自然语言的序列标注问题与解决方法(2)

孙栩
信息科学技术学院
xusun@pku.edu.cn
http://klcl.pku.edu.cn/member/sunxu/index.htm

大纲

- **链状结构即通常所说的“序列标注问题”**
- **自然语言处理的序列标注问题举例**
  - 词性标注
  - 中文切词
  - 短语识别（浅层句法分析）
  - 命名实体识别
- **代表性的序列标注方法**
  - 关键问题是什么？
  - 隐马尔科夫模型 HMM
  - 结构化感知器 structured perceptron

开始讲解具体的序列标注方法

大纲

- **链状结构即通常所说的“序列标注问题”**
- **自然语言处理的序列标注问题举例**
  - 词性标注
  - 中文切词
  - 短语识别（浅层句法分析）
  - 命名实体识别
- **代表性的序列标注方法**
  - 关键问题是什么？
  - 隐马尔科夫模型 HMM
  - 结构化感知器 structured perceptron

我们这里所讲的序列标注的每个点的分类都是多元分类，而不再是二元分类了

马尔科夫模型(Markov Model)

- 一个有限的状态集合 $\{s_1, s_2, \ldots, s_N\}$
- 从一个状态转移到另一个状态，从而产生一个状态序列

$$s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$$

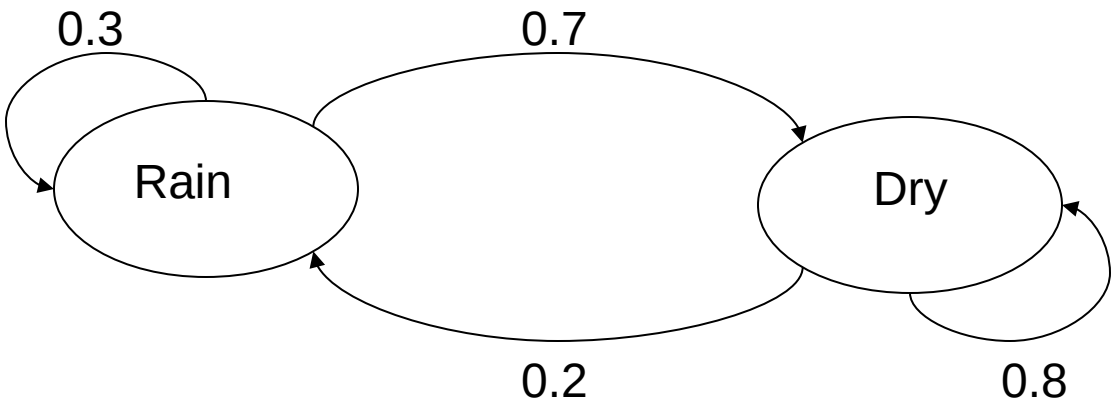
- 马尔科夫独立性假设(Markov assumption)：一个状态的概率只和之前的一个状态相关:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- 为了定义马尔科夫模型，需要定义状态和状态之间的转移概率

$$a_{ij} = P(s_i \mid s_j)$$

马尔科夫模型举例



- 两个状态 : ‘Rain’ and ‘Dry’

- 转移概率: P(‘Rain’|‘Rain’)=0.3 , P(‘Dry’|‘Rain’)=0.7 , P(‘Rain’|‘Dry’)=0.2, P(‘Dry’|‘Dry’)=0.8

- 初始概率: say P(‘Rain’)=0.4 , P(‘Dry’)=0.6

序列概率的计算

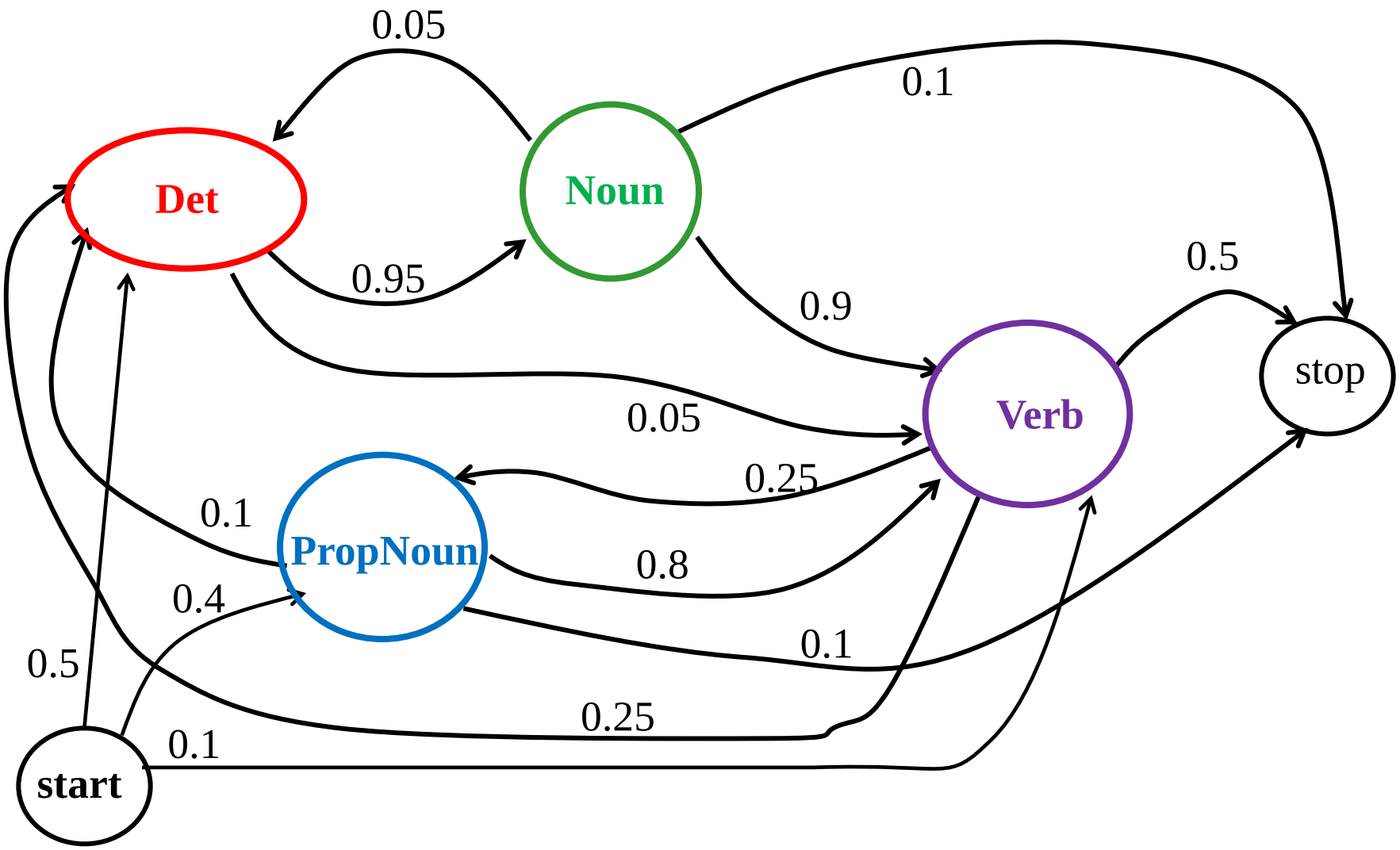
- 根据马尔科夫独立性假设，一个状态序列的概率可以这样计算：

$$P(s_{i1}, s_{i2}, \ldots, s_{ik}) = P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1})$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1}) = \ldots$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \ldots P(s_{i2} \mid s_{i1}) P(s_{i1})$$

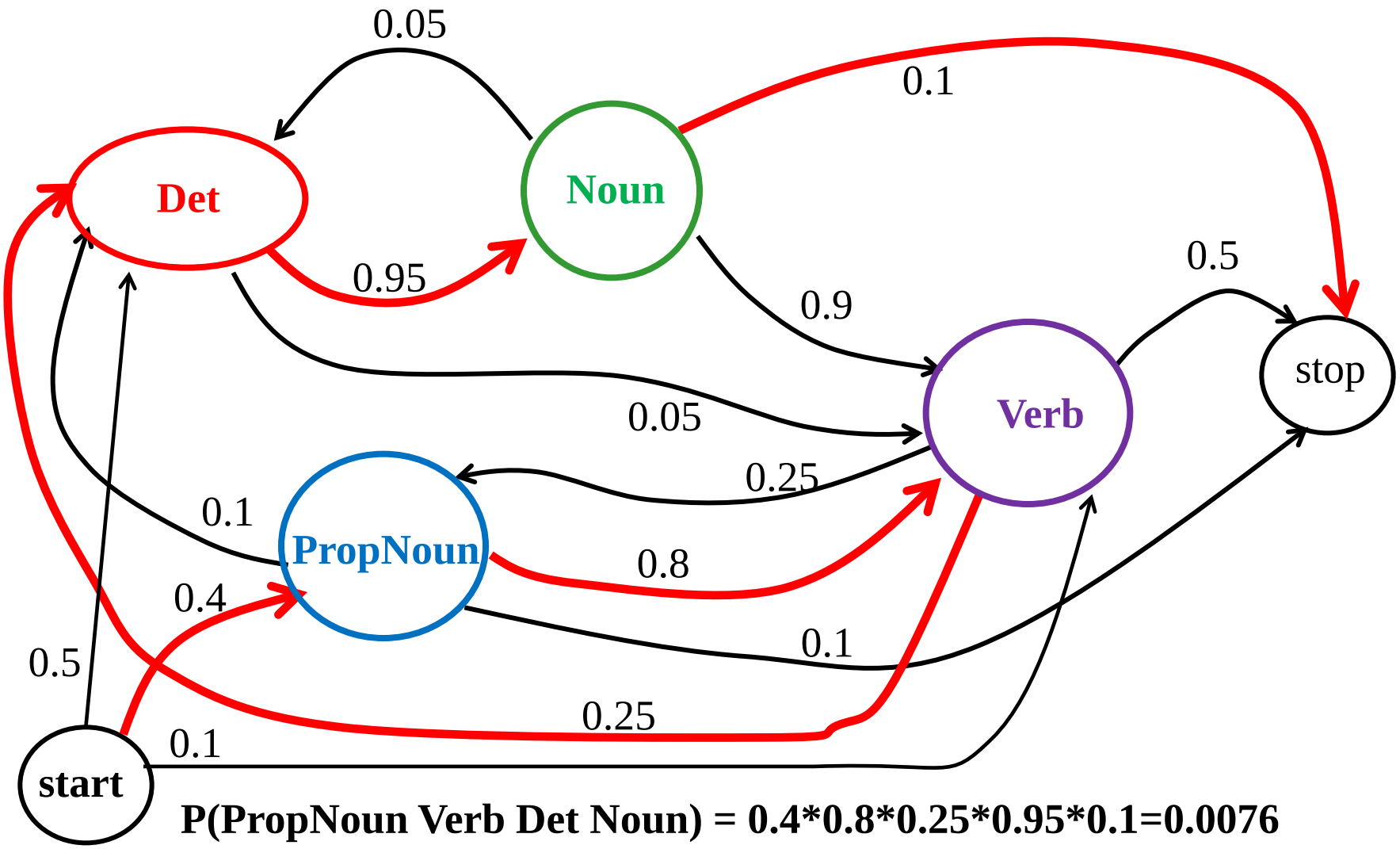
- 假设我们想计算这样的一个天气状态的序列：
{ ‘Dry’ ,‘ Dry’ ,‘ Rain’ ,Rain’ }

P({‘Dry’,’Dry’,’Rain’,Rain’})
= P(‘Rain’|’Rain’) P(‘Rain’|’Dry’) P(‘Dry’|’Dry’) P(‘Dry’)
= 0.3*0.2*0.8*0.6

用于词性标注的马尔科夫模型举例



用于词性标注的马尔科夫模型举例



P(PropNoun Verb Det Noun) = 0.4*0.8*0.25*0.95*0.1=0.0076

隐马尔科夫模型(Hidden Markov Model)

- **是对状态序列建模的一个概率生成模型(probabilistic generative model)**
- **假设一个不可见的隐藏状态集合(hidden/unobserved states)**
  - 例如，在词性标注这个任务上，一个词性(POS)就是一个隐藏状态
- **假设隐藏状态之间存在一个概率转移**
  - 例如，从一个词性概率转移到另一个词性
- **假设存在生成概率，即从一个隐藏状态可以生成若干可观测量**
  - 例如，一个词性有一定的概率生成特定的词

