序列标注问题

孙栩
信息科学技术学院
xusun@pku.edu.cn

大纲

- **序列标注问题**
- **传统方法**
  - 简单分类
  - 标签偏置问题
  - HMM、结构化感知器
- **深度学习方法**
  - 词向量
  - 全连接神经网络
  - 循环神经网络

大纲

- **序列标注问题** 
- **传统方法**
  - 简单分类
  - 标签偏置问题
  - HMM、结构化感知器
- **深度学习方法**
  - 词向量
  - 全连接神经网络
  - 循环神经网络

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

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

例1：词性标注(Part-of-Speech Tagging)

- **给定一个句子(词序列)，对每个词标注出对应的词性类别**
  - 即对每个词给出一个标签，即对每个词模式分类！在词性标注里，每个标签为一个词性(part-of-speech, POS)
  - 在句法分析、信息提取等任务上有重要作用

- **英文词性标注举例:**

例2：中文切词(word segmentation)

- **通过计算机把组成汉语文本的字串自动转换为词串的过程被称为中文切词**
  - 即给定一个中文句子（字序列），尽可能将之切分成正确的词序列
  - 是大部分中文信息处理任务的基础、第一步

- **例子**

结果：企业 /要 /真正 /具有 /用工 /的 /自主权

例3：短语切分（phrase chunking）

- **给定一个自然语言的句子，对句子中的短语进行切分、并识别短语的种类**
  - 又称为浅层句法分析(shallow parsing)
  - 对句法分析、机器翻译等任务有重要作用

- **英文短语切分举例:**

例4：命名实体识别(named entity recognition)

- **给定一个句子或篇章，定位和识别相关的命名实体(named entity)**
  - 命名实体包括：人名、地名、机构名
  - 或特定领域相关的命名实体，例如生物领域命名实体识别包括：蛋白质Protein、DNA、RNA等
  - 在信息提取、知识抽取等任务有重要作用

- **举例**

O O O B-Protein I O B-DNA I I I

We showed that interleukin-1 IL-1 and IL-2 receptor alpha gene …

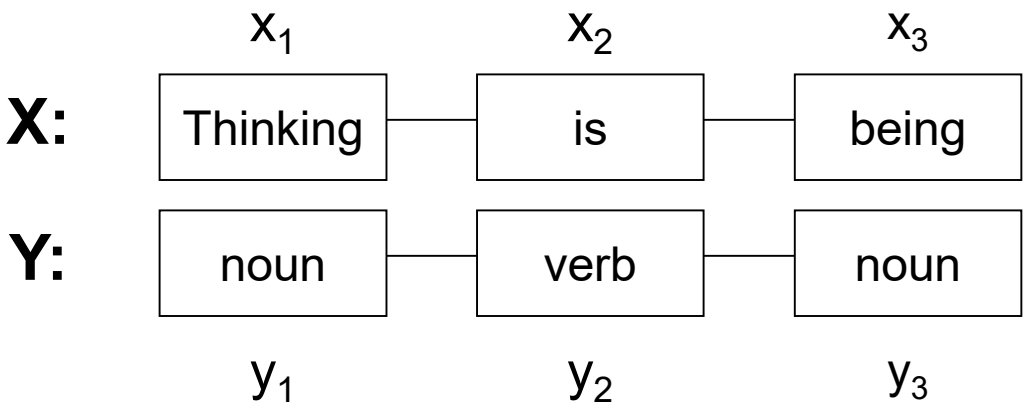
We showed that interleukin-1 IL-1 and IL-2 receptor alpha gene …

Protein DNA

序列标注问题的总结

- 对一个序列中的每一个观测量(token)进行分类，给出一个对应的标签(label)

