深度学习

司博

2019-02-24

课前约定

1.空杯心态；

2.积极参与；

3.手机静音；

4.不看电脑；

5.接打电话到室外。

目录
CONTENTS

PART 01

01

VGGNet

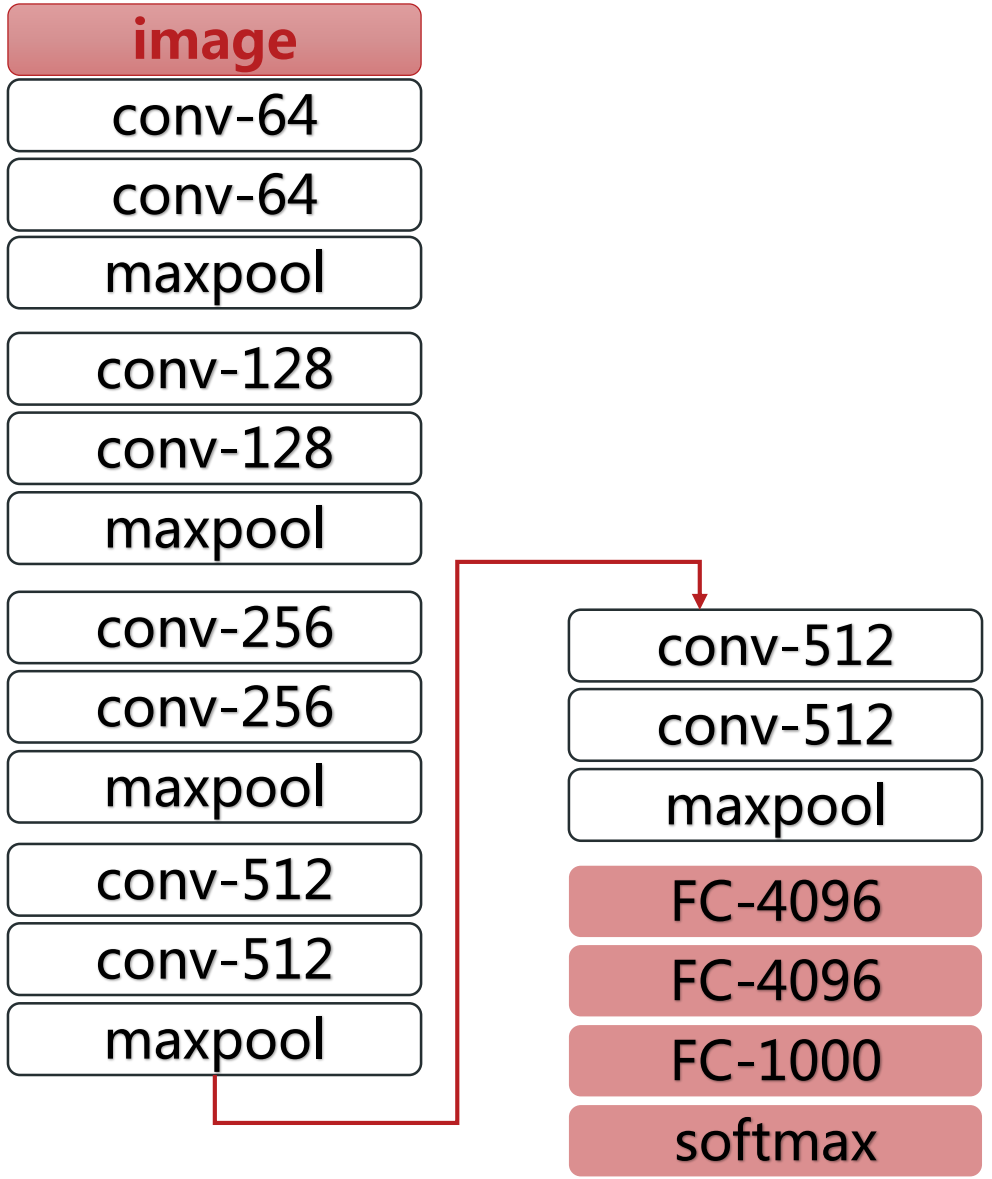
网络设计

关键设计:

- 3x3 卷积核 —— very small
- 卷积步长为1 —— no loss of information

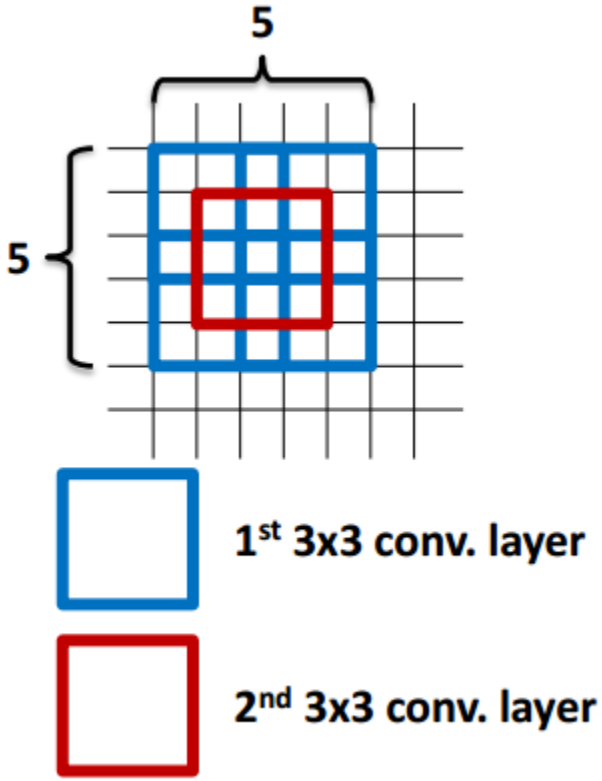
其它细节:

- ReLU
- 5个池化层 (2倍减量)
- 无归一化
- 3个全连接层



为什么使用3x3卷积层

- 堆叠的卷积层具有大的感受野
  - 2个3x3 —— 1个5x5
  - 3个3x3 —— 1个7x7
- 更多的非线性
- 更少的待学习参数
  - 每个网络约少140M



训练过程

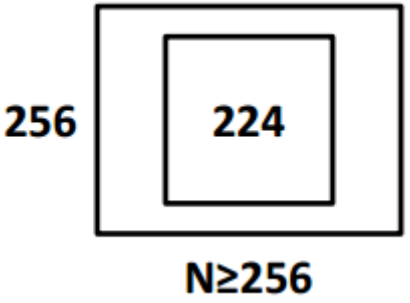
- 求解器
  - 多元逻辑回归
  - 小批量动量梯度下降
  - 随机失活和权重衰减规律化
  - 快速收敛（74次迭代）

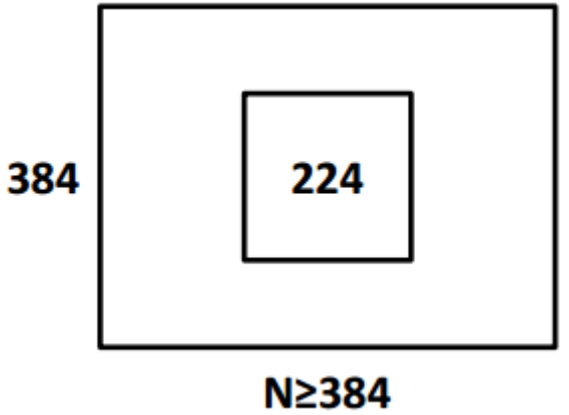
- 初始化
  - 大量ReLU层——易失速
  - 大多数浅网（11层）使用高斯初始化
  - 深层网
    - 用11层网络初始化前4个凸起和FC层
    - 其它层使用随机高斯

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

训练过程（2）

- 多尺度训练
  - 随机裁剪ConvNet输入
    - 固定大小224x224
  - 训练图像尺寸不同
    - 256xN
    - 384xN
    - 随机裁剪——随机图像尺寸（尺度抖动）
- 标准抖动
  - 随机水平翻转
  - 随机RGB颜色偏移