HMM的正式定义

- ***N* +2** 个隐藏状态 $S=\{s_0, s_1, s_2, \ldots s_N, s_F\}$
  - Distinguished start state: s_0
  - Distinguished final state: s_F
- ***M*** 个可能的观测量 $V=\{v_1, v_2 \ldots v_M\}$
- 状态转移概率分布 $A=\{a_{ij}\}$

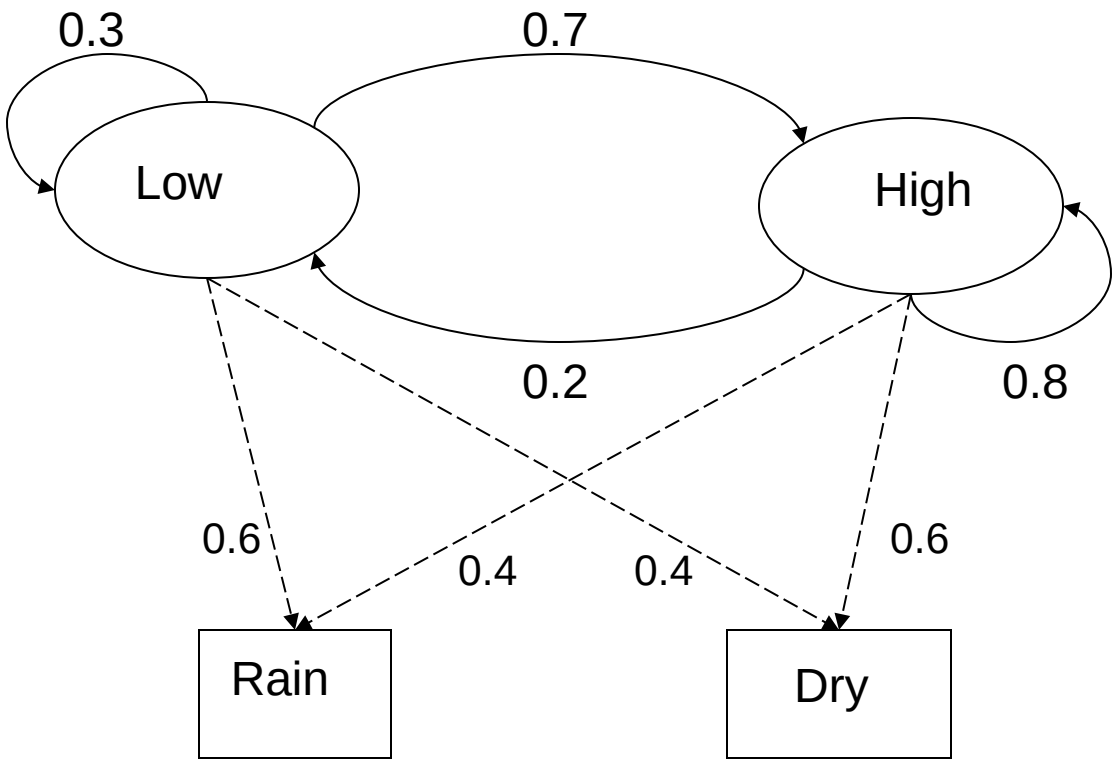
$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i) \qquad 1 \le i, j \le N \text{ and } i = 0, j = F$$

$$\sum_{j=1}^{N} a_{ij} + a_{iF} = 1 \quad 0 \le i \le N$$

- 可观测量的生成概率分布，对于状态 j，其生成观测量k的概率为 $B=\{b_j(k)\}$

$$b_j(k) = P(v_k \text{ at } t \mid q_t = s_j) \quad 1 \le j \le N \quad 1 \le \mathrm{k} \le \mathrm{M}$$

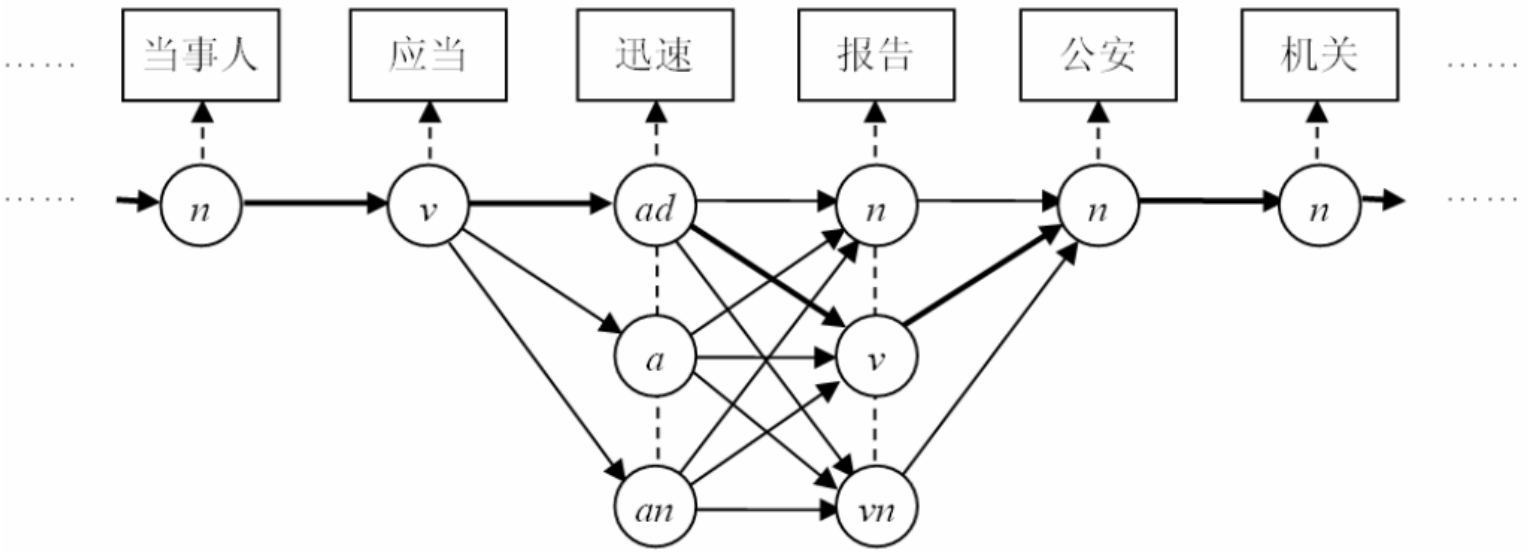
HMM举例



HMM举例

- 2个隐藏状态 : ‘Low’ and ‘High’ 气压
- 2个观测量: ‘Rain’ and ‘Dry’
- 状态转移概率: P(‘Low’|‘Low’)=0.3 , P(‘High’|‘Low’)=0.7 , P(‘Low’|‘High’)=0.2, P(‘High’|‘High’)=0.8
- 观测量生成概率: P(‘Rain’|‘Low’)=0.6 , P(‘Dry’|‘Low’)=0.4 , P(‘Rain’|‘High’)=0.4 , P(‘Dry’|‘High’)=0.3

HMM举例



HMM的观测量生成过程

- 生成有 T 个观测量的序列: $O = o_1\ o_2\ \ldots\ o_T$

Set initial state $q_1=s_0$
For t = 1 to T
    Transit to another state $q_{t+1}=s_j$ based on transition distribution a_{ij} for state q_t
    Pick an observation $o_t=v_k$ based on being in state q_t using distribution $b_{qt}(k)$

HMM的3个核心问题

- 观测概率(Observation Likelihood)
  - 给定一个观测序列，计算其概率
- 最可能的状态序列问题，即解码问题 (Decoding):
  - 对一个一个观测量序列，计算最可能的状态序列，也就是序列标注问题！
- 最大似然的参数训练，也就是学习问题(Learning)
  - 给定一个训练数据集，训练一个对应的HMM模型

观测概率问题的原始解决方案

- 对于一个给定长度为T的观测序列O，穷举所有可能的长度为T 的状态序列Q， 从而基于该状态序列HMM模型可以生成给定的观测序列
- 根据状态转移概率、观测量生成概率计算所有Q对应的概率P(O|Q)
- 把所有Q对应的概率加起来，从而得到观测序列O的总概率P(O)
- 该原始解决方法的计算复杂度为 $O(TN^T)$

观测概率问题的高效率解决方案

- 根据马尔科夫假设，时间 t 的状态只概率依赖于时间 $t-1$ 的状态

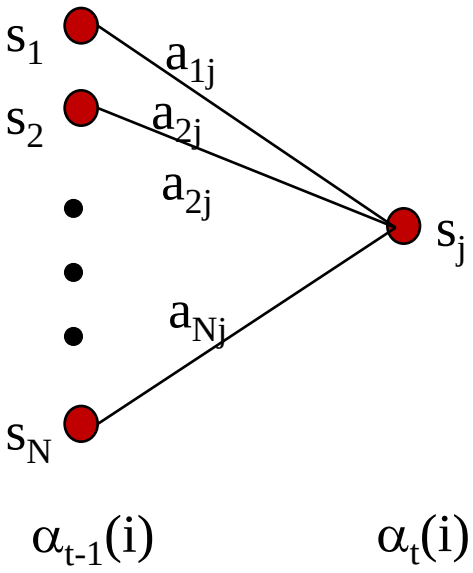
- 前向算法(Forward Algorithm): 基于马尔科夫假设，使用动态规划(dynamic programming)计算观测序列概率，时间复杂度为 $O(TN^2)$
  - 计算前向格架图(forward trellis)，从而可以以紧凑的形式计算和保存所有可能的状态转移路径

前向概率

- 令 $\alpha_t(j)$ 表示在看到了 t 个观测量之后状态为 j 的概率
  - 即累加所有导致当前状态为 j 的路径的概率

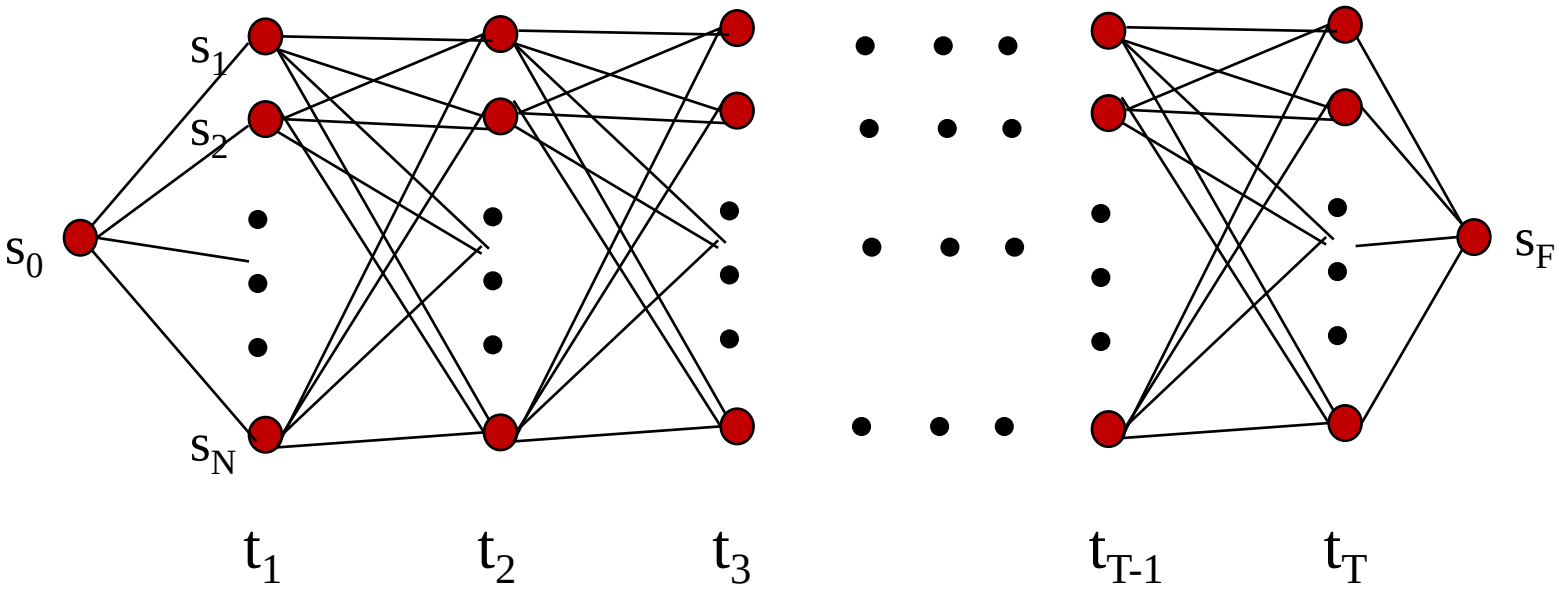
$$\alpha_t(j) = P(o_1, o_2, \ldots o_t,\ q_t = s_j \mid \lambda)$$