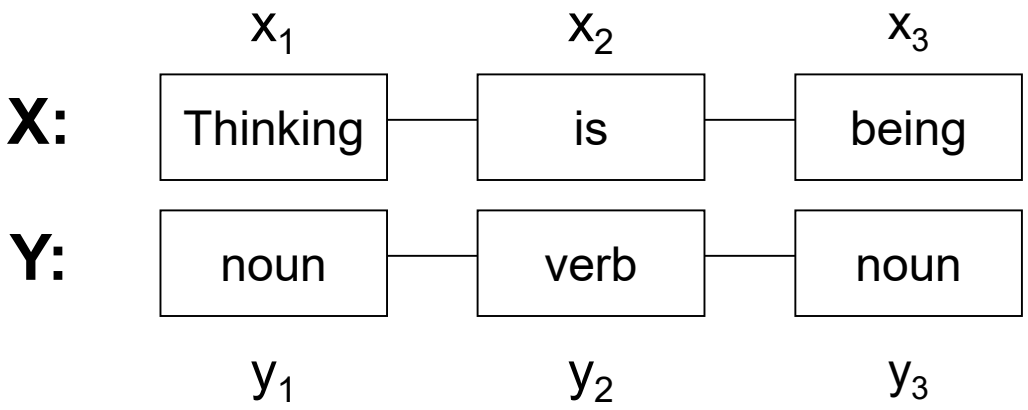
- 观测量对应的标签之间存在**结构依赖关系**，会互相影响，而不是独立分布的。这种结构依赖关系表现为较强的局部结构依赖，但是局部结构依赖会传导为全局依赖



序列标注问题的总结

- X 是一个观测向量，代表观测量序列
- Y 是一个标签向量，代表标签序列
- Y_i 是标签向量的第i个元素，值域是一个给定的有限集合：标签集合A

- **序列标注问题：**
  - 给定一个有限标签集合A，学习怎么从观测向量X映射到标签向量Y



大纲

- **序列标注问题**
- **传统方法** 
  - 简单分类
  - 标签偏置问题
  - HMM
- **深度学习方法**
  - 词向量
  - 全连接神经网络
  - 循环神经网络

大纲

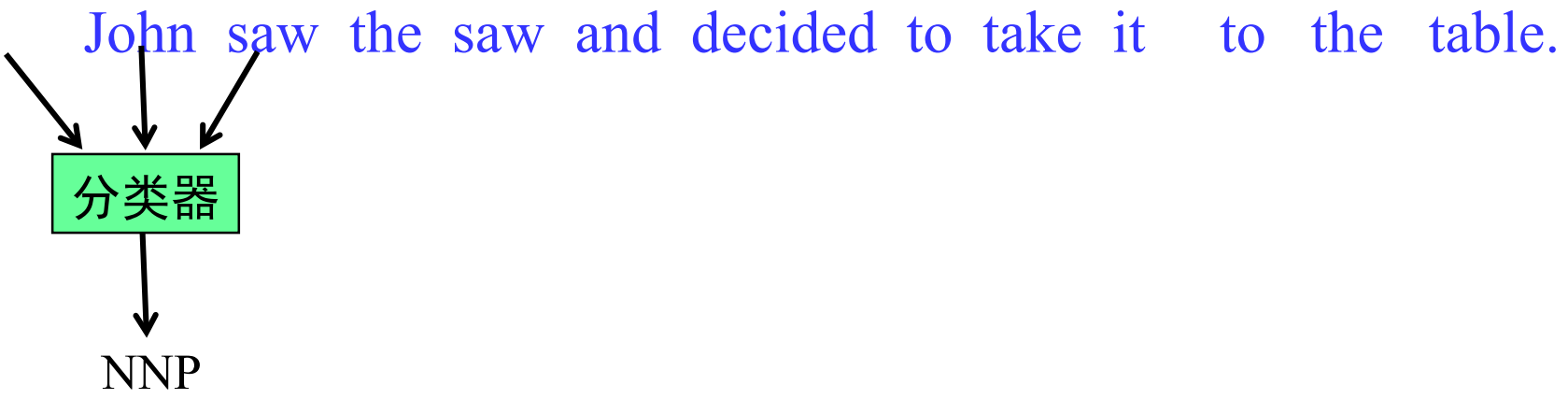
- **链状结构即通常所说的“序列标注问题”**
- **自然语言处理的序列标注问题举例**
  - 词性标注
  - 中文切词
  - 短语识别
  - 命名实体识别
- **代表性的序列标注方法**
  - 关键问题是什么?
  - 隐马尔可夫模型 HMM
  - 结构化感知器 structured perceptron

