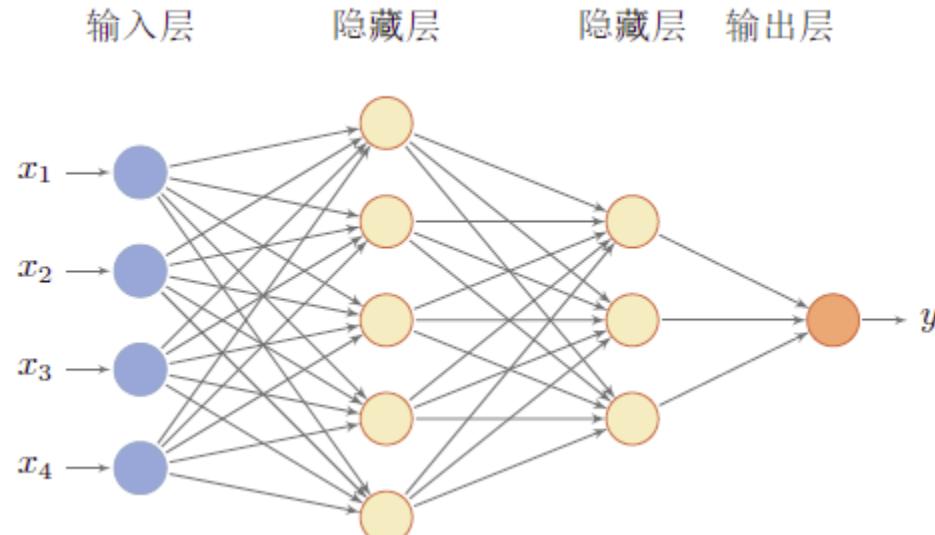
常见激活函数及其导数

激活函数	函数	导数
Logistic 函数	$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh 函数	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus 函数	$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

网络结构

- ▶ 在前馈神经网络中，各神经元分别属于不同的层。整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示

-





前馈网络

► 给定一个前馈神经网络，我们用下面的记号来描述这样网络。

- L : 表示神经网络的层数;
- n^l : 表示第 l 层神经元的个数;
- $f_l(\cdot)$: 表示 l 层神经元的激活函数;
- $W^{(l)} \in \mathbb{R}^{n^l \times n^{l-1}}$: 表示 $l - 1$ 层到第 l 层的权重矩阵;
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n^l}$: 表示 $l - 1$ 层到第 l 层的偏置;
- $\mathbf{z}^{(l)} \in \mathbb{R}^{n^l}$: 表示 l 层神经元的净输入（净活性值）;
- $\mathbf{a}^{(l)} \in \mathbb{R}^{n^l}$: 表示 l 层神经元的输出（活性值）。



前馈网络

► 前馈神经网络通过下面公式进行信息传播。

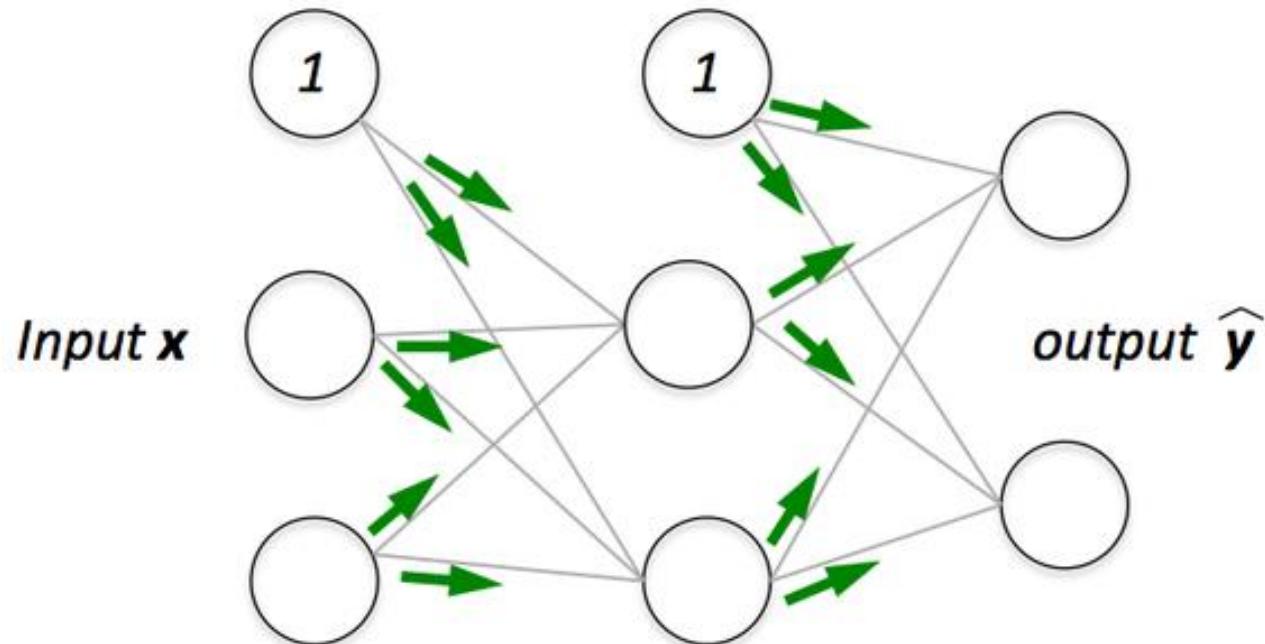
$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)})$$

► 前馈计算：

$$\mathbf{x} = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \cdots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)} = f(\mathbf{x}; W, \mathbf{b}).$$

前馈计算





应用到机器学习

► 对于多类分类问题

- 如果使用softmax回归分类器，相当于网络最后一层设置C个神经元，其输出经过softmax函数进行归一化后可以作为每个类的后验概率。
- 采用交叉熵损失函数，对于样本 (\mathbf{x}, \mathbf{y}) ，其损失函数为

$$\mathcal{L}(\mathbf{y}, f(\mathbf{x}; \mathbf{W}, \mathbf{b})) = -\mathbf{y}^\top \log \hat{\mathbf{y}}$$



反向传播算法

► 目标函数

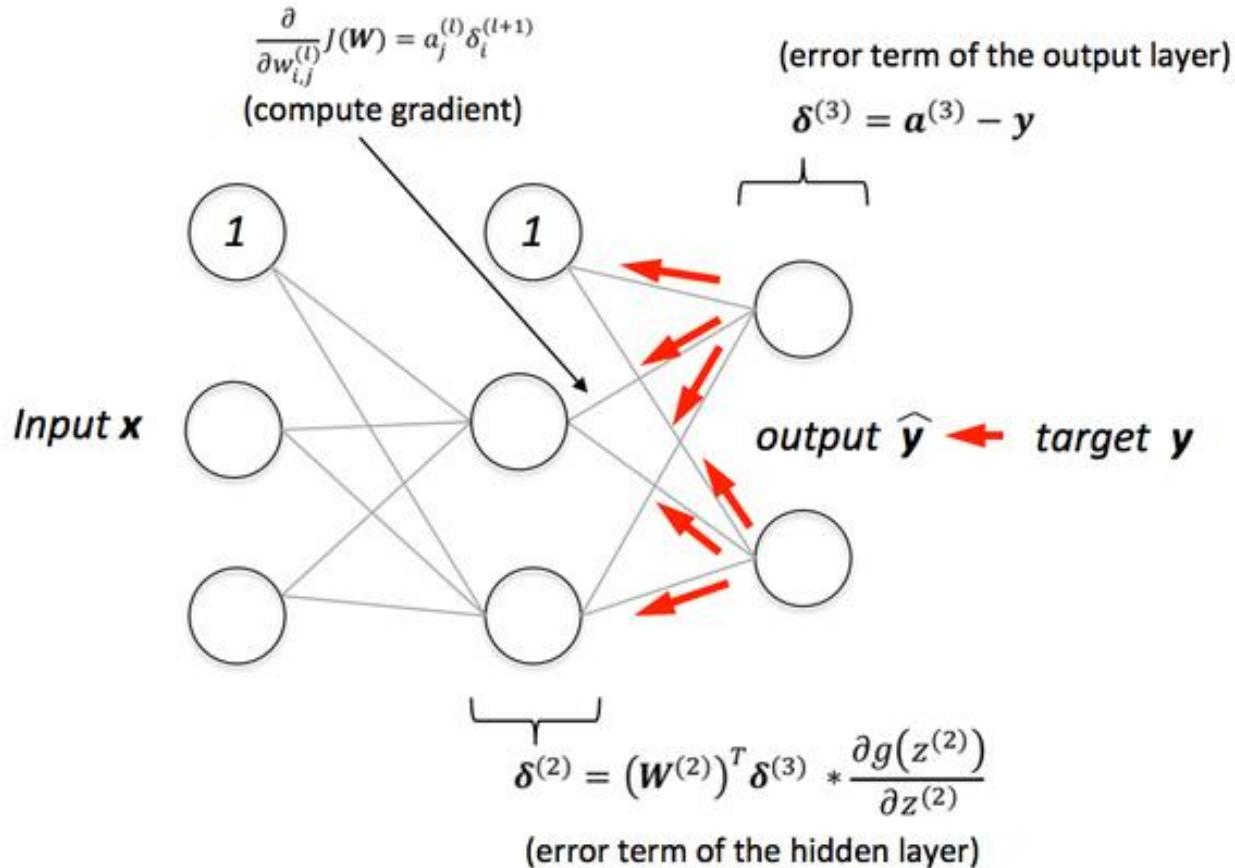
$$\begin{aligned}\mathcal{R}(W, \mathbf{b}) &= \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^{(i)}, f(\mathbf{x}^{(i)}; W, \mathbf{b})) + \frac{1}{2} \lambda \|W\|_F^2, \\ &= \frac{1}{N} \sum_{i=1}^N \mathcal{L}(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)}) + \frac{1}{2} \lambda \|W\|_F^2,\end{aligned}$$

正则化而不包含偏置

► 梯度

$$\begin{aligned}W^{(l)} &= W^{(l)} - \alpha \frac{\partial \mathcal{R}(W, \mathbf{b})}{\partial W^{(l)}}, \\ &= W^{(l)} - \alpha \left(\frac{1}{N} \sum_{i=1}^N \left(\frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial W^{(l)}} \right) - \lambda W^{(l)} \right), \\ \mathbf{b}^{(l)} &= \mathbf{b}^{(l)} - \alpha \frac{\partial \mathcal{R}(W, \mathbf{b})}{\partial \mathbf{b}^{(l)}}, \\ &= \mathbf{b}^{(l)} - \alpha \left(\frac{1}{N} \sum_{i=1}^N \frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}} \right),\end{aligned}$$

梯度链式法则





反向传播算法

► 根据链式法则，

$$\frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)}{\partial W_{ij}^{(l)}} = \left(\frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l)}} \right)^\top \frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}}.$$

- 右边第一项为目标函数关于第1层的神经元 $\mathbf{z}^{(1)}$ 的偏导数，我们称为误差项 $\delta^{(1)}$ 。
- 误差项反映了最终的输出对第1层的神经元对最终误差的敏感程度。



误差项

► 根据链式法则, $\mathbf{z}^{(l+1)} = W^{(l+1)} \mathbf{a}^{(l)}, \mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$

$$\begin{aligned}\delta^{(l)} &\triangleq \frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l)}} \\&= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l+1)}} \\&= \text{diag}(f'_l(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^\top \cdot \delta^{(l+1)} \\&= f'_l(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^\top \delta^{(l+1)}),\end{aligned}$$



反向传播算法

进一步， $\mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)$ 关于第 l 层权重 $W^{(l)}$ 的梯度为

$$\frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^T.$$

同理可得， $\mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)$ 关于第 l 层偏置 $\mathbf{b}^{(l)}$ 的梯度为

$$\frac{\partial \mathcal{L}(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}.$$



反向传播算法

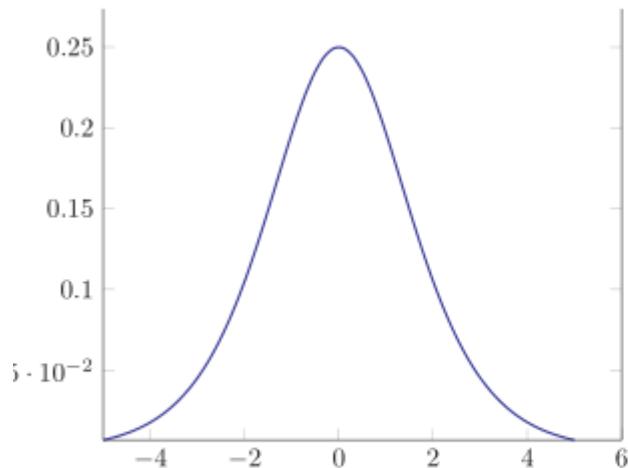
- 在计算出每一层的误差项之后，我们就可以得到每一层参数的梯度。

- 前馈神经网络的训练过程可以分为以下三步
 - 前馈计算每一层的状态和激活值，直到最后一层
 - 反向传播计算每一层的误差项；
 - 计算每一层参数的偏导数，并更新参数。

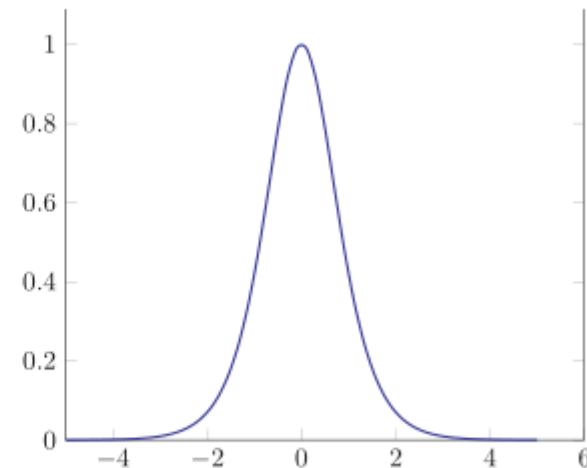
梯度消失问题

► 在神经网络中误差反向传播的迭代公式为

$$\delta^{(l)} = f'_l(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^T \delta^{(l+1)}),$$

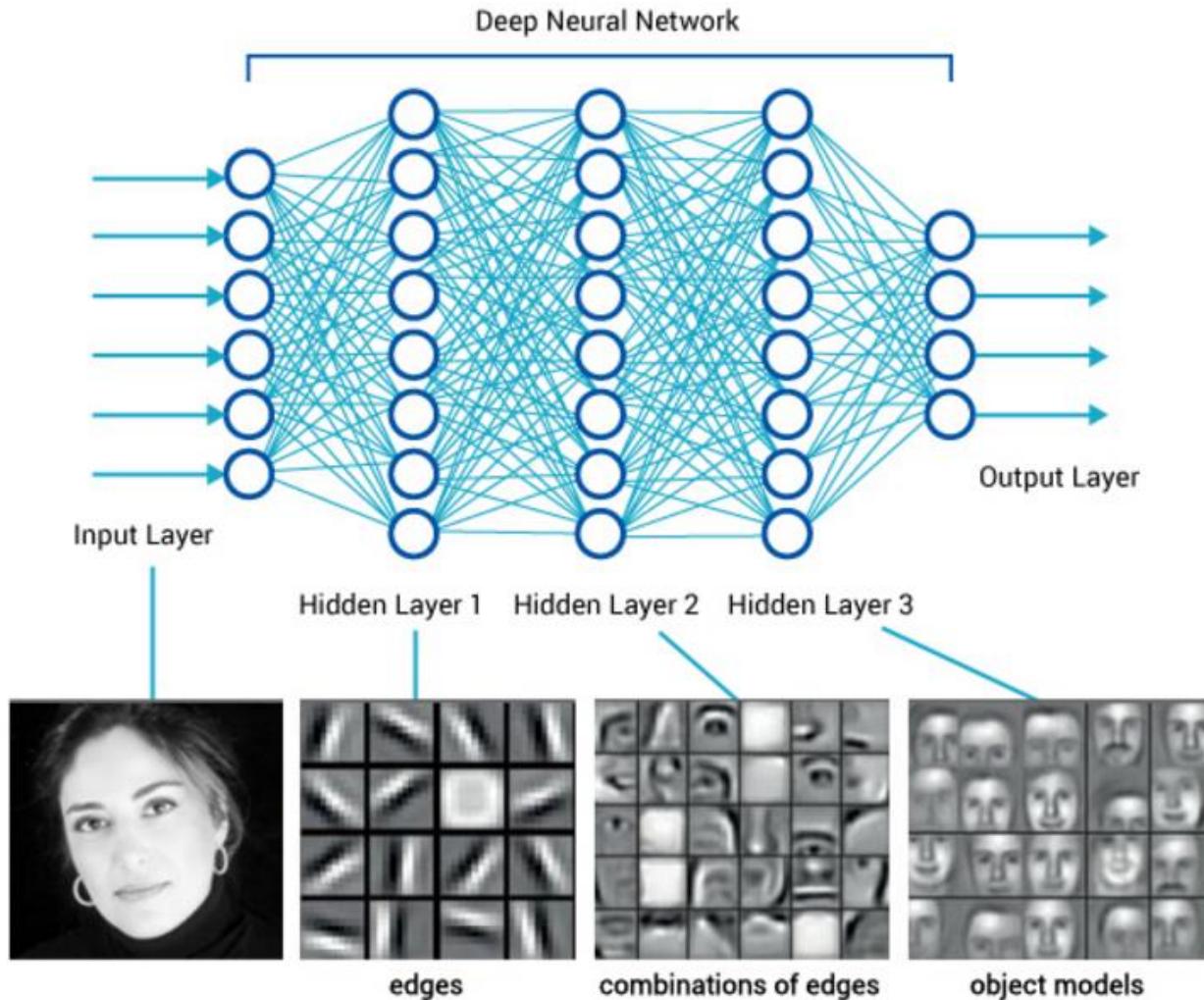


(a) logistic 函数的导数



(b) tanh 函数的导数

深层神经网络





卷积网络



卷积神经网络

- ▶ 全连接网络
 - ▶ 权重矩阵的参数非常多
- ▶ 卷积神经网络 (Convolutional Neural Networks, CNN) 是一种前馈神经网络。
 - ▶ 卷积神经网络是受生物学上感受野 (Receptive Field) 的机制而提出的。
 - ▶ 在视觉神经系统中，一个神经元的感受野是指视网膜上的特定区域，只有这个区域内的刺激才能够激活该神经元。



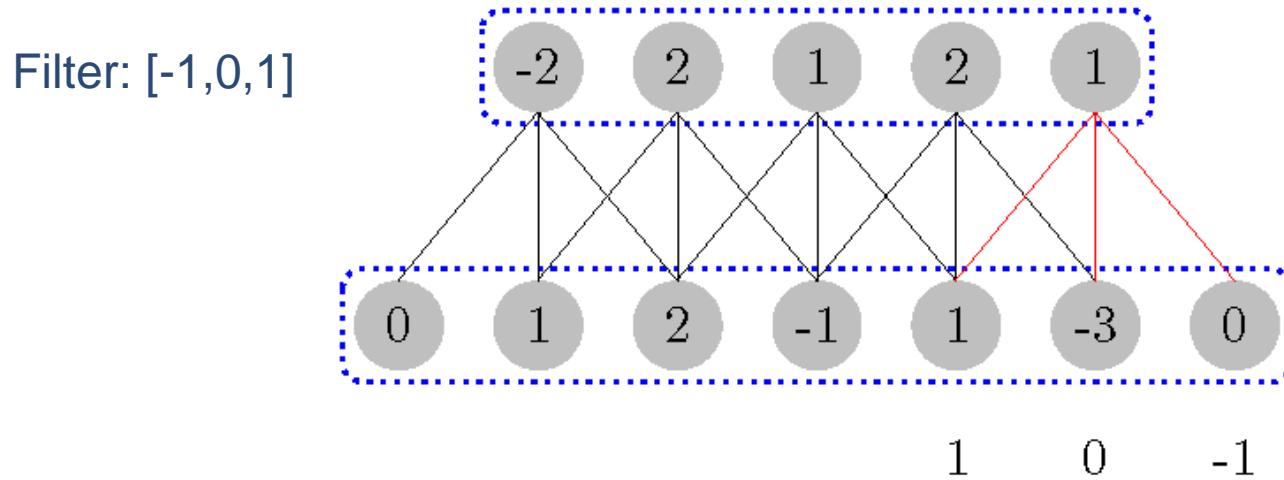
卷积神经网络

- ▶ 卷积神经网络有三个结构上的特性：
 - ▶ 局部连接
 - ▶ 权重共享
 - ▶ 空间或时间上的次采样

卷积

- ▶ 给定一个输入信号序列 x 和滤波器 f , 卷积的输出为:

$$y_t = \sum_{k=1}^m f_k x_{t-k+1}$$

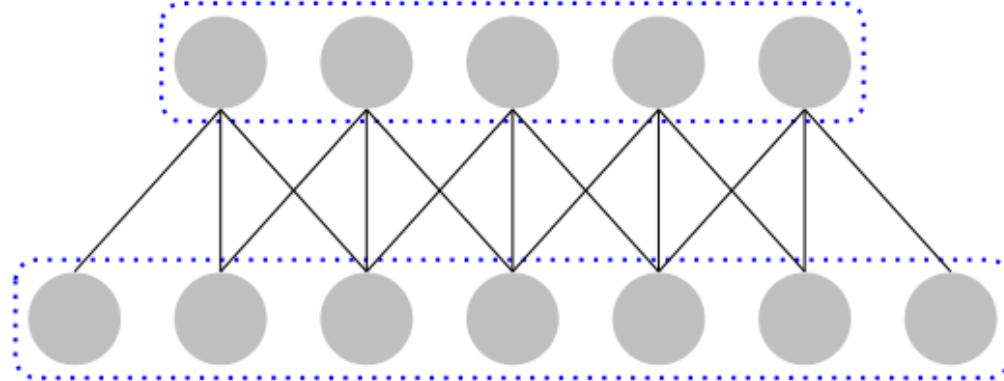




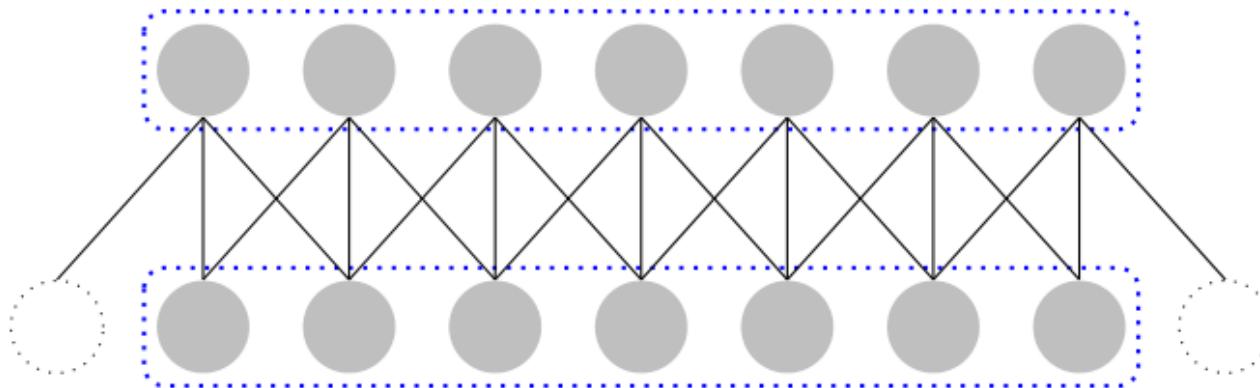
卷积类型

- ▶ 卷积的结果按输出长度不同可以分为两类：
 - ▶ 窄卷积：输出长度 $n - m + 1$ ，不补零。
 - ▶ 宽卷积：输出长度 $n+m-1$ ，对于不在 $[1,n]$ 范围之外的 x 用零补齐 (zero-padding)。($\text{Padding}=m-1$)
 - ▶ 等长卷积：输出长度 n ，对于不在 $[1,n]$ 范围之外的 x_t 用零补齐 (zero-padding)。($\text{Padding}=(m-1)/2$)
- ▶ 在这里除了特别声明，我们一般说的卷积默认为窄卷积。

两端填补 (Padding)