前向概率计算步骤



- 核心问题：假设 $\alpha_{t-1}(i)$ 已知，怎么计算 $\alpha_t(i)$？
- 考虑所有从时间 **t-1** 的状态转移到时间 **t** 的状态的转移概率
- 将这些转移概率分别和 $\alpha_{t-1}(i)$ 对应的值相乘，然后相加
- 再乘以时间 **t** 的观测量的生成概率，即得到 $\alpha_t(i)$ 对应点的值

通过前向概率计算观测概率



- 把前向概率的计算步骤，即从$\alpha_{t-1}(i)$ 计算$\alpha_t(i)$的步骤从开始点（时间0）计算到终止点（时间T），即完成观测概率P(O)的计算

通过前向概率计算观测概率

- 初始化

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$

- 递归计算

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 1 \le j \le N, \ 1 < t \le T$$

利用了马尔科夫假设，从而能够设计此动态规划算法。时间复杂度是 O(TN2)

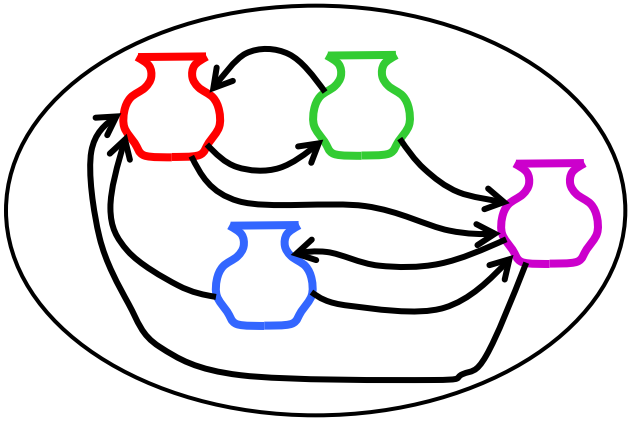
- 结束

$$P(O|\lambda) = \alpha_{T+1}(s_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

最大概率状态序列问题(解码问题)

- 给定一个观测序列O, 如何寻找最大概率的状态序列 $Q=q_1,q_2,...q_T$?

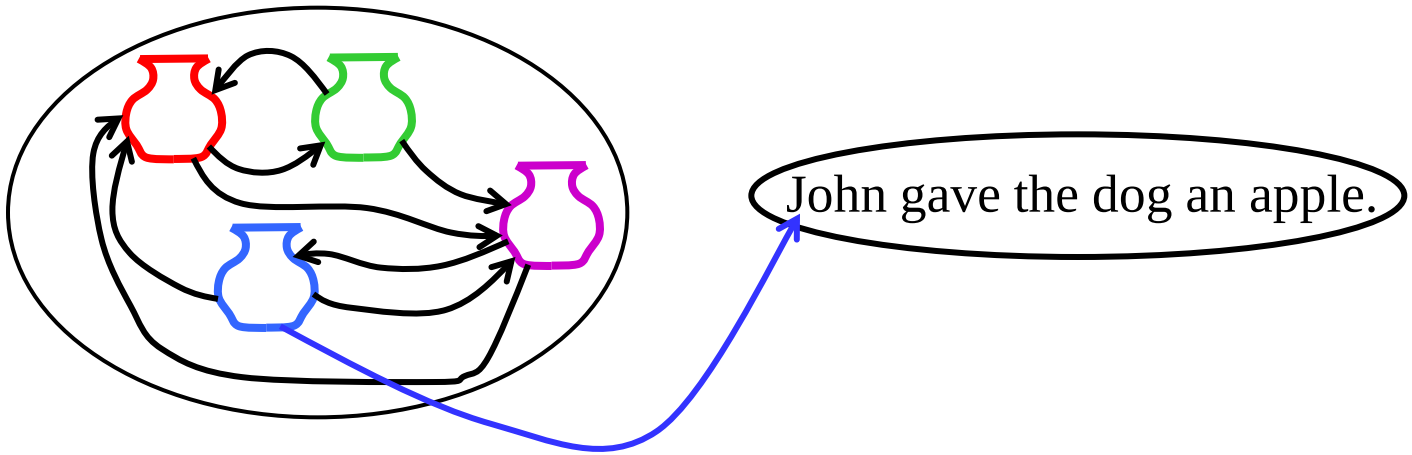
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



Most Likely State Sequence

- 给定一个观测序列O, 如何寻找最大概率的状态序列 $Q=q_1,q_2,...q_T$?

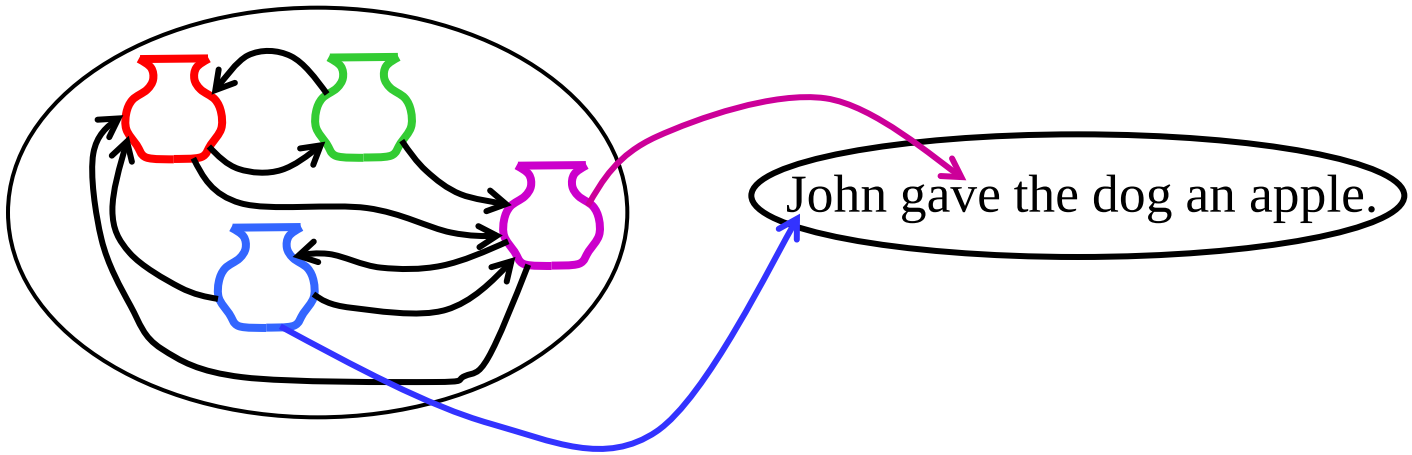
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



Most Likely State Sequence

- 给定一个观测序列O, 如何寻找最大概率的状态序列 $Q=q_1,q_2,...q_T$?

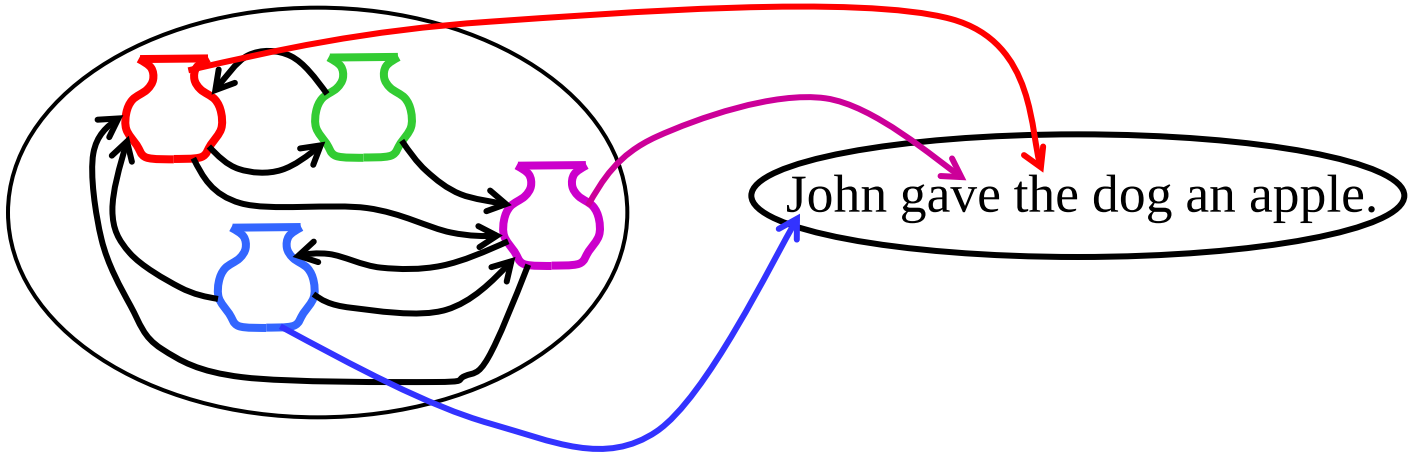
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



Most Likely State Sequence

- 给定一个观测序列O, 如何寻找最大概率的状态序列 $Q=q_1,q_2,...q_T$?

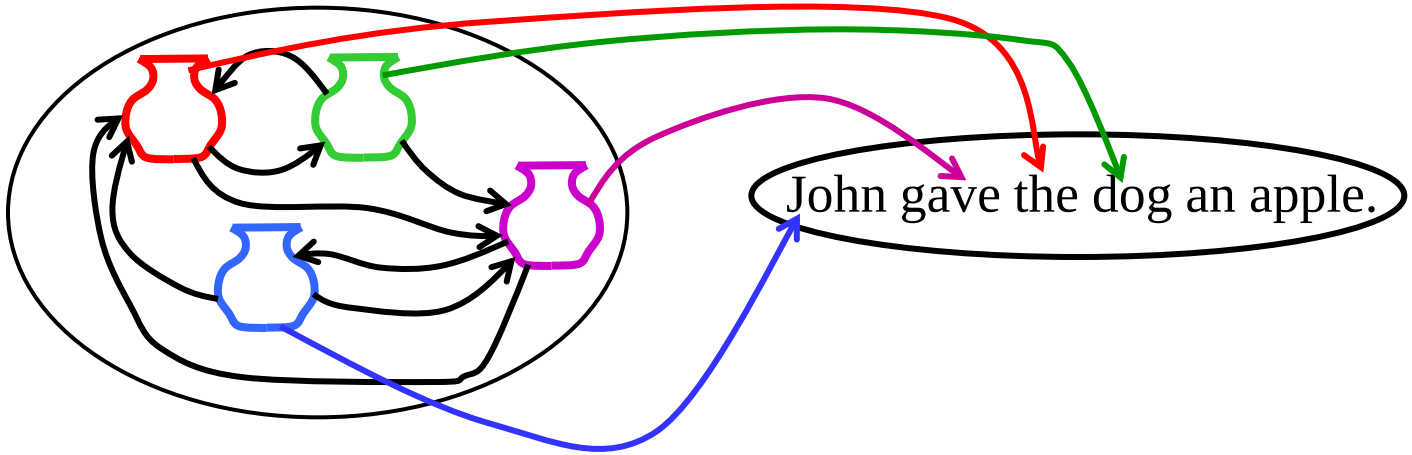
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



Most Likely State Sequence

- 给定一个观测序列O, 如何寻找最大概率的状态序列 $Q=q_1,q_2,...q_T$?