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

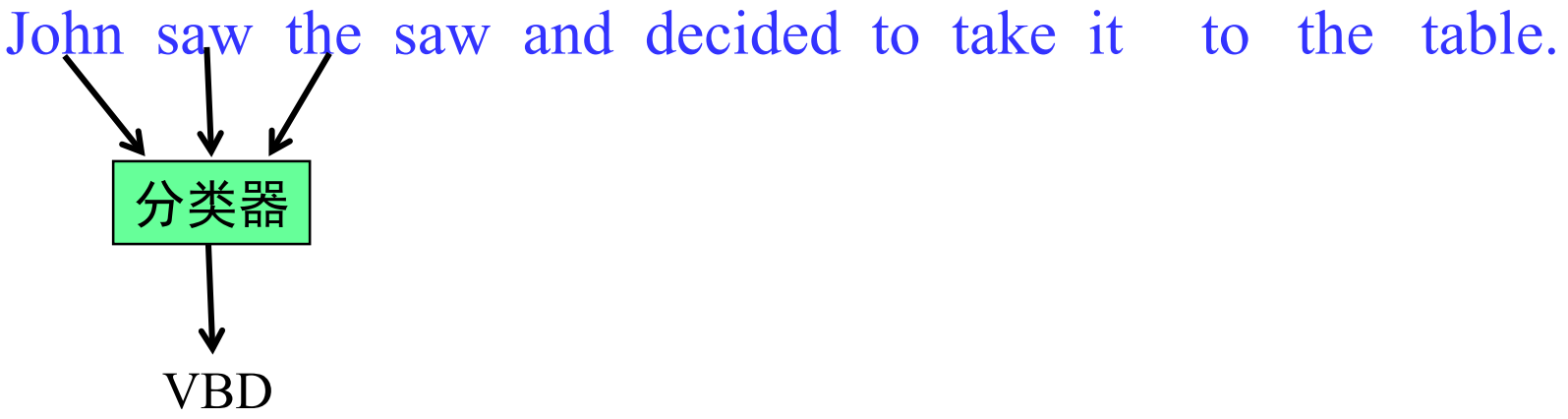
直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）



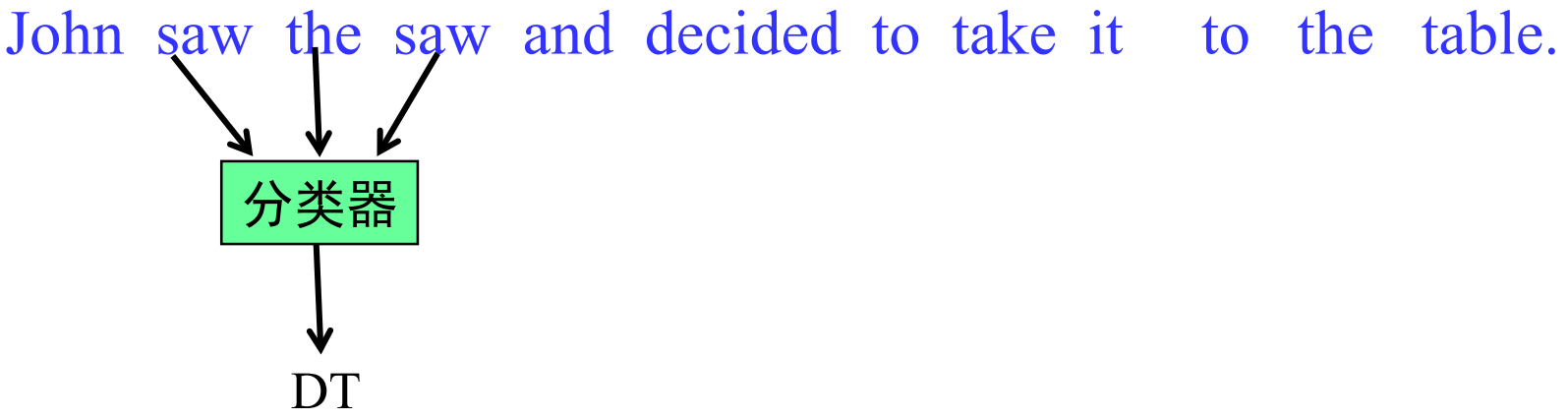
直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）