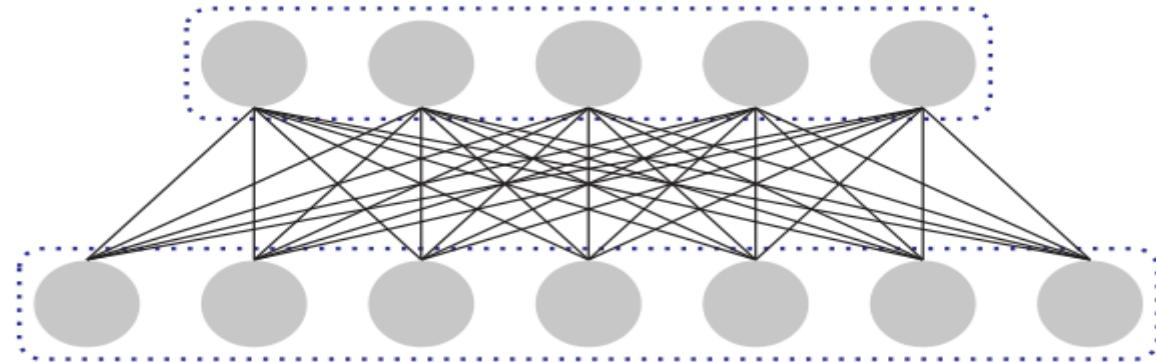
(a) 窄卷积 Padding=0



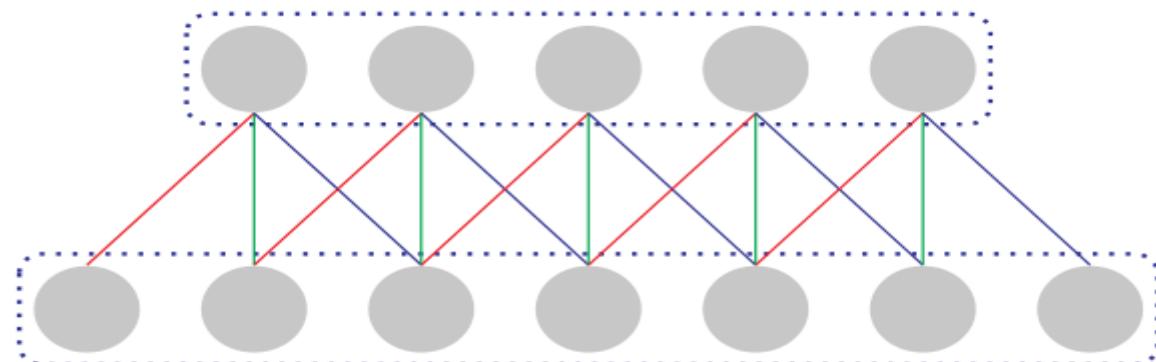
(b) 等长卷积 Padding=1

卷积神经网络

► 用卷积层代替全连接层



(a) 全连接层



(b) 卷积层



二维卷积

► 在图像处理中，图像是以二维矩阵的形式输入到神经网络中，因此我们需要二维卷积。

$$X_{s,t}^{(l)} = f \left(\sum_{i=1}^u \sum_{j=1}^v W_{i,j}^{(l)} \cdot X_{s-i+u, t-j+v}^{(l-1)} + b^{(l)} \right),$$



$$\underline{X^{(l)} = f \left(W^{(l)} \otimes X^{(l-1)} + b^{(l)} \right)},$$

步长1 filter个数1 3*3 不填充

Input (zero-padding) (5x5)

$x[:, :]$

1	0	0	0	0
2	1	1	2	1
1	1	2	2	0
2	2	1	0	0
2	1	2	1	1

Filter W (3x3)

$w[:, :]$

1	1	1
0	-1	0
0	-1	1

Output (3x3)

$o[:, :]$

1		

步长2 filter个数1 3*3 填充

Input (zero-padding) (7x7)

Filter W (3x3)

Output (3x3)

$x[:, :]$

$w[:, :]$

$o[:, :]$

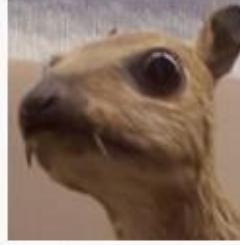
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	2	1	1	2	1	0

1	1	1
0	-1	0
0	-1	1

-2		

0	1	1	2	2	0	0
0	2	2	1	0	0	0
0	2	1	2	1	1	0
0	0	0	0	0	0	0

二维卷积示例

Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

二维卷积示例

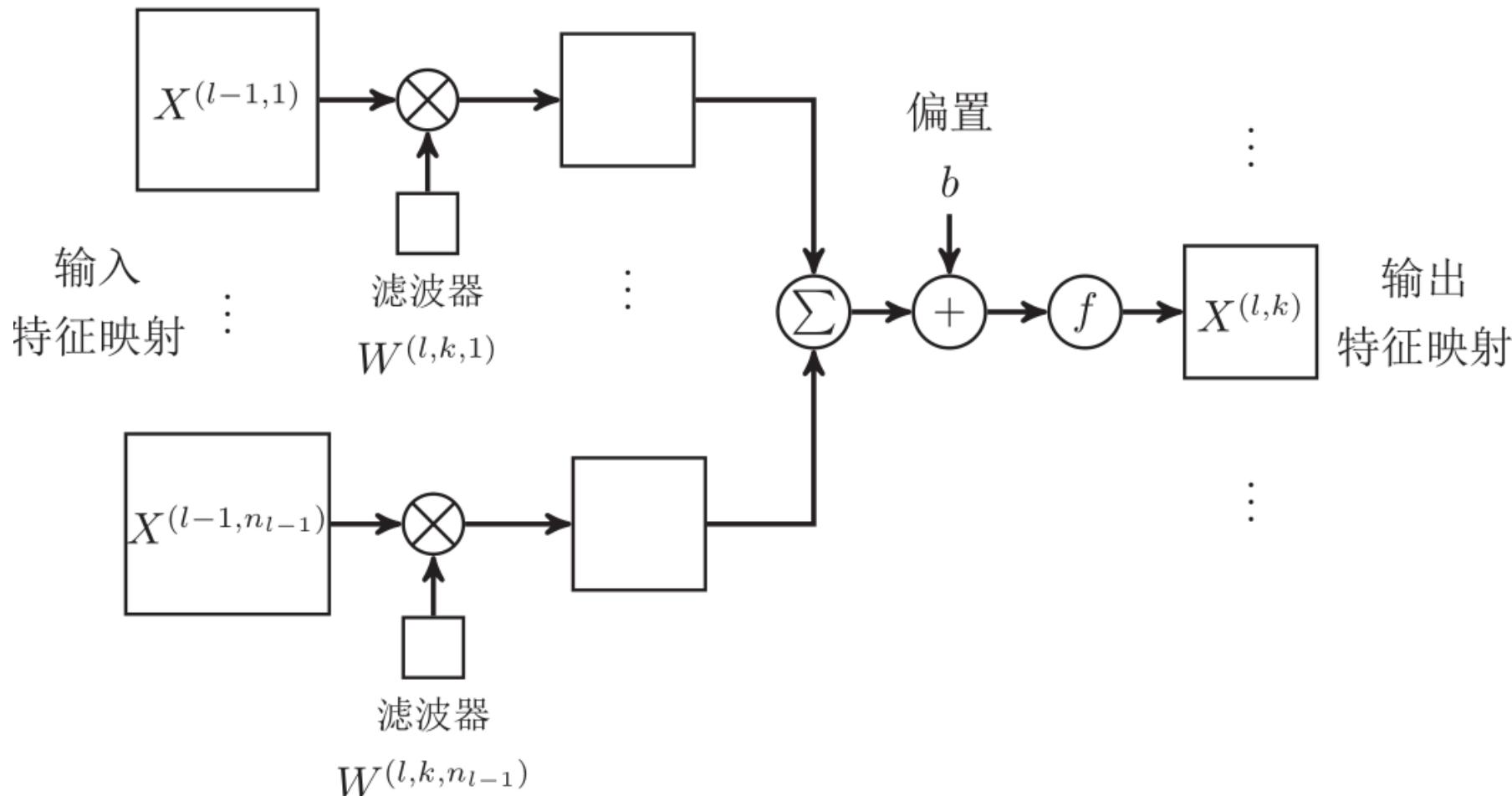
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	



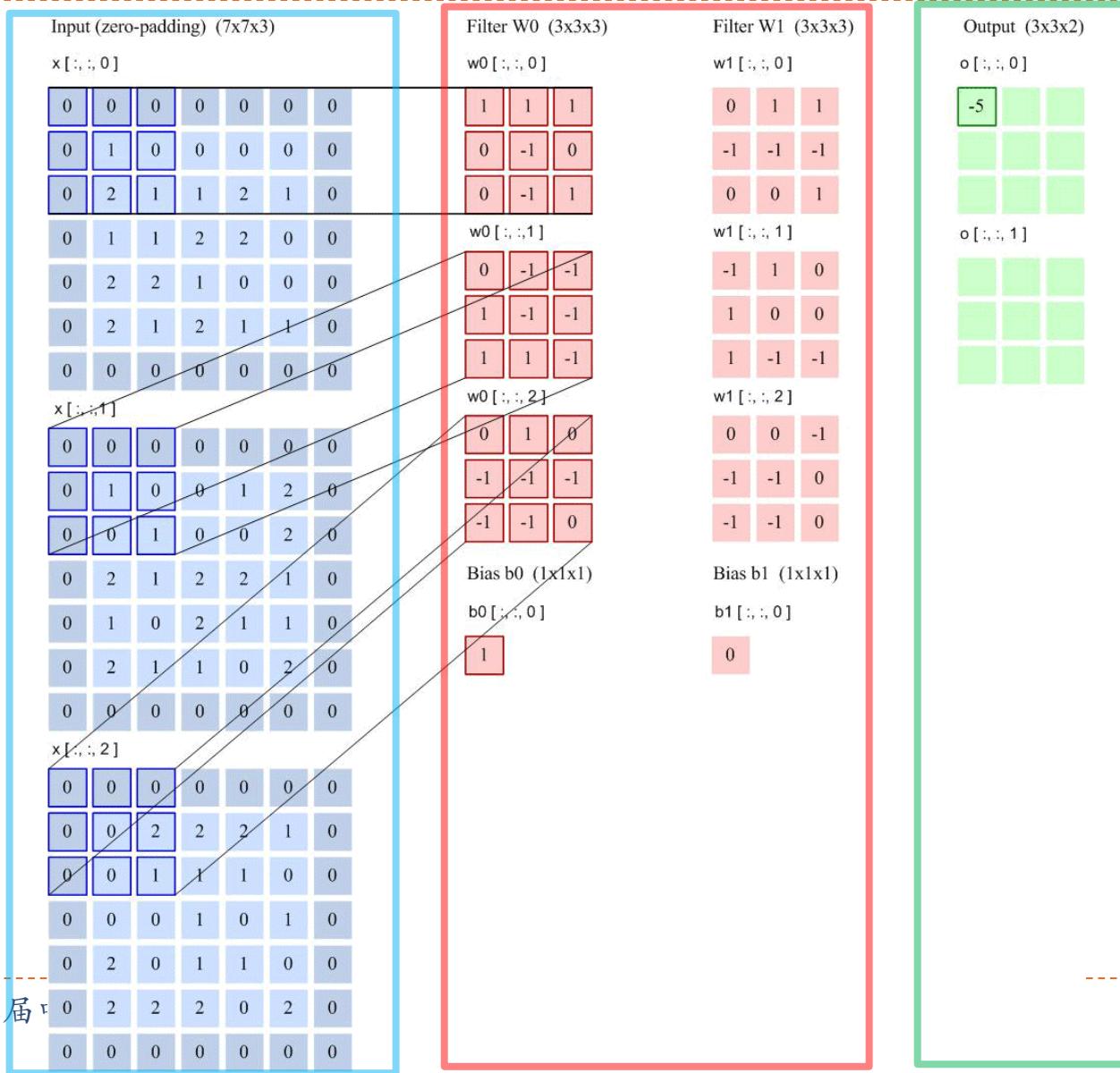
特征映射 (Feature Map)

- ▶ 为了增强卷积层的表示能力，我们可以使用K组不同的滤波器来得到K组输出。
- ▶ 如果我们把滤波器看成一个特征提取器，每一组输出都可以看成是输入图像经过一个特征抽取后得到的特征。
- ▶ 因此，在卷积神经网络中每一组输出也叫作一组特征映射。

两维卷积层的映射关系

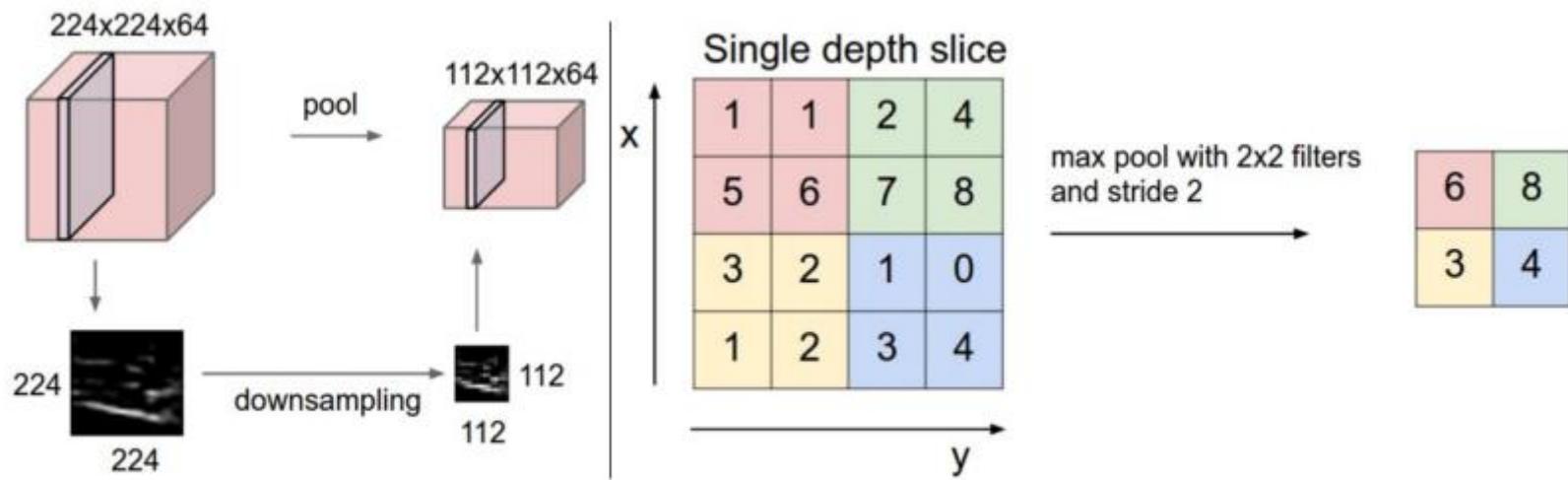


步长2 filter个数3 3*3 填充

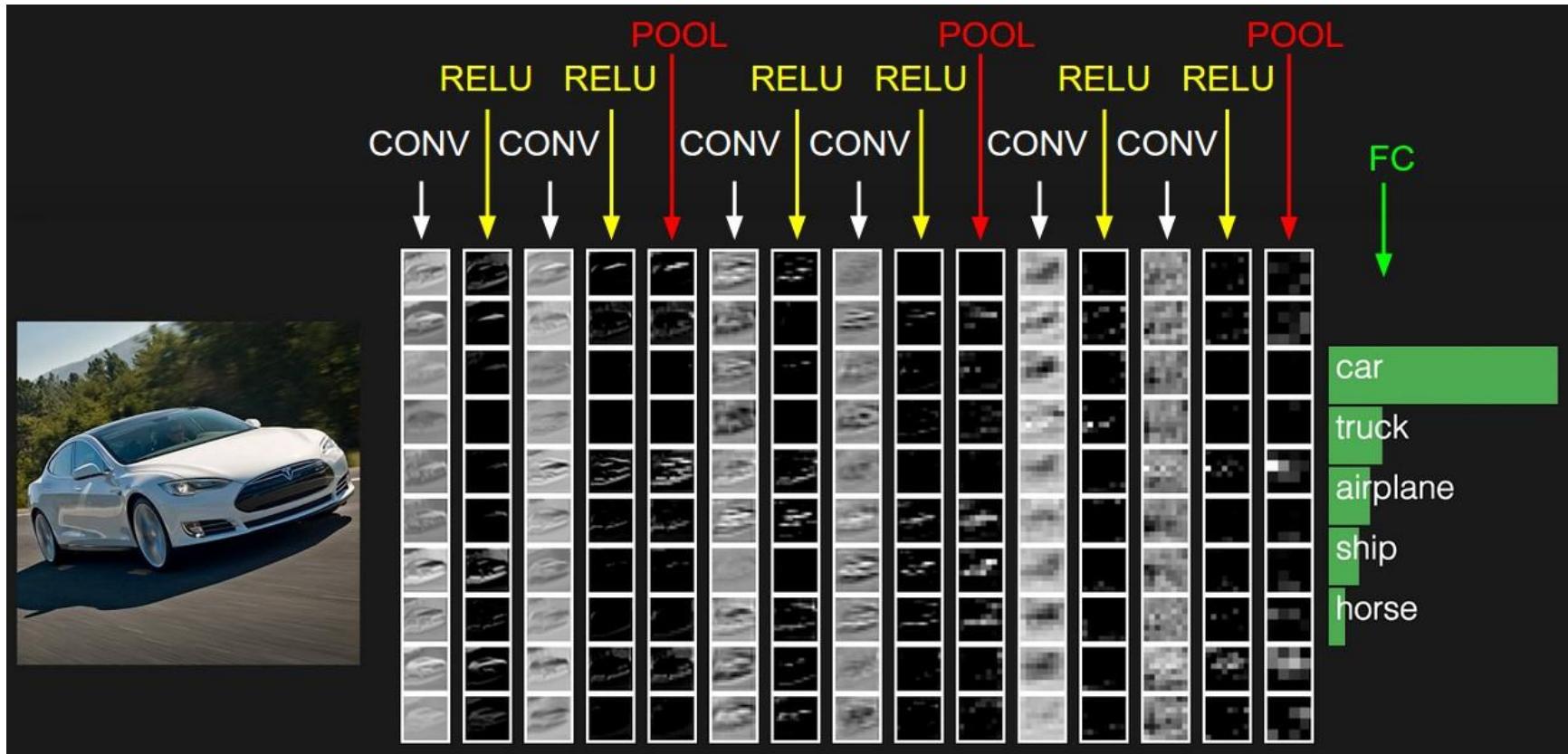


子采样层

- ▶ 卷积层虽然可以显著减少连接的个数，但是每一个特征映射的神经元个数并没有显著减少。



表示学习与深度学习

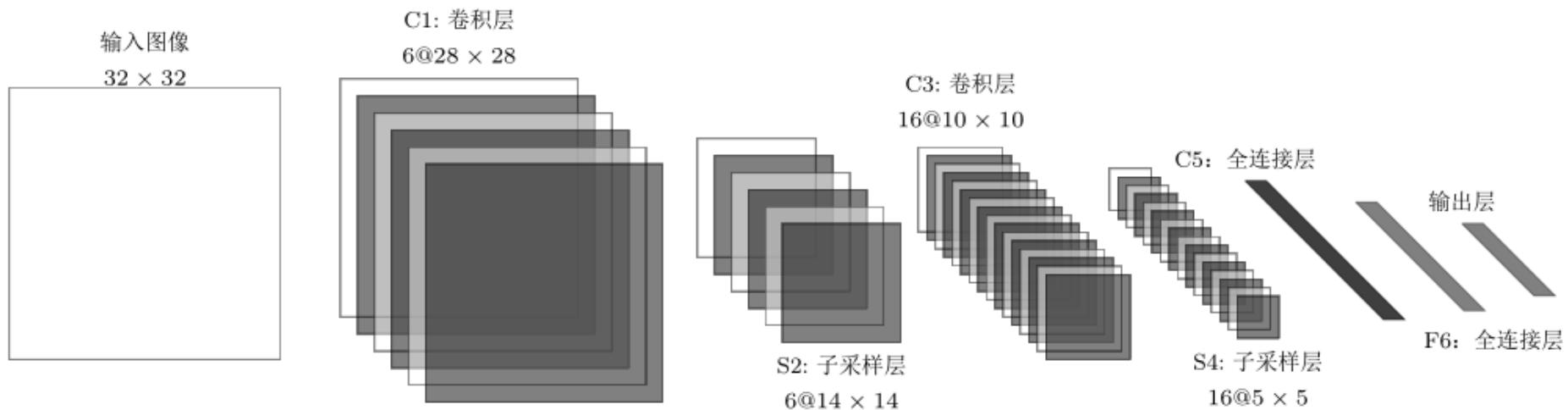




典型的卷积网络

LeNet-5

- ▶ LeNet-5 是一个非常成功的神经网络模型。
- ▶ 基于 LeNet-5 的手写数字识别系统在 90 年代被美国很多银行使用，用来识别支票上面的手写数字。LeNet-5 的网络结构如图15所示。不计输入层，LeNet-5 共有 7 层。



Large Scale Visual Recognition Challenge

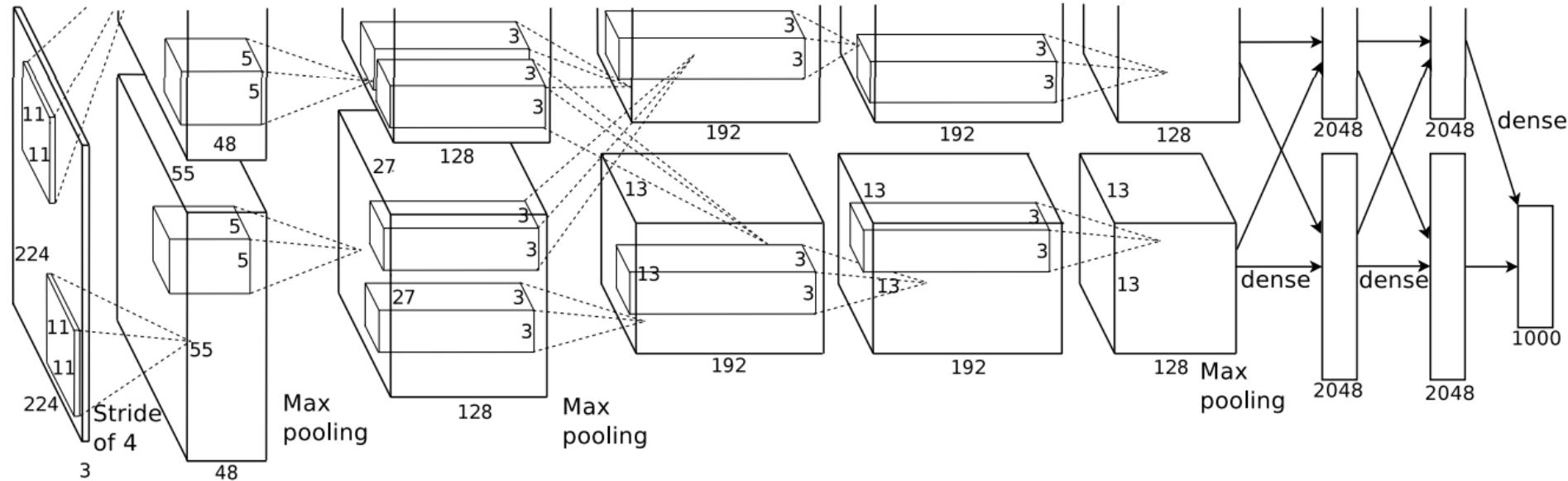


2012 Teams	%error	2013 Teams	%error	2014 Teams	%error
Supervision (Toronto)	15.3	Clarifai (NYU spinoff)	11.7	GoogLeNet	6.6
ISI (Tokyo)	26.1	NUS (singapore)	12.9	VGG (Oxford)	7.3
VGG (Oxford)	26.9	Zeiler-Fergus (NYU)	13.5	MSRA	8.0
XRCE/INRIA	27.0	A. Howard	13.5	A. Howard	8.1
UvA (Amsterdam)	29.6	OverFeat (NYU)	14.1	DeeperVision	9.5
INRIA/LEAR	33.4	UvA (Amsterdam)	14.2	NUS-BST	9.7
		Adobe	15.2	TTIC-ECP	10.2
		VGG (Oxford)	15.2	XYZ	11.2
		VGG (Oxford)	23.0	UvA	12.1