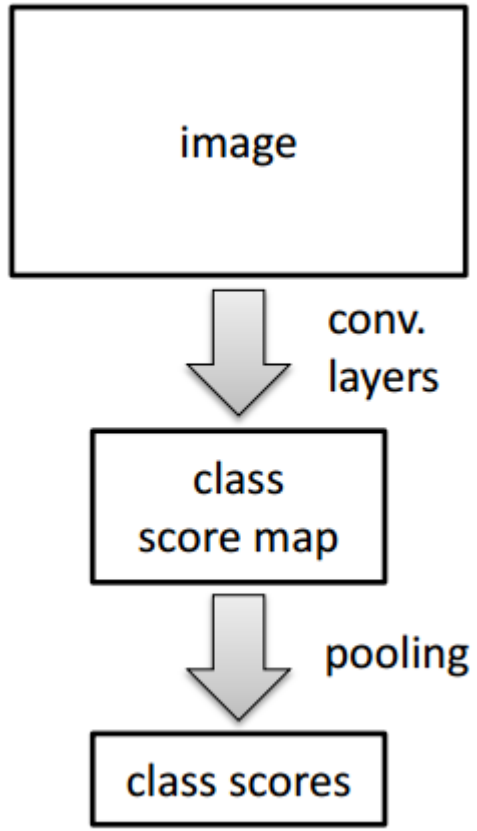




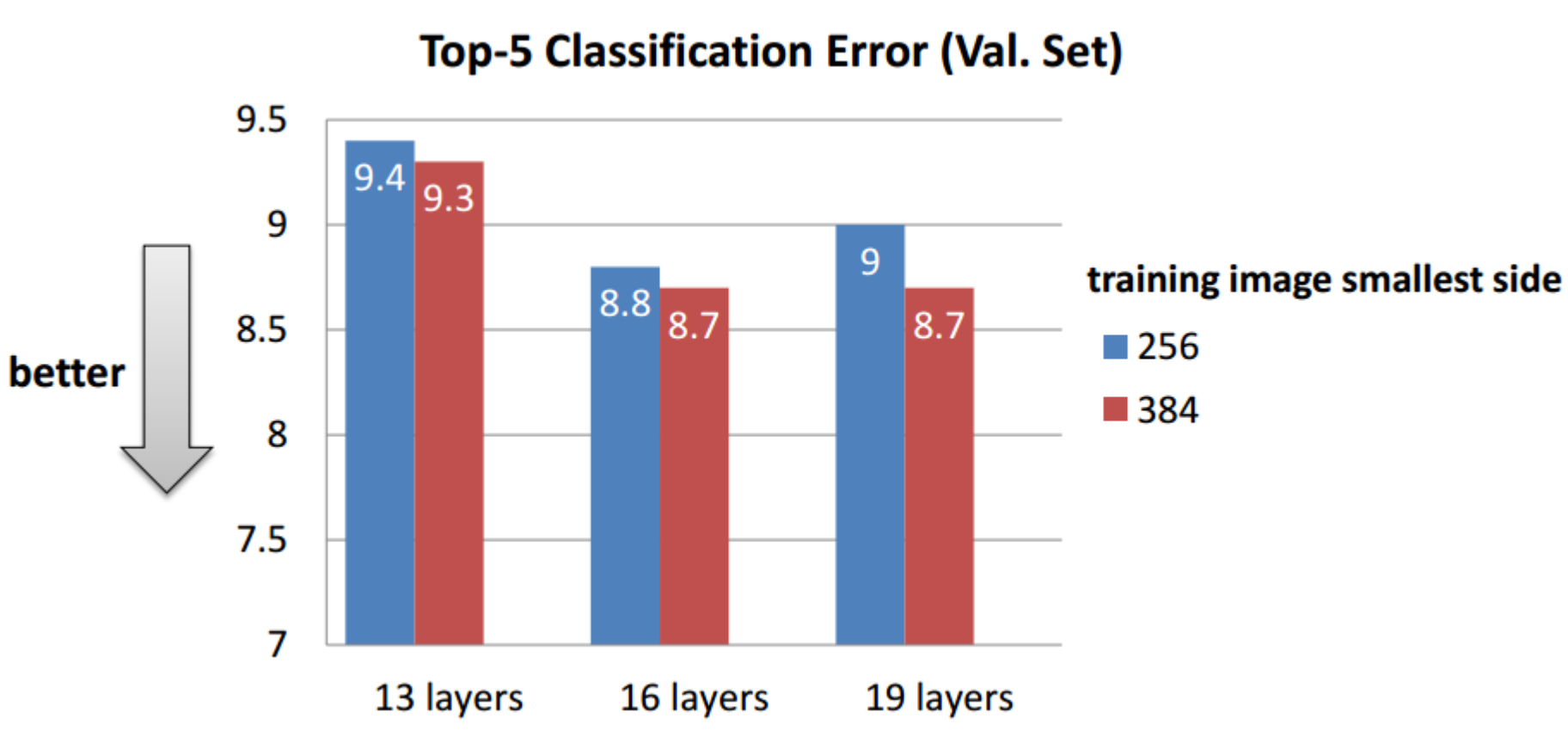
测试

- 在整个图像上的密集应用
  - 全连接层转换为卷积层
  - 和池化类得分图
  - 比将网络应用于多裁剪图像更有效

- 抖动
  - 多图像尺寸：256xN，384xN等
  - 水平翻转
  - 类得分平均化

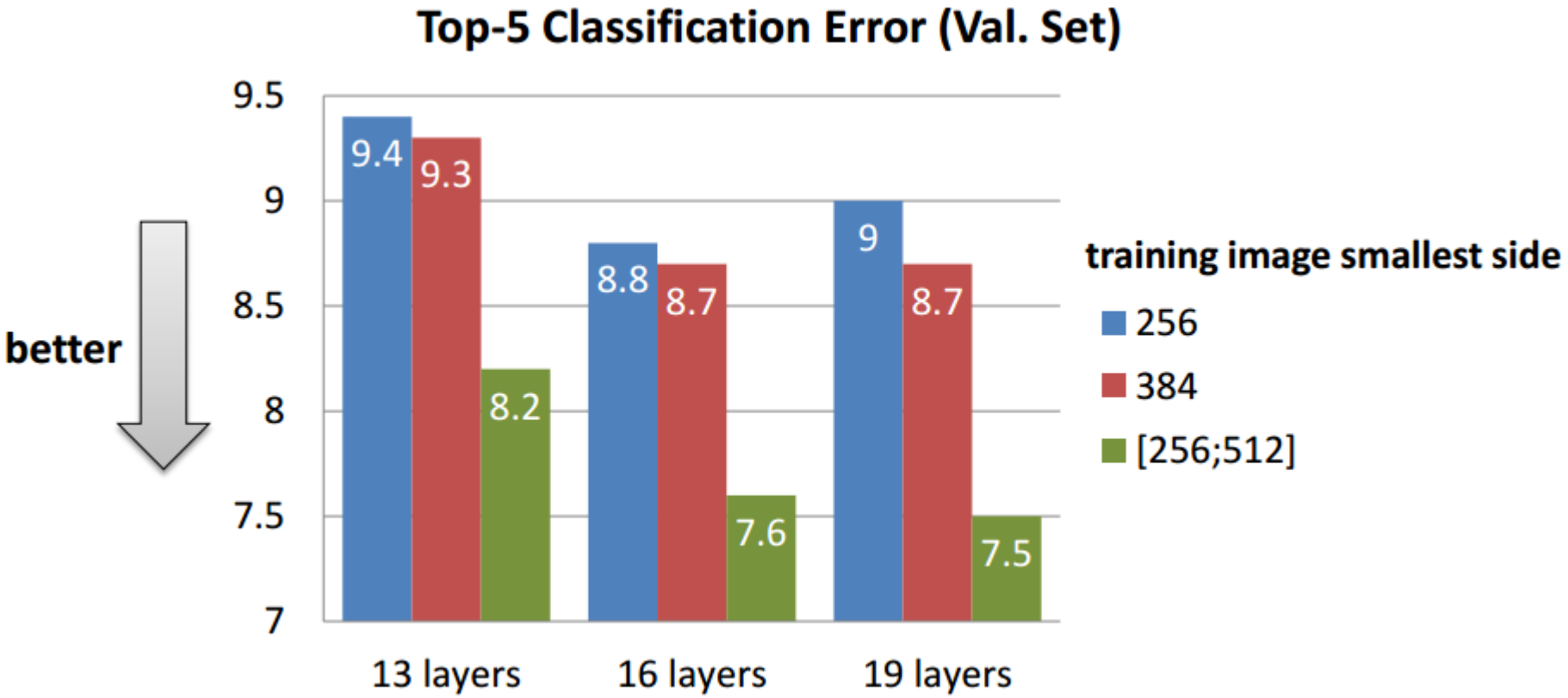


比较——固定训练大小



- 使用16或19层网络在384xN图像上训练，效果最好

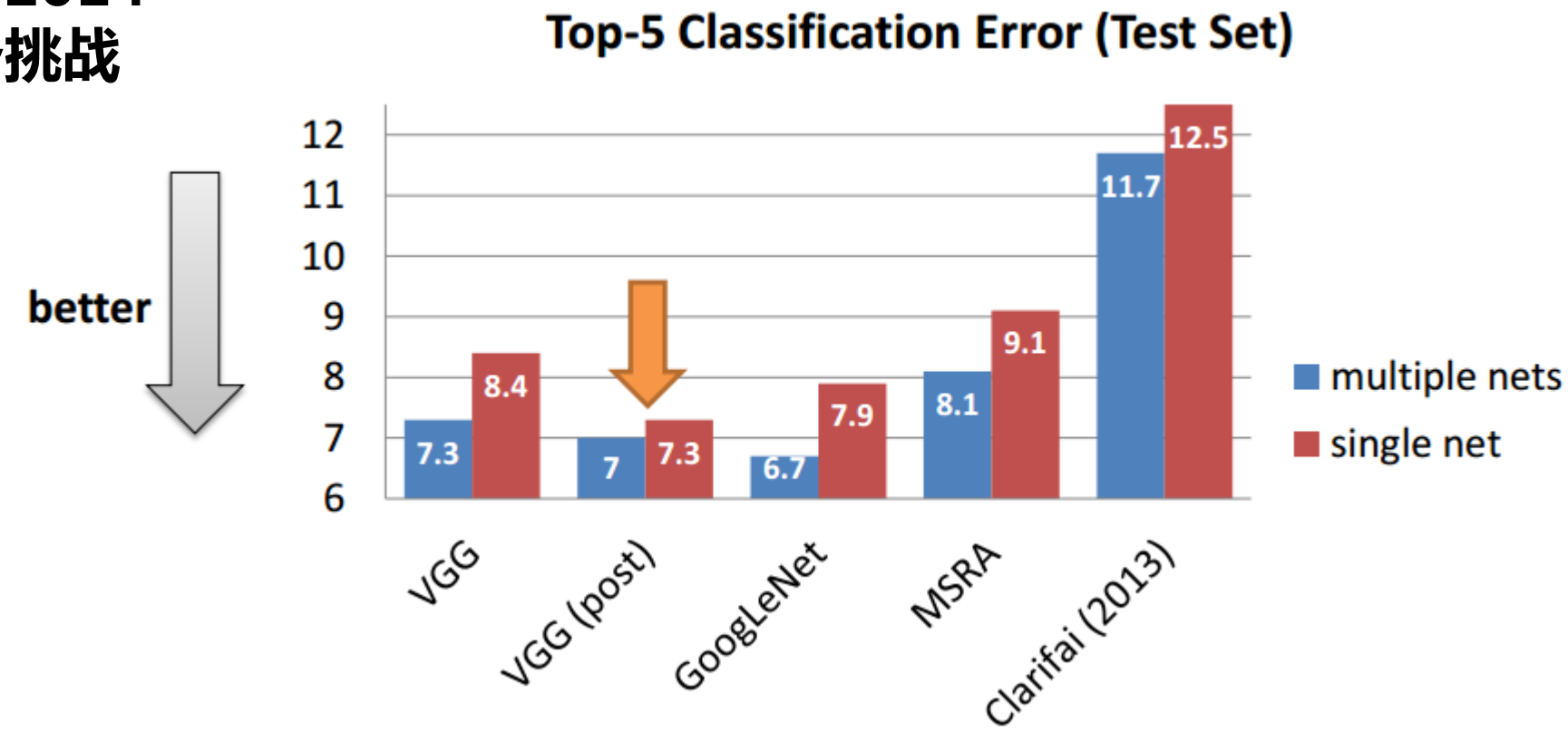
比较——随机训练大小



- 训练尺度抖动优于固定尺度
- 3个网络，每层均微调

最终结果

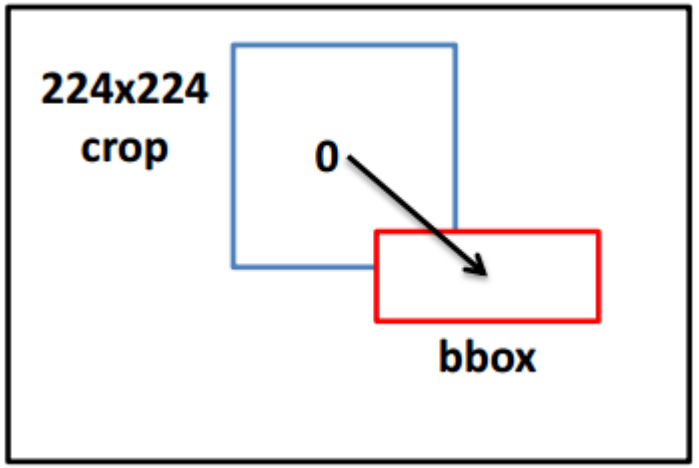
ILSVRC-2014 分类任务挑战



- 错误率7.0%，排名第2
  - 使用2个多尺度模型的组合（16和19层网络）
- 单模型错误率：7.3%

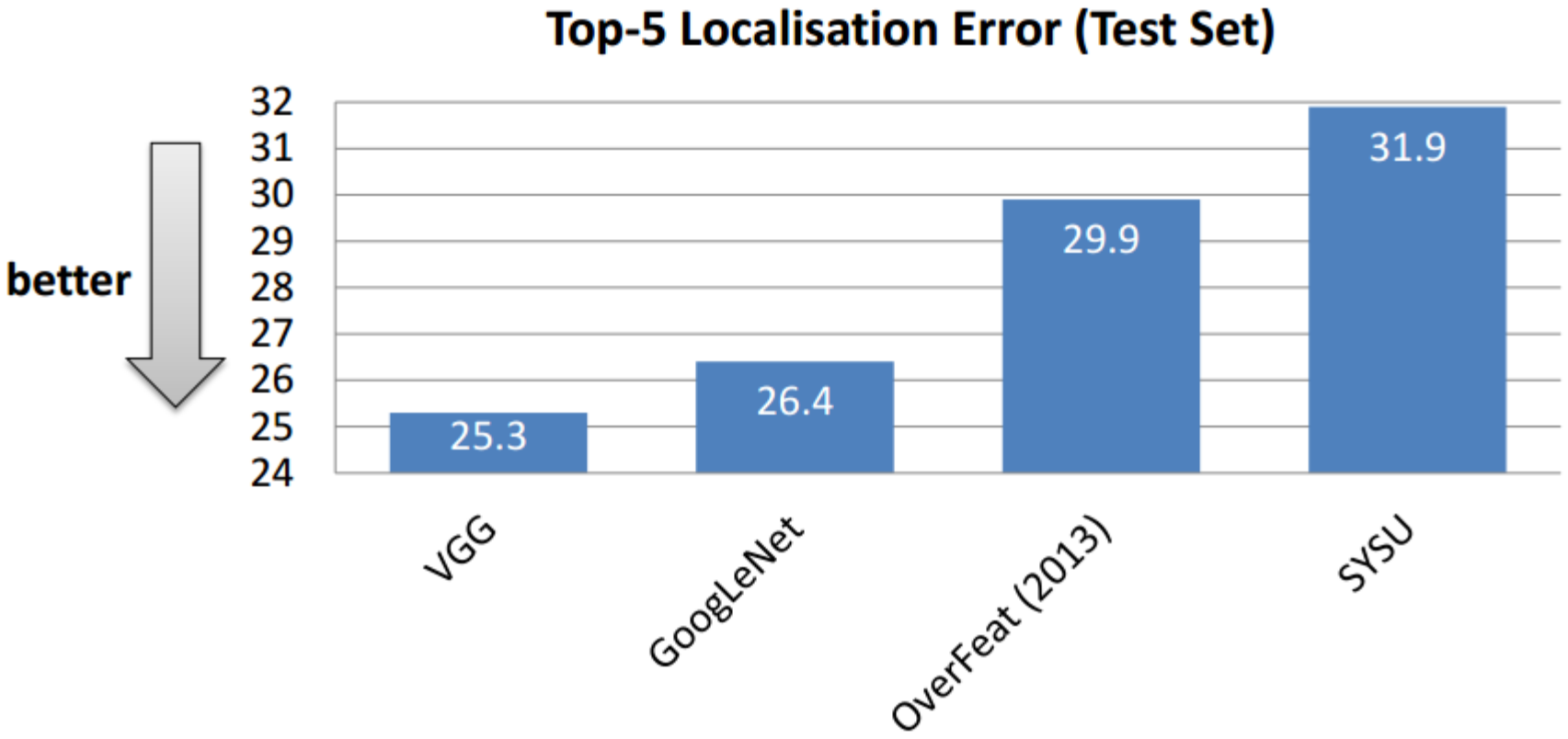
目标检测

- 构建于深层次的分类卷积神经网络上
- 类似于OverFeat
  - 目标检测神经网络预测出一组边界框
  - 边界框是混合的
  - 结果框由分类卷积神经网络评分
- 最后一层对每类预测边界框
  - 边界框参数：（x,y,w,h）