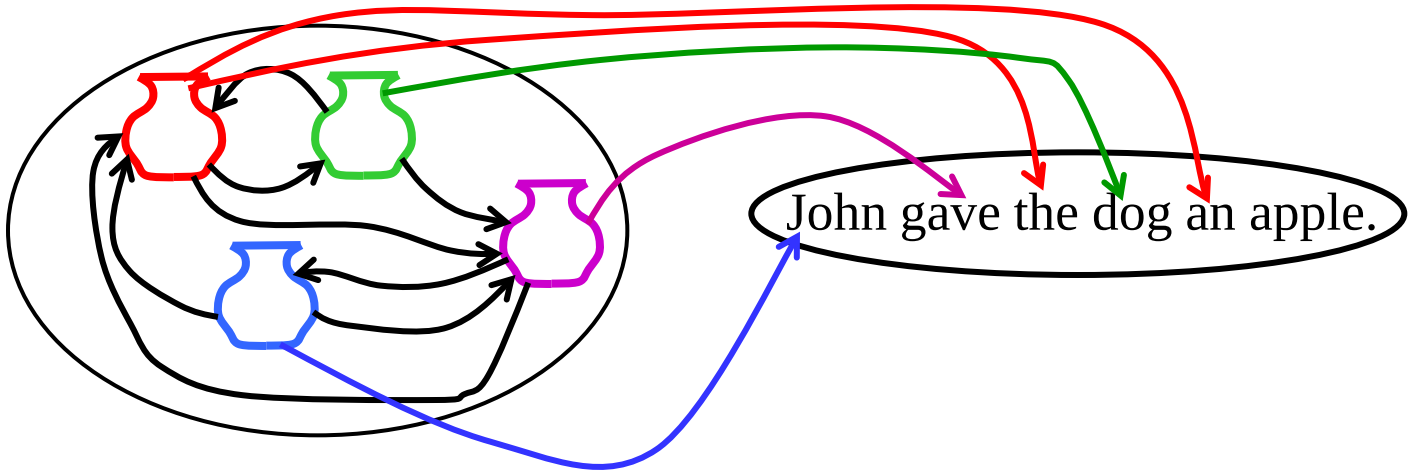
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



Most Likely State Sequence

- 给定一个观测序列O, 如何寻找最大概率的状态序列 $Q=q_1,q_2,...q_T$?

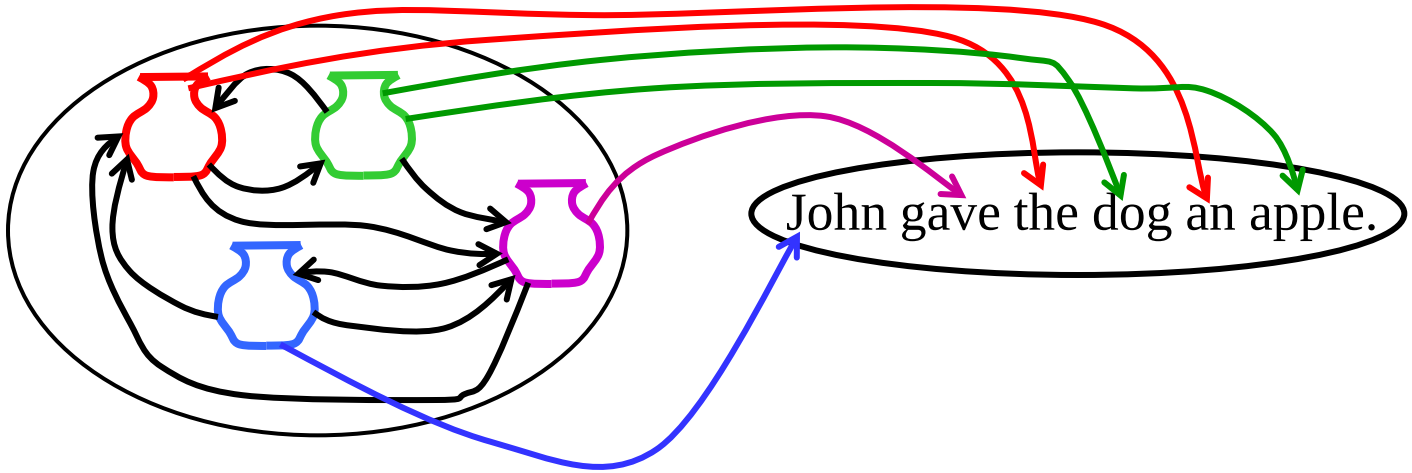
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



Most Likely State Sequence

- 给定一个观测序列O, 如何寻找最大概率的状态序列 $Q=q_1,q_2,...q_T$?

- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



解码问题的高效率算法

- **基于马尔科夫假设，同样可以使用动态规划算法**
- **有一种标准的解码动态规划算法，叫做维特比算法(Viterbi algorithm) (Viterbi, 1967)**
  - 跟前向算法一样，时间复杂度为$O(N^2T)$

维特比算法

- 递归计算“到目前为止(时间t)状态为 s_j 的最大概率状态序列”

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1, \ldots, q_{t-1}, o_1, \ldots, o_t, q_t = s_j \mid \lambda)$$

- 在递归计算过程中，记录“回溯指针”(backpointers)，从而能够回溯寻找最大概率的状态序列
  - 回溯指针 $bt_t(j)$ 记录了时间 t-1 所对应的状态，从而使得时间 t 的状态 s_j 达到其对应的最大概率

维特比算法：递归计算过程

- 初始化

$$v_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$

- 递归计算

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \ 1 < t \le T$$

- 算法结束

$$P^* = v_{T+1}(s_F) = \max_{i=1}^{N} v_T(i) a_{iF}$$

跟前向算法很相似，只是把 求和 步骤改成了 max 步骤

维特比算法：回溯指针的记录过程

- 初始化

$$bt_1(j) = s_0 \quad 1 \le j \le N$$

- 递归计算

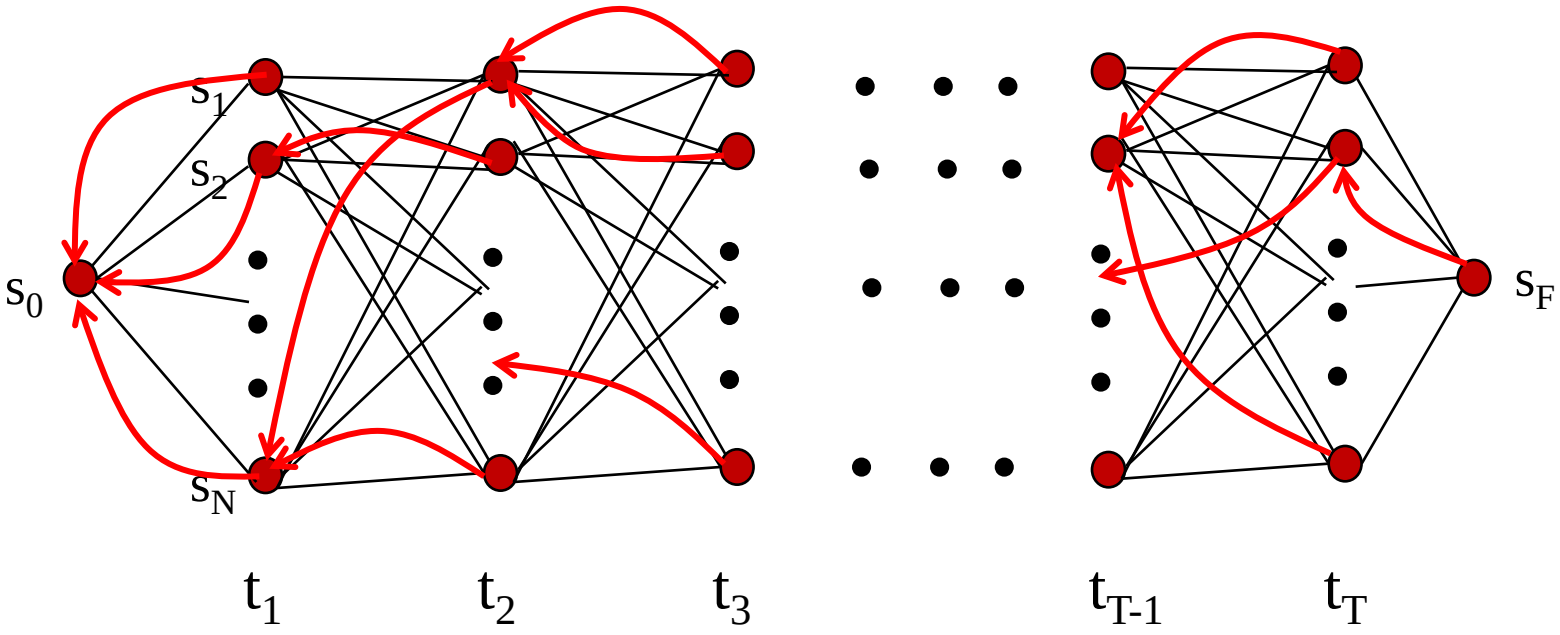
$$bt_t(j) = \underset{i=1}{\overset{N}{\operatorname{argmax}}} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \quad 1 \le t \le T$$

- 算法结束

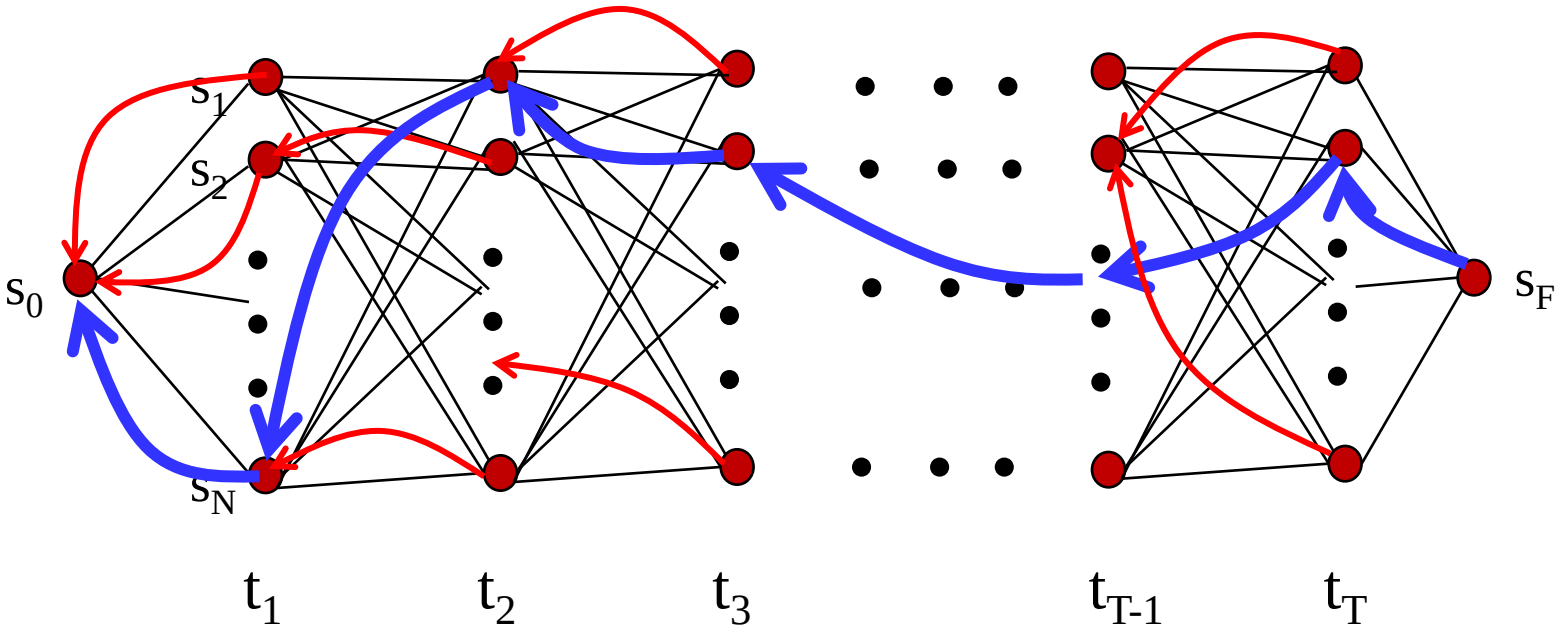
$$q_T^* = bt_{T+1}(s_F) = \underset{i=1}{\overset{N}{\operatorname{argmax}}} v_T(i) a_{iF}$$