AlexNet

► 2012 ILSVRC winner

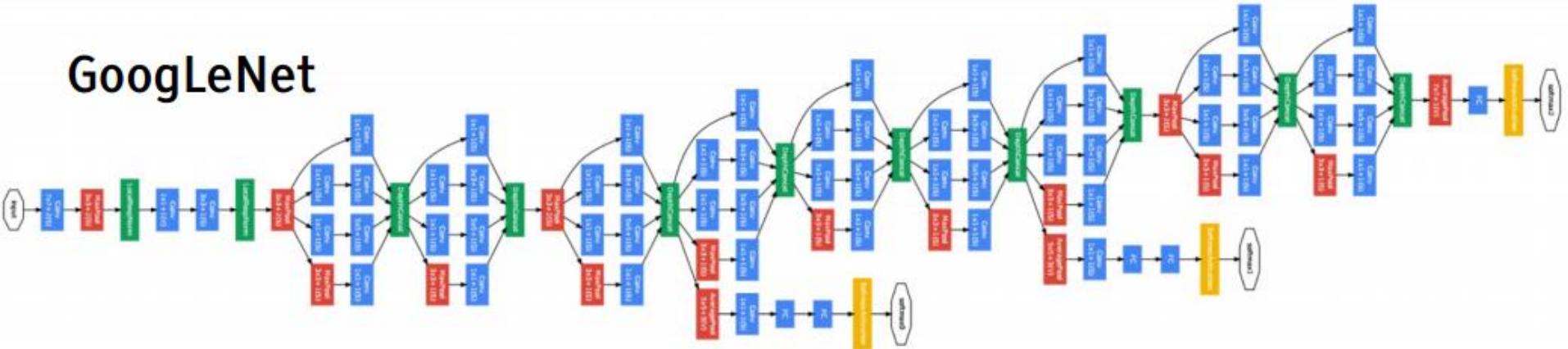
- (top 5 error of 16\% compared to runner-up with 26\% error)
- 共有8层，其中前5层卷积层，后边3层全连接层



GoogLeNet

- ▶ 2014 ILSVRC winner (22层)
- ▶ 参数： GoogLeNet: 4M VS AlexNet: 60M
- ▶ 错误率： 6.7%

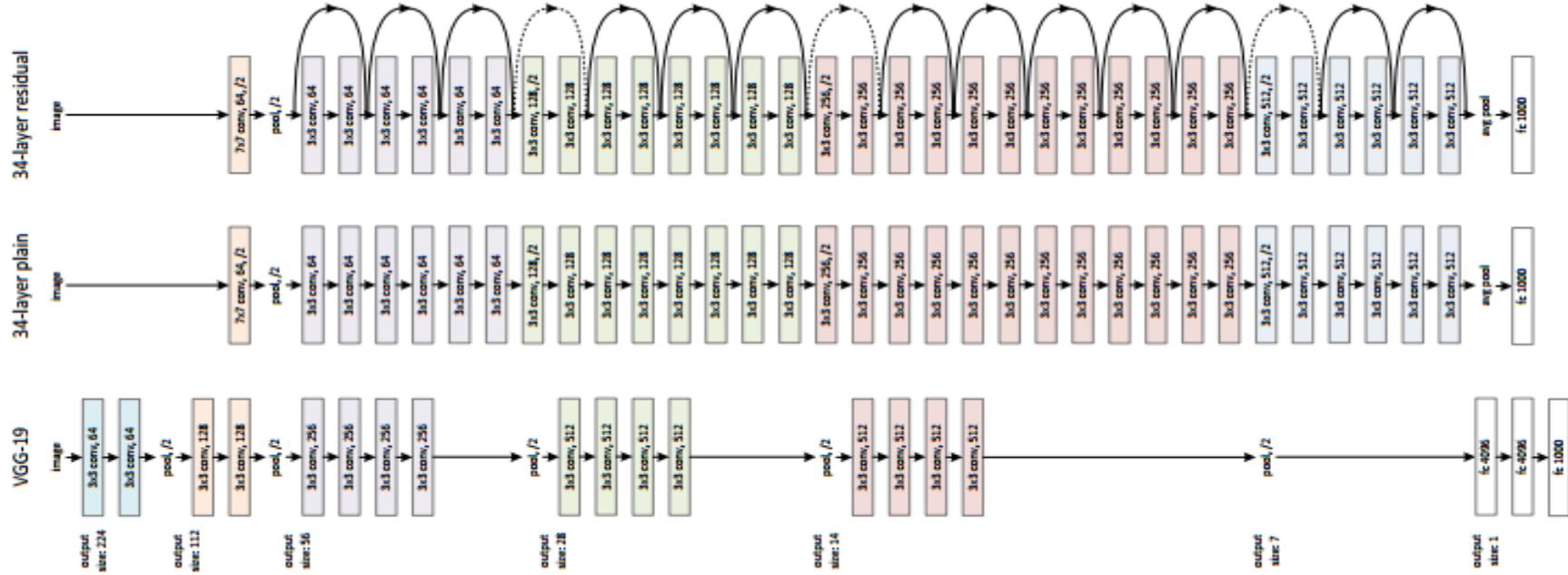
GoogLeNet



ResNet

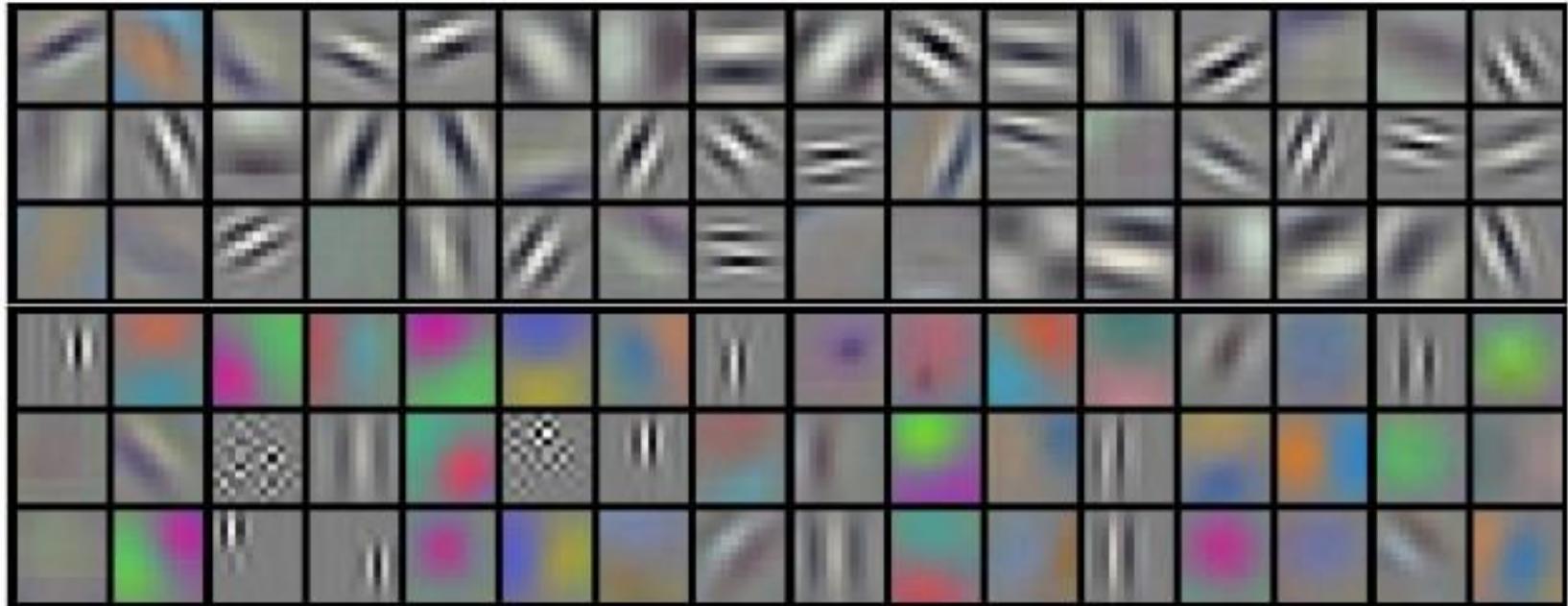
► 2015 ILSVRC winner (152层)

► 错误率: 3.57%

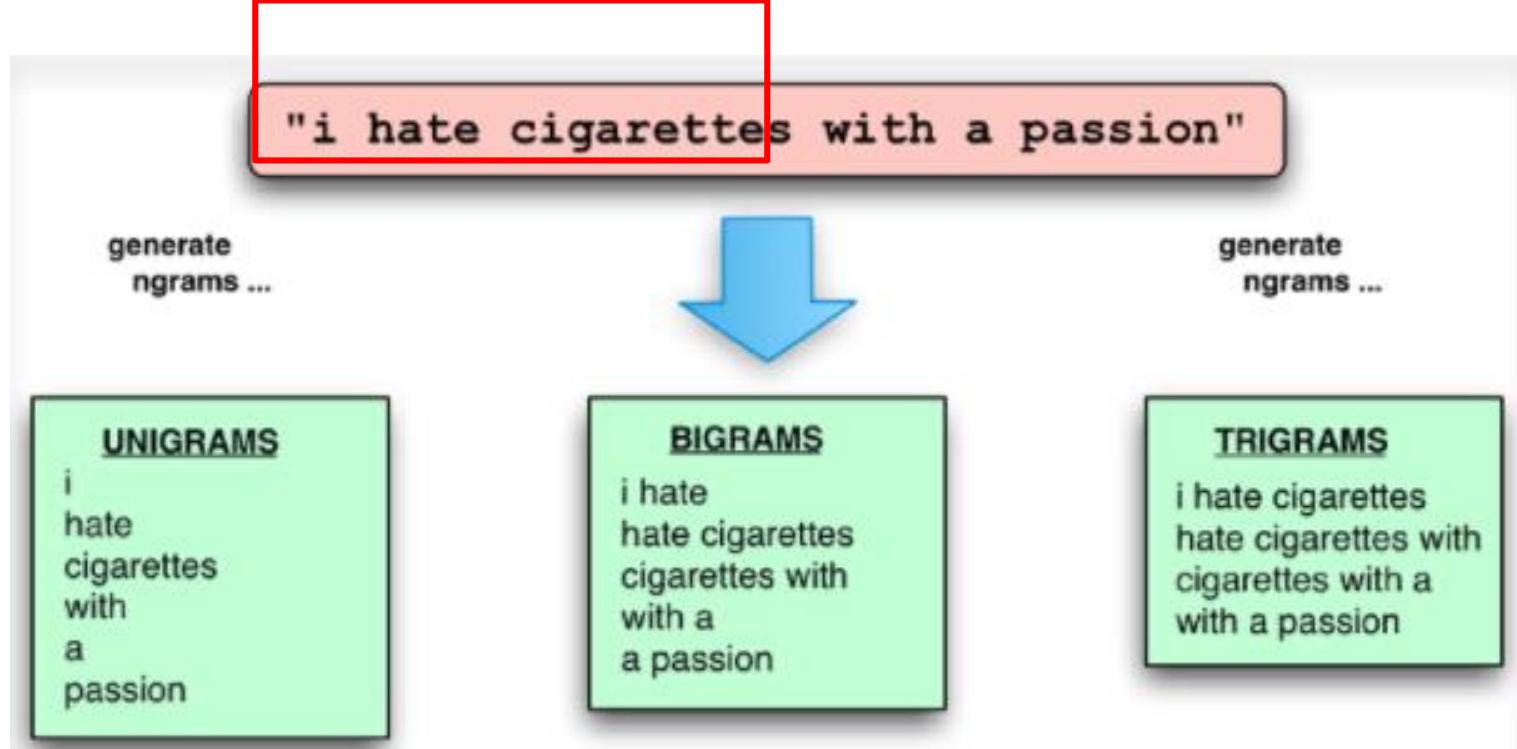


CNN 可视化：滤波器

► AlexNet中的滤波器 (96 filters [11x11x3])

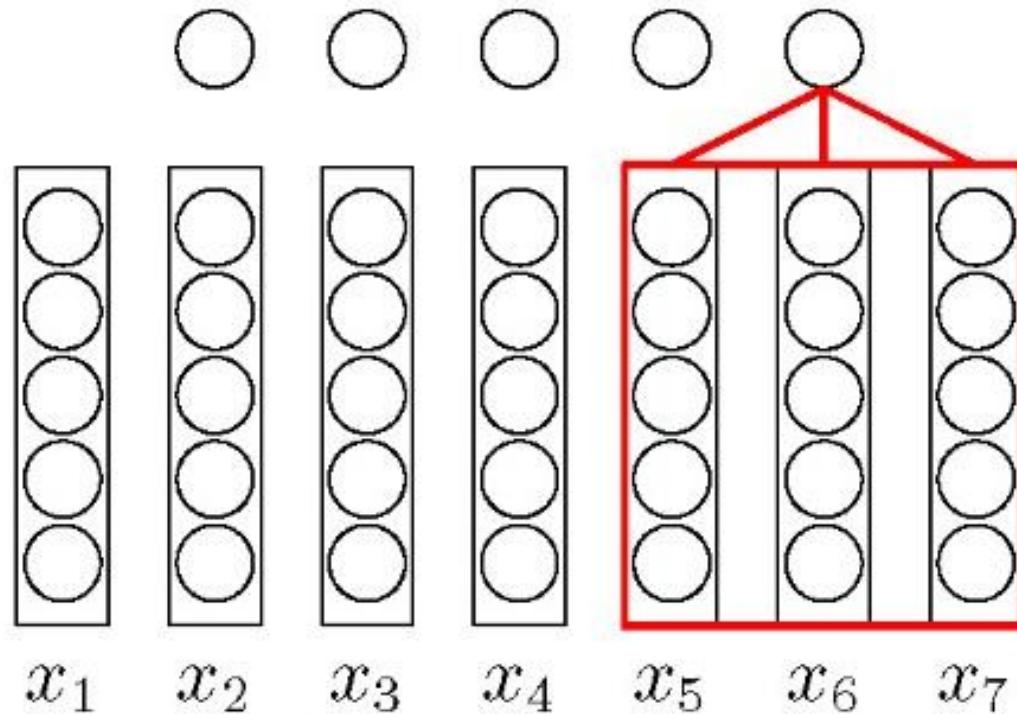


Ngram特征与卷积

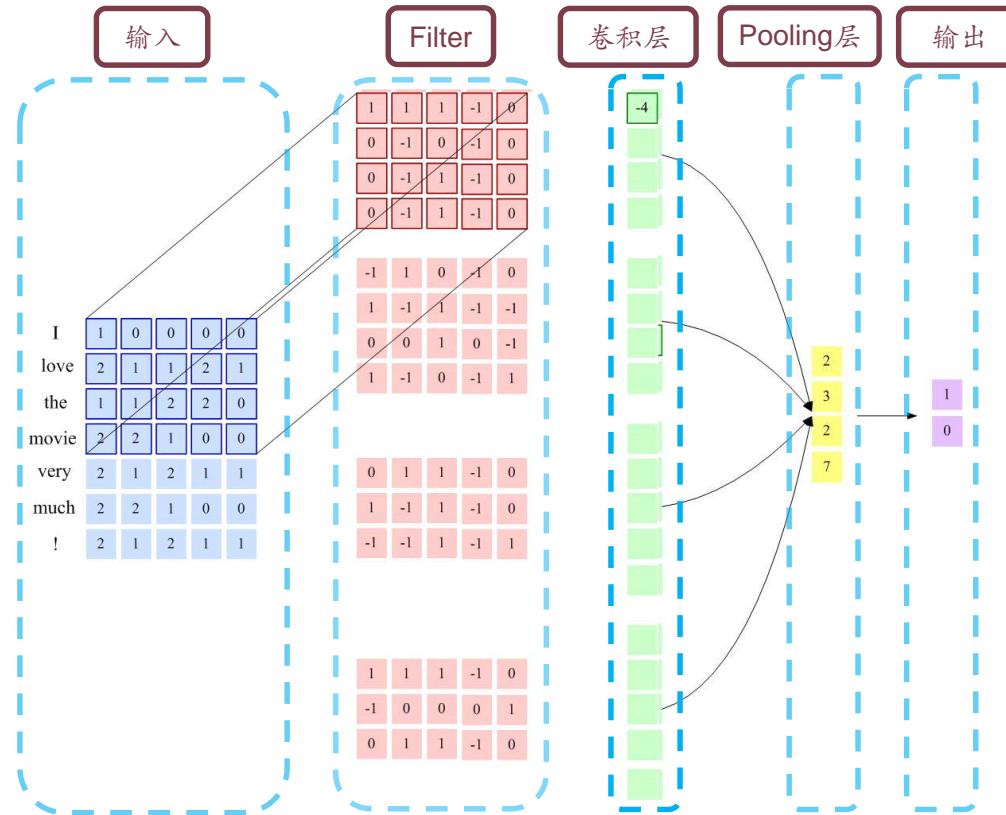


如何用卷积操作来实现?

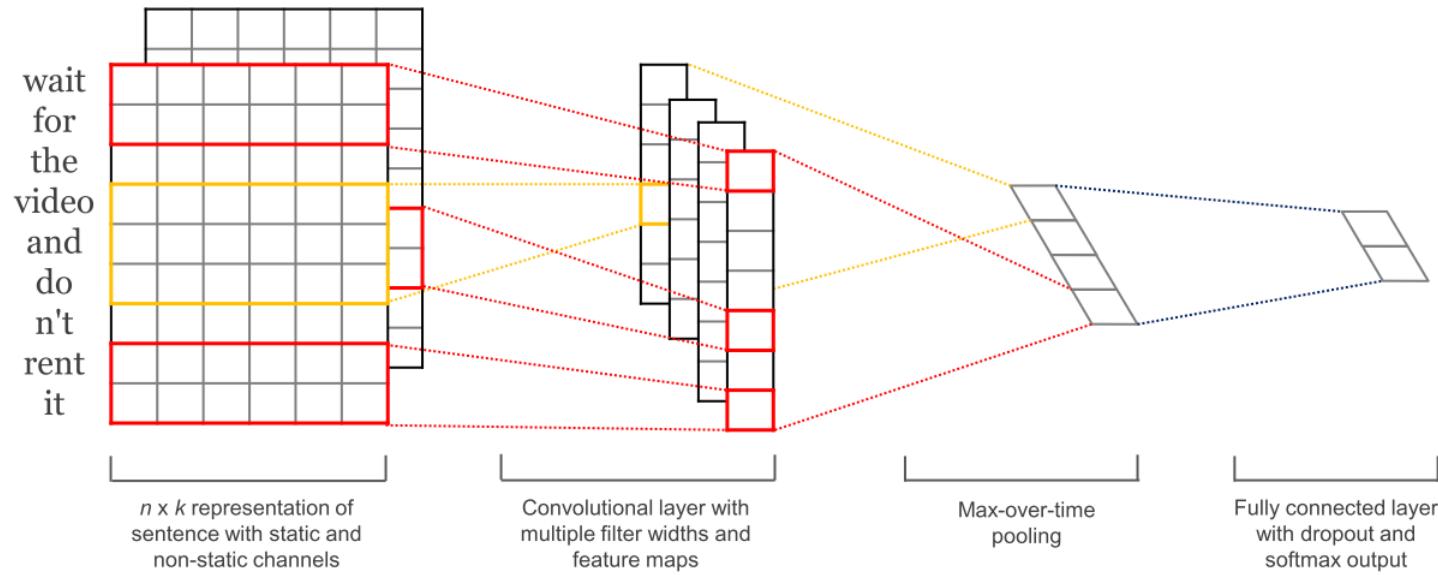
文本序列的卷积



文本序列的卷积模型

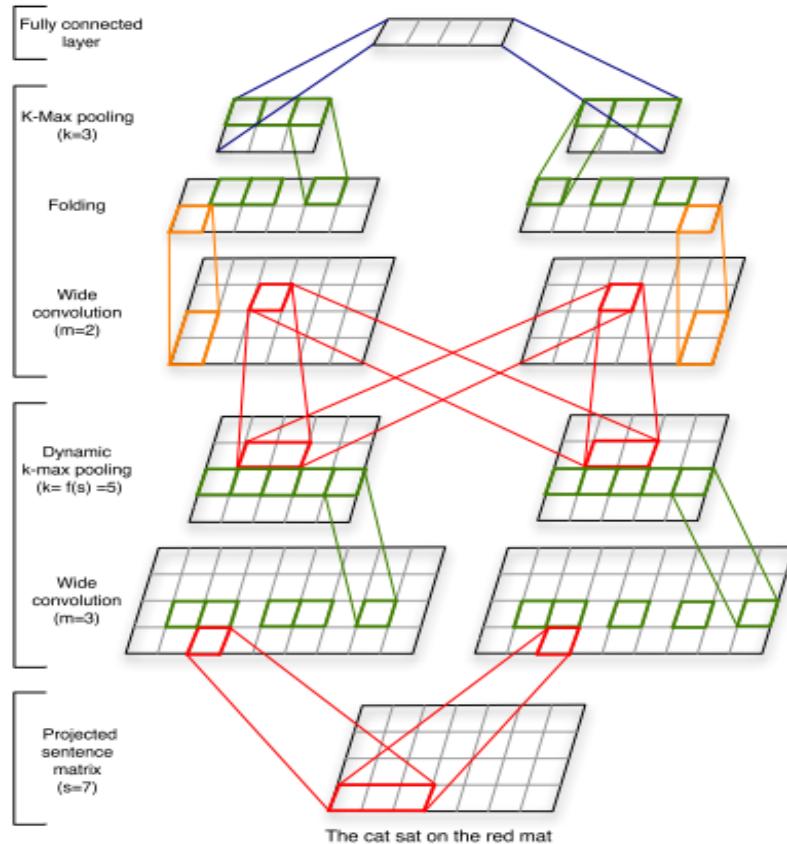


基于卷积模型的句子表示



Y. Kim. "Convolutional neural networks for sentence classification" . In: *arXiv preprint arXiv:1408.5882* (2014).

基于卷积模型的句子表示



N. Kalchbrenner, E. Grefenstette, and P. Blunsom. "A Convolutional Neural Network for Modelling Sentences". In:
Proceedings of ACL. 2014



循环网络



前馈网络不足

- ▶ 连接存在层与层之间，每层的节点之间是无连接的。
- ▶ 输入和输出的维数都是固定的，不能任意改变。无法处理变长的序列数据。
- ▶ 假设每次输入都是独立的，也就是说每次网络的输出只依赖于当前的输入。



循环神经网络

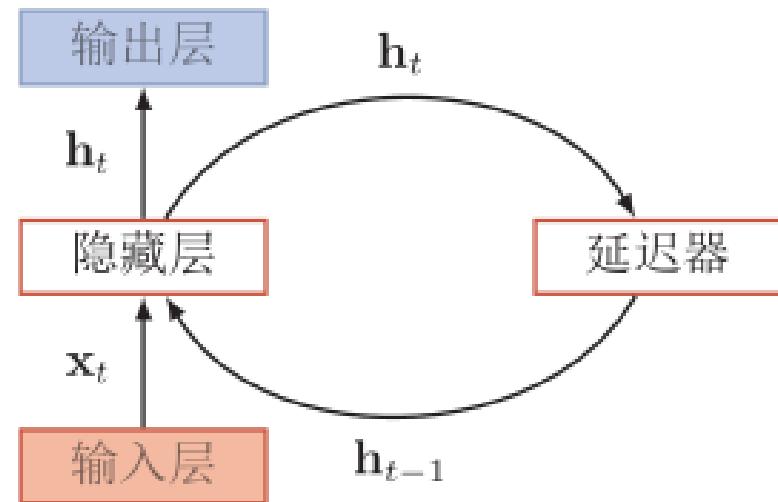
▶ 循环神经网络

- ▶ 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的序列。
- ▶ 循环神经网络比前馈神经网络更加符合生物神经网络的结构。
- ▶ 循环神经网络已经被广泛应用在语音识别、语言模型以及自然语言生成等任务上。

循环神经网络

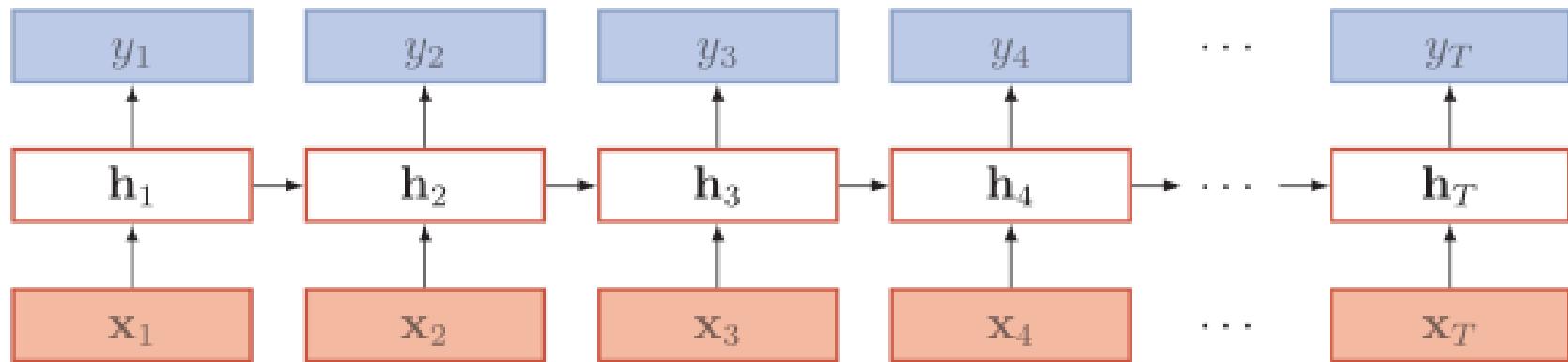
给定一个输入序列 $\mathbf{x}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t, \dots, \mathbf{x}_T)$, 循环神经网络通过下面公式更新带反馈边的隐藏层的活性值 \mathbf{h}_t :

$$\mathbf{h}_t = \begin{cases} 0 & t = 0 \\ f(\mathbf{h}_{t-1}, \mathbf{x}_t) & \text{otherwise} \end{cases}$$