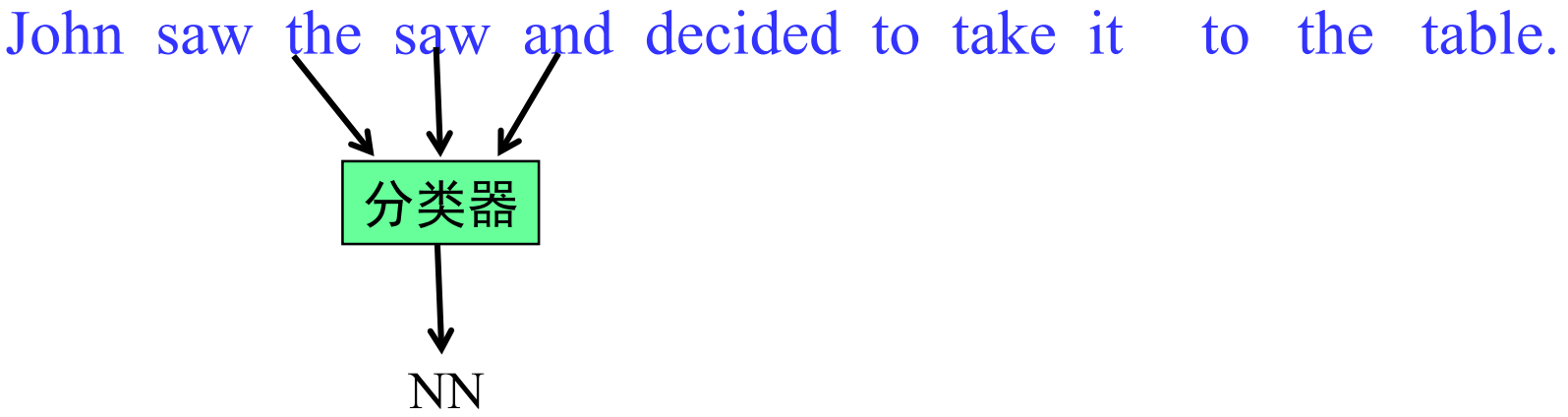
直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）



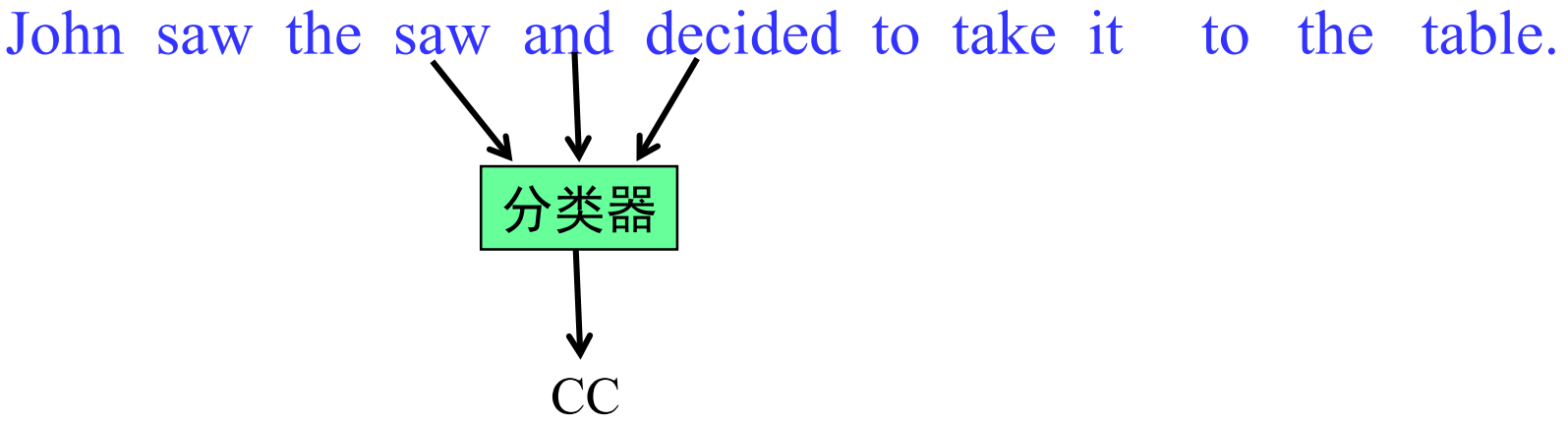
直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）