- 训练
  - 欧氏损失
  - 使用分类网络初始化
  - 对所有网络进行微调

最终结果



- 错误率：25.3%，排名第1
  - 使用2个目标检测模型的组合

PART 02

02

马尔科夫链与隐马尔科夫模型

马尔可夫链

一个系统有N个状态$s_1, s_2, \ldots, s_N$，随着时间推移，系统从某一状态转移到另一状态，设 q_t 为时刻 t 的状态，若其仅与系统在时刻$t-1$的状态有关，则该系统构成一个马尔可夫链（又称马尔可夫过程），即系统在时刻t的状态为s_j的概率为

$$P(q_t = s_j | q_{t-1})$$

马尔可夫性质：当一个随机过程在给定现在状态及所有过去状态情况下，其未来状态的条件概率分布仅依赖于当前状态。

隐式马尔可夫模型

- 隐马尔可夫模型（Hidden Markov Model，HMM）是统计模型，它用来描述一个含有隐含未知参数的马尔可夫过程。
- 在HMM中，状态并不是直接可见的，但受状态影响的某些变量则是可见的。
- 每个状态对应一个可观测的事件，即状态的随机函数
- HMM是一双重随机过程，其中状态转移过程是不可观察的

HMM的三个假设

对于一个随机事件，有一观察值序列：$O = o_1, o_2, \dots, o_T$，该事件隐含着一个状态序列：$Q = q_1, q_2, \dots, q_T$

- 假设1：马尔可夫性假设（状态构成一阶马尔可夫链）

$$P(q_i|q_{i-1}q_{i-2}\dots q_1) = P(q_i|q_{i-1})$$

- 假设2：不动性假设（状态与具体时间无关）

$P(q_{i+1}|q_i) = P(q_{j+1}|q_j)$，对任意$i, j$成立

- 假设3：输出独立性假设（输出仅与当前状态有关）

$$P(o_1, o_2, \dots, o_T|q_1, q_2, \dots, q_T) = \prod P(o_t|q_t)$$

HMM定义

一个隐式马尔可夫模型是由一个五元组描述的：

$$\lambda = (N, M, A, B, \pi)$$

其中：

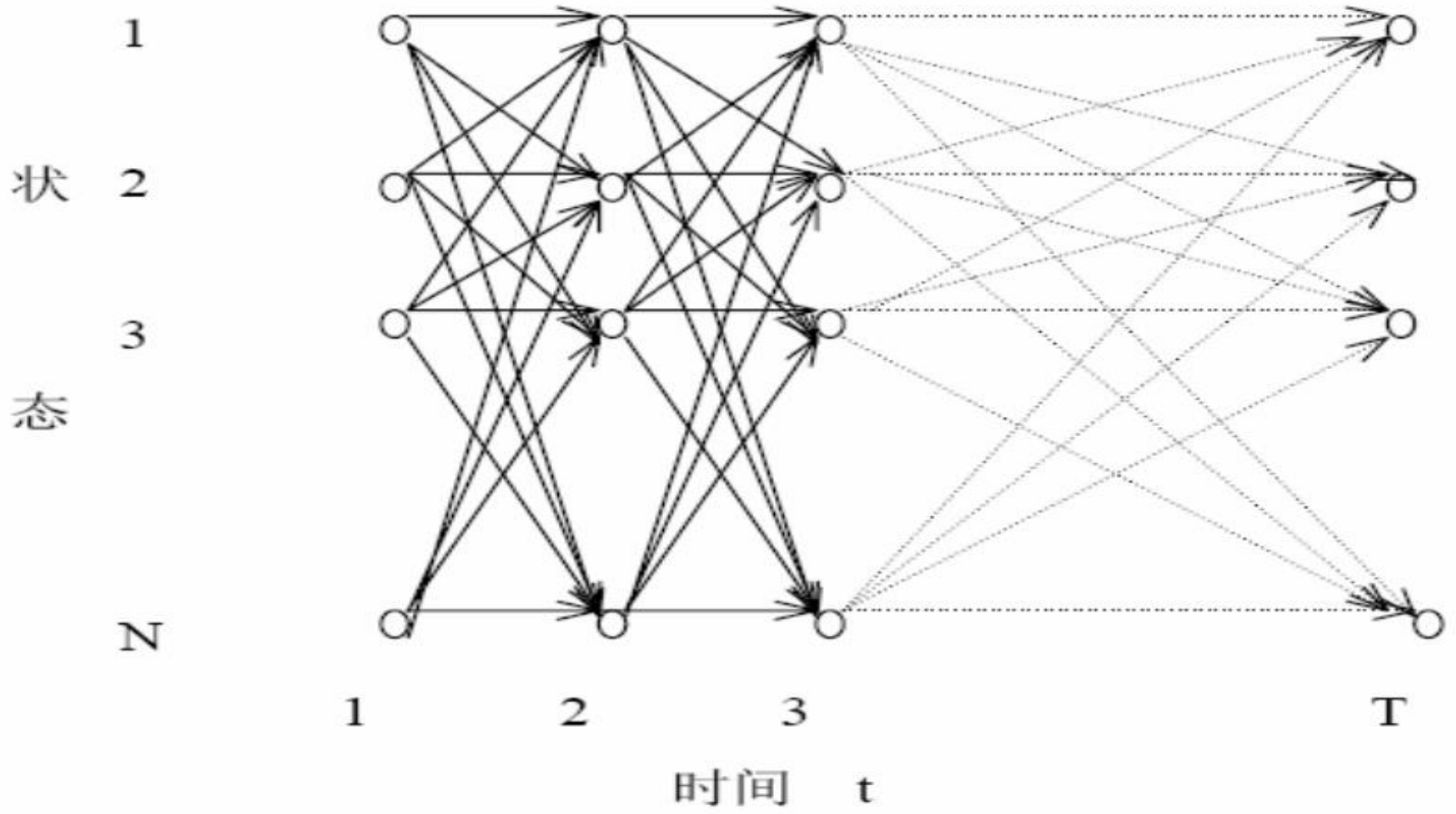
- $N = \{s_1, s_2, \dots, s_N\}$：状态的有限集合
- $M = \{v_1, v_2, \dots, v_M\}$：观察值的有限集合
- $A = \{a_{ij}\}, a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$：状态转移概率矩阵
- $B = \{b_{jk}\}, b_{jk} = P(o_t = v_k | q_t = s_j)$：观察值概率分布矩阵
- $\pi = \{\pi_i\}, \pi_i = P(q_1 = s_i)$：初始状态概率分布
- $b_j(o_t) = P(o_t | q_t = j)$：状态为$j$时观察值的概率

观察序列产生步骤

给定HMM模型$\lambda = (N, M, A, B, \pi)$，则观察序列$O = o_1, o_2, \ldots, o_T$，可由以下步骤产生：

- 1. 根据初始状态概率分布$\pi = \pi_i$，选择一初始状态$q_1 = s_i$
- 2. 设$t = 1$
- 3. 根据状态s_i的输出概率分布b_{jk}，输出$o_t = v_k$
- 4. 根据状态转移概率分布a_{ij}，转移到新状态$q_{t+1} = s_j$
- 5. 设$t = t + 1$，若$t < T$，重复步骤3和4，否则结束

HMM的网格结构



HMM的三个基本问题

给定HMM模型$\lambda = (A, B, \pi)$，令$O = o_1, o_2, \ldots, o_T$为观察值序列，则有关于HMM的三个基本问题：

- 评估问题：对于给定模型，求某个观察值序列的概率$P(O|\lambda)$
- 解码问题：对于给定模型和观察值序列，求可能性最大的状态序列$max_Q\{P(Q|O, \lambda)\}$
- 学习问题：对于给定的一个观察值序列O，调整参数λ，使得观察值出现的概率$P(O|\lambda)$最大

三个基本问题的求解算法

- 评估问题：前向算法
  - 定义前向变量
  - 采用动态规划算法，复杂度$O(N^2T)$
- 解码问题：韦特比（Viterbi）算法
  - 采用动态规划算法，复杂度$O(N^2T)$
- 学习问题：向前向后算法
  - EM算法的一个特例，带隐变量的最大似然估计

定义前向变量为：

给定HMM模型$\lambda = (A, B, \pi)$，“在时刻t的状态是s_i且观察值序列为$o_1 o_2 \dots o_t$” 这一事件的概率，记为

$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, q_t = s_i | \lambda)$$

则

- $\alpha_1(i) = P(o_1, q_1 = s_i | \lambda) = \pi_i b(i, o_1)$
- $\alpha_T(i) = P(o_1, o_2, \dots, o_T, q_T = s_i | \lambda)$
- $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

算法过程

- 1. 初始化

$$\alpha_1(i) = \pi_i b(i, o_1)$$

- 2. 递归

$$\alpha_{t+1}(j) = b(j, o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}$$

- 3. 终结

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

解决问题二：Viterbi算法

定义$\delta(t,i)$沿状态序列$q_1 q_2 \dots q_t$且$q_t = s_i$的随机过程产生出观察值$o_1 o_2 \dots o_t$的最大概率，即：

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P(q_1 q_2 \dots q_{t-1}, q_t = s_i, o_1 o_2 \dots o_t | \lambda)$$

Viterbi 算法也是类似于前向算法的一种网格结构

Viterbi算法（续）

- 目标：给定一个观察序列和HMM模型，如何有效选择“最优”状态序列，以“最好地解释”观察序列

- “最优”⟶概率最大：

$$Q^* = arg\max_Q P(Q|O,\lambda)$$

- Viterbi变量：

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = s_i, o_1 o_2 \ldots o_t|\lambda)$$

- 递归关系：

$$\delta_{t+1}(i) = b(i, o_{t+1}) \max_j \delta_t(j) a_{ji}$$

- 记忆变量：$\varphi_t(i)$记录概率最大路径上当前状态的前一个状态

Viterbi算法（续）

- 初始化：

$$\delta_1(i) = \pi_i b_i(o_1)\ ,\ \varphi_1(i) = 0\ ,\ 1 \le i \le N$$

- 递归：

$$\delta_t(j) = b_j(o_t) \max_{1 \le i \le N} \delta_{t-1}(i) a_{ij}$$

$$\varphi_t(j) = b_j(o_t) arg \max_{1 \le i \le N} \delta_{t-1}(i) a_{ij}$$

其中$2 \le t \le T\ ,\ 1 \le j \le N$

- 终结：

$$p^* = \max_{1 \le i \le N} \delta_T(i)\ ,\ q_T^* = arg \max_{1 \le i \le N} \delta_T(i)$$

- 路径回溯：

$$q_t^* = \varphi_{t+1}(q_{t+1}^*)\ ,\ t = T-1, T-2, \dots, 1$$

EM算法

- EM(Expectation-Maximization)算法可用于含有隐变量的统计模型的最大似然估计。
- EM算法是一个由交替进行的“期望(E过程)”和“极大似然估计(M过程)”两部分组成的迭代过程：
  - 对于给定的不完全数据和当前的参数值，“E过程”从条件期望中相应地构造完全数据的似然函数值，“M过程”则利用参数的充分统计量，重新估计概率模型的参数，使得训练数据的对数似然最大。
- EM算法的每一次迭代过程必定单调地增加训练数据的对数似然值，于是迭代过程渐进地收敛于一个局部最优值。

后向变量

定义后向变量为：

给定HMM模型$\lambda = (A, B, \pi)$且在时刻t的状态是s_i时，“部分观察值序列为$o_{t+1}o_{t+2}\dots o_T$”这一事件的概率，记为

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = s_i, \lambda)$$

则

- $\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1}), (1 \le i \le N, 1 \le t \le T-1)$
- $\beta_T(i) = 1, (1 \le i \le N)$
- $P(O, q_t = i|\lambda) = \alpha_t(i)\beta_t(i), (1 \le i \le N, 1 \le t \le T)$
- $P(O|\lambda) = \sum_{i=1}^{N} P(O, q_t = i|\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$

期望值

- 给定HMM和观察序列， $\xi_t(i,j)$为在时刻t位于状态i，时刻$t+1$位于状态j的概率：

$$\xi_t(i,j) = P\left(q_t = s_i, q_{t+1} = s_j \middle| O, \lambda\right)$$
$$= \frac{P(q_t = s_i, q_{t+1} = s_j, O|\lambda)}{P(O|\lambda)}$$
$$= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}$$

- 给定HMM和观察序列， 在时刻t位于状态i的概率：

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j)$$

重估公式

$$\pi_i = q_1 \text{为} S_i \text{的概率} = \gamma_1(i)$$

$$a_{ij} = \frac{Q\text{中从状态}q_i\text{转移到}q_j\text{的期望次数}}{Q\text{中从状态}q_i\text{转移到另一状态(包括}q_i\text{本身)的期望次数}} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}$$

$$b_j(k) = \frac{Q\text{中由状态}q_j\text{输出}v_k\text{的期望次数}}{Q\text{到达}q_j\text{的期望次数}} = \frac{\sum_{t=1}^{T}\gamma_t(j)\times\delta(O_t, v_k)}{\sum_{t=1}^{T}\gamma_t(j)}$$