算法结束后，可以通过记录的回溯指针从后往前寻找最大概率的状态序列

维特比算法：回溯



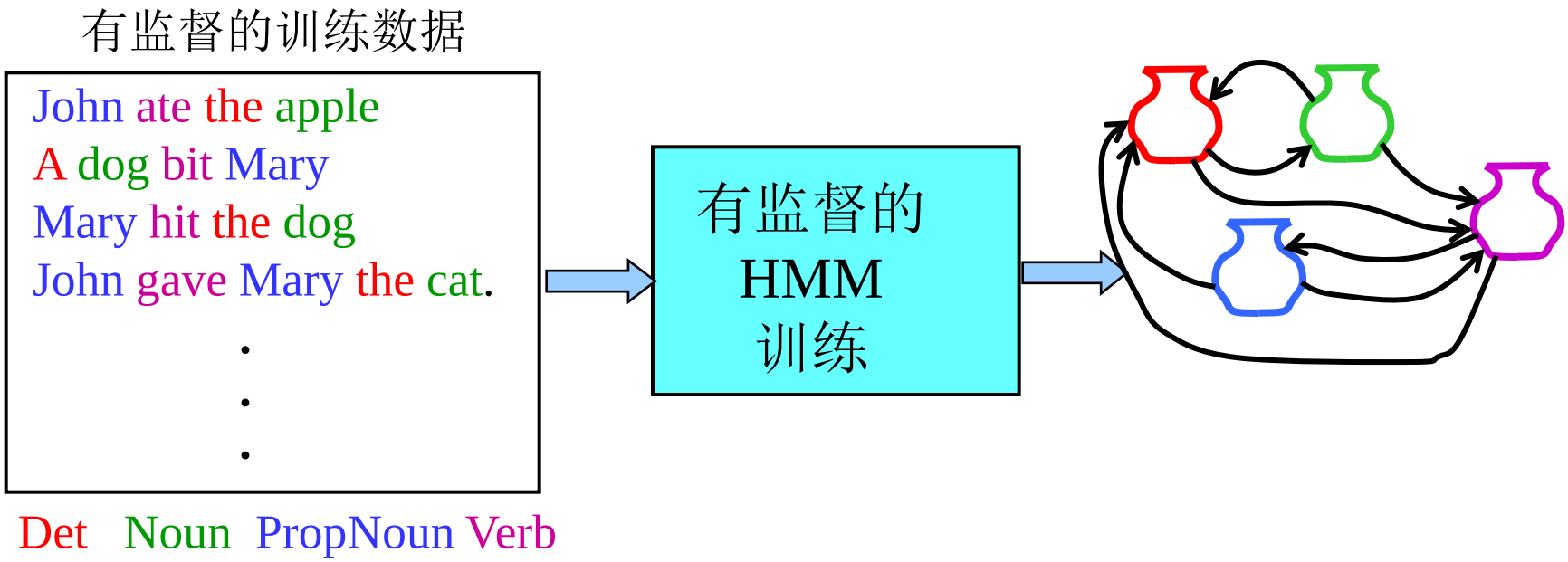
维特比算法：回溯



最大概率的状态序列: $s_0\ s_N\ s_1\ s_2\ \ldots s_2\ s_F$

HMM的有监督学习问题

- 如果训练数据已经都标注好了标准答案，则HMM的相关参数λ={A,B} 可以之间进行计算



HMM的有监督学习问题

- 状态转移概率可以直接通过bigram 和 unigram 信息进行计算

$$a_{ij} = \frac{C(q_t = s_i, \mathrm{q}_{\mathrm{t+1}} = s_j)}{C(q_t = s_i)}$$

- 生成概率也可以通过tag/word 的共现信息直接计算

$$b_j(k) = \frac{C(q_i = s_j, o_i = v_k)}{C(q_i = s_j)}$$

- 如果训练数据比较稀疏，可以采用一些 smoothing 算法

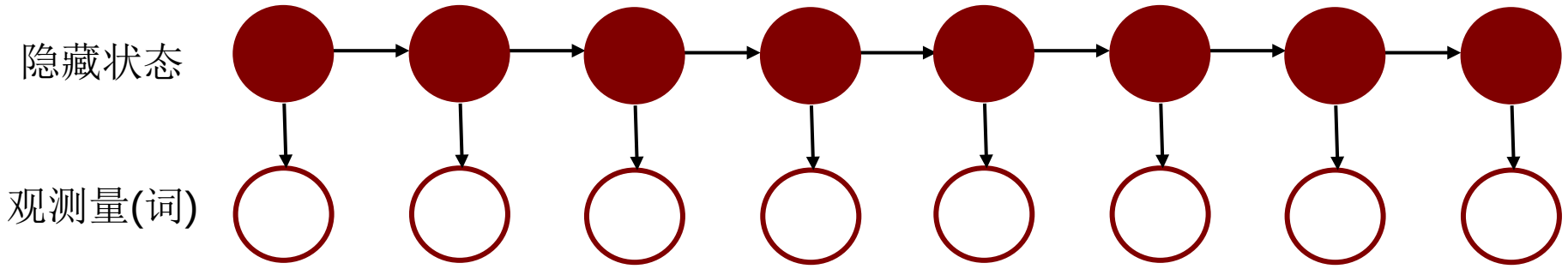
HMM优缺点

- 优点:
  - 一个可以寻找全局最优的模型，适合序列标注问题
  - 参数训练相对简单

HMM优缺点

- 优点:
  - 一个可以寻找全局最优的模型，适合序列标注问题
  - 参数训练相对简单

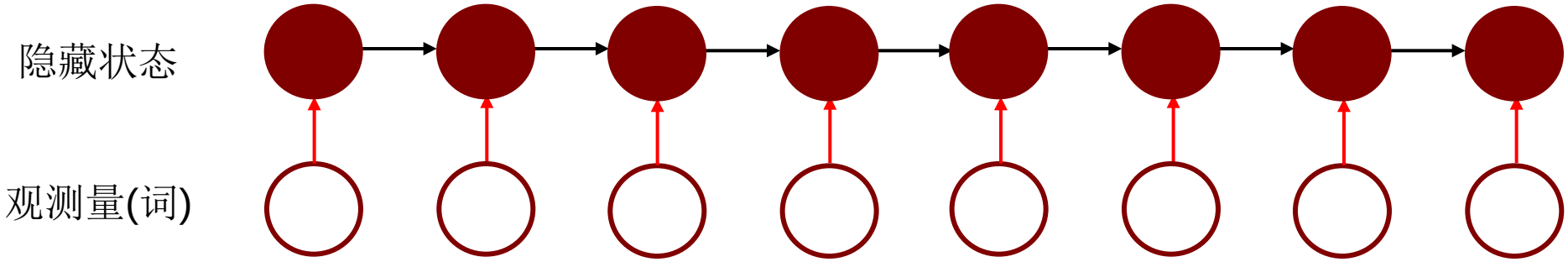
- 缺点:
  - HMM是一个**生成模型(generative model)**，解码过程需要建模 (x,y) 的联合概率分布，以及生成概率。
    → 对于序列标注问题来说，这相当于绕弯路



大纲

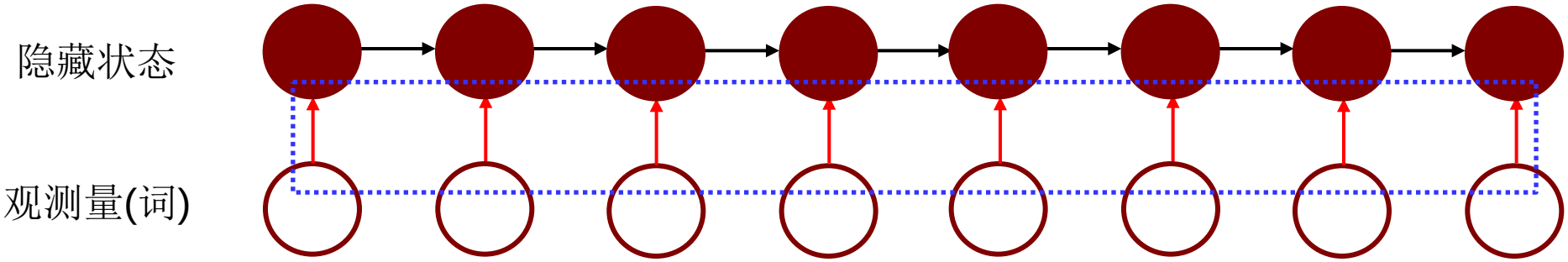
- **链状结构即通常所说的“序列标注问题”**
- **自然语言处理的序列标注问题举例**
  - 词性标注
  - 中文切词
  - 短语识别（浅层句法分析）
  - 命名实体识别
- **代表性的序列标注方法**
  - 关键问题是什么？
  - 隐马尔可夫模型 HMM
  - 结构化感知器 structured perceptron

判别模型 Discriminative Models



判别模型(discriminative models)是自然语言处理序列标注问题的更好的选择，不需要绕弯路

判别模型 Discriminative Models



只解决需要解决的问题！ “Solve the problem you need to solve”

判别模型(discriminative models)是自然语言处理序列标注问题的更好的选择，不需要绕弯路

感知器模型

- **感知器模型(perceptron)**

回顾之前讲的内容

- **假设问题是线性可分的**
  - 我们需要一种学习方法，能够较快速地收敛到稳定状态，实现自动分类(classification)

- **主要思路**
  - 如果遇到一个新实例（比如句子、文本），跟原有实例(已知分类结果)相似的实例更有可能被分类为相似的类

早期思想由Rosenblatt在1950年代提出，但是现有的perceptron模型和原来早期的模型已经有了较大的不同。经过了大幅度算法改进，如今使用很广泛。

感知器模型

- **主要步骤：**
  - 随机初始化一个超平面
  - 一个接一个扫描训练数据（已经标注了正确的分类结果），基于现有的模型参数(weight vector)，计算分类结果
  - 如果分类结果正确，则继续
  - 如果分类错误，则修改模型参数，加上正确分类结果对应的特征向量，减去错误分类结果对应的特征向量
  - 如果达到收敛状态（稳定状态），则结束

回顾之前讲的内容

主要受到神经元网络的启发

- 生物学的解释：
- 有点像大脑神经元的正向反馈和负向反馈

感知器模型

具体算法

回顾之前讲的内容

Input: *example* $\mathbf{x}_i$ *with gold standard label sequence* $\mathbf{y}_i^*$, *weight vector* Θ, *and feature vector* $\mathbf{f}(\mathbf{y}, \mathbf{x})$
Initialization: *set parameters* $\Theta^1 = 0$
for $i = 1 \ldots d$ **do**
  $\mathbf{y}_i = \text{argmax}_{\mathbf{y}} F(\mathbf{y}|\mathbf{x}_i, \Theta^i)$
  if $\mathbf{y}_i \neq \mathbf{y}_i^*$ **then**
    $\Theta^{i+1} = \Theta^i + \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}_i, \mathbf{x}_i)$
  else
    $\Theta^{i+1} = \Theta^i$
Output: *parameter vectors* Θ^{i+1} *for* $i = 1 \ldots d$

- 感知器模型是基于简单的加减法！
  - 优点一：非常容易实现
  - 优点二：而且实际效果好

结构化感知器模型

具体算法

Input: *example* $\mathbf{x}_i$ *with gold standard label sequence* $\mathbf{y}_i^*$, *weight vector* $\mathbf{\Theta}$, *and feature vector* $\mathbf{f}(\mathbf{y}, \mathbf{x})$
Initialization: *set parameters* $\mathbf{\Theta}^1 = 0$
for $i = 1 \ldots d$ **do**
  $\mathbf{y}_i = \text{argmax}_{\mathbf{y}} F(\mathbf{y}|\mathbf{x}_i, \mathbf{\Theta}^i)$
  if $\mathbf{y}_i \neq \mathbf{y}_i^*$ **then**
    $\mathbf{\Theta}^{i+1} = \mathbf{\Theta}^i + \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}_i, \mathbf{x}_i)$
  else
    $\mathbf{\Theta}^{i+1} = \mathbf{\Theta}^i$
Output: *parameter vectors* $\mathbf{\Theta}^{i+1}$ *for* $i = 1 \ldots d$