循环神经网络 (RNN)

缺点：长距离依赖问题



RNN是图灵完全等价的 (Siegelmann and Sontag, 1995)

FNN: 模拟任何函数

RNN: 模拟任何程序（计算过程）。



简单循环网络

► 状态更新：

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

► 机器学习

- 给定一个训练样本 (\mathbf{x}, \mathbf{y}) ，
 - 其中 $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$ 为长度是 T 的输入序列，
 - $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_T)$ 是长度为 T 的标签序列。
- 时刻 t 的损失函数为

$$\mathcal{L}_t = \mathcal{L}(\mathbf{y}, f(\mathbf{h}_{t-1}, \mathbf{x}_t)),$$

► 总的损失函数

$$\mathcal{L} = \sum_{t=1}^T \mathcal{L}_t.$$



梯度

► 链式法则

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k} \mathbf{h}_k^T.$$

► 其中， $\delta_{t,k}$ 为第t时刻的损失对第k步隐藏神经元的净输入 \mathbf{z}_k 的导数

$$\begin{aligned}\delta_{t,k} &= \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} \\ &= \frac{\partial \mathbf{h}_k}{\partial \mathbf{z}_k} \frac{\partial \mathbf{z}_{k+1}}{\partial \mathbf{h}_k} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_{k+1}} \\ &= \text{diag}(f'(\mathbf{z}_k)) U^T \delta_{t,k+1}.\end{aligned}$$



梯度消失/爆炸

► 梯度

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^T \sum_{k=1}^t \left(\prod_{i=k}^{t-1} \text{diag}(f'(\mathbf{z}_i)) U^\top \right) \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_t} \mathbf{h}_k^\top$$

我们定义 $\gamma = \|\text{diag}(f'(\mathbf{z}_i))U^\top\|$, 则在上面公式中的括号里面为 γ^{t-k} 。如果 $\gamma > 1$, 当 $t - k \rightarrow \infty$ 时, $\gamma^{t-k} \rightarrow \infty$, 会造成系统不稳定, 也就是所谓的梯度爆炸问题 (Gradient Exploding Problem); 相反, 如果 $\gamma < 1$, 当 $t - k \rightarrow \infty$ 时, $\gamma^{t-k} \rightarrow 0$, 会出现和深度前馈神经网络类似的梯度消失问题 (Gradient Vanishing Problem)。

由于梯度爆炸或消失问题, 实际上只能学习到短周期的依赖关系。这就是所谓的长期依赖问题。



长期依赖问题

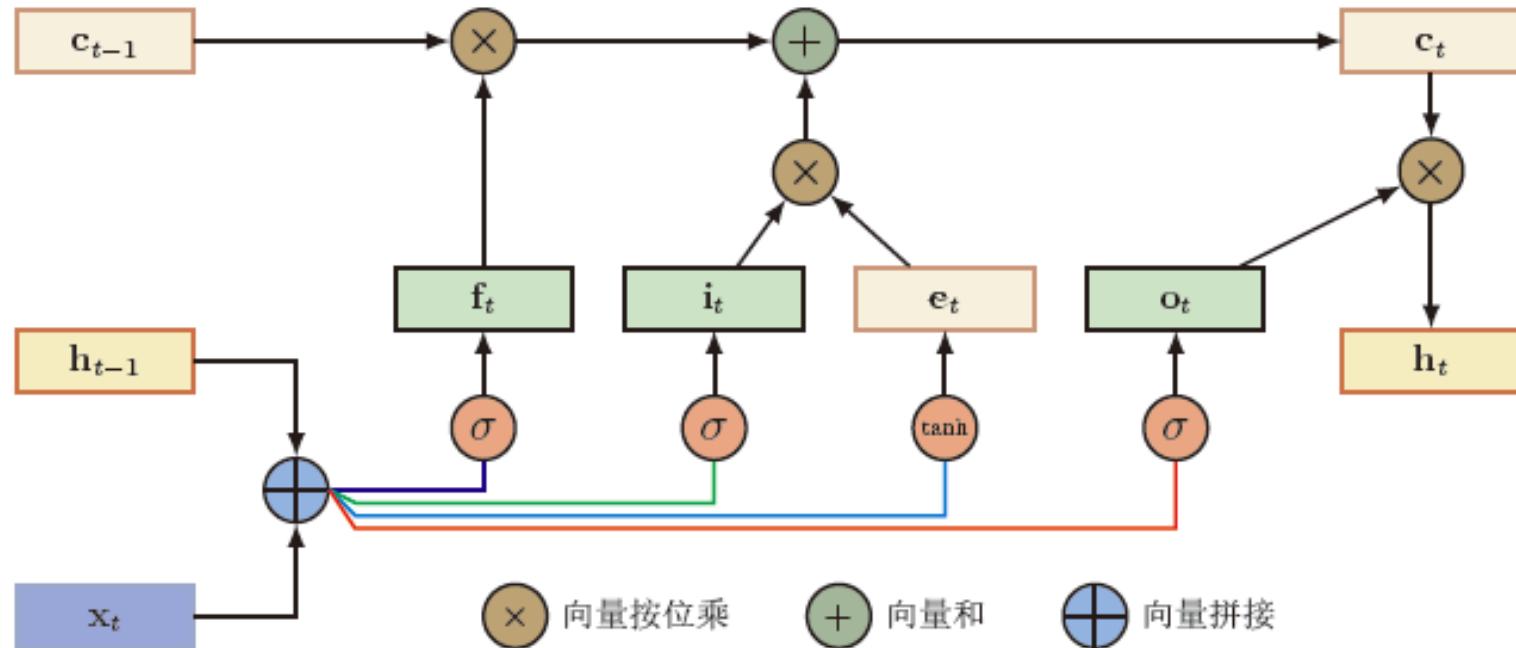
- ▶ 循环神经网络在时间维度上非常深!
 - ▶ 梯度消失或梯度爆炸
- ▶ 改进方法
 - ▶ 循环边改为线性依赖关系
 - ▶ 增加非线性

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

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{W}\mathbf{g}(\mathbf{x}_t)$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t).$$

长短时记忆神经网络：LSTM



$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o),$$

$$\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$



LSTM的各种变体

► 没有遗忘门

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t.$$

► 耦合输入门和遗忘门

$$\mathbf{f}_t + \mathbf{i}_t = 1.$$

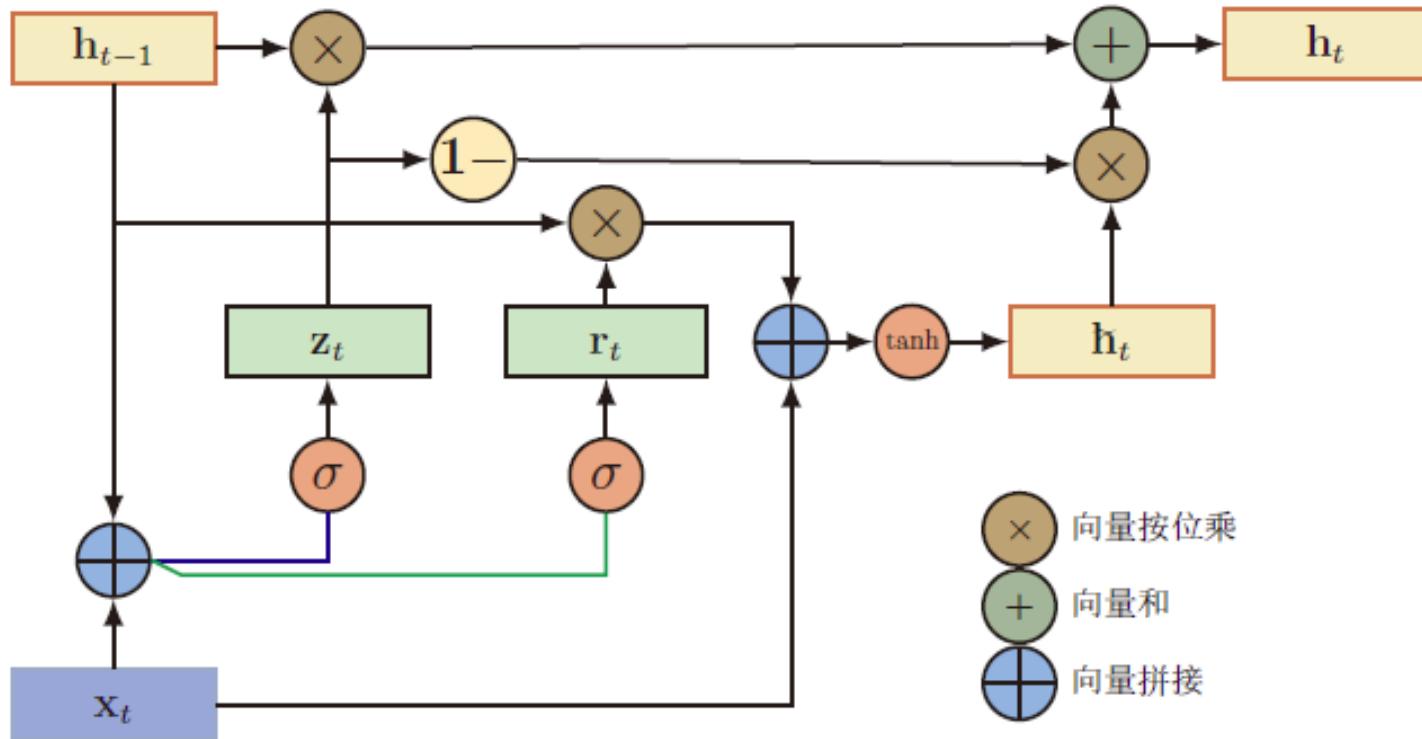
► peephole连接

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t + \mathbf{b}_o),$$

GRU



重置门

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r),$$

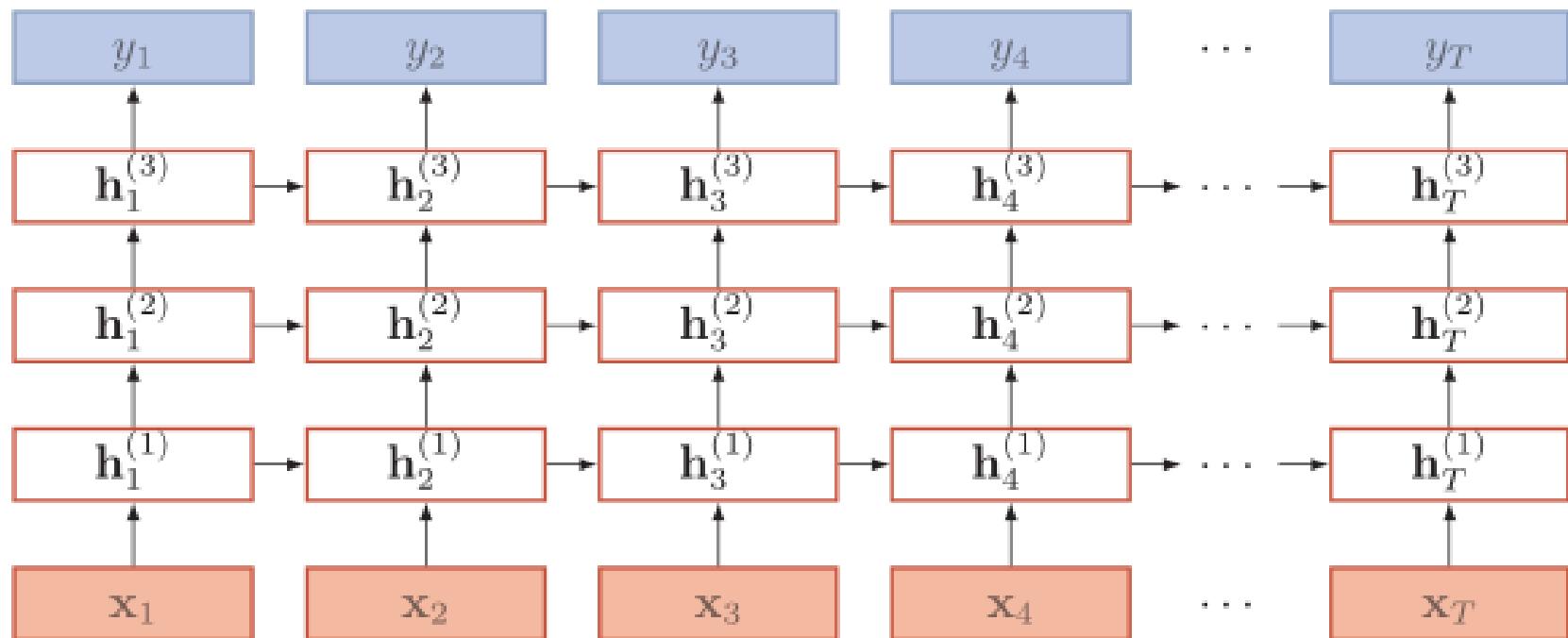
$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z),$$

更新门

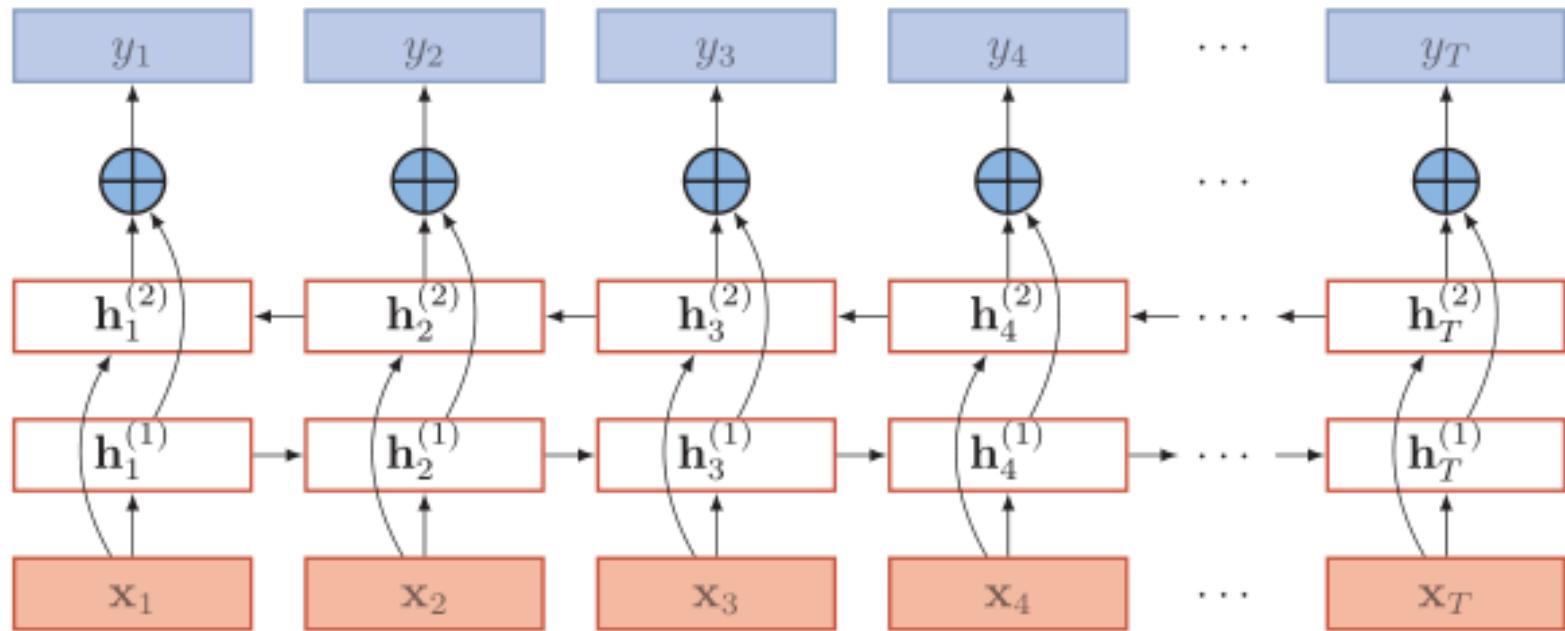
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t,$$

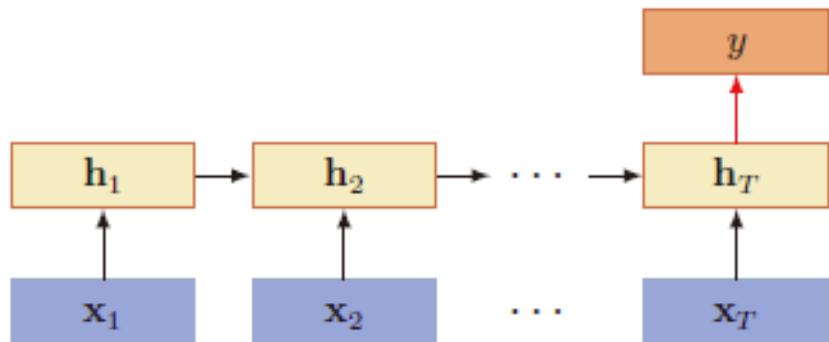
堆叠循环神经网络



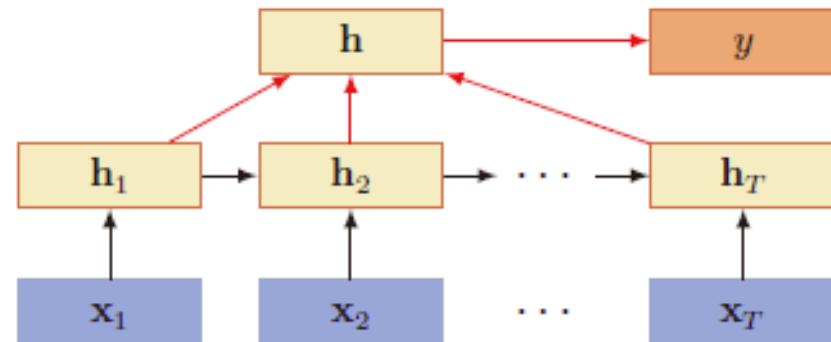
双向循环神经网络



序列模型：RNN

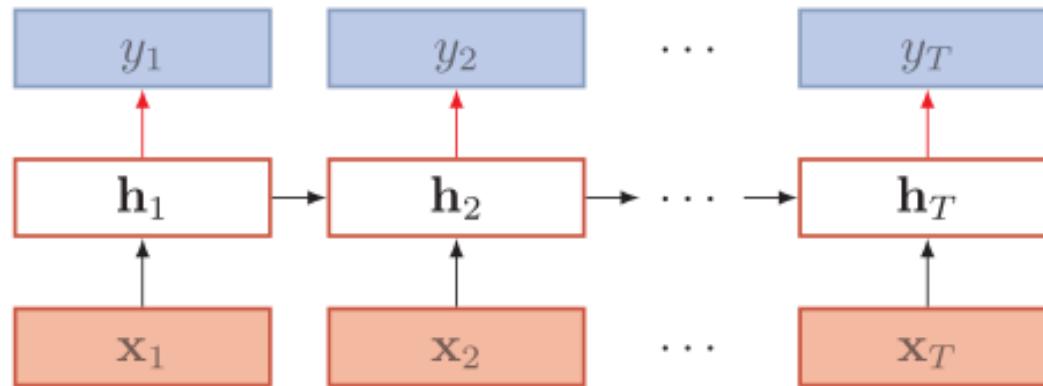


(a) 正常模式

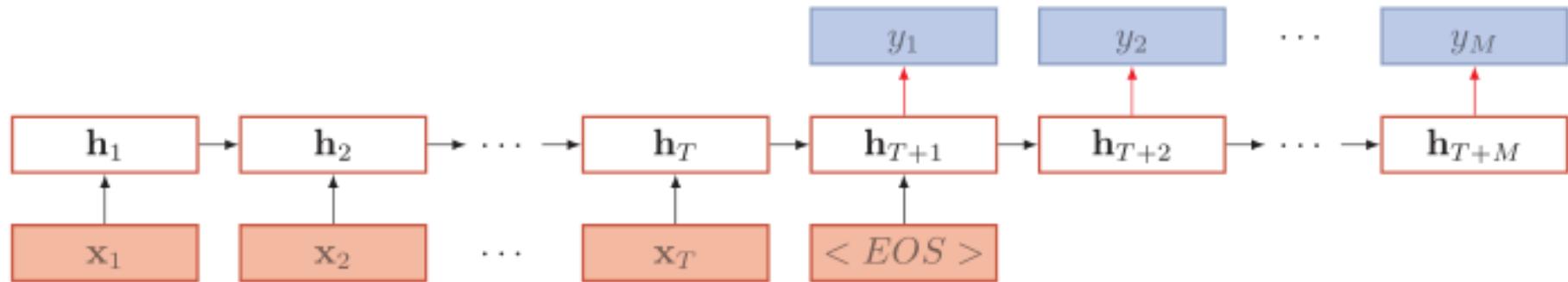


(b) 按时间进行平均采样模式

同步的序列到序列模式



序列到序列模型



递归神经网络

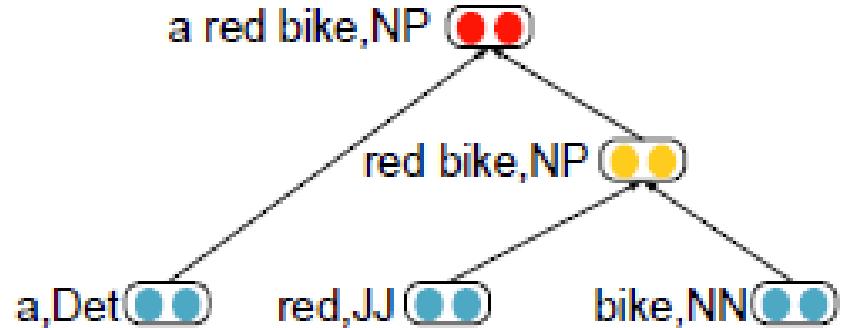
给定一个语法树，

$$p_2 \rightarrow ap_1,$$

$$p_1 \rightarrow bc.$$

$$p_1 = f(W \begin{bmatrix} b \\ c \end{bmatrix}),$$

$$p_2 = f(W \begin{bmatrix} a \\ p_1 \end{bmatrix}).$$





优化与正则化



优化

- ▶ 数据增强
- ▶ 数据预处理
- ▶ 网络参数初始化
- ▶ 正则化
- ▶ 超参数优化

<http://playground.tensorflow.org/>



超参数优化

- ▶ 层数
- ▶ 每层神经元个数
- ▶ 激活函数
- ▶ 学习率（以及动态调整算法）
- ▶ 正则化系数
- ▶ mini-batch 大小



超参数优化

► 网格搜索 (Grid Search)

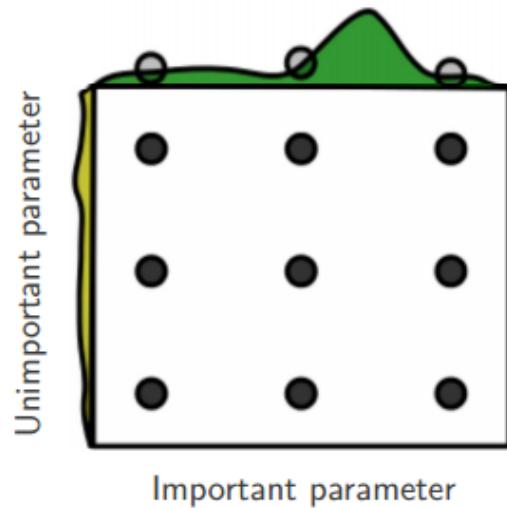
- 假设总共有K个超参数，第k个超参数的可以取 m_k 个值。
- 如果参数是连续的，可以将参数离散化，选择几个“经验”值。比如学习率 α ，我们可以设置

$$\alpha \in \{0.01, 0.1, 0.5, 1.0\}.$$

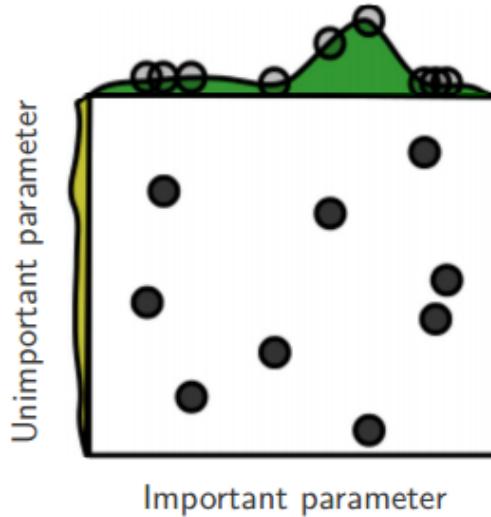
- 这些超参数可以有 $m_1 \times m_2 \times \dots \times m_K$ 个取值组合。