向前向后算法(Baum-Welch算法)

- 1. 初始化：随机地为π_i，a_{ij}，b_{jk}赋初值（满足概率条件），得到模型λ_0，设$i=0$
- 2. EM步骤：
  - E步骤：由λ_i计算期望值$\xi_t(i,j)$和$\gamma_t(i)$
  - M步骤：用E步骤所得的期望值，重新估计π_i，a_{ij}，b_{jk}，得到模型λ_{i+1}
- 3.循环设计：$i=i+1$；重复EM步骤，直至π_i，a_{ij}，b_{jk}值收敛。

HMM的一些实际问题

- 初始概率分布的选择
  - 1. 随机选择
  - 2. 利用先验信息
  - 3. 来自多序列比对的结果
- 数值计算中的防溢出处理
  - 在前向算法、Viterbi算法以及Baum-Welch算法中，概率值的连续乘法运算很容易导致下溢现象。
- 解决办法：
  - 1. 前向算法中，每一个时间步的运算中都乘以一 个比例因子
  - 2. Viterbi算法中，对概率值取对数后计算

HMM的一些实际问题

- 向算法:引入比例因子

$$\alpha_{t+1}(j) = b(j, o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}$$

$$\alpha'_{t+1}(j) = c(t) b(j, o_{t+1}) \sum_{i=1}^{N} \alpha'_t(i) a_{ij}$$

其中，$c(t) = {}^{1}/_{\sum_{i=1}^{N} \alpha_t(i)}$

- Viterbi算法:连乘积→对数求和

$$\delta_t(j) = b_j(o_t) \max_{1 \le i \le N} \delta_{t-1}(i) a_{ij}$$

$$\delta'_t(j) = \log b_j(o_t) + \max_{1 \le i \le N} \{\delta'_{t-1}(i) + \log a_{ij}\}$$

HMM总结

- HMM模型可以看作一种特定的Bayes Net
- HMM模型等价于概率正规语法或概率有限状态自动机
- HMM模型可以用一种特定的神经网络模型来模拟
- 优点：研究透彻，算法成熟，效率高，效果好，易于训练
- 局限性：
  - 模型定义的是联合概率，必须列举所有观察序列的可能值，这对多数领域来说是比较困难的。
  - 基于观察序列中的每个元素都相互条件独立。即在任何时刻观察值仅仅与状态（即要标注的标签）有关。大多数现实世界中的真实观察序列是由多个相互作用的特征和观察序列中较长范围内的元素之间的依赖而形成的。

PART 03

03

条件随机场

条件随机场概述

- 条件随机场（Conditional Random Fields，CRF）模型是Lafferty等人于2001年在最大熵模型和隐马尔可夫模型的基础上提出的一种无向图学习模型，是一种用于标注和切分有序数据的条件概率模型。
- 概率图模型：是一类用图的形式表示随机变量之间条件依赖关系的概率模型。是概率论与图论的结合。

$$G = (V, E)$$

  - V：顶点/节点，表示随机变量
  - E：边/弧，表示随机变量间的条件依赖关系
- 根据图中边有无方向，常用的概率图模型分为两类：
  - 有向图：亦称贝叶斯网络或信念网络
  - 无向图：亦称马尔可夫随机场或马尔可夫网络

条件随机场的定义

- 随机场可以看成是定义在同一样本空间上的一组随机变量的集合：$(X_1, X_2, \ldots, X_n)$
- 马尔科夫随机场（MRF）是具有马尔可夫性质的随机场，对应一个无向图模型。MRF的结构本质上反应了我们的先验知识：哪些变量之间有依赖关系需要考虑，而哪些可以忽略。
- 如果给定的MRF中每个随机变量下面还有观察值，我们要确定的是给定观察集合下，MRF的条件分布，那么这个MRF就称为CRF。它的条件分布形式完全类似于MRF的分布形式，只不过多了一个观察集合X。
- 从通用角度来看，CRF本质上是给定了观察值 (observations) 集合的MRF。

条件随机场的定义

- 设$G = (V, E)$是一个无向图，$Y = \{Y_v | v \in V\}$是以G中节点为索引的随机变量Y_v构成的集合。
- 在给定X的条件下，如果每个随机变量Y_v服从马尔可夫属性，即$P(Y_v | X, Y_u, u \neq v) = P(Y_v | X, Y_u, u \sim v)$，$u \sim v$表示$u$和$v$是相邻的节点
- 则(X, Y)构成一个条件随机场。

- CRFs 是在给定需要标记的观察序列的条件下，计算整个标记序列的联合概率，即求条件分布：$P(Y|O)$
- 而不是在给定当前状态条件下，定义下一个状态的分布(HMM)，即求联合分布：$P(Y, O)$

Linear-chain CRFs模型

令$x = (x_1, x_2, \dots, x_n)$表示观察序列，$y = (y_1, y_2, \dots, y_n)$是有限状态的集合，根据随机场的基本理论：

$$p(y|x,\lambda) \propto \exp\left(\sum_j \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k(y_i, x, i)\right)$$

- $t_j(y_{i-1}, y_i, x, i)$：对于观察序列的标记位置$i-1$与i之间的转移特征函数；定义在边上的函数，权值为λ_j
- $s_k(y_i, x, i)$：观察序列的i位置的状态特征函数；定义在节点上的函数，权值为μ_k
- 一般来说，特征函数的取值为1或0，当满足规定好的特征条件时取值为1，否则为0。

Linear-chain CRFs模型

将两个特征函数统一为： $f_j(y_{i-1}, y_i, x, i)$

则有：

$$p(y|x,\lambda) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n} \sum_{j} \lambda_j f_j(y_{i-1}, y_i, x, i) \right)$$

其中：

$$Z(x) = \sum_{j} \exp\left(\sum_{i=1}^{n} \sum_{j} \lambda_j f_j(y_{i-1}, y_i, x, i) \right)$$

关键问题

- 1. 特征函数的选择
  - 特征函数的选取直接关系模型的性能

- 2. 参数估计
  - 从已经标注好的训练数据集学习条件随机场模型的参数，即各特征函数的权重向量λ

- 3. 模型推断
  - 在给定条件随机场模型参数λ下，预测出最可能的状态序列。

特征函数的选择

- CRFs模型中特征函数的形式定义： $f_j(y_{i-1}, y_i, x, i)$
- 它是状态特征函数和转移特征函数的统一形式表示。特征函数通常是二值函数，取值要么为1要么为0
- 在定义特征函数的时候，首先构建观察序列的实数值特征 $b(x,i)$集合来描述训练数据的经验分布特征。例如：

$$b(x,i)=\begin{cases}1 & \textit{如果时刻 i观察值x是大写开头}\\ 0 & \textit{otherwise}\end{cases}$$

特征函数的选择

- 每个特征函数表示为观察序列的实数值特征$b(x,i)$集合中的一个元素
- 如果前一个状态和当前状态具有特定的值，则所有的特征函数都是实数值

$$f(y_{i-1},y_i,x,i)=\begin{cases}b(x,i) & if \quad y_{i-1}=<title>, y_i=<author> \\ 0 & otherwise\end{cases}$$

参数估计

- 建立条件随机场模型的主要任务是从训练数据中估计特征的权重λ
- 假设给定训练集$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$，采用极大似然估计法来估计参数$\lambda$。条件概率$p(y|x, \lambda)$的对数似然函数形式为：

$$L(\lambda) = \sum_{x,y} \tilde{p}(x, y) \sum_{i=1}^{n} \left(\sum_{j} \lambda_j f_j((y_{i-1}, y_i, x, i)) \right) - \sum_{x} \tilde{p}(x) \log Z(x)$$

其中：

- $\tilde{P}(x, y)$ 为训练样本中(x, y)的经验概率

$$\tilde{P}(x, y) = \frac{(x, y)\text{在样本中同时出现的次 数}}{\text{样本空间的容量}}$$

- $\tilde{P}(x)$ 是随机变量X在训练样本中的经验分布

$$\tilde{P}(x) = \frac{x\text{在样本中出现的次数}}{\text{样本空间的容量}}$$

分别对对数似然函数$L(\lambda)$中的λ_j求偏导

$$\frac{\partial L(\lambda)}{\lambda_j} = \sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} f_j(y_{i-1}, y_i, x) - \sum_{x,y} \tilde{p}(x) p(y|x,\lambda) \sum_{i=1}^{n} f_j(y_{i-1}, y_i, x)$$

令上式等于0，求出λ_j

上述方法直接使用对数最大似然估计，可能会发生过度学习问题，通常引入惩罚函数的方法解决这一问题。

参数估计

使用惩罚项$\frac{\sum_j \lambda_j^2}{2\sigma^2}$将对数似然函数改造为：

$$L(\lambda) = \sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} \left(\sum_j \lambda_j f_j((y_{i-1}, y_i, x, i)) \right) - \sum_x \tilde{p}(x) \log Z(x) - \frac{\sum_j \lambda_j^2}{2\sigma^2}$$

对上式中每个λ_j求偏导，并令导数为0，求λ_j

- 由于极大似然估计并不一定能得到一个近似解，因而Lafferty提出两个迭代缩放的算法用于估计条件随机场的极大似然参数：
  - GIS算法（Generalised Iterative Scaling）
  - IIS算法（Improved Iterative Scaling）

模型推断

$$p(y|x,\lambda)=\frac{1}{Z(x)}\exp\left(\sum_{i=1}^{n}\sum_{j}\lambda_j f_j(y_{i-1},y_i,x,i)\right)$$

$$Z(x)=\sum_{j}\exp\left(\sum_{i=1}^{n}\sum_{j}\lambda_j f_j(y_{i-1},y_i,x,i)\right)$$

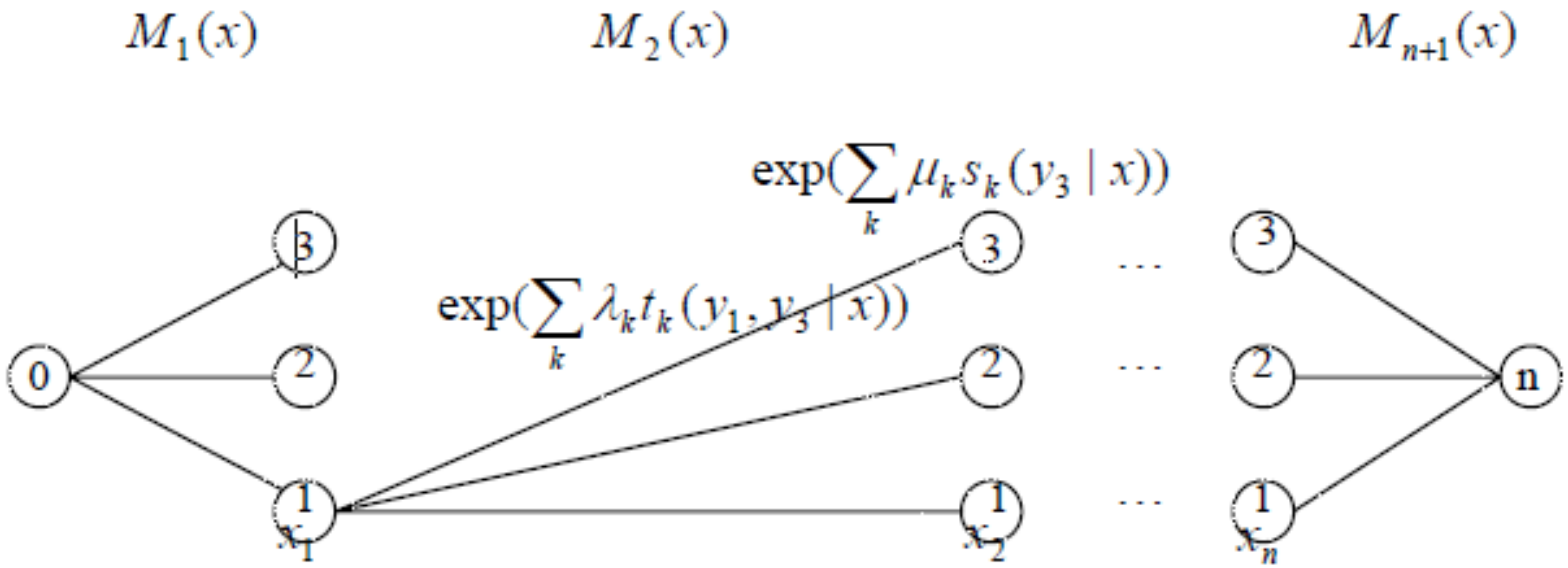
- 对于一个给定观察序列$x(x_1,x_2,\dots,x_n)$，求使得该观察序列出现概率最大的标记序列(状态序列)$y(y_1,y_2,\dots,y_n)$