是的，跟原来的非结构化的情况相比，算法基本上一样！
只有2个不同的地方

- (1) 计算 **argmax_y** 的具体实现不一样
- (2) y 从一个单变量变成了一个向量，从而计算特征向量 **f(y, x)** 的具体实现不一样

结构化感知器模型

具体算法

Input: *example* $\mathbf{x}_i$ *with gold standard label sequence* $\mathbf{y}_i^*$, *weight vector* Θ, *and feature vector* $\mathbf{f}(\mathbf{y}, \mathbf{x})$
Initialization: *set parameters* $\Theta^1 = 0$
for $i = 1 \ldots d$ **do**
  $\mathbf{y}_i = \text{argmax}_{\mathbf{y}} F(\mathbf{y}|\mathbf{x}_i, \Theta^i)$
  if $\mathbf{y}_i \neq \mathbf{y}_i^*$ **then**
    $\Theta^{i+1} = \Theta^i + \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}_i, \mathbf{x}_i)$
  else
    $\Theta^{i+1} = \Theta^i$
Output: *parameter vectors* Θ^{i+1} *for* $i = 1 \ldots d$

是的，跟原来的非结构化的情况相比，算法基本上一样！
只有2个不同的地方

- (1) 计算 **argmax_y** 的具体实现不一样
- (2) y 从一个单变量变成了一个向量，从而计算特征向量 **f(y, x)** 的具体实现不一样

结构化感知器模型

具体算法

Input: *example* $\mathbf{x}_i$ *with gold standard label sequence* $\mathbf{y}_i^*$, *weight vector* Θ, *and feature vector* $\mathbf{f}(\mathbf{y}, \mathbf{x})$
Initialization: *set parameters* $\Theta^1 = 0$
for $i = 1 \ldots d$ **do**
  $\mathbf{y}_i = \text{argmax}_{\mathbf{y}} F(\mathbf{y}|\mathbf{x}_i, \Theta^i)$
  if $\mathbf{y}_i \neq \mathbf{y}_i^*$ **then**
    $\Theta^{i+1} = \Theta^i + \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}_i, \mathbf{x}_i)$
  else
    $\Theta^{i+1} = \Theta^i$
Output: *parameter vectors* Θ^{i+1} *for* $i = 1 \ldots d$

是的，跟原来的非结构化的情况相比，算法基本上一样！
只有2个不同的地方

- (1) 计算 **argmax_y** 的具体实现不一样
- (2) y 从一个单变量变成了一个向量，从而计算特征向量 **f(y, x)** 的具体实现不一样

结构化感知器模型

不同点1： 计算 argmax_y

Input: *example* $\mathbf{x}_i$ *with gold standard label sequence* $\mathbf{y}_i^*$, *weight vector* Θ, *and feature vector* $\mathbf{f}(\mathbf{y}, \mathbf{x})$
Initialization: *set parameters* $\Theta^1 = 0$
for $i = 1 \ldots d$ **do**
  $\mathbf{y}_i = \text{argmax}_{\mathbf{y}} F(\mathbf{y}|\mathbf{x}_i, \Theta^i)$
  if $\mathbf{y}_i \neq \mathbf{y}_i^*$ **then**
    $\Theta^{i+1} = \Theta^i + \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}_i, \mathbf{x}_i)$
  else
    $\Theta^{i+1} = \Theta^i$
Output: *parameter vectors* Θ^{i+1} *for* $i = 1 \ldots d$

- 直接通过前面介绍的通用动态规划算法 → 维特比算法计算 y* = argmax_y
- 时间复杂度为O(N2T)

维特比算法：递归计算过程

- 初始化

$$v_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$

- 递归计算

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \ 1 < t \le T$$

- 算法结束

$$P^* = v_{T+1}(s_F) = \max_{i=1}^{N} v_T(i) a_{iF}$$

跟前向算法很相似，只是把 求和 步骤改成了 max 步骤

维特比算法：回溯指针的记录过程

- 初始化

$$bt_1(j) = s_0 \quad 1 \le j \le N$$

- 递归计算

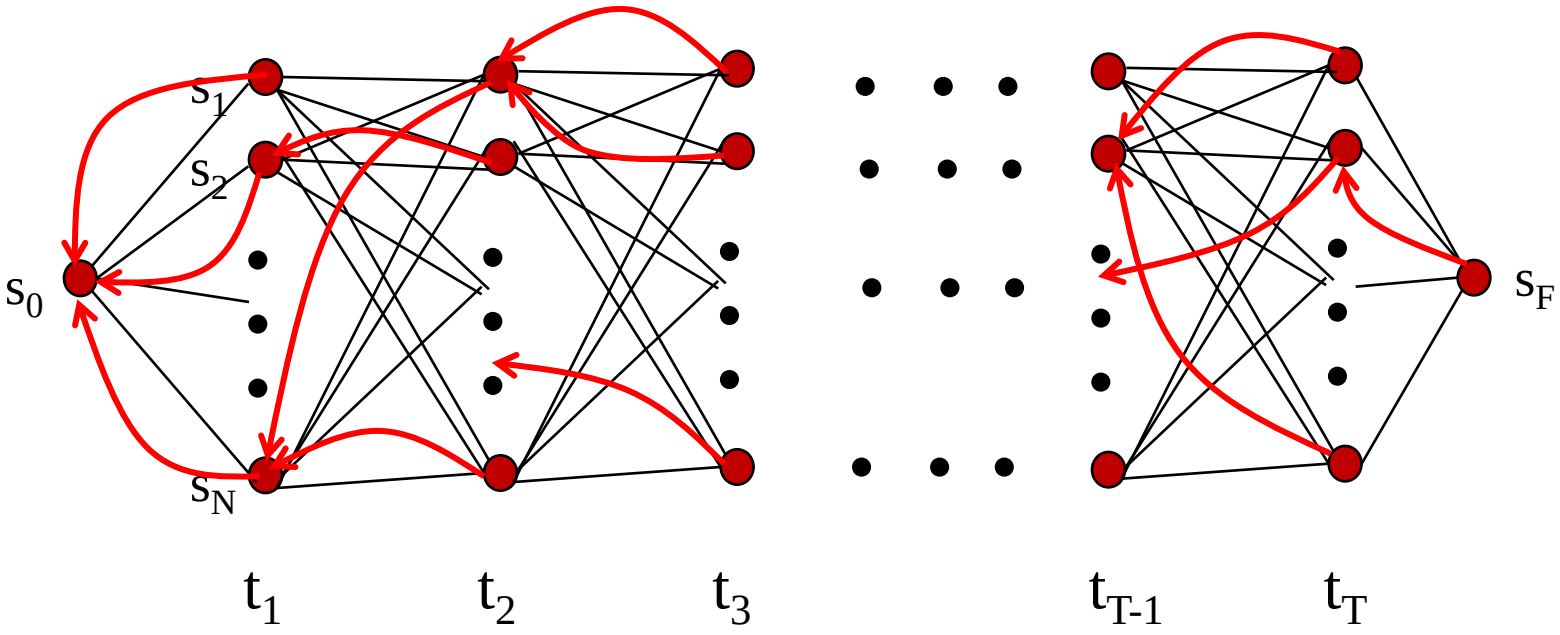
$$bt_t(j) = \underset{i=1}{\overset{N}{\operatorname{argmax}}} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \quad 1 \le t \le T$$

- 算法结束

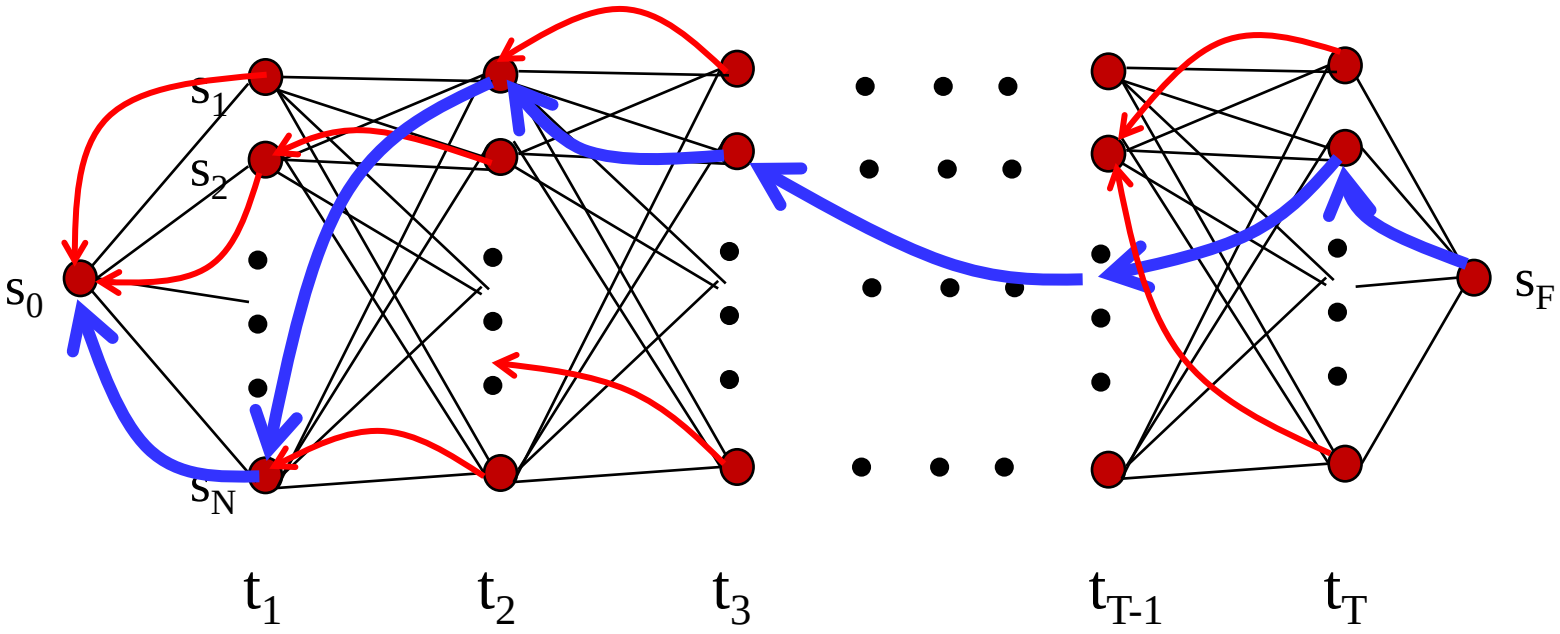
$$q_T^* = bt_{T+1}(s_F) = \underset{i=1}{\overset{N}{\operatorname{argmax}}} v_T(i) a_{iF}$$

算法结束后，可以通过记录的回溯指针从后往前寻找最大概率的状态序列

维特比算法：回溯



维特比算法：回溯



最大概率的状态序列: $s_0\ s_N\ s_1\ s_2\ \ldots s_2\ s_F$

结构化感知器模型

不同点2：计算特征向量 f(y, x)

Input: *example* $\mathbf{x}_i$ *with gold standard label sequence* $\mathbf{y}_i^*$, *weight vector* Θ, *and feature vector* $\mathbf{f}(\mathbf{y}, \mathbf{x})$
Initialization: *set parameters* $\Theta^1 = 0$
for $i = 1 \ldots d$ **do**
  $\mathbf{y}_i = \text{argmax}_{\mathbf{y}} F(\mathbf{y}|\mathbf{x}_i, \Theta^i)$
  if $\mathbf{y}_i \neq \mathbf{y}_i^*$ **then**
    $\Theta^{i+1} = \Theta^i + \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}_i, \mathbf{x}_i)$
  else
    $\Theta^{i+1} = \Theta^i$
Output: *parameter vectors* Θ^{i+1} *for* $i = 1 \ldots d$

- 结构化感知器的特征向量又称为全局特征向量，因为x和y都是一个向量了
- 全局特征向量是每个点上的特征向量的累加

$$\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}) = \sum_{k=1}^{T} \boldsymbol{f}(y_{(k)}, x_{(k)})$$