超参数优化

Grid Layout



Random Layout

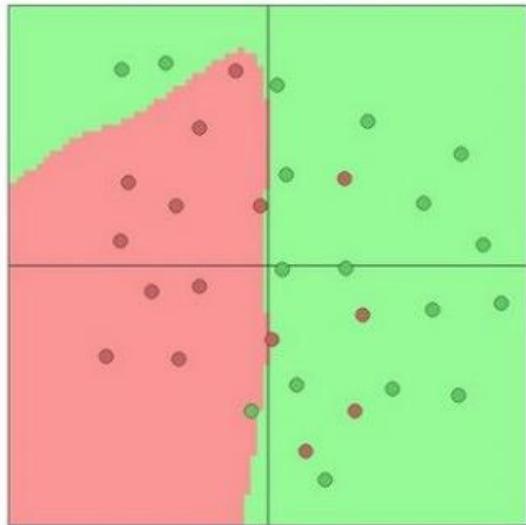


Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g . This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

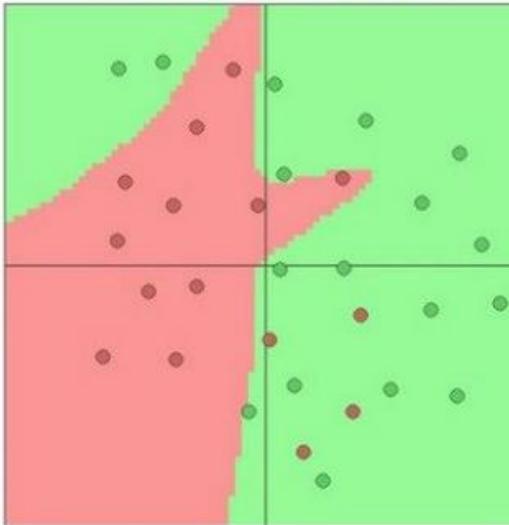
神经网络示例

► 隐藏层的不同神经元个数

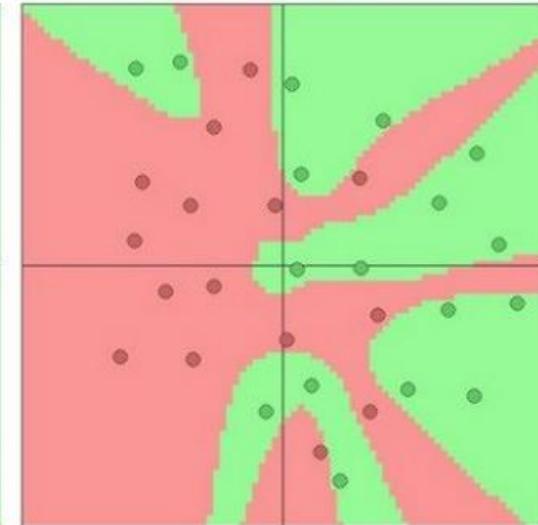
3 hidden neurons



6 hidden neurons

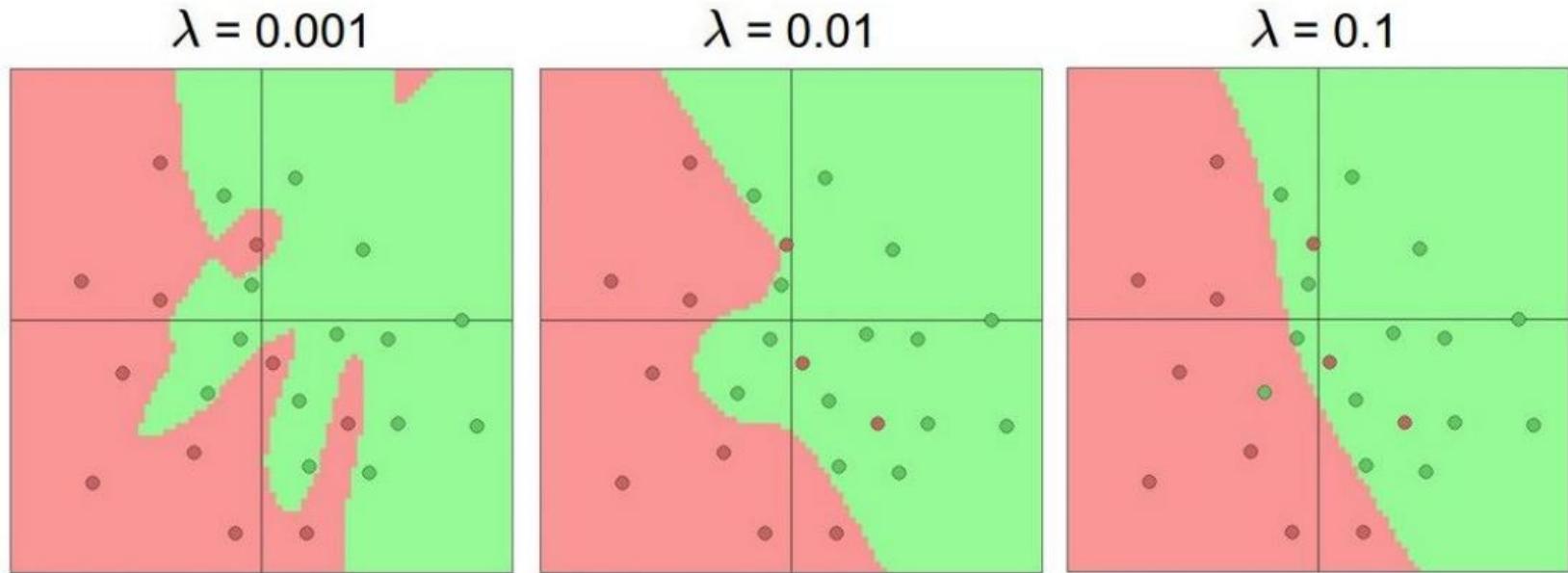


20 hidden neurons



神经网络示例

► 不同的正则化系数





参数初始化

► 参数不能初始化为0！为什么？

► 对称权重问题！

► Gaussian初始化方法

► Gaussian初始化方法是最简单的初始化方法，参数从一个固定均值（比如0）和固定方差（比如0.01）的Gaussian分布进行随机初始化。

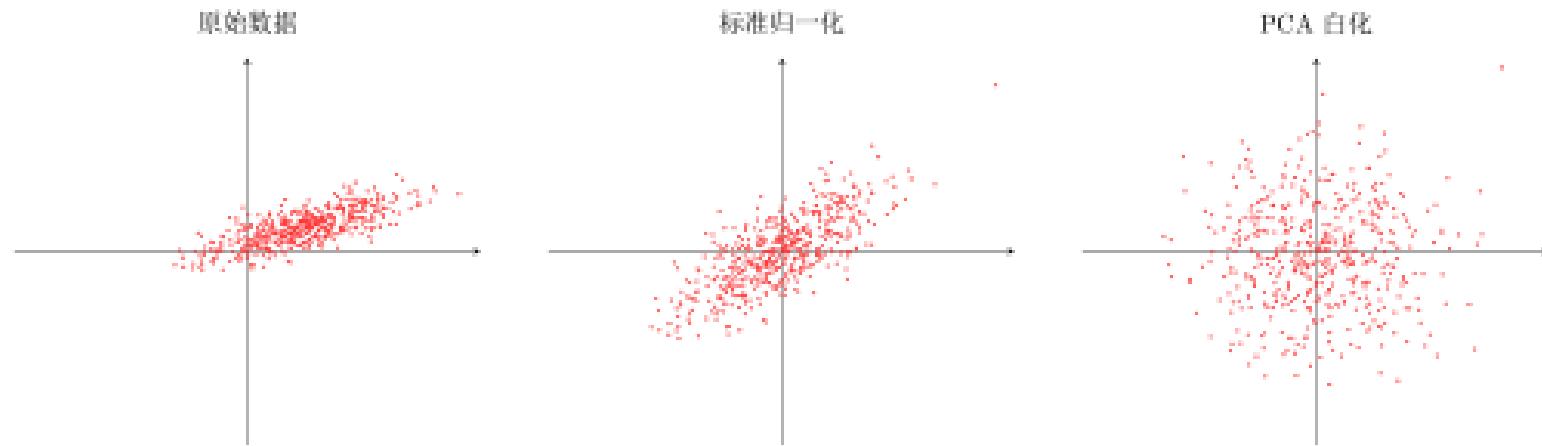
► Xavier初始化

► 参数可以在区间 $[-r, r]$ 内采用均匀分布进行初始化。

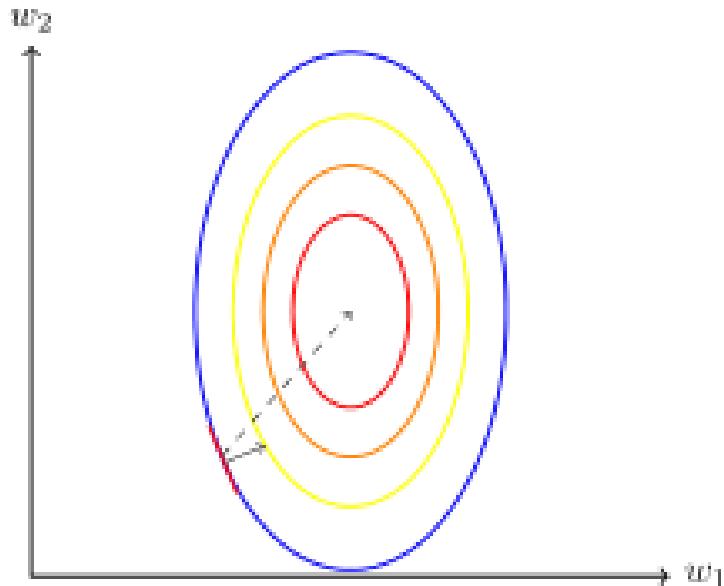
$$r = \sqrt{\frac{6}{n^{l-1} + n^l}},$$

数据预处理

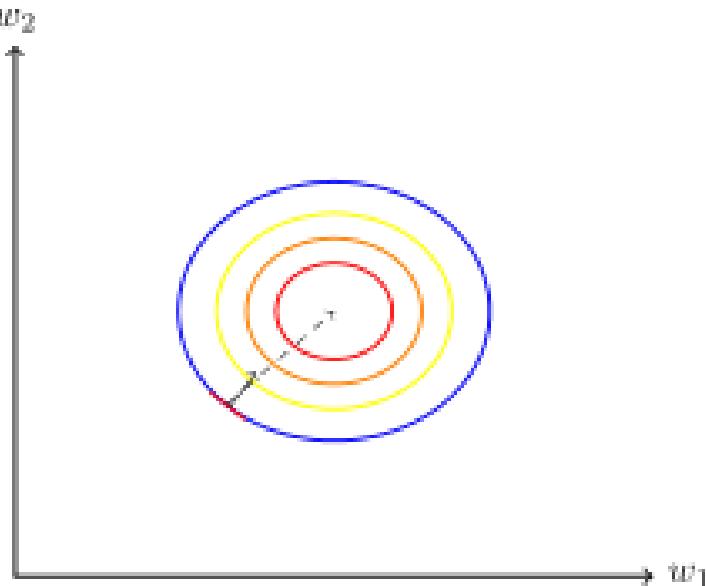
- ▶ 数据归一化
 - ▶ 标准归一化
 - ▶ 缩放归一化
 - ▶ PCA



数据归一化对梯度的影响



(a) 未归一化数据的梯度



(b) 归一化数据的梯度



批量归一化

输入: 一次 mini-batch 中的所有样本集合:

$$\mathcal{B} = \{\mathbf{x}^{(i)}\}, i = 1, \dots, m;$$

参数: γ, β ;

输出: $\{y^{(i)} = \text{BN}_{\gamma, \beta}(\mathbf{x}^{(i)})\}$

1 for $k = 1 \dots K$ do

2

$$\mu_{\mathcal{B}, k} = \frac{1}{m} \sum_{i=1}^m x_k^{(i)}, \quad // \text{mini-batch 均值}$$

$$\sigma_{\mathcal{B}, k}^2 = \frac{1}{m} \sum_{i=1}^m (x_k^{(i)} - \mu_{\mathcal{B}, k})^2. \quad // \text{mini-batch 方差}$$

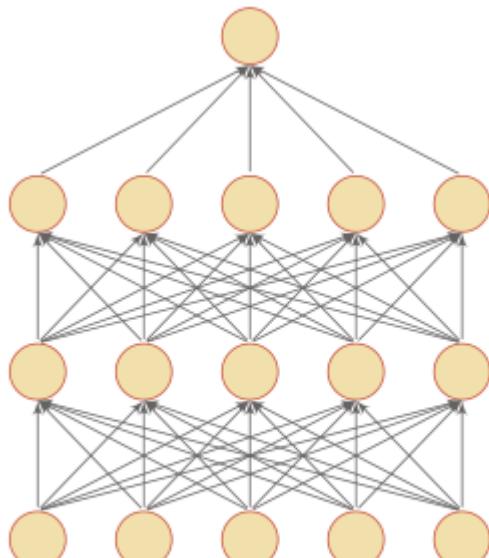
$$\hat{x}_k^{(i)} = \frac{x_k^{(i)} - \mu_{\mathcal{B}, k}}{\sqrt{\sigma_{\mathcal{B}, k}^2 + \epsilon}}, \forall i \quad // \text{归一化}$$

3

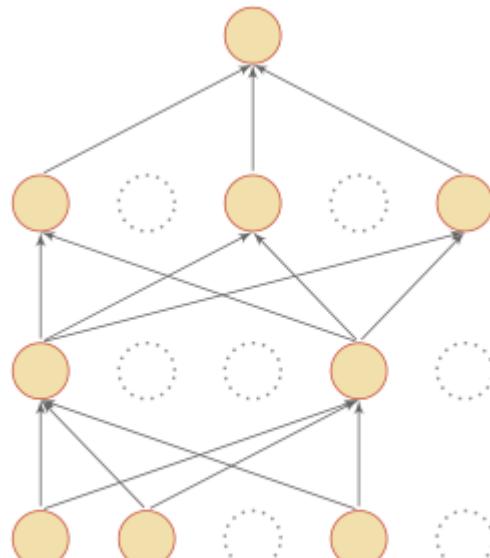
$$y_k^{(i)} = \gamma \hat{x}_k^{(i)} + \beta \equiv \text{BN}_{\gamma, \beta}(\mathbf{x}^{(i)}), \forall i \quad // \text{缩放和平移}$$

4 end

Dropout



(a) 标准网络



(b) Dropout 神经网络



经验

- ▶ 用 ReLU 作为激活函数
- ▶ 分类时用交叉熵作为损失函数
- ▶ SGD+mini-batch
- ▶ 每次迭代都重新随机排序
- ▶ 数据预处理（标准归一化）
- ▶ 动态学习率（越来越小）
- ▶ 用 L1 或 L2 正则化（跳过前几轮）
- ▶ dropout
- ▶ 数据增强

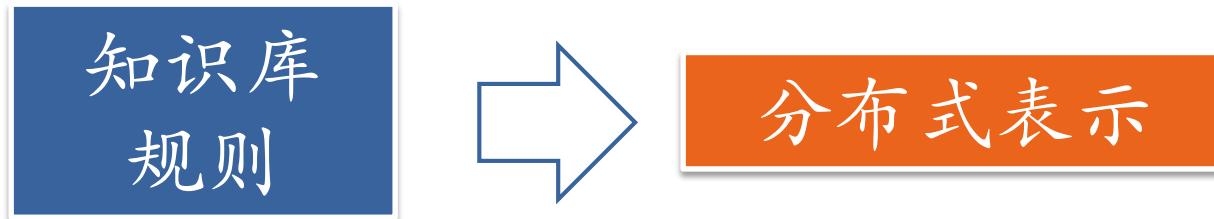


应用



语言表示

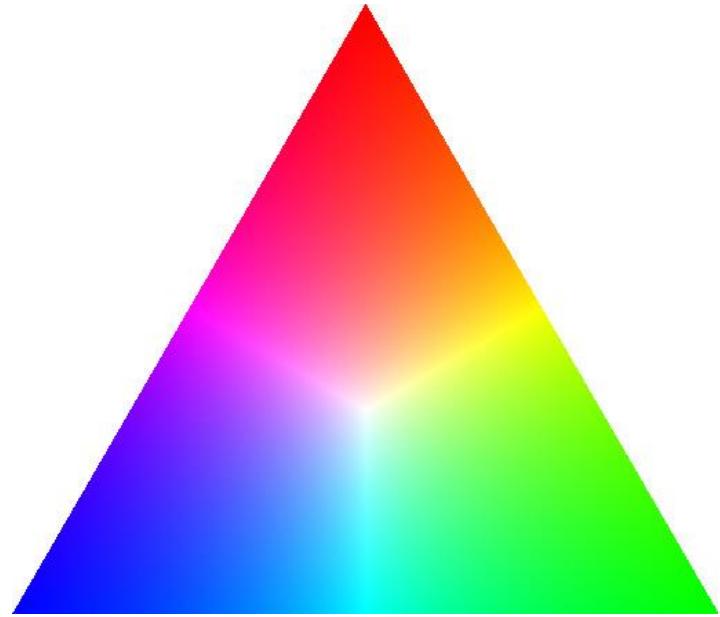
► 如何在计算机中表示语言的语义?



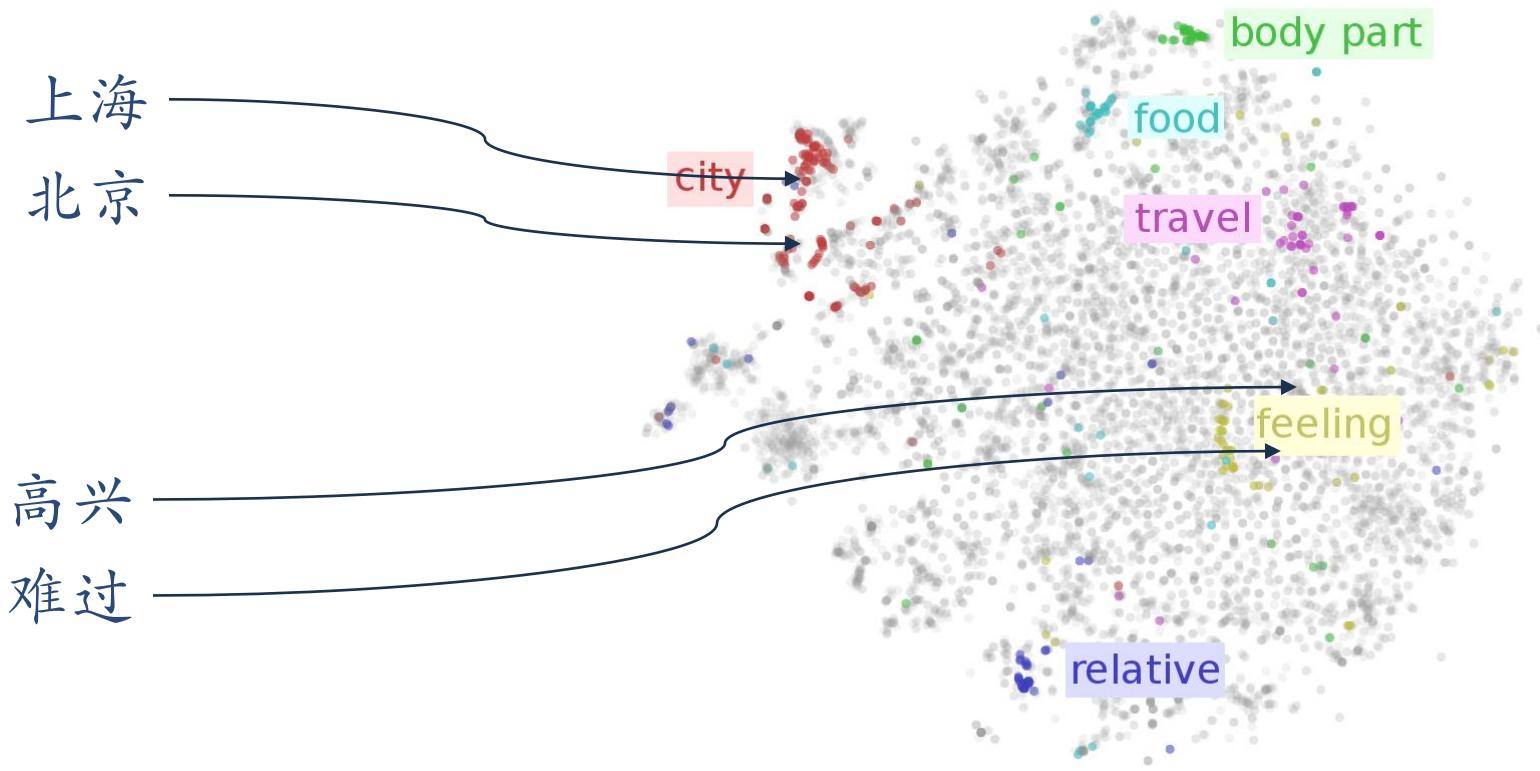
- 压缩、低维、稠密向量
- 用 $O(N)$ 个参数表示 $O(2^k)$ 区间
 - k 为非0参数， $k < N$

一个生活中的例子：颜色

命名	RGB值
红	[1,0,0]
绿	[0,1,0]
蓝	[0,0,1]
中国红	[0.67, 0.22, 0.12]
咖啡色	[0.64, 0.16, 0.16]



词嵌入 (Word Embeddings)



<https://indico.io/blog/visualizing-with-t-sne/>

分布式表示

--来自神经科学的证据



<http://www.nature.com/nature/journal/v532/n7600/full/nature17637.html>

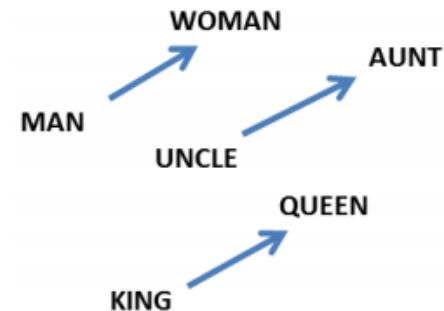
词嵌入



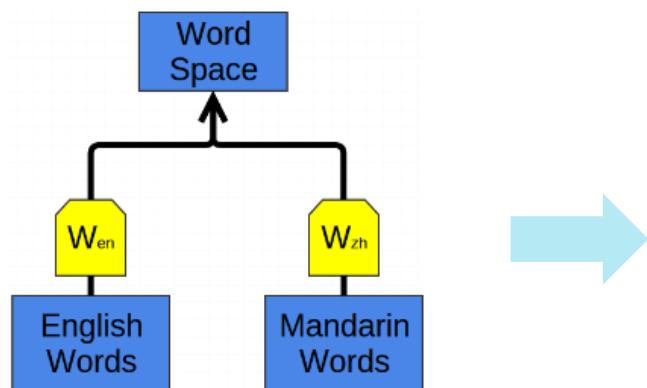
$$W("woman") - W("man") \approx W("aunt") - W("uncle")$$

$$W("woman") - W("man") \approx W("queen") - W("king")$$

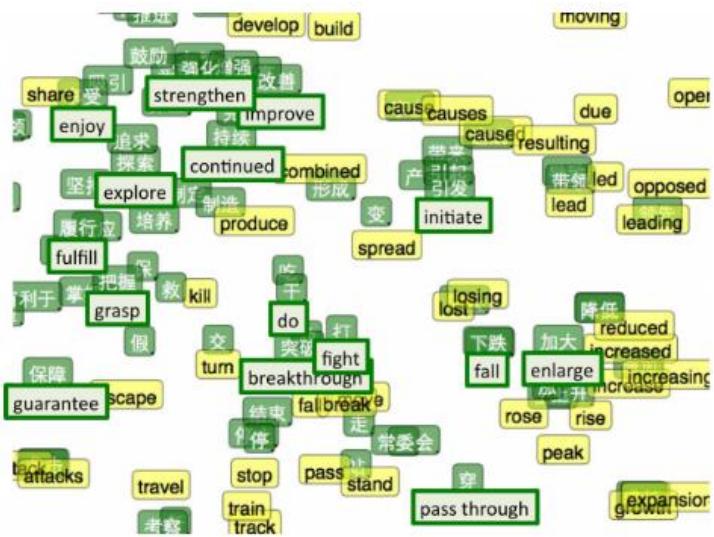
$$W("中国") - W("北京") \approx W("英国") - W("伦敦")$$



From Mikolov et al. (2013)