- 对于一个链式条件随机场，在图的模型中添加一个开始状态Y_0和一个结束状态Y_{n+1}
- 定义一组矩阵$\{M_i(x)|i = 1,2,\dots,n+1\}$，其中每个$M_i(x)$是$N \times N$阶的随机变量矩阵。$M_i(x)$中的每个元素$M_i(y_{i-1}, y|x)$定义如下：

$$M_i(y_{i-1} = y', y_i = y|x) = exp\left(\sum_{i=1}^{n}\sum_{j}\lambda_j f_j(y_{i-1}, y, x, i)\right)$$

$$= exp\left(\sum_{k}\lambda_k t_k(y_{i-1}, y_i, x, i)\right) + \sum_{k}\mu_k s_k(y_i, x, i)$$

模型推断



$$M_2(y_1, y_3 \mid x) = \exp\left(\sum_k \lambda_k t_k(y_1, y_3 \mid x) + \sum_k \lambda_k s_k(y_3 \mid x)\right)$$

- 因为$p(y|x,\lambda)$实际上是从开始节点到结点节点的一条路径的概率，所以有：

$$P(y \mid x, \lambda) = \frac{1}{Z(x)} \prod_{i=1}^{n+1} M_i(y_{i-1}, y_i \mid x)$$

- 其中 $Z(x)$ 为归一化因子，为所有路径概率的和，表达式如下：

$$Z(x) = \prod_{i=1}^{n+1} M_i(x)$$

CRF总结

- 条件随机场的本质是通过给定的观察序列求一组对应的状态序列的过程。主要应用于词性标注等
- 优点
  - 条件随机场使用一种概率图模型，具有表达长距离依赖性和交叠性特征的能力，能够较好地解决标注（分类）偏置等问题的优点
  - 所有特征可以进行全局归一化，能够求得全局的最优解。
- 缺点：
  - 模型训练时收敛速度比较慢

PART 04

04

命名实体识别

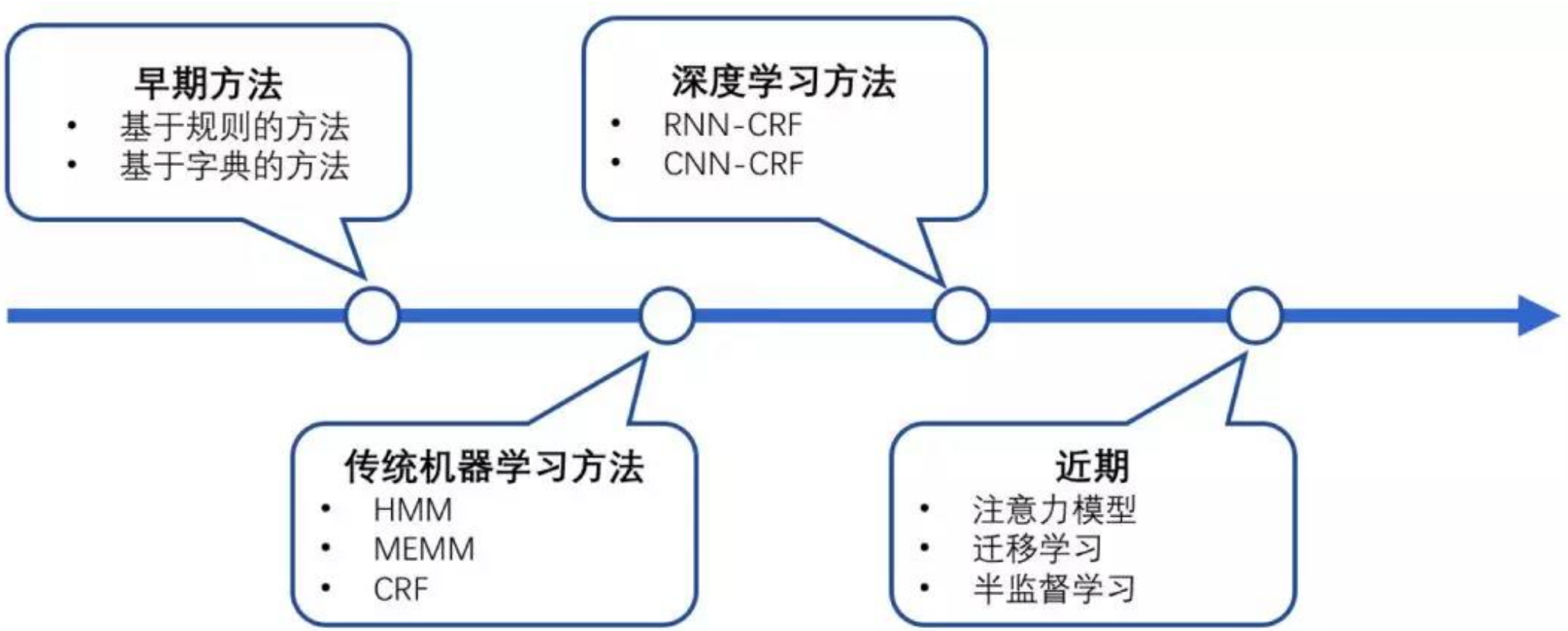
基本概念

- 命名实体识别（Named Entity Recognition，NER），又称作“专名识别”，是指识别文本中具有特定意义的实体，如人名、机构名、地名等专有名词和有意义的时间等
- NER是信息检索、问答系统等技术的基础任务。
- NER通常包括两部分：
  - （1）实体边界识别；
  - （2）确定实体类别（人名、地名、机构名或其他）。

基本概念

- 学术上NER所涉及的命名实体一般包括
  - 3大类（实体类、时间类、数字类）
  - 7小类（人名、地名、组织机构名、时间、日期、货币、百分比）。
- NER系统就是从非结构化的输入文本中抽取出上述实体，并且可以按照业务需求识别出更多类别的实体，比如产品名称、型号、价格等。

发展趋势


早期方法
基于规则的方法
基于字典的方法
深度学习方法
RNN-CRF
CNN-CRF
传统机器学习方法
HMM
MEMM
CRF
近期
注意力模型
迁移学习
半监督学习

主要技术方法

- 命名实体识别的主要技术方法分为：
  - 基于规则和词典的方法
  - 基于统计的方法
  - 规则字典和统计混合的方法
  - 基于深度学习的方法等。
- 条件随机场（ConditionalRandom Field，CRF）是NER目前的主流模型。优点在于其为一个位置进行标注的过程中可以利用丰富的内部及上下文特征信息。

基于规则和词典的方法

- 多采用语言学专家构造规则模板
- 选用特征包括统计信息、标点符号、关键字、指示词和方向词、位置词、中心词等方法
- 以模式和字符串相匹配为主要手段
- 当提取的规则能比较精确地反映语言现象时，基于规则的方法性能要优于基于统计的方法
- 容易产生错误，系统可移植性差，对于不同的系统需要语言学专家重新书写规则。
- 代价太大，存在系统建设周期长、需要建立不同领域知识库作为辅助以提高系统识别能力等问题。

基于统计的方法

- 基于统计机器学习的方法主要包括：隐马尔可夫模型、最大熵、支持向量机、条件随机场等。
- 最大熵模型结构紧凑，具有较好的通用性，主要缺点是训练时间复杂性非常高，有时甚至导致训练代价难以承受，另外需要明确的归一化计算，导致开销比较大。
- 条件随机场为命名实体识别提供了一个特征灵活、全局最优的标注框架，但同时存在收敛速度慢、训练时间长的问题。
- 隐马尔可夫模型更适用于一些对实时性有要求以及像信息检索这样需要处理大量文本的应用,如短文本命名实体识别。

基于统计的方法

- 基于统计的方法对特征选取的要求较高，需要从文本中选择对该项任务有影响的各种特征，并将这些特征加入到特征向量中。
- 基于统计的方法对语料库的依赖也比较大，而可以用来建设和评估命名实体识别系统的大规模通用语料库又比较少。

混合方法

- 统计学习方法之间或内部层叠融合。
- 规则、词典和机器学习方法之间的融合，其核心是融合方法技术。
- 将各类模型、算法结合起来，将前一级模型的结果作为下一级的训练数据，并用这些训练数据对模型进行训练，得到下一级模型。

基于深度学习的方法

- 基于深度学习的方法主要包括：BiLSTM-CRF、IDCNN-CRF等。
- BiLSTM-CRF模型主要由Embedding层（主要有词向量，字向量以及一些额外特征），双向LSTM层，以及最后的CRF层构成。
- 实验结果表明BiLSTM-CRF已经达到或者超过了基于丰富特征的CRF模型，成为目前基于深度学习的NER方法中的最主流模型。
- 在特征方面，该模型继承了深度学习方法的优势，无需特征工程，使用词向量以及字符向量就可以达到很好的效果，如果有高质量的词典特征，能够进一步获得提高。

PART 05

05

使用BiLSTM-CRF进行命名实体识别

使用BiLSTM-CRF进行命名实体识别

PART 06

06

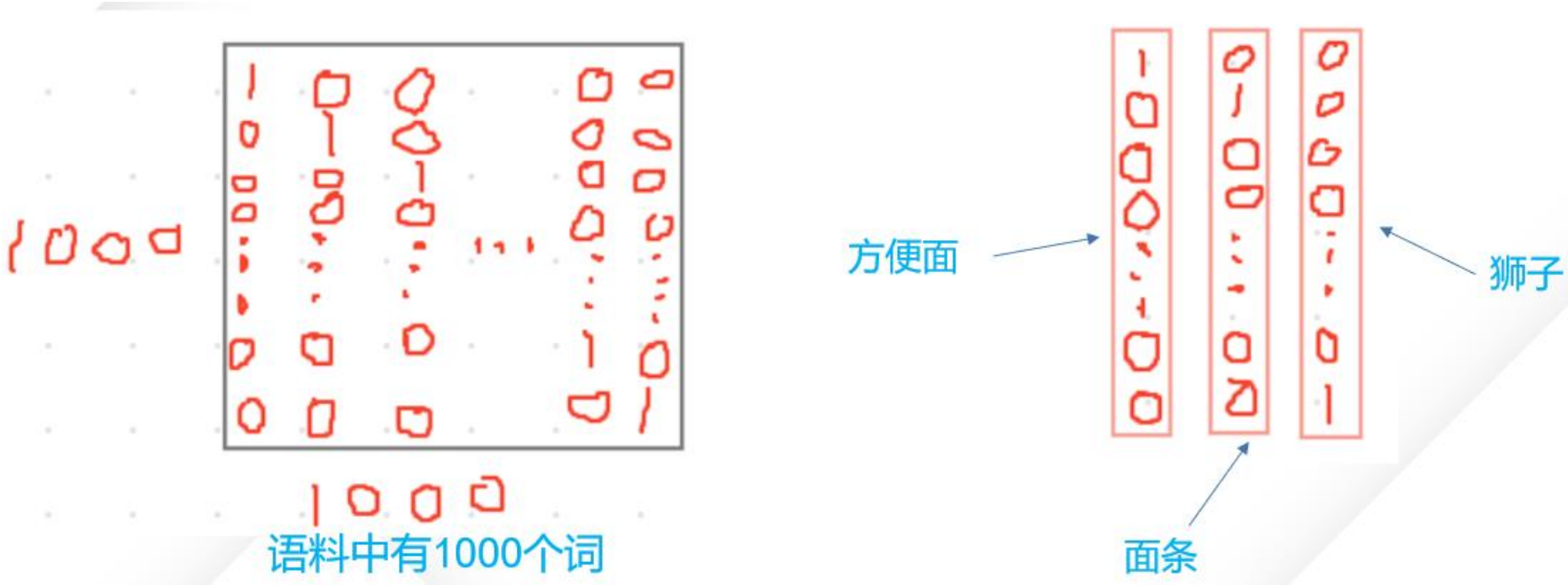
词向量训练word2vec

词向量

- 词向量，是指单词或者汉字的向量化表示。
- 词向量一般有两种表示方式，分别是稀疏向量和密集向量。
  - 稀疏向量，就是用一个很长的向量来表示一个词，向量的长度为词典的大小N，向量的分量只有一个1，其他全为0，1的位置对应该词在词典中的索引。
  - 密集向量，也可以称之为word embedding，基本思路是通过训练将每个词映射成一个固定长度的短向量，所有这些向量就构成一个词向量空间，每一个向量可视为该空间上的一个点。

稀疏向量(one-hot representation)

- 假设一段文本有1000个词，如果用一个矩阵来表示这个文本，那么这个矩阵的维度为1000*1000。假设文本中有“方便面”，“面条”，“狮子”这三个词，用one-hot向量表示的话，可以表示为下面这种形式。