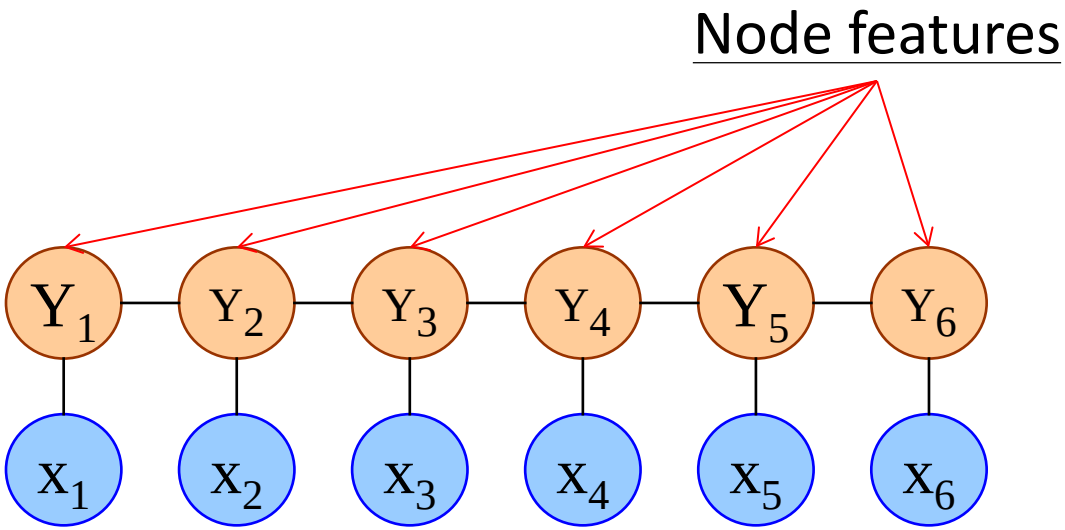
结构化感知器模型

- 结构化感知器对过拟合的控制方法

Input: *example* $\mathbf{x}_i$ *with gold standard label sequence* $\mathbf{y}_i^*$, *weight vector* Θ, *and feature vector* $\mathbf{f}(\mathbf{y}, \mathbf{x})$
Initialization: *set parameters* $\Theta^1 = 0$
for $i = 1 \ldots d$ **do**
  $\mathbf{y}_i = \text{argmax}_{\mathbf{y}} F(\mathbf{y}|\mathbf{x}_i, \Theta^i)$
  if $\mathbf{y}_i \neq \mathbf{y}_i^*$ **then**
    $\Theta^{i+1} = \Theta^i + \mathbf{f}(\mathbf{y}_i^*, \mathbf{x}_i) - \mathbf{f}(\mathbf{y}_i, \mathbf{x}_i)$
  else
    $\Theta^{i+1} = \Theta^i$
Output: *parameter vectors* Θ^{i+1} *for* $i = 1 \ldots d$

- 投票方法(voted perceptron)
  - 对于所有的 i 对应的参数向量，进行投票
- 参数求平均方法(averaged perceptron)
  - 对于所有的 i 对应的参数向量，计算平均值

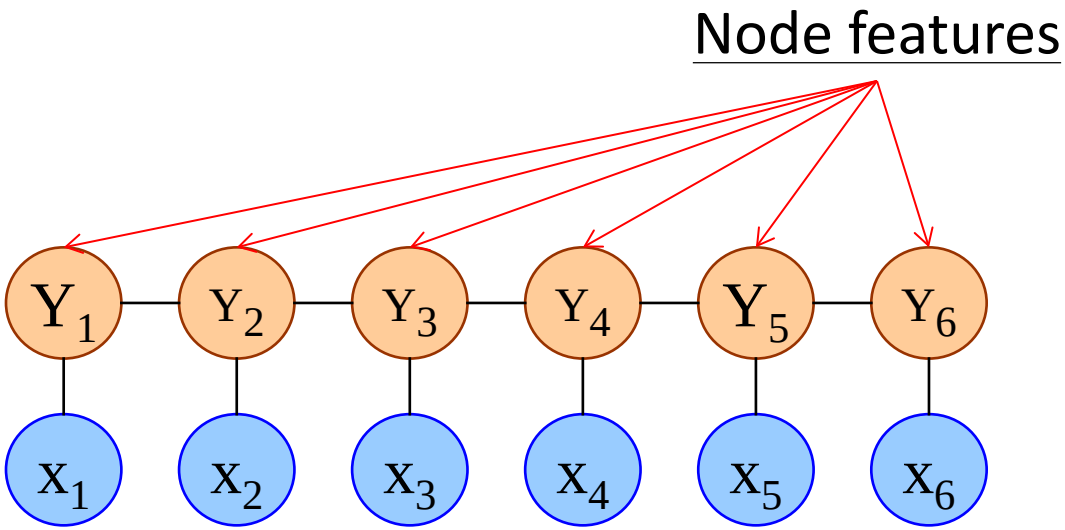
怎么提取结构相关特征(Feature)



节点特征

Node feature

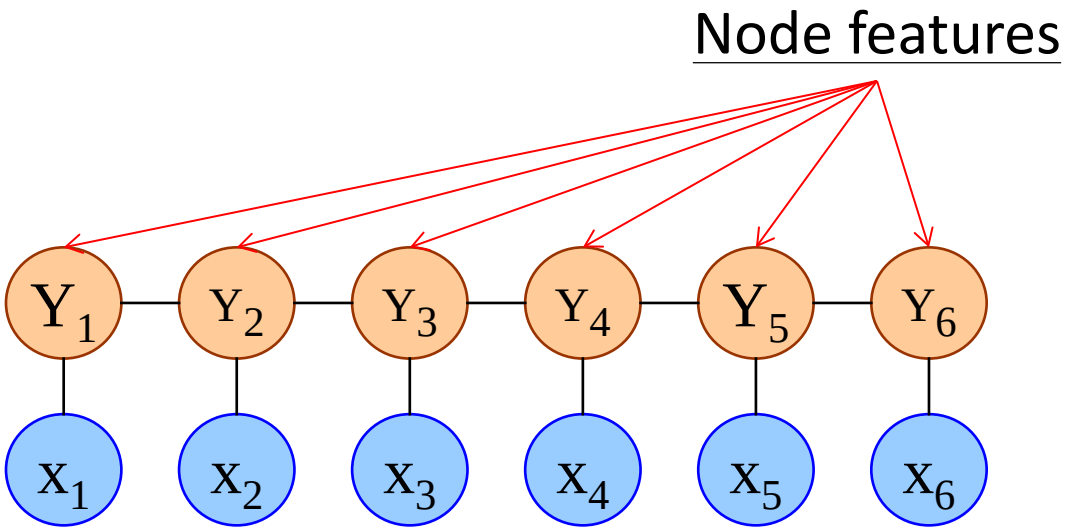
怎么提取结构相关特征(Feature)



节点特征

Node feature

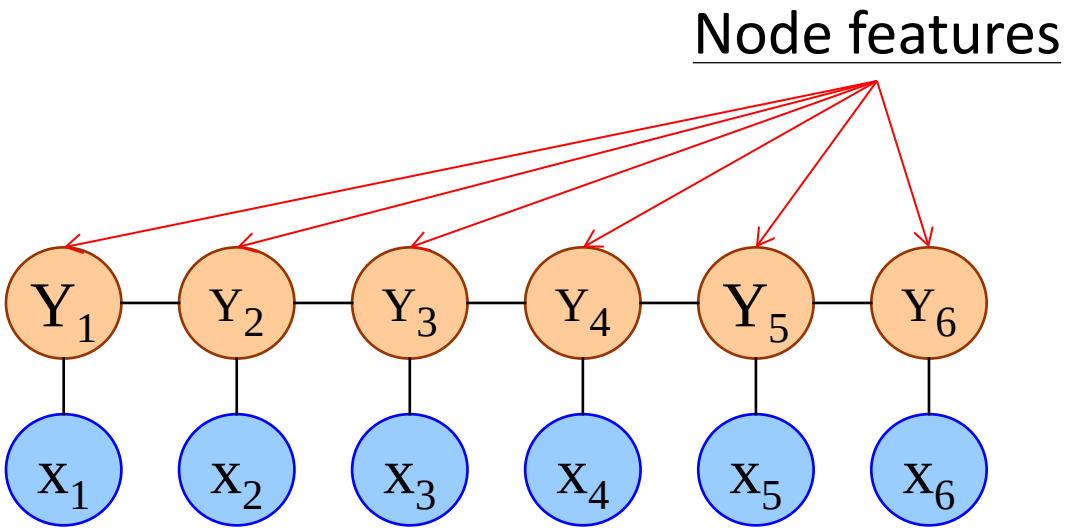
怎么提取结构相关特征(Feature)



节点特征

Node feature

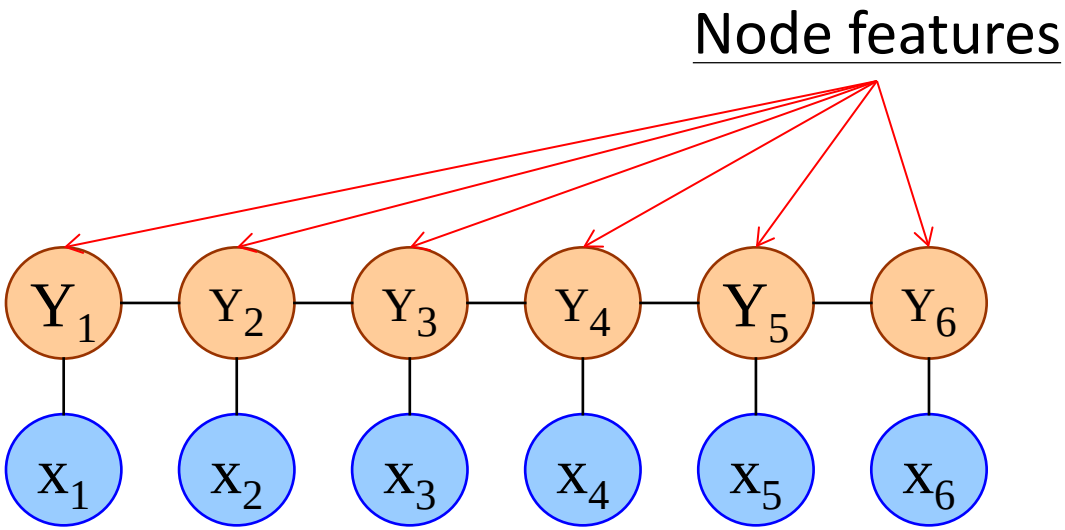
怎么提取结构相关特征(Feature)



节点特征

Node feature

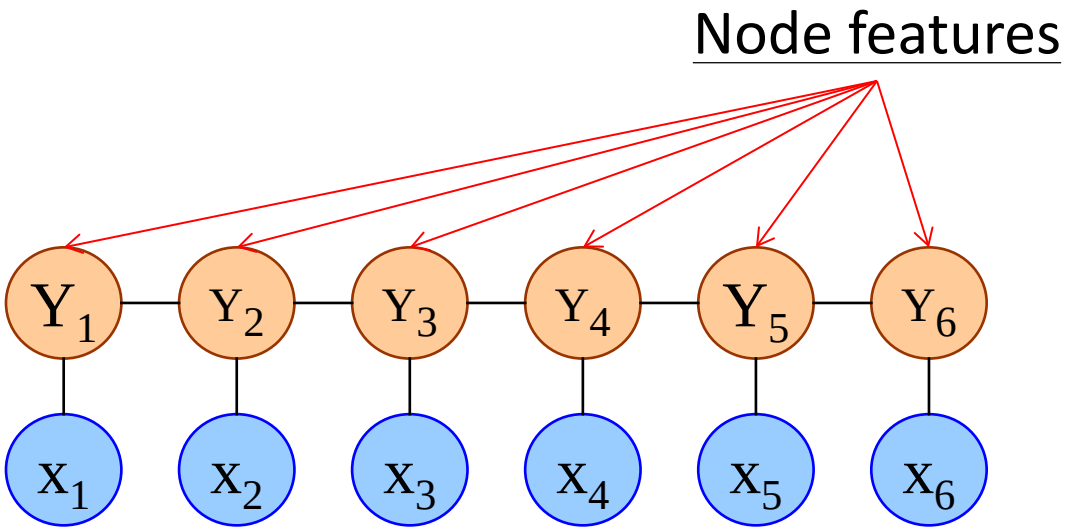
怎么提取结构相关特征(Feature)