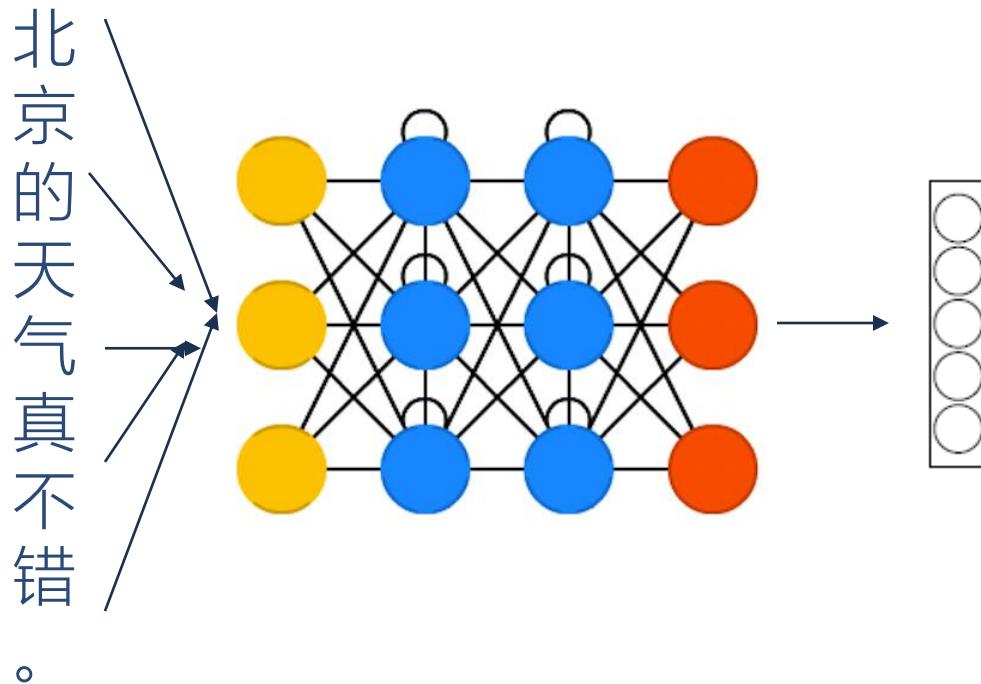


Socher et al. (2013)



语言表示学习

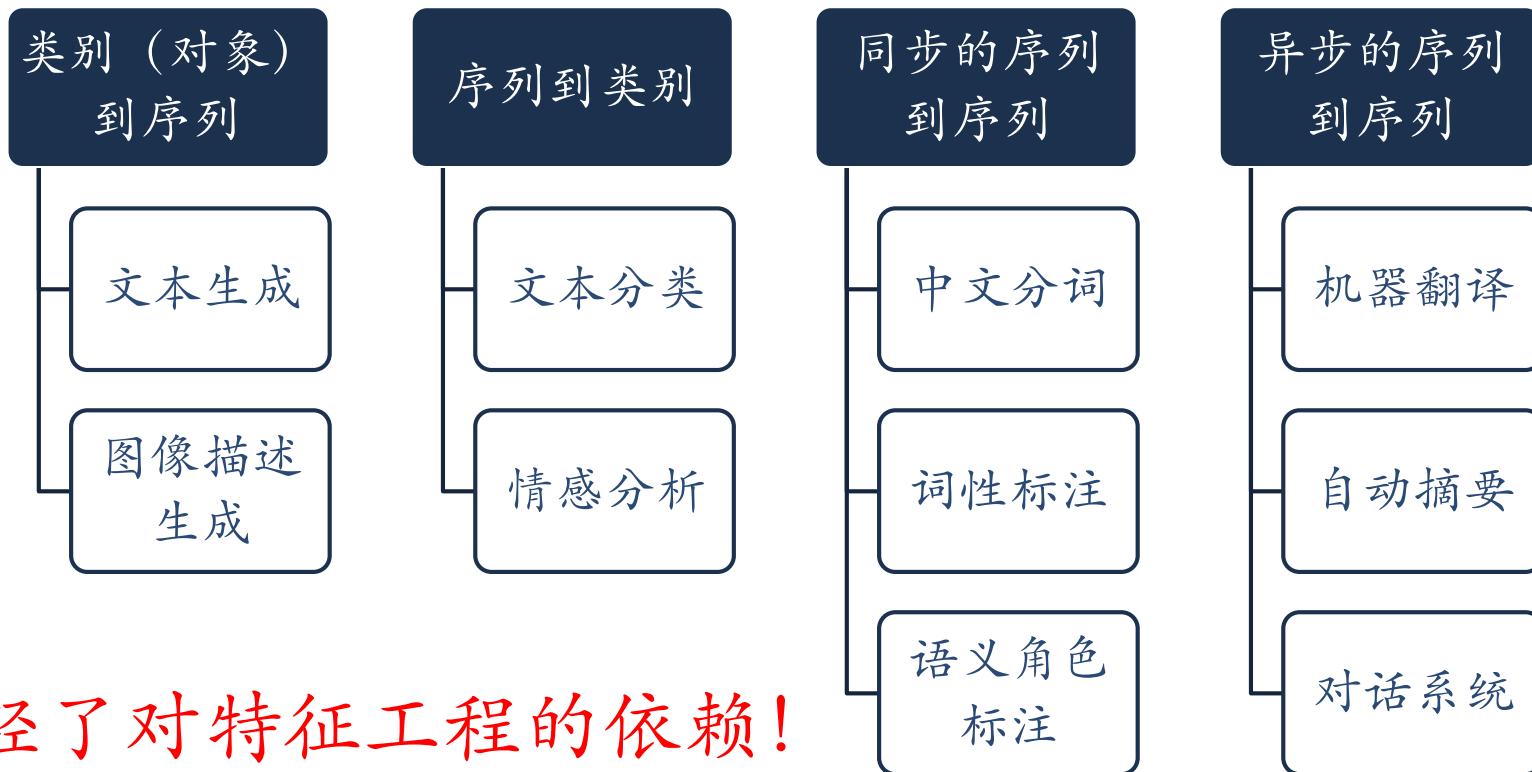
- ▶ 词
- ▶ 短语
- ▶ 组合语义模型
- ▶ 句子
 - ▶ 连续词袋模型
 - ▶ 序列模型
 - ▶ 递归组合模型
 - ▶ 卷积模型
- ▶ 篇章
- ▶ 层次模型





自然语言处理任务

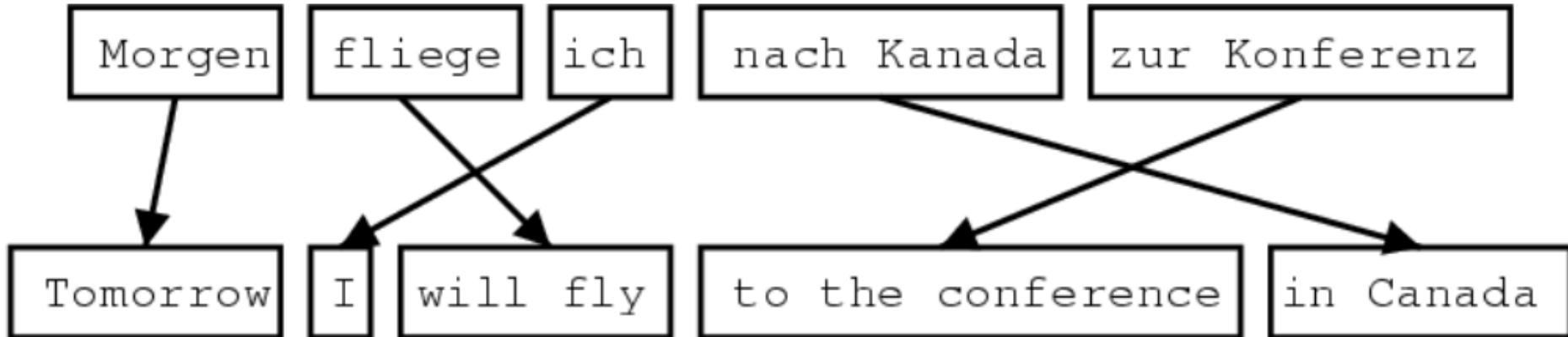
► 在得到字、句子表示之后，自然语言处理任务类型划分为



减轻了对特征工程的依赖！

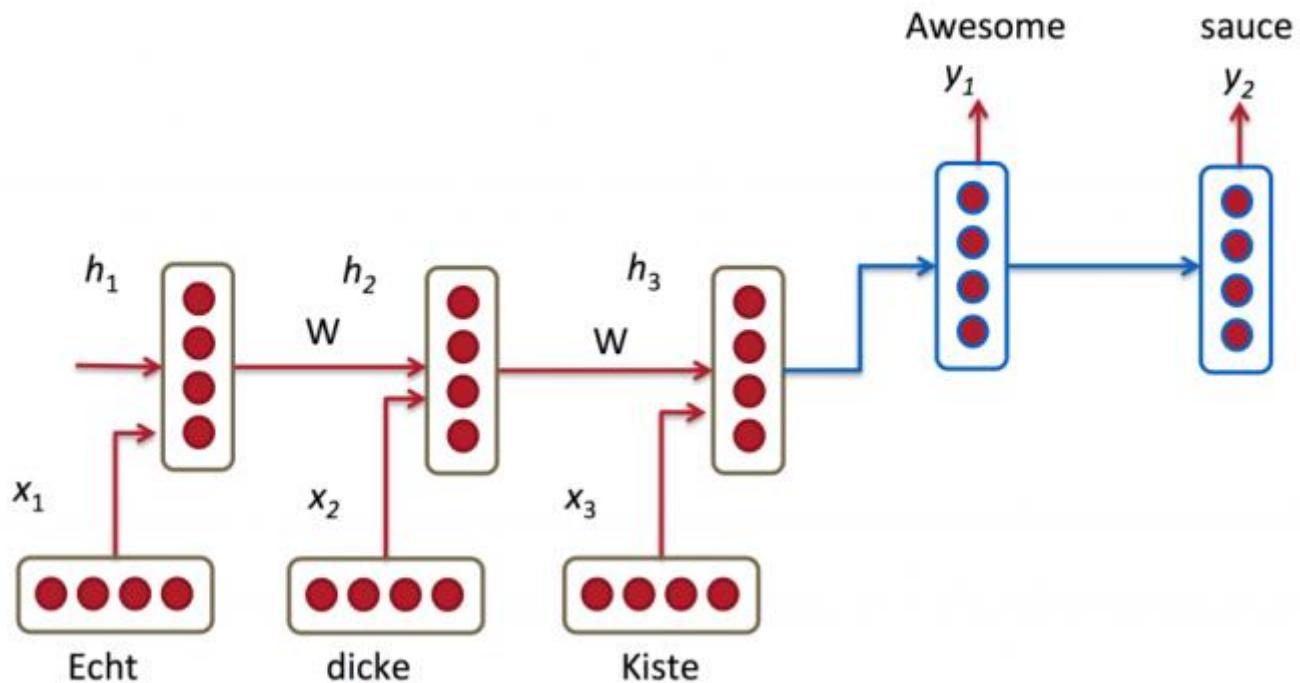
传统统计机器翻译

- ▶ 源语言: f
- ▶ 目标语言: e
- ▶ 模型: $\hat{e} = \operatorname{argmax}_e p(e|f) = \operatorname{argmax}_e p(f|e)p(e)$
- ▶ $p(f|e)$: 翻译模型
- ▶ $p(e)$: 语言模型

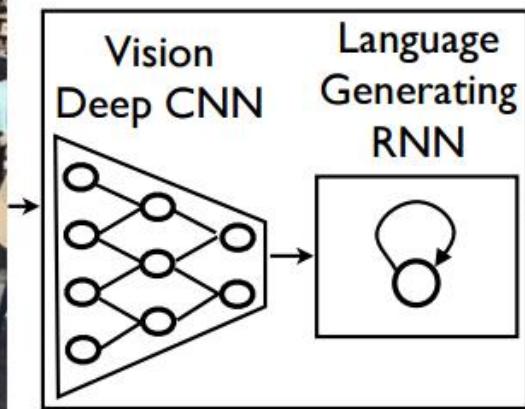


基于序列到序列的机器翻译

- ▶ 一个RNN用来编码
- ▶ 另一个RNN用来解码



看图说话



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

看图说话



Two dogs play in the grass.



A skateboarder does a trick on a ramp.



A dog is jumping to catch a frisbee.



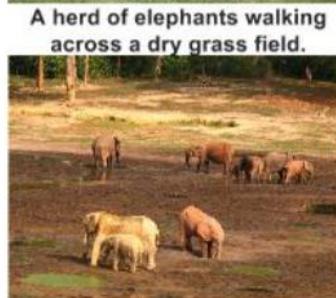
Two hockey players are fighting over the puck.



A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.



A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.



A yellow school bus parked in a parking lot.



Describes without errors

Describes with minor errors

Somewhat related to the image

Unrelated to the image

Figure 5. A selection of evaluation results, grouped by human rating.



生成LINUX内核代码

```
/*
 * If this error is set, we will need anything right after that BSD.
 */

static void action_new_function(struct s_stat_info *wb)
{
    unsigned long flags;
    int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
    buf[0] = 0xFFFFFFFF & (bit << 4);
    min(inc, slist->bytes);
    printk(KERN_WARNING "Memory allocated %02x/%02x, "
           "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
    div_u64_w(val, inb_p);
    spin_unlock(&disk->queue_lock);
    mutex_unlock(&s->sock->mutex);
    mutex_unlock(&func->mutex);
    return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
    if (tty == tty)
        disable_single_st_p(dev);
    pci_disable_spool(port);
```





作词机

► RNN在“学习”过汪峰全部作品后自动生成的歌词

- 我在这里中的夜里
- 就像一场是一种生命的意叶
- 就像我的生活变得在我一样
- 可我们这是一个知道
- 我只是一天你会怎吗
- 可我们这是我们的是不要为你
- 我们想这有一种生活的时候

<https://github.com/phunterlau/wangfeng-rnn>



作诗

白鹭窥鱼立，

Egrets stood, peeping fishes.

青山照水开。

Water was still, reflecting mountains.

夜来风不动，

The wind went down by nightfall,

明月见楼台。

as the moon came up by the tower.

满怀风月一枝春，

Budding branches are full of romance.

未见梅花亦可人。

Plum blossoms are invisible but adorable.

不为东风无此客，

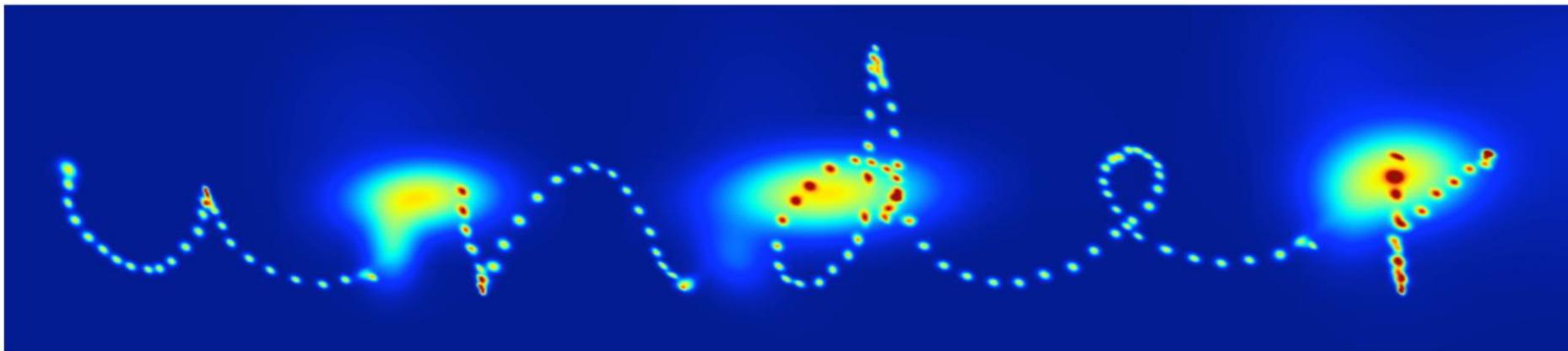
With the east wind comes Spring.

世间何处是前身。

Where on earth do I come from?

写字

- ▶ 把一个字母的书写轨迹看作是一连串的点。一个字母的“写法”其实是每一个点相对于前一个点的偏移量，记为 $(\text{offset } x, \text{offset } y)$ 。再增加一维取值为0或1来记录是否应该“提笔”。



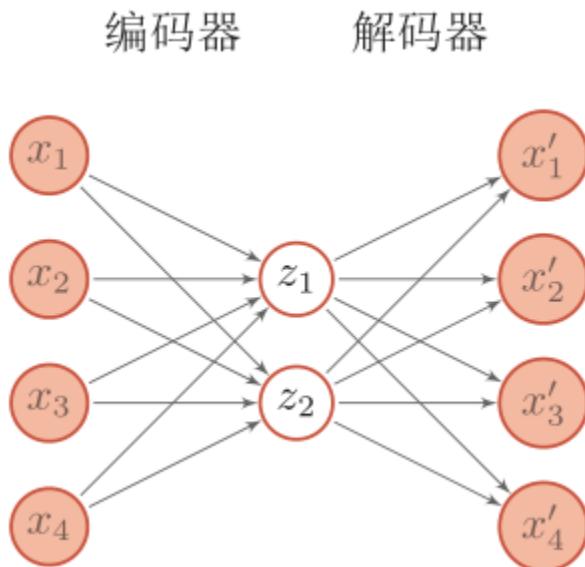


无监督模型

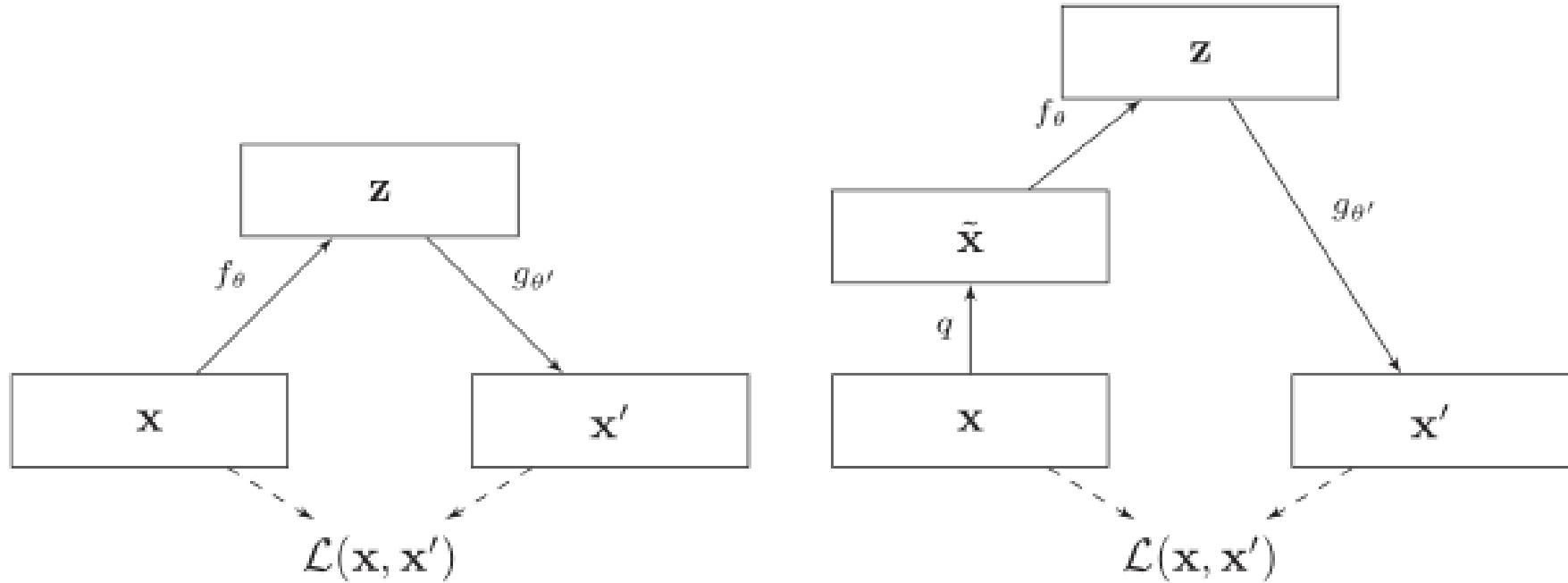
自编码器

- ▶ 编码器 (Encoder) $f: \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$
- ▶ 解码器 (Decoder) $g: \mathbb{R}^{d'} \rightarrow \mathbb{R}^d.$
- ▶ 自编码器的学习目标是最小化重构错误：

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^N \|\mathbf{x}^{(i)} - g(f(\mathbf{x}^{(i)}))\|^2 \\ &= \sum_{i=1}^N \|\mathbf{x}^{(i)} - f \circ g(\mathbf{x}^{(i)})\|^2\end{aligned}$$



降噪自编码器





注意力机制与外部记忆



长期依赖

- ▶ 简单循环神经网络存在长期依赖问题
 - ▶ (LSTM网络) 引入一个近似线性依赖的记忆单元来存储远距离的信息。
 - ▶ 记忆单元的存储能力和其大小相关。如果增加记忆单元的大小，网络的参数也随之增加。
- ▶ 改进方法
 - ▶ 注意力机制
 - ▶ 外部记忆



注意力

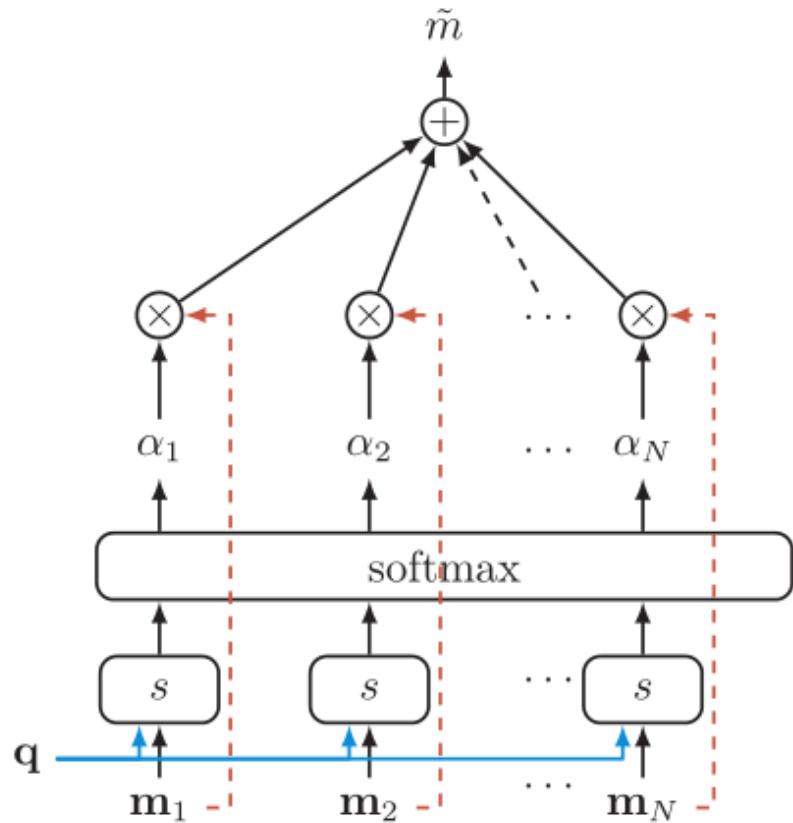
- ▶ 当一个人在吵闹的鸡尾酒会上和朋友聊天时，尽管周围噪音干扰很多，他还是可以听到朋友的谈话内容，而忽略其他人的声音。
- ▶ 同时，如果未注意到的背景声中有重要的词（比如他的名字），他会马上注意到。