稀疏向量的缺点

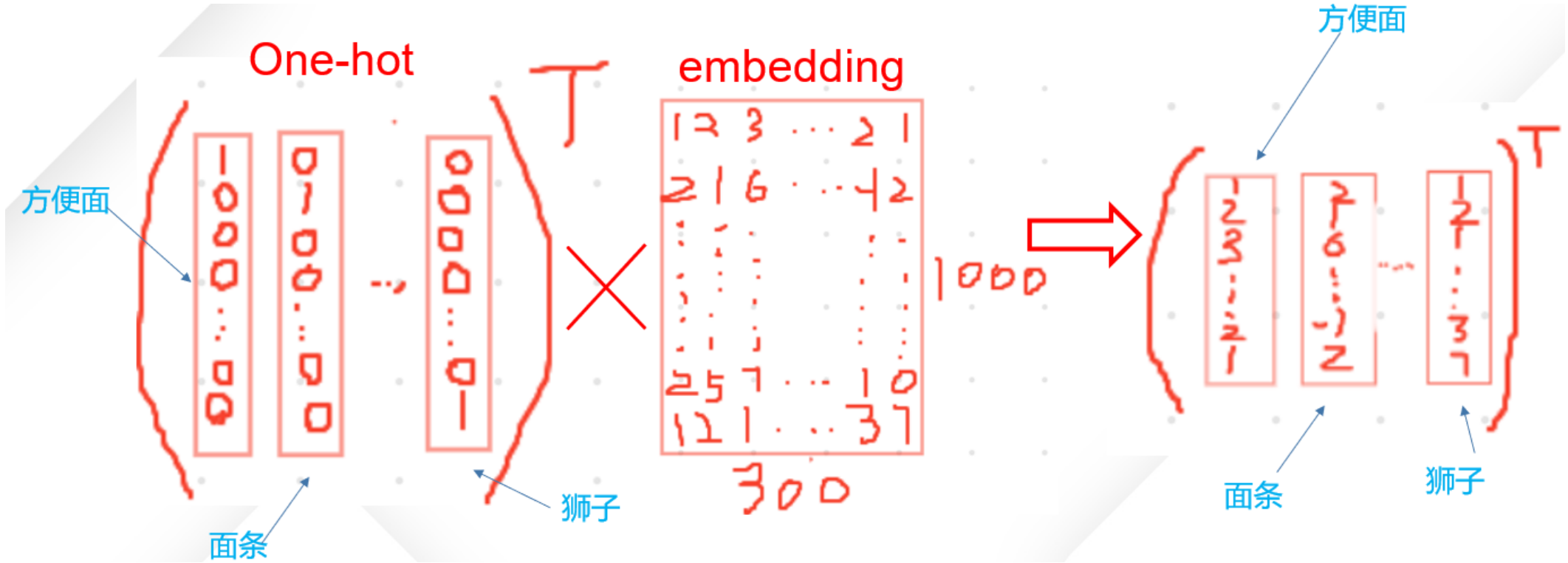
- 1. 当语料长度过长的时候，容易引发维数灾难。举个例子，如果一个文本有10万个词，需要用一个10万×10万的稀疏向量来表示它。
- 2. 无法体现出近义词之间的关系。例如“方便面”，“面条”，在中文中两个词是有一定关系的，但是如果用稀疏向量就无法表示出他们之间的关系，如果采用余弦相似度计算的话，它们之间的相似度为0。

$$\text{Sim}(\text{方便面}, \text{面条}) = \cos\Theta = \frac{\overrightarrow{\text{方便面}} \cdot \overrightarrow{\text{面条}}}{||\text{方便面}|| \cdot ||\text{面条}||} = 0$$

- 3. 不支持可微分优化(即不能使用经典的反向传播算法)。

密集向量(distributed representation)

- 通过one-hot矩阵乘以一个embedding层就可以得到一个词的向量表示，同时也达到了降维的效果。
- 此时向量长度可以自由选择，与词典规模无关。
- 这种表示方式能更精准的表现出近义词之间的关系。



word2vec

- word2vec是从大量文本中以无监督学习的方式学习语义知识的模型，其本质就是通过学习文本来用词向量的方式表征词的语义信息，通过嵌入空间将语义上相似的单词映射到距离相近的地方。即将单词从原先所属的空间映射到新的多维空间中。
- 举例来讲：
  - smart和intelligent意思相近，target和goal在意思上比较相近，target与apple离的较远
  - 映射到新的空间中后smart和intelligent、target和goal，它们的词向量比较相近，而target与apple的词向量相差较远。

word2vec

- word2vec采用分布式表征
- 在向量维数比较大的情况下，每一个词都可以用元素的分布式权重来表示
- 向量的每一维都表示一个特征向量，作用于所有的单词，而不是简单的元素和值之间的一一映射
- 将独热编码转化为低维度的连续值即稠密向量，并且其中意思相近的词被映射到向量空间中相近的位置

word2vec

- 常见的word2vec词向量有两种模式，CBOW(continuous bag of words)和skip-gram
- CBOW是根据目标单词所在原始语句的上下文来推测目标单词本身，
- skip-gram则是利用该目标单词推测原始语句信息即它的上下文。
- 举个例子：美国对中国进口钢铁采取了反倾销调查。
  - CBOW的目标可以是：{中国，钢铁}—>进口，{采取了，调查}—>反倾销，{美国，中国}—>对
  - skip-gram的目标可以是：{进口}—>{中国，钢铁}，{了}—>{采取，反倾销}。

PART 07

07

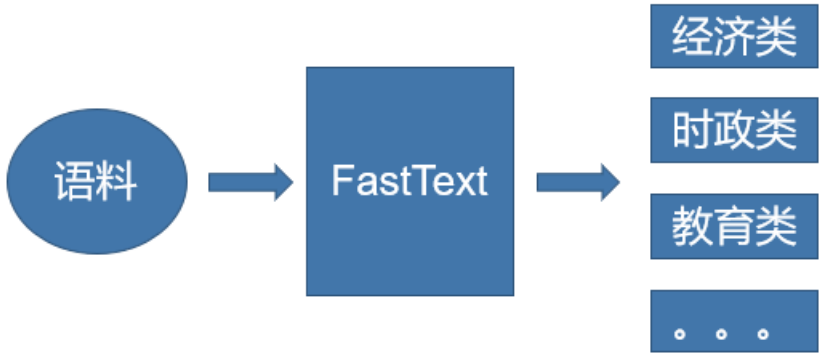
Fasttext分类

FastText简述

- FastText是一个快速文本分类算法
- 与基于神经网络的文本分类算法相比，它主要由两个优点：
  - FastText在保持高精度的同时极大地加快了训练速度和测试速度
  - 不需要使用预先训练好的词向量，因为FastText会自己训练词向量。

FastText应用场景

- 文本分类



- 情感分析

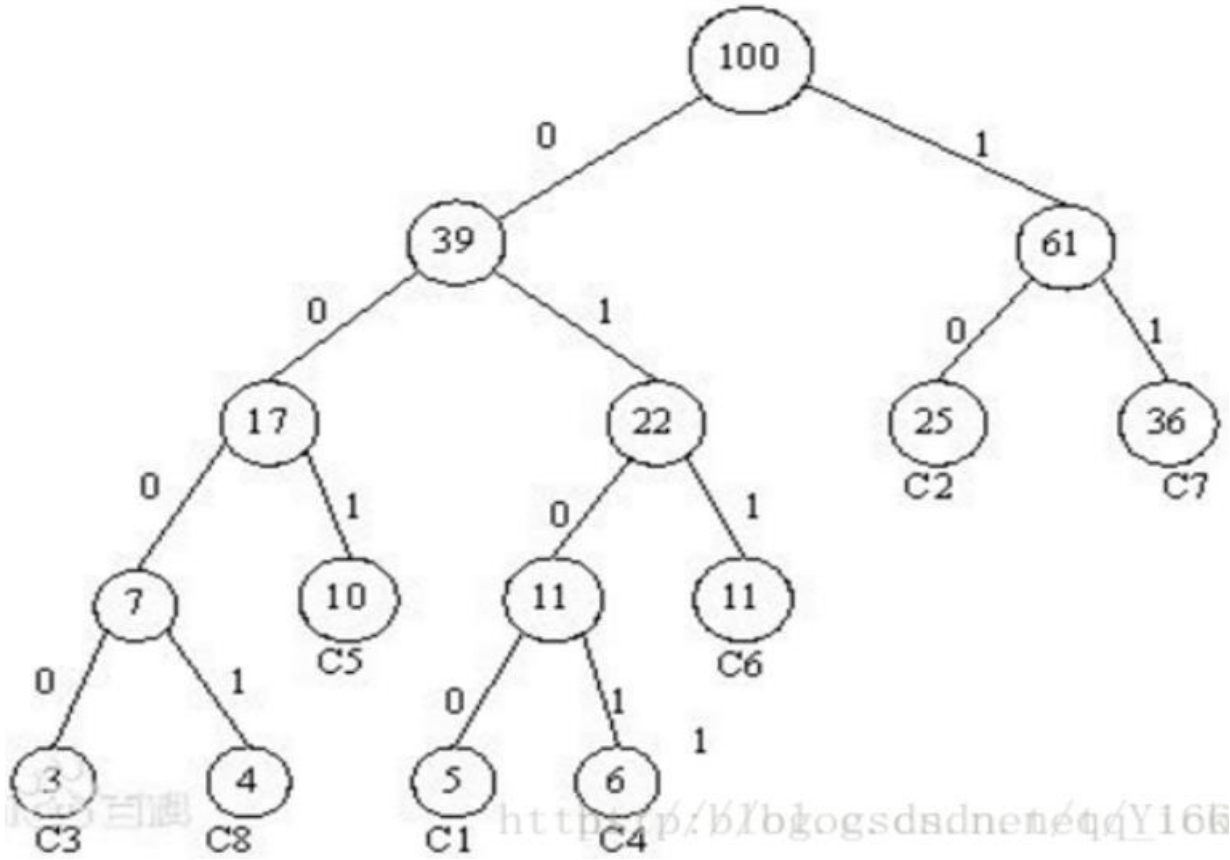
- Web搜索、信息检索、网页排名等

FastText应用的技术

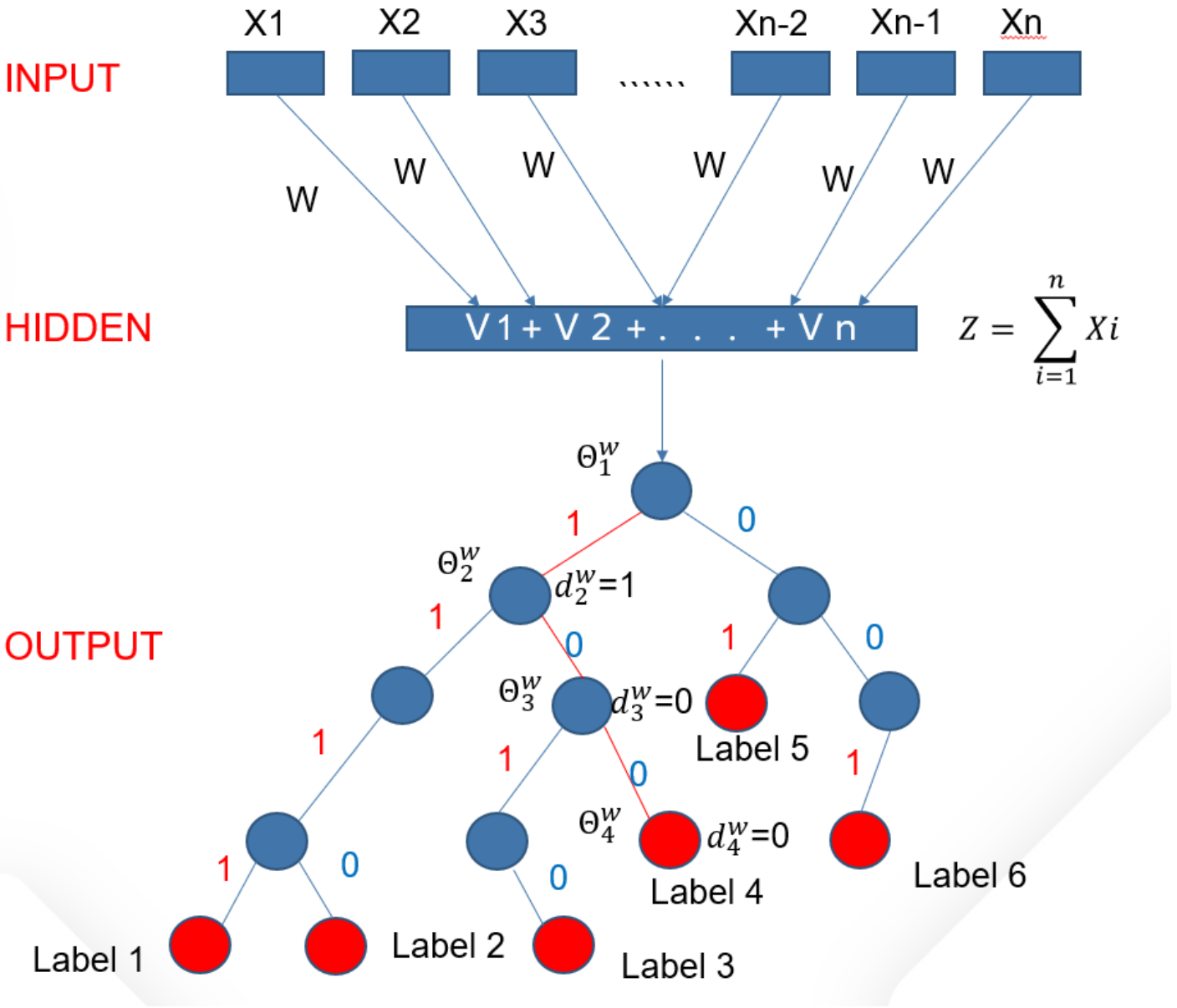
- FastText使用了n-gram来给文本添加额外的特征来得到关于局部词顺序的部分信息
- n-gran feature的优点:
  - 1. 为罕见的单词生成更好的单词嵌入
  - 2. 在词汇单词中，即使单词没有出现在训练语料库中，它们也可以从字符级n-gram中构造单词的向量。
  - 3. 采用n-gram额外特征来得到关于局部词顺序的部分信息 。
- n-gran feature的缺点:
  - 随着语料库大小的增长，内存需求也会增长 。
- 解决方案：
  - 1. 过滤掉出现次数过少的词 。
  - 2. 使用hash存储。
  - 3. 由采用字粒度变为采用词粒度