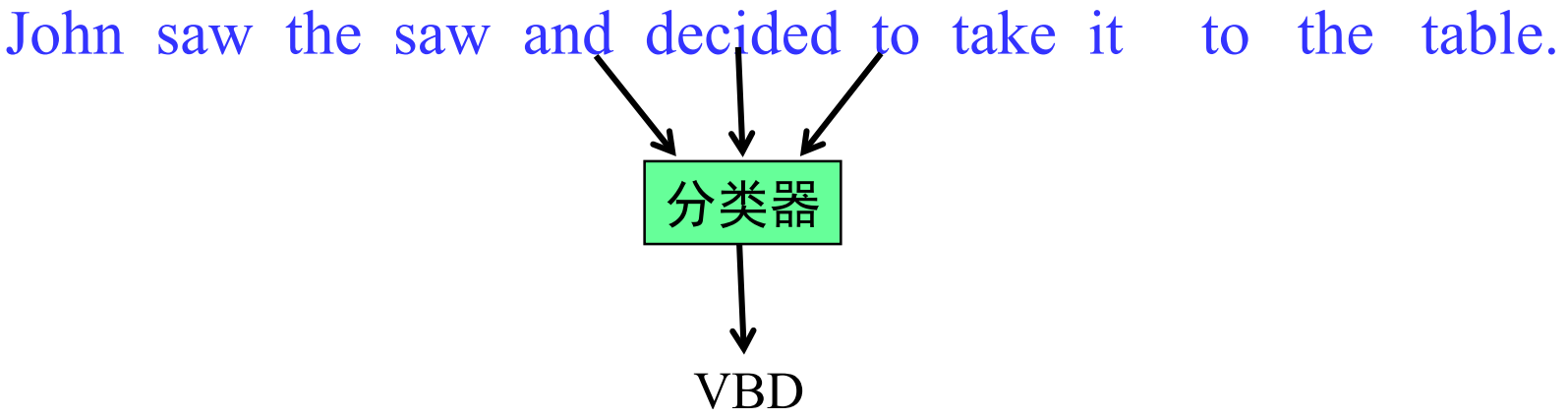
直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）



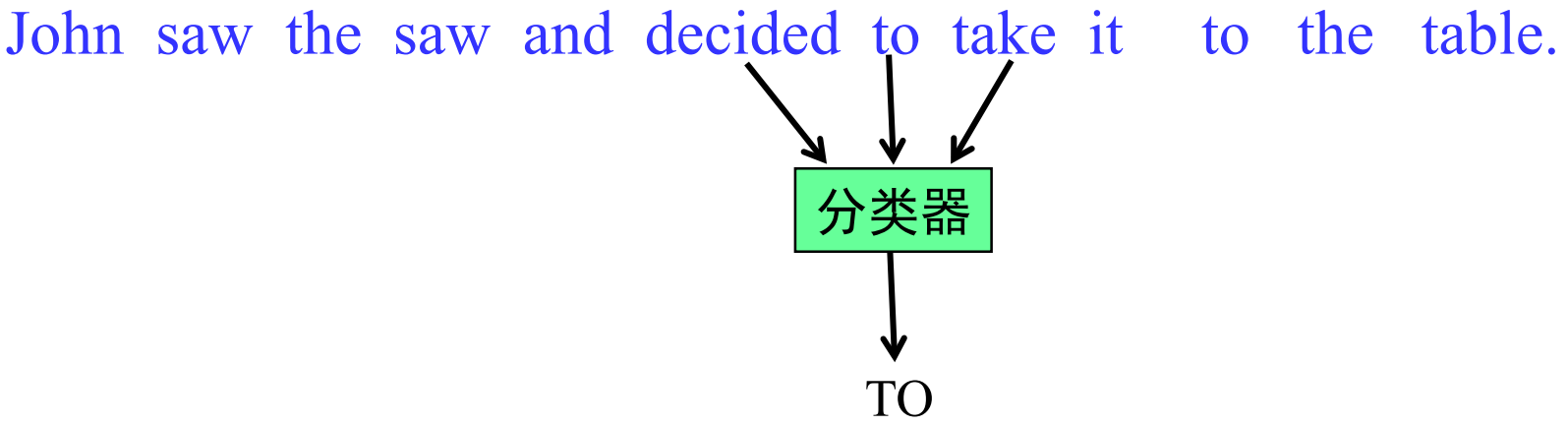
直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）