鸡尾酒会效应

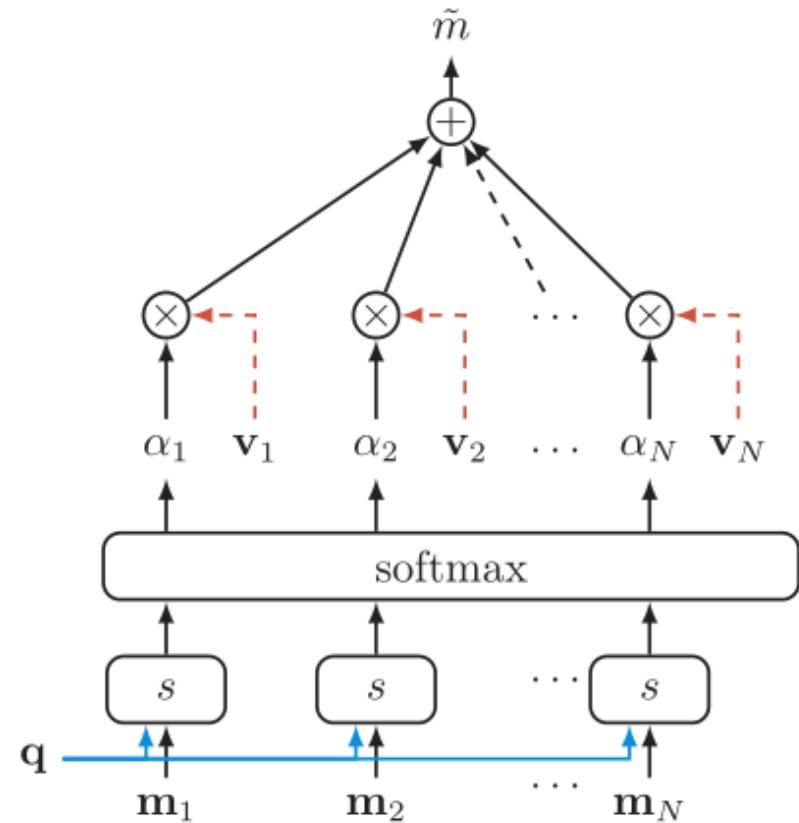
注意力



注意力模型



(a) 普通模式



(b) 键值对模式

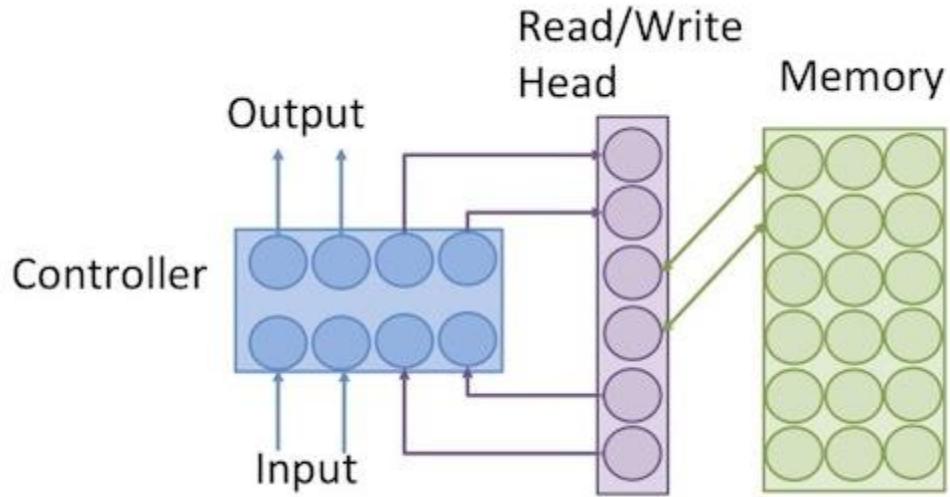


外部记忆

- ▶ 外部记忆定义为矩阵 $M \in R^{K \times D}$
 - ▶ K 是记忆片段的数量， D 是每个记忆片段的大小
 - ▶ 外部记忆类型
 - ▶ 只读
 - ▶ Memory Network
 - ▶ RNN 中的 h_t
 - ▶ 可读写
 - ▶ NTM
 - ▶ 如何读写?
注意力机制

神经图灵机

- ▶ 组件
 - ▶ 控制器
 - ▶ 外部记忆
 - ▶ 读写操作
- ▶ 整个架构可微分

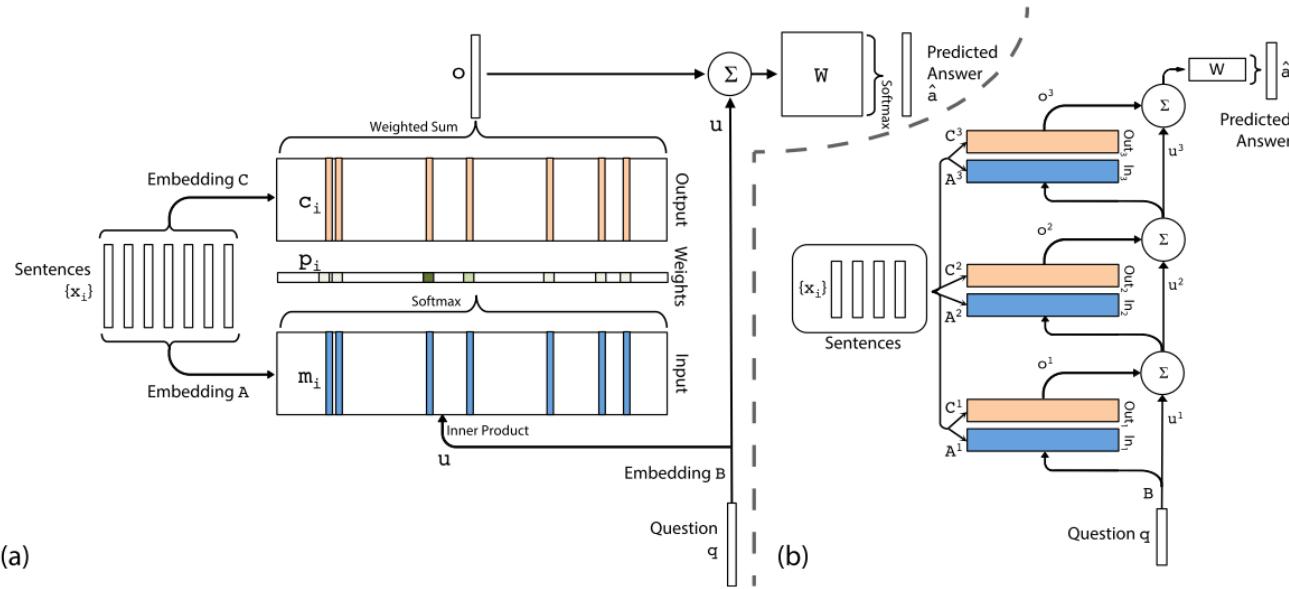


图片来源：

<http://cpmarkchang.logdown.com/posts/279710-neural-network-neural-turing-machine>

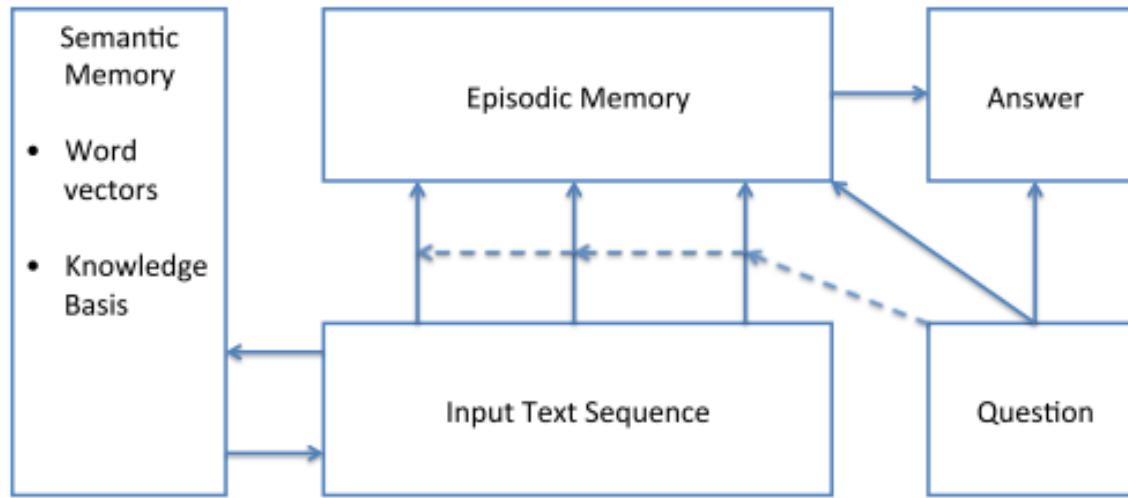
Graves, A., Wayne, G., & Danihelka, I. (2014). Neural Turing Machines. Arxiv, 1–26.
<http://arxiv.org/abs/1410.5401>

记忆网络

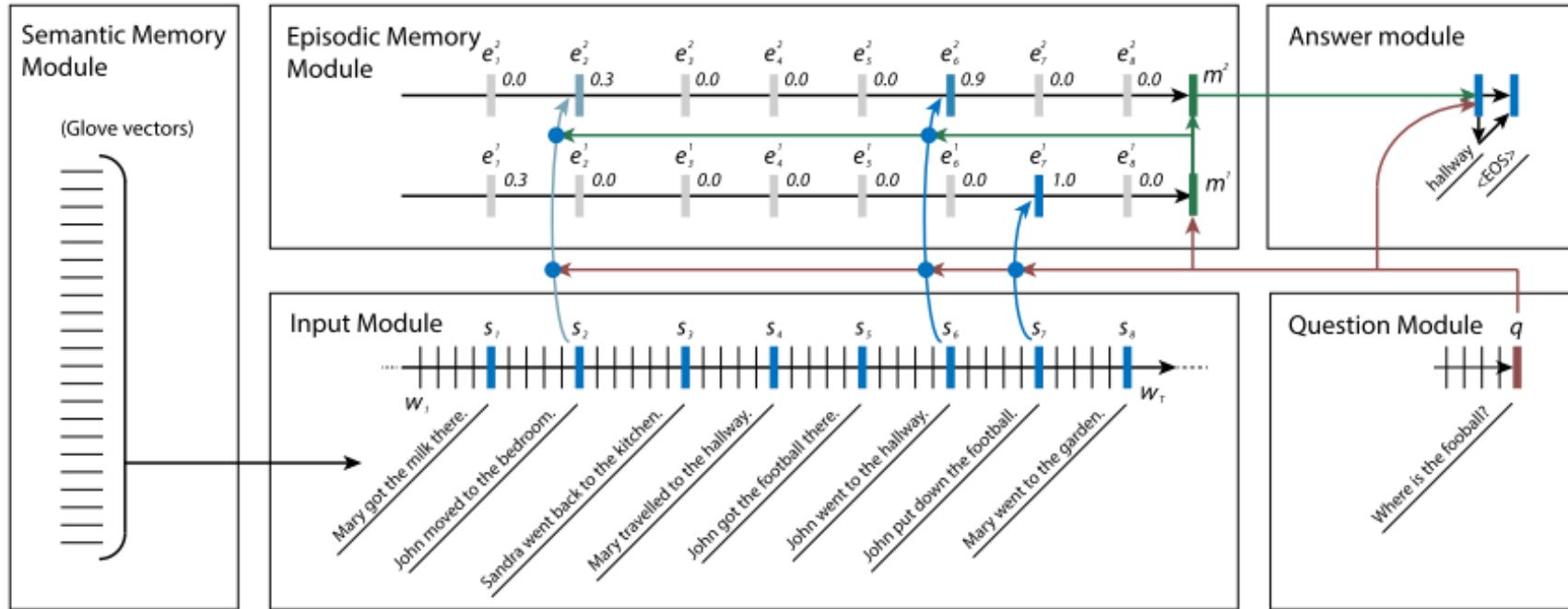


Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). End-To-End Memory Networks, 1–11. <http://arxiv.org/abs/1503.08895>

Dynamic Memory Networks



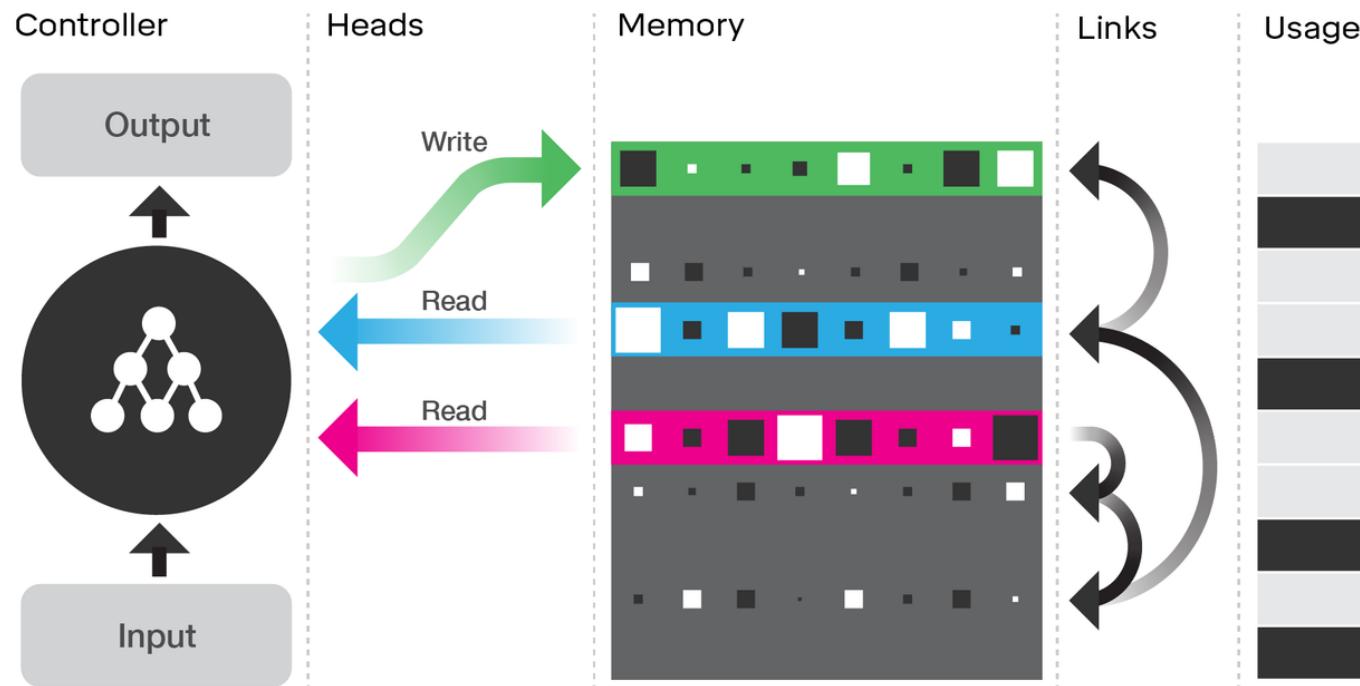
Dynamic Memory Networks



Differentiable neural computers (DeepMind)



Illustration of the DNC architecture

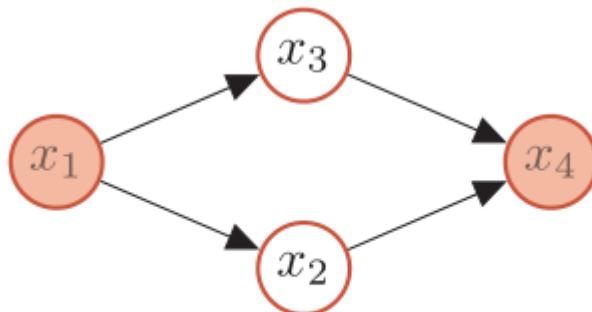




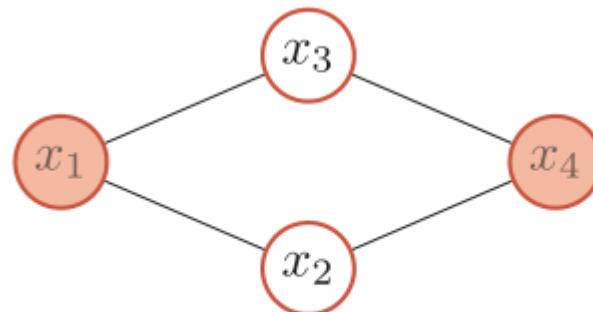
概率图模型

概率图模型

▶ 概率图模型是指一种用图结构来描述多元随机变量之间条件独立关系的概率模型。



(a) 有向图：贝叶斯网络



(b) 无向图：马尔可夫随机场



局部马尔可夫性质

- ▶ 利用图模型的局部马尔可夫性，我们可以对多元变量的联合概率进行简化，从而降低建模的复杂度。
- ▶ 以贝叶斯网络为例，

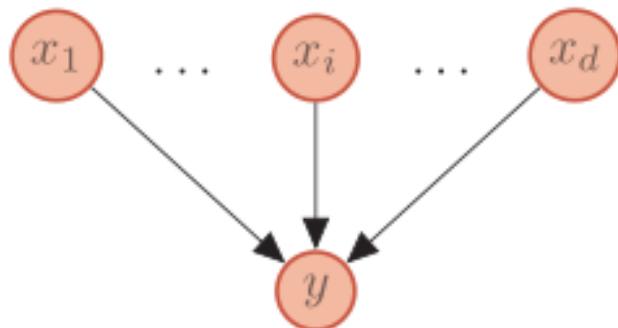
$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3),$$

- ▶ 是4个局部条件概率的乘积，这样只需要 $1 + 2 + 2 + 4 = 9$ 个独立参数。

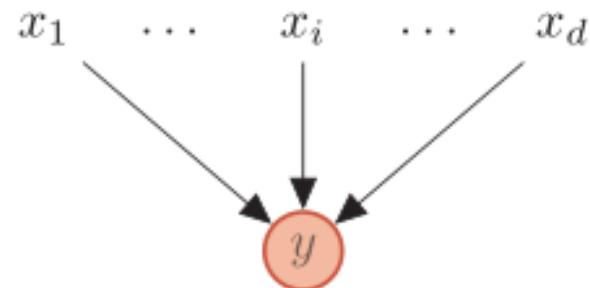
Sigmoid信念网络

- Sigmoid信念网络中的变量为二值变量，取值为 {0,1}。

$$p(x_k = 1 | x_{\pi_k}) = \sigma(w_0 + \sum_{x_i \in \pi_k} w_i x_i),$$



(a) Sigmoid 信念网络



(b) Logistic 回归



图模型的几个要素

► 图结构

► 推断

给定一组变量, **推断**(Inference)是指在观测到部分变量 $\mathbf{e} = \{e_1, e_2, \dots, e_m\}$ 时, 计算其它变量的某个子集 $\mathbf{q} = \{q_1, q_2, \dots, q_n\}$ 的后验概率分布 $p(\mathbf{q}|\mathbf{e})$ 。

► 参数学习



参数估计

- ▶ 在贝叶斯网络中，所有变量 x 的联合概率分布可以分解为每个随机变量 x_k 的局部条件概率的连乘形式。
- ▶ 假设每个局部条件概率 $p(x_k | x_{\pi(k)})$ 的参数为 θ_k ，则 x 的对数似然函数为

$$\log p(\mathbf{x}, \Theta) = \sum_{k=1}^K \log p(x_k | x_{\pi_k}, \theta_k),$$



马尔可夫网络

► 马尔可夫网络的联合分布可以表示为

$$\begin{aligned} P(\mathbf{X} = \mathbf{x}) &= \frac{1}{Z} \prod_{c \in \mathcal{C}} \exp(-E(X_c)) \\ &= \frac{1}{Z} \exp\left(\sum_{c \in \mathcal{C}} -E(X_c)\right) \end{aligned}$$

► 其中， $E(X_c)$ 为能量函数， Z 是配分函数。



图模型的几个要素

► 图结构

► 推断

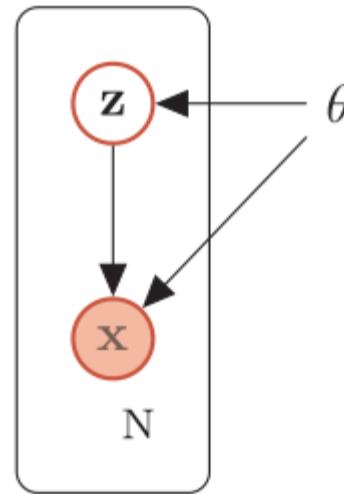
给定一组变量,推断(Inference)是指在观测到部分变量 $\mathbf{e} = \{e_1, e_2, \dots, e_m\}$ 时, 计算其它变量的某个子集 $\mathbf{q} = \{q_1, q_2, \dots, q_n\}$ 的后验概率分布 $p(\mathbf{q}|\mathbf{e})$ 。

► 参数估计

隐变量->EM算法

EM算法

- ▶ 假设有一组变量，有部分变量是不可观测的，如何进行参数估计呢？
- ▶ 期望最大化算法
 - ▶ Expectation-Maximum, EM算法



- E步 (Expectation): 固定参数 θ , 找到一个分布 $q(\mathbf{z})$ 使得 $L(q, \theta; \mathbf{x})$ 最大。
- M步 (Maximization): 固定 $q(\mathbf{z})$, 找到一组参数 θ , 使得 $L(q, \theta; \mathbf{x})$ 最大。

EM: 1-d example



$$P(x_i | b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b | x_i) = \frac{P(x_i | b)P(b)}{P(x_i | b)P(b) + P(x_i | a)P(a)}$$

$$a_i = P(a | x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \dots + b_n x_n}{b_1 + b_2 + \dots + b_n}$$

$$\sigma_b^2 = \frac{b_1 (x_1 - \mu_b)^2 + \dots + b_n (x_n - \mu_b)^2}{b_1 + b_2 + \dots + b_n}$$

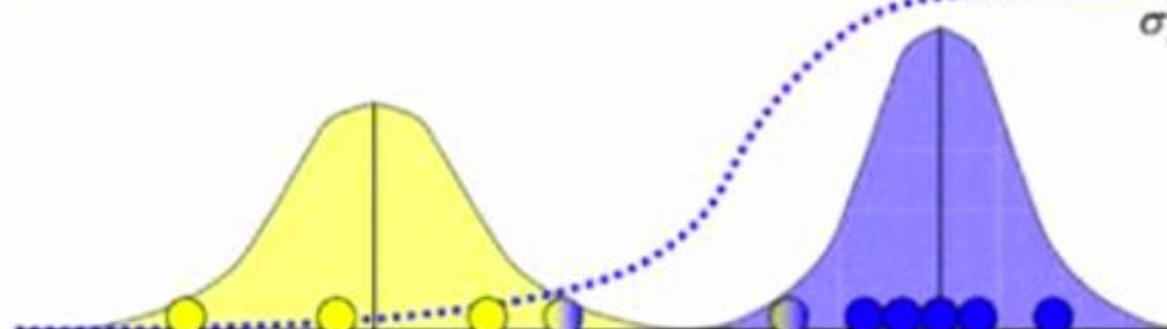
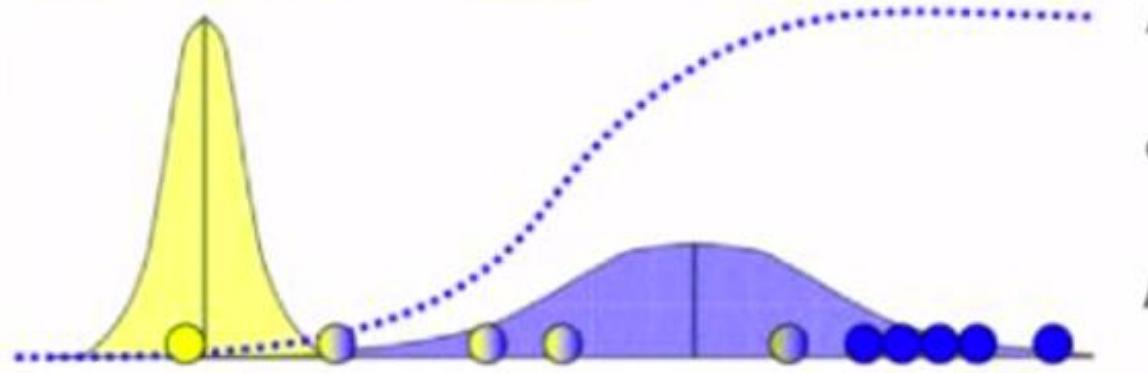
$$\mu_a = \frac{a_1 x_1 + a_2 x_2 + \dots + a_n x_n}{a_1 + a_2 + \dots + a_n}$$

$$\sigma_a^2 = \frac{a_1 (x_1 - \mu_a)^2 + \dots + a_n (x_n - \mu_a)^2}{a_1 + a_2 + \dots + a_n}$$

could also estimate priors:

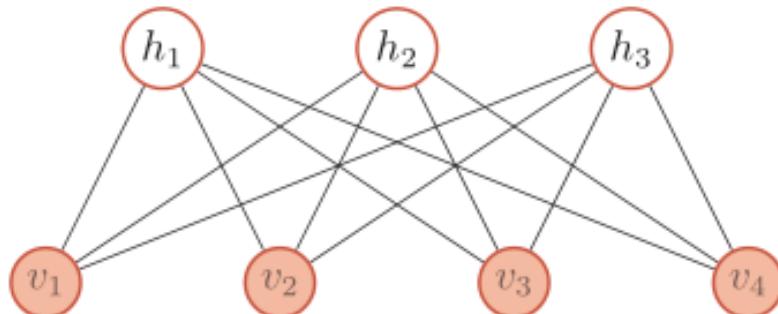
$$P(b) = (b_1 + b_2 + \dots + b_n) / n$$

$$P(a) = 1 - P(b)$$



受限玻尔兹曼机

- ▶ 受限玻尔兹曼机（Restricted Boltzmann Machines，RBM）是一个二分图结构的无向图模型。
- ▶ 在受限玻尔兹曼机中，变量可以为两组，分别为隐藏层和可见层（或输入层）。
- ▶ 节点变量的取值为0或1。
- ▶ 和两层的全连接神经网络的结构相同。