FastText应用的技术

- FastText使用了Hierarchical softmax来对大数据集进行加速。
- FastText根据语料中每个标签出现的次数构造一颗Huffman树，Huffman树的结构如下：



FastText的模型结构



FastText的模型结构

- 模型总共分为三层，输入层输入的是一个已经分词后的短文本。
- 如果用普通softmax训练，每个标签都需要计算，而采用霍夫曼树，标签数量多，权重就高，霍夫曼编码自然就越短，这样按照霍夫曼编码路径计算标签，就可以极大减少计算量。
- 短文本中每个词的词向量是由该短文本的ont-hot矩阵乘以一个初始化的矩阵w得到的。

FastText的INPUT->HIDDEN

- X1、X2…Xn是单词以及字符的n-gram特征的one-hot表示。
- V1、V2…Vn是单词以及字符的n-gram特征的密集向量表示 。
- Z为V1、V2…Vn的累加和。
- W是一个矩阵其长度与one-hot向量的长度一致，但是其宽度为50~300之间的一个数。开始的W中的值是随机初始化的。

FastText的HIDDEN->OUTPUT

- p^w：从根节点出发到达w对应叶子节点的路径。
- l^w：路径p^w中包含节点的个数 。
- $p_1^w, p_2^w, \dots, p_{l^w}^w$：路径上的节点，$p_{l^w}^w$表示词$w$对应的节点。
- $d_2^w, d_3^w, \dots, d_{l^w}^w \in \{0,1\}$：词$w$的Huffman编码，有$l^w - 1$个。
- $\Theta_1^w, \Theta_2^w, \dots, \Theta_{l^w}^w \in R^m$：路径上非叶子节点对应的向量

- 这棵Huffman树的叶子节点代表的是标签，根据语料中每个标签出现的次数构成的。

FastText Vs word2vec

- 相似点：
  - 图模型结构都是采用embedding向量的形式，得到word的隐向量表达
  - 都采用很多相似的优化方法，比如使用Hierarchical softmax优化训练和预测中的打分速度。
- 不同点：
  - 输出层：word2vec的输出层，对应的是每一个term，计算某term的概率最大；而fasttext的输出层对应的是 分类的label。
  - 输入层：word2vec的输入层，是 context window 内的term；而fasttext对应的是整个sentence的内容，包括term，也包括 n-gram的内容。
  - h-softmax的使用：Word2vec目的是得到词向量，最终是在输入层得到，输出层对应的h-softmax也会生成一系列向量，但不会使用；fasttext则充分利用h-softmax的分类功能，遍历分类树的所有叶节点，找到概率最大的label（一个或者N个）。

PART 08

08

基于CNN分类：textcnn

TextCNN的网络结构

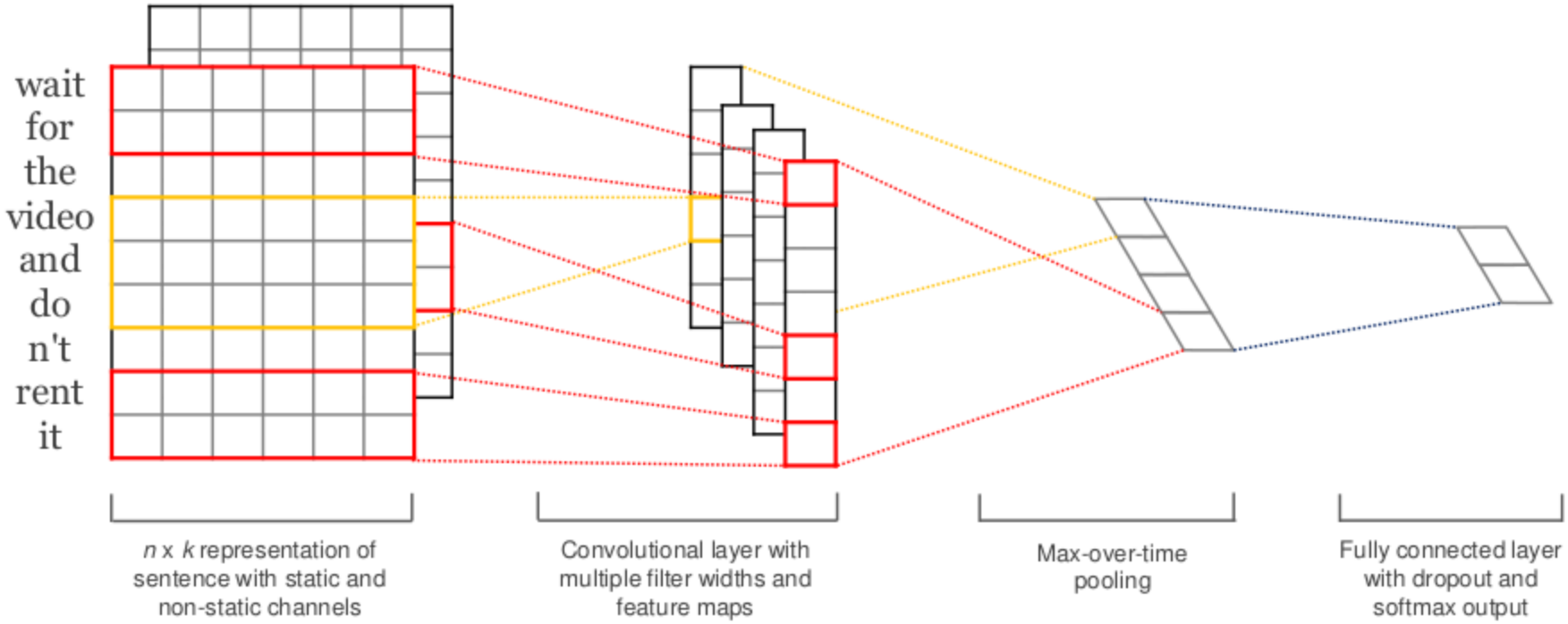


Figure 1: Model architecture with two channels for an example sentence.

TextCNN的网络结构

- 输入
  - 输入是预训练好的词向量（Word2Vector或者glove）
  - 每一个词向量都是通过无监督的方法训练得到的。
- 卷积(convolution)
  - 卷积核的宽度一般需要与词向量的维度一样，图上的维度是6 。
  - 卷积核的高度则是一个超参数可以设置，比如设置为2、3等如图。
- 池化(pooling)
  - 这里的池化操作是max-overtime-pooling，其实就是在对应的feature map求一个最大值。
- 优化、正则化
  - 加上全连接层和SoftMax层做分类任务，同时防止过拟合，一般会添加L2和Dropout正则化方法。最后整体使用梯度法进行参数的更新模型的优化。

TextCNN的注意事项

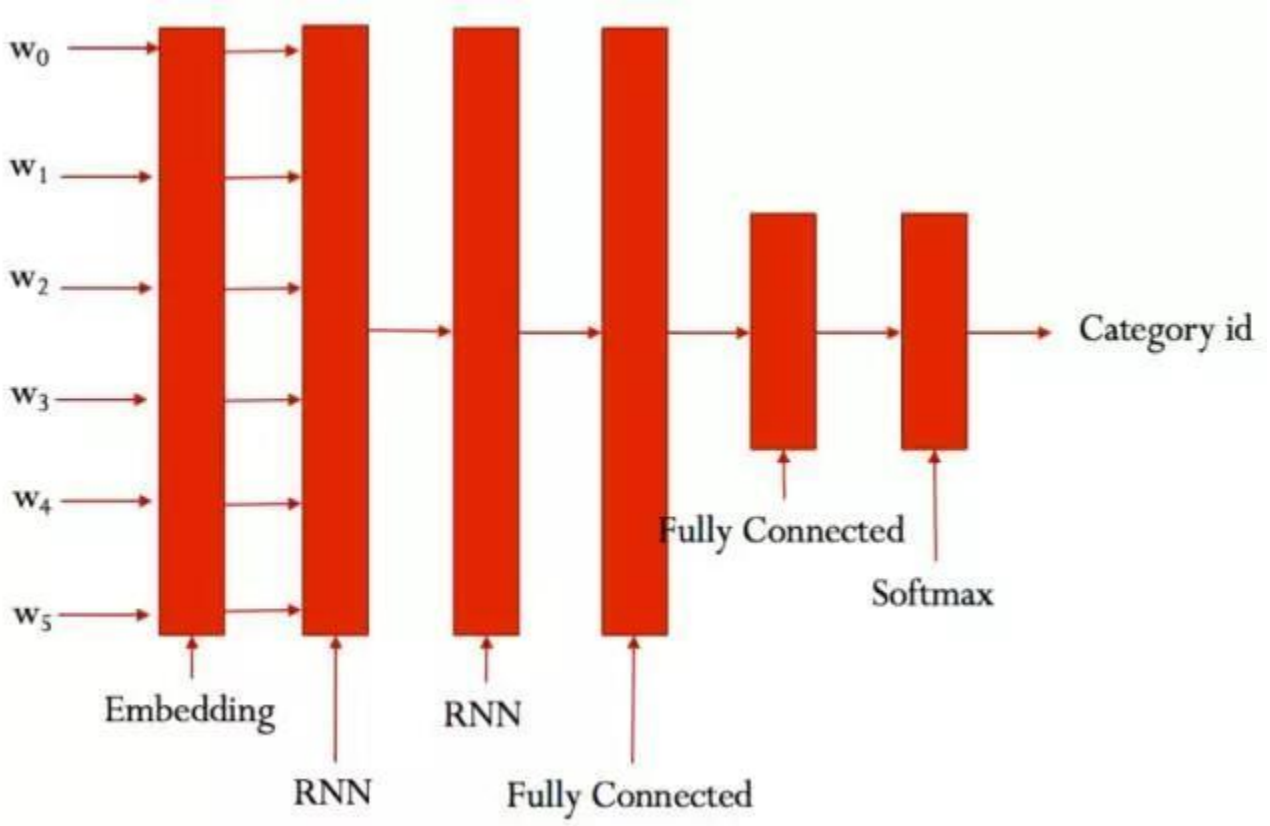
- （1）使用预训练的word2vec 、 GloVe初始化效果会更好。一般不直接使用One-hot。
- （2）卷积核的大小影响较大，一般取1~10，对于句子较长的文本，则应选择大一些。
- （3）卷积核的数量也有较大的影响，一般取100~600 ，同时一般使用Dropout（0~0.5）。
- （4）激活函数一般选用ReLU 和 tanh。
- （5）池化使用1-max pooling。
- （6）随着feature map数量增加，性能减少时，试着尝试大于0.5的Dropout。
- （7）评估模型性能时，记得使用交叉验证。

PART 09

09

基于RNN分类：textrnn

TextRNN的网络结构



TextRNN的介绍

- 循环神经网络（RNN, Recurrent Neural Network），能够更好的表达上下文信息。
- 与TextCNN相比，将Conv+Pooling替换成了Bi-LSTM使用的是双向LSTM，最后将两个方向上的输出进行拼接再传给输出层即可。
- RNN模型在训练时速度很慢。