节点特征

Node feature

怎么提取结构相关特征(Feature)

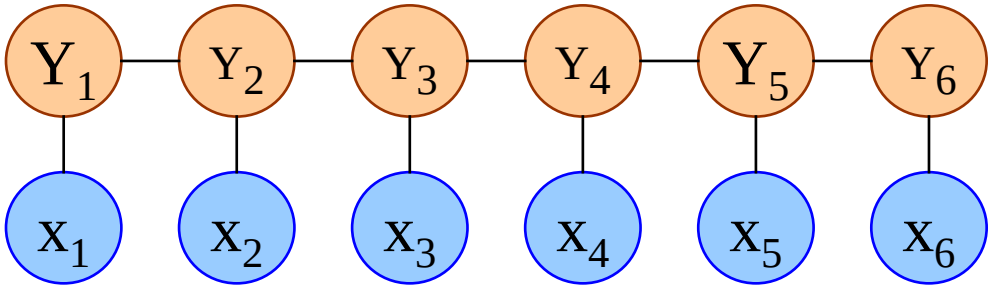


节点特征

Node feature

怎么提取结构相关特征(Feature)

Edge feature

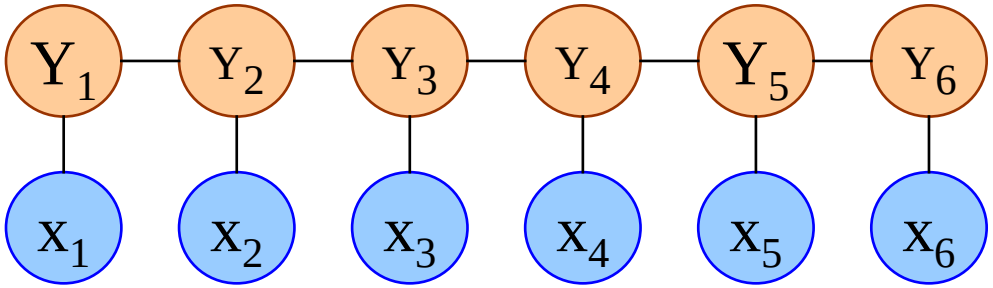


边特征

Edge feature

怎么提取结构相关特征(Feature)

Edge feature

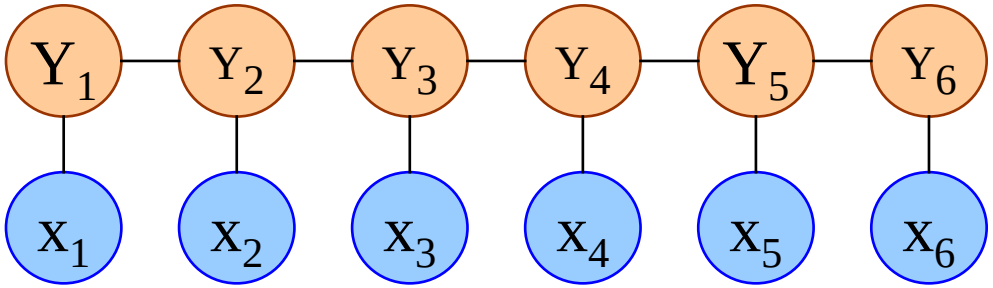


边特征

Edge feature

怎么提取结构相关特征(Feature)

Edge feature

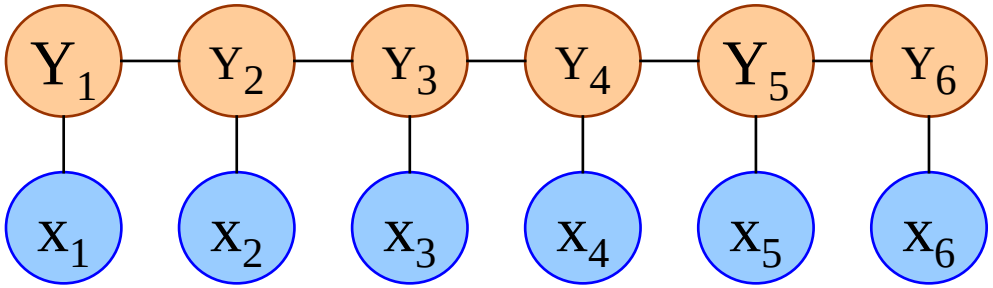


边特征

Edge feature

怎么提取结构相关特征(Feature)

Edge feature

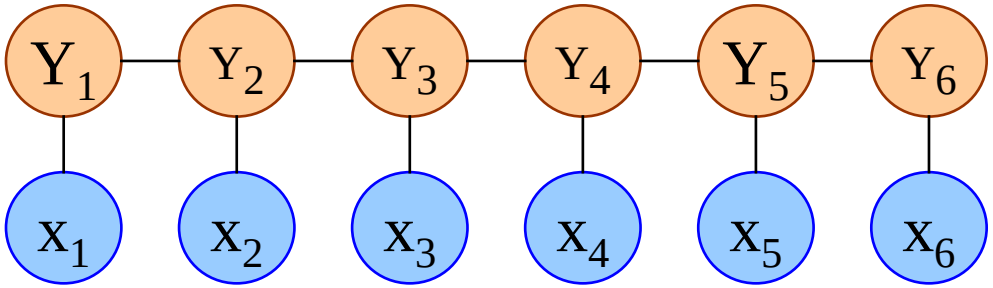


边特征

Edge feature

怎么提取结构相关特征(Feature)

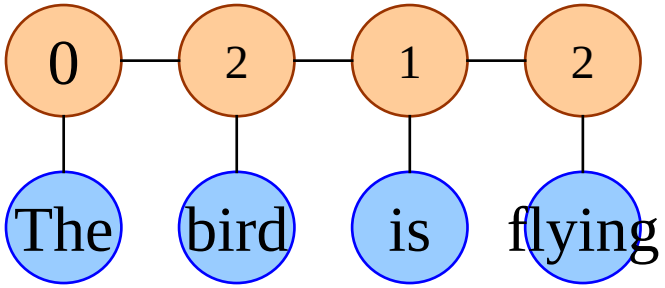
Edge feature



边特征

Edge feature

特征提取举例



假设我们采用如下特征模板：

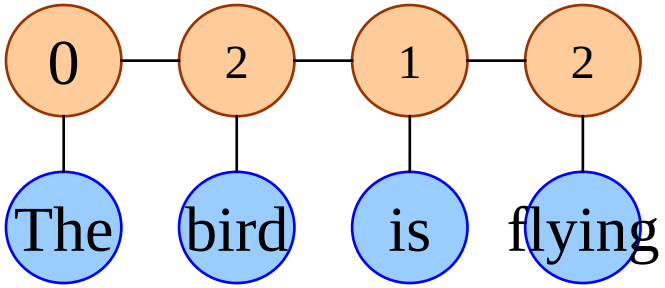
1，节点特征：

x(i)y(i), x(i-1)x(i)y(i)

2，边特征：

y(i-1)y(i)

特征提取举例



该标签序列的总特征**f(y, x)**如下（基于字符串）：

1，节点特征:

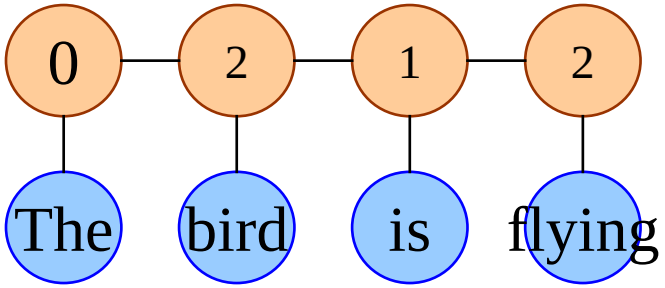
the_0, bird_2, is_1, flying_2

*_the_0, the_bird_2, bird_is_1, is_flying_2

2，边特征:

*_0, 0_2, 2_1, 1_2

特征提取举例



该标签序列的总特征**f(y, x)**如下（基于数字）：

1，节点特征:

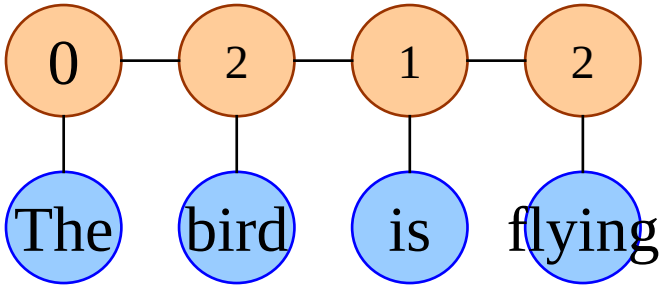
2, 4, 5, 7

10, 11, 15, 16

2，边特征:

19, 20, 23, 24

特征提取举例



$$\boldsymbol{f}(\boldsymbol{y},\boldsymbol{x}) = \sum_{k=1}^{T} \boldsymbol{f}(y_{(k)}, x_{(k)})$$

该标签序列的总特征**f(y, x)**如下（基于向量）：

<0,1,0,1,1,0,1,0,0,1,1,0,0,0,1,1,0,0,1,1,0,1,1>

假设目前的结构化感知器的权重向量**theta**如下：

<0,0,0,2,0,1,3,0,2,5,1,0,0,1,0,0,0,0,0,2,0,2,4>

则对应的分数**F(y|x,theta)**如下：

2+3+5+1+2+2+4=19

结构化感知器模型

- **流程：**
  - 1，建立特征模板，提取特征
  - 2，把特征映射为整数，该整数对应其模型参数的下标
  - 3，基于训练数据，用结构化感知器的算法训练模型参数
  - 4，基于训练得到的模型参数，在测试数据上测试效果

结构化感知器模型

- 实验对比

- 2组序列标注问题的实验
  - 词性标注任务POS tagging
    - Using the Adwait’ s features
  - 短语切分NP chunking
    - Using BIO tags (Start, Continue, Outside tags)

NP-chunking任务的特征选择

Current word	w_i	& t_i
Previous word	w_{i-1}	& t_i
Word two back	w_{i-2}	& t_i
Next word	w_{i+1}	& t_i
Word two ahead	w_{i+2}	& t_i
Bigram features	w_{i-2}, w_{i-1}	& t_i
	w_{i-1}, w_i	& t_i
	w_i, w_{i+1}	& t_i
	w_{i+1}, w_{i+2}	& t_i
Current tag	p_i	& t_i
Previous tag	p_{i-1}	& t_i
Tag two back	p_{i-2}	& t_i
Next tag	p_{i+1}	& t_i
Tag two ahead	p_{i+2}	& t_i
Bigram tag features	p_{i-2}, p_{i-1}	& t_i
	p_{i-1}, p_i	& t_i
	p_i, p_{i+1}	& t_i
	p_{i+1}, p_{i+2}	& t_i
Trigram tag features	p_{i-2}, p_{i-1}, p_i	& t_i
	p_{i-1}, p_i, p_{i+1}	& t_i
	p_i, p_{i+1}, p_{i+2}	& t_i

结构化感知器模型

实验结果

NP Chunking Results

Method	F-Measure	Numits
Perc, avg, cc=0	93.53	13
Perc, noavg, cc=0	93.04	35
Perc, avg, cc=5	93.33	9
Perc, noavg, cc=5	91.88	39
ME, cc=0	92.34	900
ME, cc=5	92.65	200

POS Tagging Results

Method	Error rate/%	Numits
Perc, avg, cc=0	2.93	10
Perc, noavg, cc=0	3.68	20
Perc, avg, cc=5	3.03	6
Perc, noavg, cc=5	4.04	17
ME, cc=0	3.4	100
ME, cc=5	3.28	200

总结

	模型类别	特征	训练速度	准确度
隐马尔可夫模型	生成模型	固定特征	快速	较低
结构化感知器	判别模型	任意特征	快速	较高

进一步阅读/学习

- **参考书**
  - 《统计自然语言处理》第6章：
    - 概率图模型
    - Page 104 - 127
  - 《统计自然语言处理》第7章：
    - 自动分词、命名实体识别与词性标注
    - Page 129 - 177