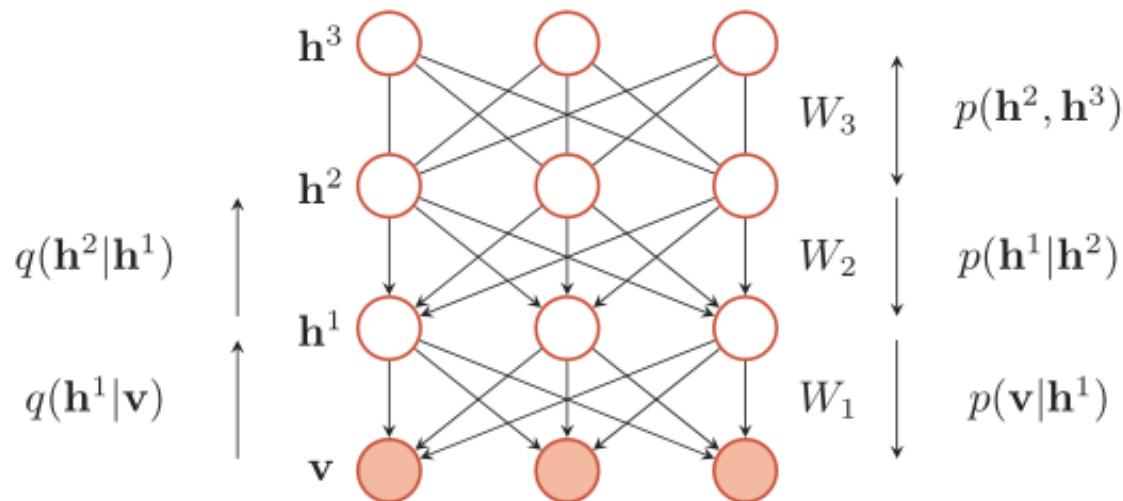


$$p(v_i = 1 | \mathbf{h}) = \sigma \left(a_i + \sum_j w_{i,j} h_j \right),$$

$$p(h_j = 1 | \mathbf{v}) = \sigma \left(b_j + \sum_i w_{i,j} v_i \right),$$

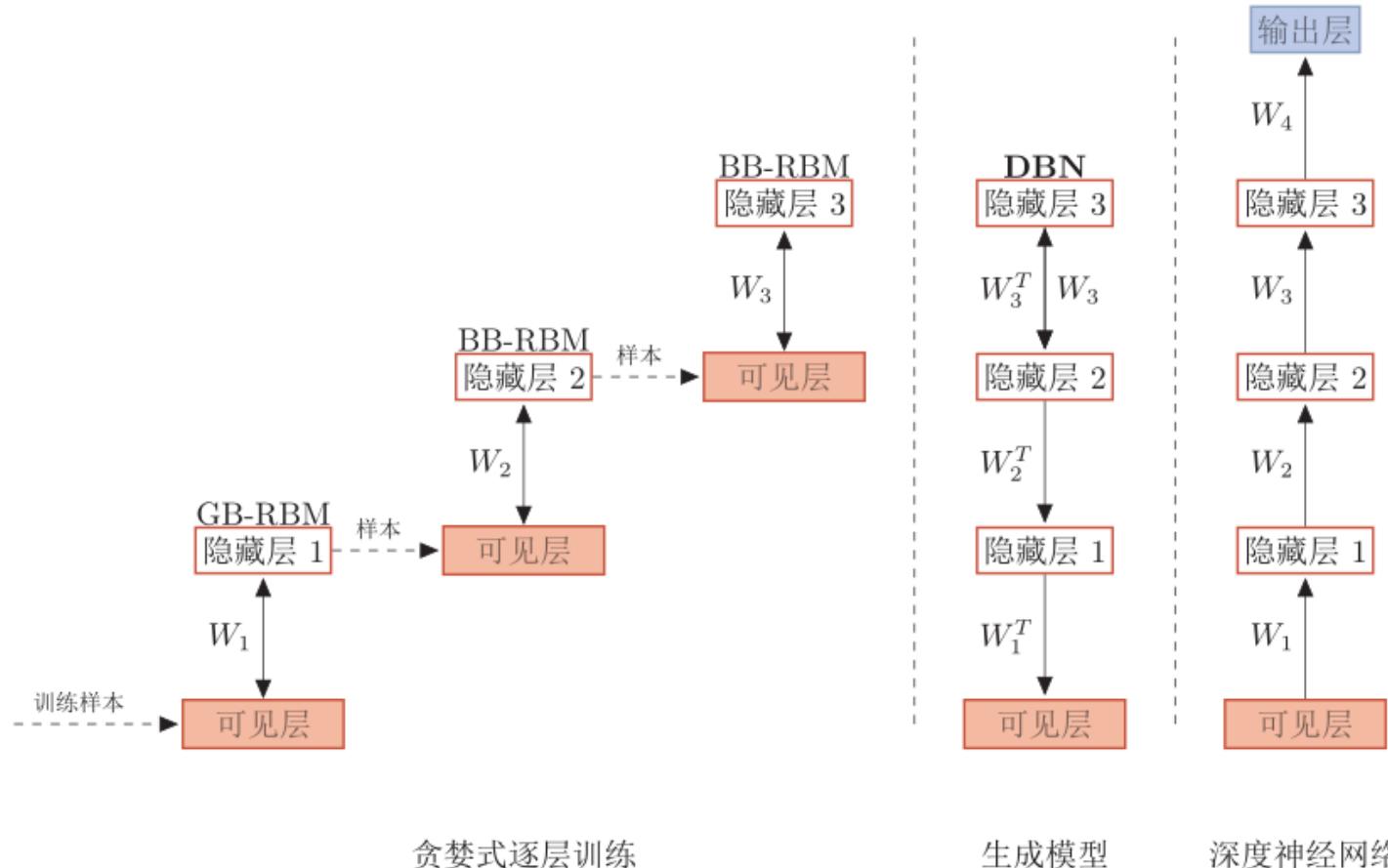
深度信念网络

- ▶ 深度信念网络 (Deep Belief Network, DBN) 是深度的有向的概率图模型，其图结构由多层的节点构成。
- ▶ 和全连接的神经网络结构相同。
- ▶ 顶部的两层为一个无向图，可以看做是一个受限玻尔兹曼机。



逐层训练

▶ 逐层训练是能够有效训练深度模型的最早的方法。





► 图模型与神经网络的关系？

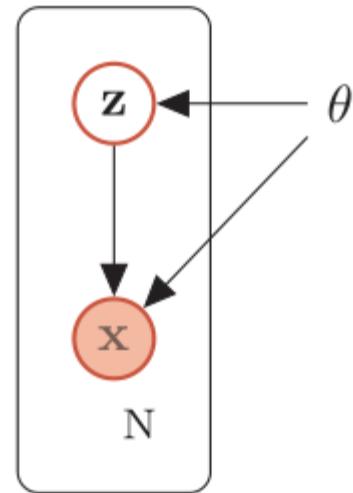
- 网络结构类似
- 节点
 - 图模型的节点是随机变量，其图结构的主要功能是用来描述变量之间的依赖关系，一般是稀疏连接。
 - 神经网络中的节点是神经元，是一个计算节点。
 - 如果将神经网络中每个神经元看做是一个二值随机变量，那神经网络就变成一个sigmoid信念网络。
- 解释性
 - 图模型中的每个变量一般有着明确的解释，变量之间依赖关系一般是人工来定义。
 - 而神经网络中的神经元则没有直观的解释。



深度生成模型

生成模型

- ▶ 生成模型指一系列用于随机生成可观测数据的模型。
- ▶ 生成数据 x 的过程可以分为两步进行：
 - ▶ 根据隐变量的先验分布 $p(z;\theta)$ 进行采样，得到样本 z ；
 - ▶ 根据条件分布 $p(x|z;\theta)$ 进行采样，得到 x 。





深度生成模型

- ▶ 深度生成模型就是利用神经网络来建模条件分布 $p(x|z;\theta)$ 。
- ▶ 对抗生成式网络 (Generative Adversarial Network, GAN)
[Goodfellow et al., 2014]
- ▶ 变分自编码器 (Variational Autoencoder, VAE) [Kingma and Welling, 2013, Rezende et al., 2014]。

变分自编码器(VAE)

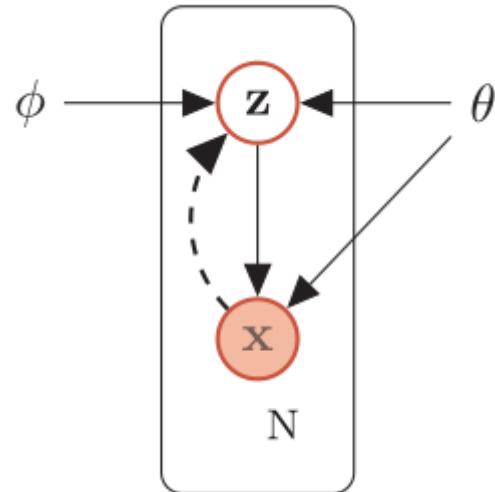
▶ 变分自编码器的模型结构可以分为两个部分：

▶ 寻找后验分布 $p(z|x;\theta)$ 的变分近似 $q(z|x;\phi^*)$ ；

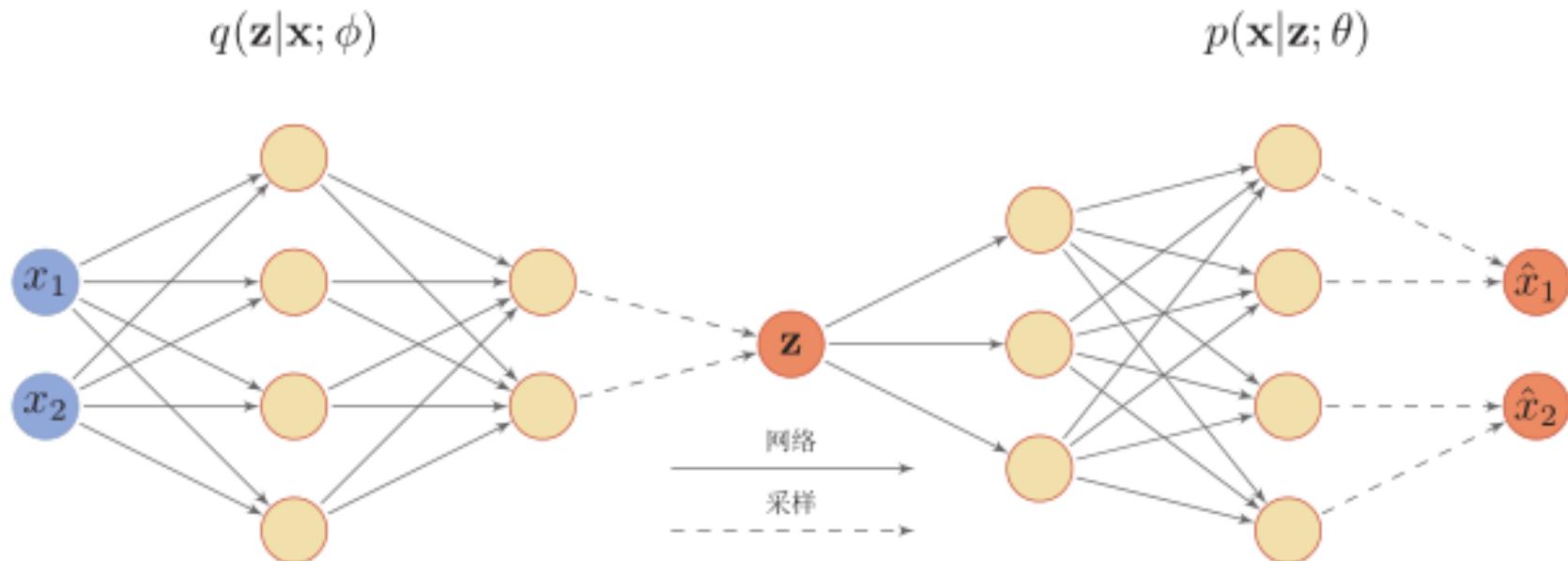
□ 变分推断：用简单的分布 q 去近似复杂的分 $p(z|x;\theta)$

▶ 在已知 $q(z|x;\phi_*)$ 的情况下，估计更好的生成 $p(x|z;\theta)$ 。

用神经网络来替代

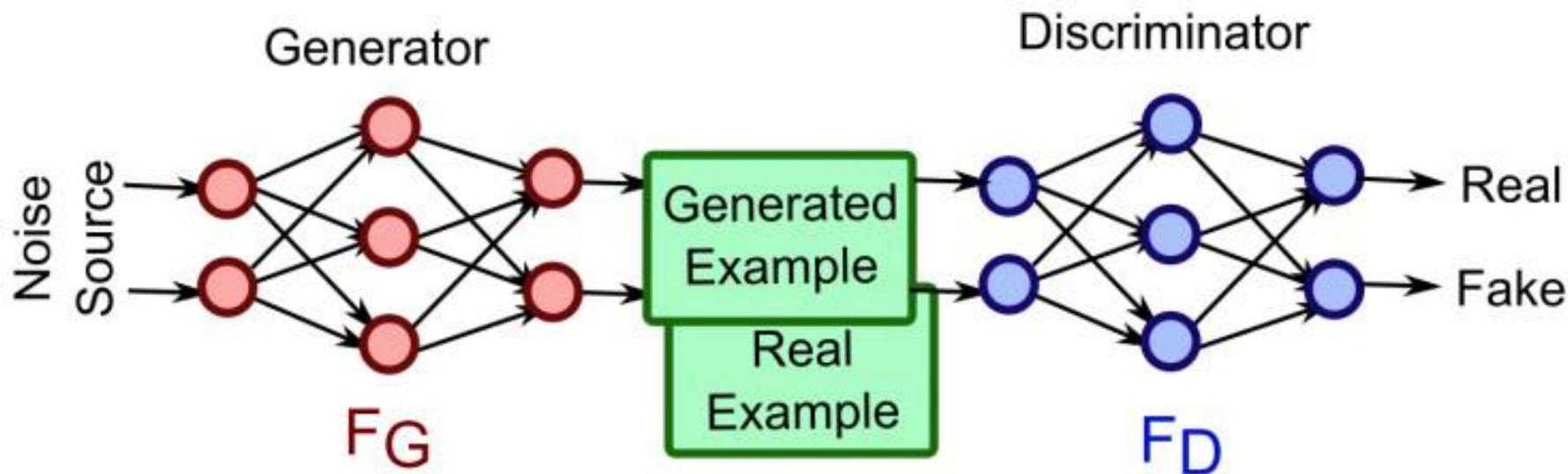


变分自编码器



生成对抗网络

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$



例子

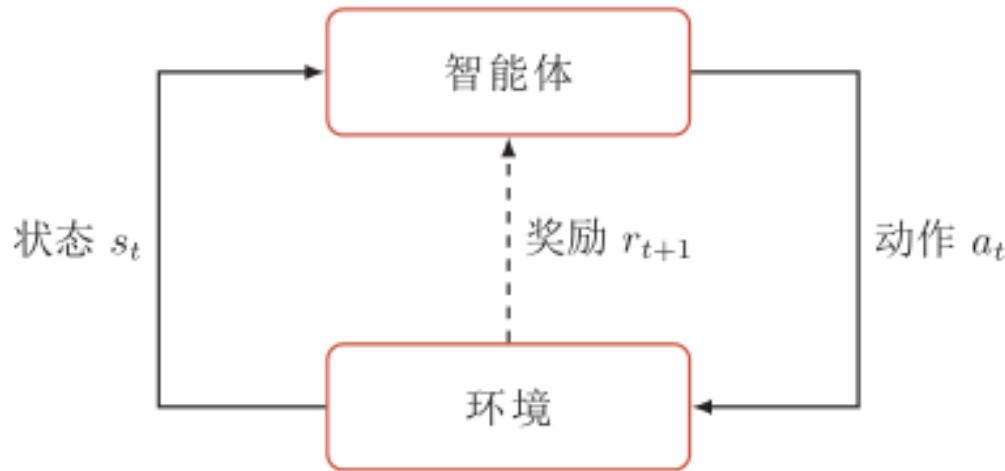




深度强化学习

强化学习

- ▶ 强化学习问题可以描述为一个智能体从与环境的交互中不断学习以完成特定目标（比如取得最大奖励值）。
- ▶ 强化学习就是智能体不断与环境进行交互，并根据经验调整其策略来最大化其长远的所有奖励的累积值。





强化学习

► 智能体 (Agent)

► 感知外界环境的状态 (State) 和奖励反馈 (Reward) , 并进行学习和决策。智能体的决策功能是指根据外界环境的状态来做出不同的动作 (Action) , 而学习功能是指根据外界环境的奖励来调整策略。

► 环境 (Environment)

► 智能体外部的所有事物, 并受智能体动作的影响而改变其状态, 并反馈给智能体相应的奖励。



强化学习中的基本要素

- 环境的状态集合: \mathcal{S} ;
- 智能体的动作集合: \mathcal{A} ;
- 状态转移概率: $p(s'|s, a)$, 即智能体根据当前状态 s 做出一个动作 a 之后, 下一个时刻环境处于不同状态 s' 的概率;
- 即时奖励: $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, 即智能体根据当前状态做出一个动作之后, 环境会反馈给智能体一个奖励, 这个奖励和动作之后下一个时刻的状态有关。



马尔可夫决策过程

► 马尔可夫过程

$$p(s_{t+1}|s_t, a_t, \dots, s_0, a_0) = p(s_{t+1}|s_t, a_t),$$

► 值函数：从状态s开始，执行策略 π 得到的期望总回报

$$\begin{aligned} V^\pi(s) &= \sum_{(s_0=s, a_0, s_1, a_1, \dots)} p(s_0=s, a_0, s_1, a_1, \dots) \sum_{t=0}^{\infty} \gamma^t r_{t+1} \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid \pi, s_0 = s \right], \end{aligned}$$



状态值函数

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots | \pi, s_0 = s] \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} p(s'|s, a) \mathbb{E}_\pi[R(s, a, s') + \gamma(r_2 + \gamma^1 r_3 + \cdots) | \pi, s_1 = s'] \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} p(s'|s, a) \left(R(s, a, s') + \gamma \mathbb{E}_\pi[r_2 + \gamma^1 r_3 + \cdots | \pi, s_1 = s'] \right) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} p(s'|s, a) \left(R(s, a, s') + \gamma V^\pi(s') \right). \end{aligned}$$



最优策略

- ▶ 最优策略：存在一个最优的策略 π_* ，其在所有状态上的期望回报最大

$$\forall s, \quad \pi^* = \arg \max_{\pi} V^\pi(s).$$

- ▶ 状态-动作值函数：即初始状态为 s ，并执行动作 a 后的所能得到的期望总回报。

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid \pi, s_0 = s, a_0 = a \right] \\ &= \sum_{s' \in \mathcal{S}} p(s'|s, a) \left(R(s, a, s') + \gamma V^\pi(s') \right). \end{aligned}$$



强化学习算法

▶ 基于模型的强化学习算法

- ▶ 基于MDP过程：状态转移概率 $p(s' | s, a)$ 和奖励函数 $R(s, a, s')$
- ▶ 值迭代
- ▶ 策略迭代

▶ 模型无关的强化学习

- ▶ 无MDP过程
- ▶ 蒙特卡罗采样方法
- ▶ 时序差分学习



深度强化学习

- ▶ 深度强化学习是将强化学习和深度学习结合在一起，用强化学习来定义问题和优化目标，用深度学习来解决状态表示、策略表示等问题。
- ▶ 三种不同的结合强化学习和深度学习的方式，分别用深度神经网络来建模强化学习中的**值函数**、**策略**、**模型**，然后用误差反向传播算法来优化目标函数。



基于值函数的深度强化学习

- 为了在连续的状态和动作空间中计算值函数 $Q_\pi(s, a)$ ，我们可以用一个函数 $Q_\theta(s, a)$ 来表示近似计算，称为值函数近似（Value Function Approximation）

$$Q_\theta(s) = \begin{bmatrix} Q_\theta(s, a_1) \\ \vdots \\ Q_\theta(s, a_m) \end{bmatrix} \approx \begin{bmatrix} Q^\pi(s, a_1) \\ \vdots \\ Q^\pi(s, a_m) \end{bmatrix}.$$



深度Q网络

输入: 环境 E , 动作空间 \mathcal{A} , 折扣率 γ , 学习率 α

```
1 初始化经验池  $\mathcal{D}$ , 容量为  $N$ ;  
2 随机初始化  $Q$  网络的参数  $\theta$ ;  
3 随机初始化目标  $Q$  网络的参数  $\theta^- = \theta$ ;  
4 repeat  
5   初始化起始状态  $s$ ;  
6   repeat  
7     在状态  $s$ , 选择动作  $a = \pi^\epsilon$ ;  
8     执行动作  $a$ , 观测环境, 得到即时奖励  $r$  和新的状态  $s'$ ;  
9     将  $s, a, r, s'$  放入  $\mathcal{D}$  中;  
10    从  $\mathcal{D}$  中采样  $ss, aa, rr, ss'$ ;  
11     $y = \begin{cases} rr, & ss' \text{ 为终止状态,} \\ r_j + \gamma \max_{a'} Q_{\theta^-}(ss', a'), & \text{否则} \end{cases};$   
12    以  $(y - Q_\theta(s, a))^2$  为损失函数来训练  $Q$  网络;  
13     $s \leftarrow s'$ ;  
14    每隔  $C$  步,  $\theta^- \leftarrow \theta$ ;  
15 until  $s$  为终止状态;  
16 until  $\forall s, a, Q_\theta(s, a)$  收敛;
```

输出: Q 网络



基于策略函数的深度强化学习

- ▶ 可以直接用深度神经网络来表示一个参数化的从状态空间到动作空间的映射函数： $a = \pi_\theta(s)$ 。
- ▶ 最优的策略是使得在每个状态的总回报最大的策略，因此策略搜索的目标函数为

$$\mathcal{L}(s; \theta) = \mathbb{E}[G(\tau) | \pi_\theta] = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | \pi_\theta \right],$$



策略梯度 (Policy Gradient)

- ▶ 策略搜索是通过寻找参数 θ 使得目标函数 $L(s; \theta)$ 最大。
- ▶ 梯度上升：



$$\begin{aligned}\frac{\partial \mathcal{L}(s; \theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \int p_\theta(\tau) G(\tau) d\tau \\ &= \int \nabla_\theta p_\theta(\tau) G(\tau) d\tau \\ &= \int p_\theta(\tau) \frac{1}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau) G(\tau) d\tau \\ &= \mathbb{E} [\nabla_\theta \log p_\theta(\tau) G(\tau)],\end{aligned}$$



未来研究方向



未来研究方向

- ▶ 任意结构网络
- ▶ 非监督学习（如何评价？）
- ▶ 长期依赖问题
- ▶ 自然语言理解和推理
- ▶ 优化
- ▶ 分布式训练、专用硬件
- ▶ 生物学
- ▶ 深度增强学习

参考：Geoff Hinton, Yoshua Bengio, Yann LeCun,
NIPS 2015 Deep Learning Tutorial



推荐课程

► CS224d: Deep Learning for Natural Language Processing

► <http://cs224d.stanford.edu/>

► 斯坦福大学 Richard Socher

► 主要讲解自然语言处理领域的各种深度学习模型

► CS231n: Convolutional Neural Networks for Visual Recognition

► <http://cs231n.stanford.edu/>

► 斯坦福大学 Fei-Fei Li Andrej Karpathy

► 主要讲解CNN、RNN在图像领域的应用



推荐配置 (4卡)

品名	规格配置	数量	单价	总价
机箱	Carbide Series® Air 540	1	800	800
CPU	英特尔 酷睿i7 5960X 八核X99	1	7609	7609
主板	Asus X99-E WS	1	4400	4400
内存	16GB DDR4ECC	8	1499	11992
硬盘	3TB 7200转64M SATA3	2	620	1240
电源	海盗船 (CORSAIR) AX1500i	1	3500	3500
GPU	1080Ti	4	6000	24000
固态硬盘	512GB	1	1000	1000
风扇	九州风神 captain	1	1299	1299
				55840



推荐配置 (8卡)

品名	型号	数量	单价	合计
服务器	超微4028准系统	1	32000	32000
CPU	至强 E5 2640 V4	2	5100	10200
内存	64G	4	3700	14800
固态硬盘	512G SATA3	1	1000	1000
硬盘	3TB SATA3	2	620	1240
GPU	1080Ti	8	6000	48000
			合计	107240



谢 谢

<https://nndl.github.io/>