PART 10

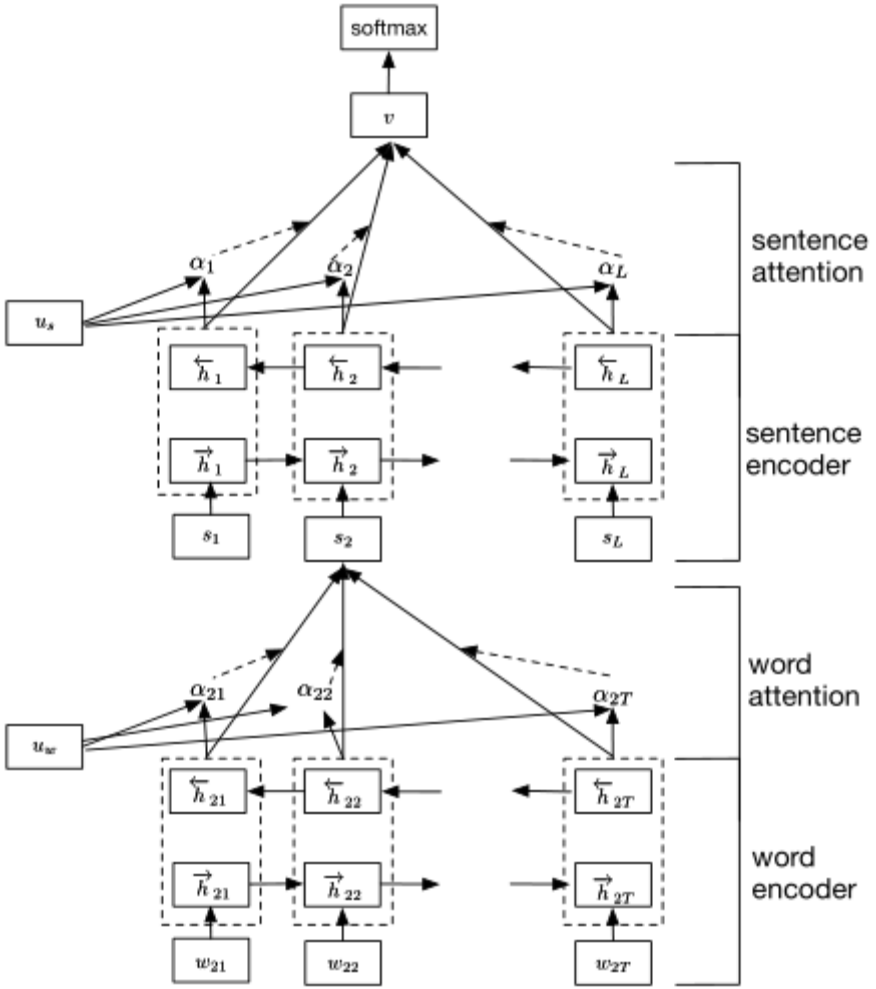
10

基于注意力机制HAN

HAN简介

- Attention机制应用的假设是对句子的含义，观点，情感等任务，每个单词的贡献是不相同的。
- Attention机制的优点：
  - 提升分类性能
  - 识别出影响最终分类决策的单词或句子的重要性。
- HAN（Hierarchical Attention Network）的基本思路：
  - 将文本按层次分成单词、句子、文本三层关系，
  - 分别用两个Bi-LSTM模型去建模word-sentence、sentence-doc的模型。
  - 在每个模型中都引入Attention机制来捕获更长的依赖关系，从而得出不同词/句子在构建句子/文本时的重要程度。

HAN的网络模型


softmax
v
sentence attention
sentence encoder
word attention
word encoder

注意力机制：加性注意力、乘性注意力、自注意力

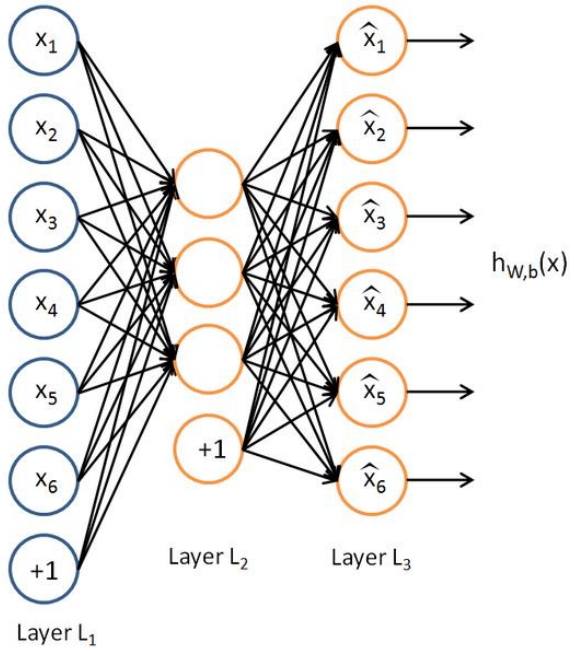
PART 11

11

Autoencoder介绍

Autoencoder介绍

- 自编码器(Autoencoder，AE)是一种前馈无返回的神经网络，有一个输入层，一个隐含层，一个输出层，可以学到输入数据的高效表示。
- 自编码器的作用：
  - 降维和数据压缩
  - 可作为强大的特征检测器（feature detectors），应用于深度神经网络的预训练
  - 可以随机生成与训练数据类似的数据



常用Autoencoder

- Sparse AutoEncoder稀疏自动编码器
  - 对自动编码器加入一个正则化项，约束隐含层神经元节点大部分输出0，少部分输出非0。
  - 稀疏编码器大大减小了需要训练的参数的数目，降低了训练的难度，同时克服了自动编码器容易陷入局部及小值和存在过拟合的问题。
- Denoising AutoEncoders降噪自动编码器
  - 对输入数据进行部分“摧毁”，然后通过训练自动编码器模型，重构出原始输入数据，以提高自动编码器的鲁棒性。对输入数据进行“摧毁”的过程其实类似于对数据加入噪声。
  - 降噪编码器采用有噪声的输入数据来训练网络参数，提高了自动编码器的泛化能力。

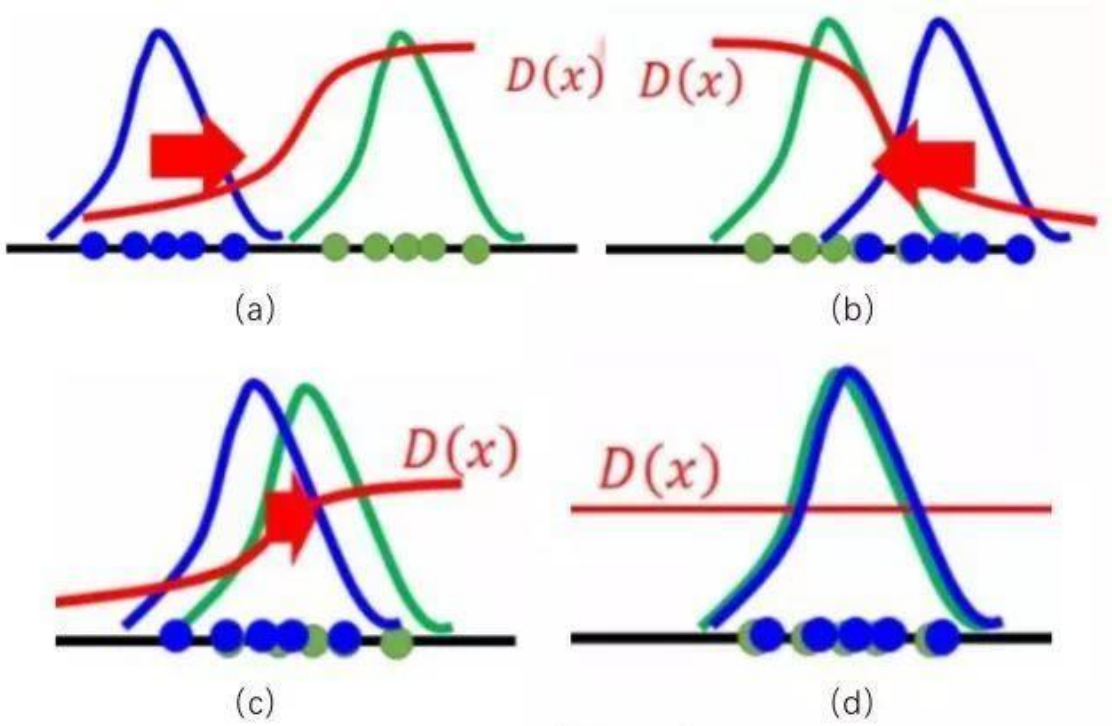
PART 12

12

GAN

GAN

- 生成式对抗网络（Generative Adversarial Networks，GANs）应用于数据的生成，其训练2个模型：
  - 仿照原始数据分布生成数据的模型G和评估数据来源（原始数据/生成数据）的模型D。
  - 训练G的目标是最大化D犯错的概率
  - 训练D的目标是最大化区分真实训练样本与G生成的样本的能力。
- GAN的基本思想：



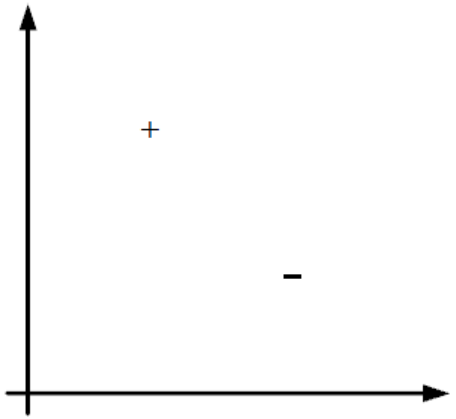
PART 13

13

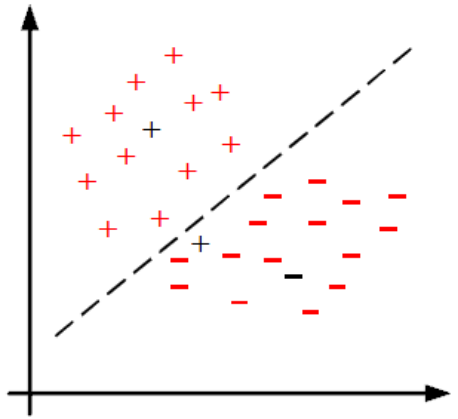
迁移学习

迁移学习

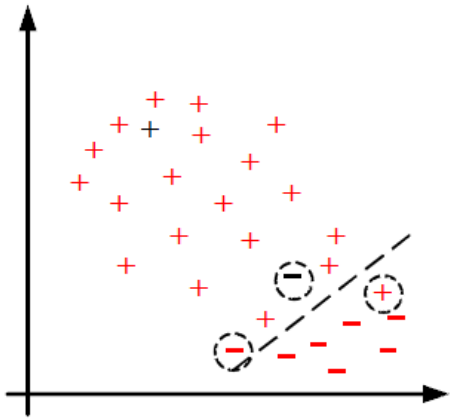
- 迁移学习（Transfer Learning）目标是将从一个环境中学到的知识用来帮助新环境中的学习任务。
- 经典算法 TrAdaBoost



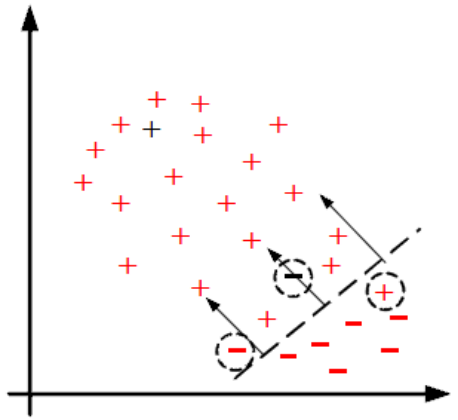
(a) 当有标注的训练样本很少的时候，分类学习是非常困难的



(b) 如果能有大量的辅助训练数据(红色的“+”和“−”),则可能可以根据辅助数据估计出分类面



(c) 有时辅助数据也可能会误导分类结果，例如图中黑色的“−”即被分错



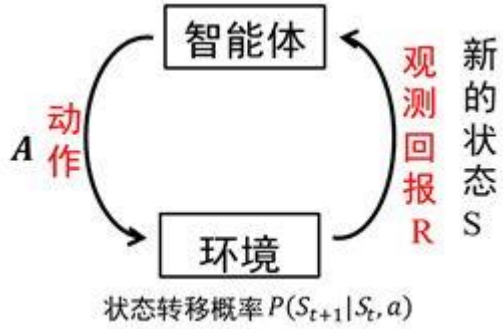
(d) TrAdaBoost算法通过增加误分类的源训练数据的权重,同时减小误分类的目标训练数据的权重,使得分类面朝正确的方向移动

PART 14

14

强化学习

强化学习



- 强化学习(reinforcement learning)的基本原理：
  - 智能体在完成某项任务时，首先通过动作A与周围环境进行交互
  - 在动作A和环境的作用下，智能体会产生新的状态，同时环境会给出一个立即回报
  - 如此循环下去，智能体与环境进行不断地交互从而产生很多数据。
  - 强化学习算法利用产生的数据修改自身的动作策略，再与环境交互，产生新的数据，并利用新的数据进一步改善自身的行为
  - 经过数次迭代学习后，智能体能最终地学到完成相应任务的最优动作（最优策略）。、
- 强化学习 Vs 监督学习和非监督学习
  - 在监督学习和非监督学习中，数据是静态的不需要与环境进行交互，只要给足够的差异样本，将数据输入到深度网络中进行训练即可。
  - 强化学习的学习过程是个动态的，不断交互的过程，所需要的数据也是通过与环境不断地交互产生的。

课后行动

1.满意度评估；

2.制定行动计划；

3.转训周围同事；

4.同学互助答疑。

助力京东基业长青
成就员工事业发展
赋能社会价值共创

京东大学公众号

Thanks！