直接用简单分类方法会怎样?

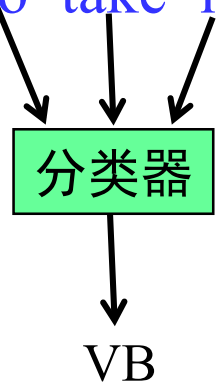
- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）



直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）

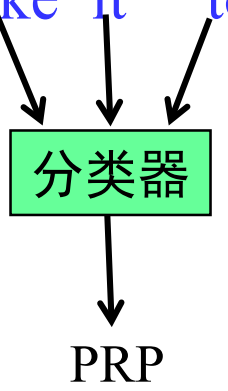
John saw the saw and decided to take it to the table.



直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）

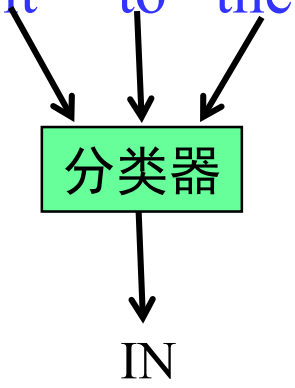
John saw the saw and decided to take it to the table.



直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）

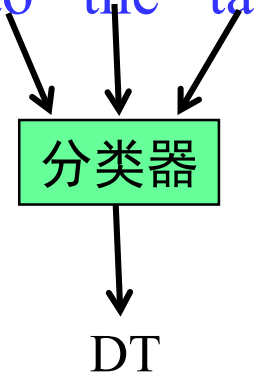
John saw the saw and decided to take it to the table.



直接用简单分类方法会怎样?

- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）

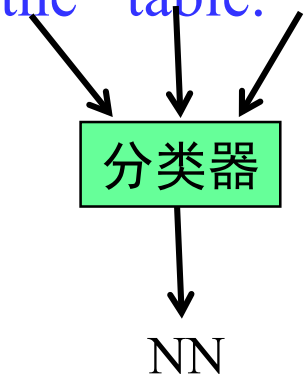
John saw the saw and decided to take it to the table.



直接用简单分类方法会怎样?

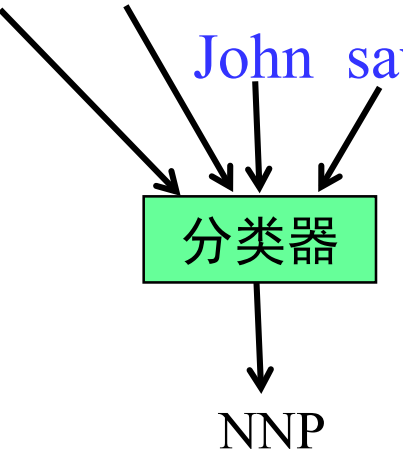
- **基于滑动窗口(sliding window)的简单分类方法**
  - 对每个观测量（词）进行独立的分类，使用周围的观测量（滑动窗口范围内的词）作为分类器的信息输入（提取的特征）

John saw the saw and decided to take it to the table.



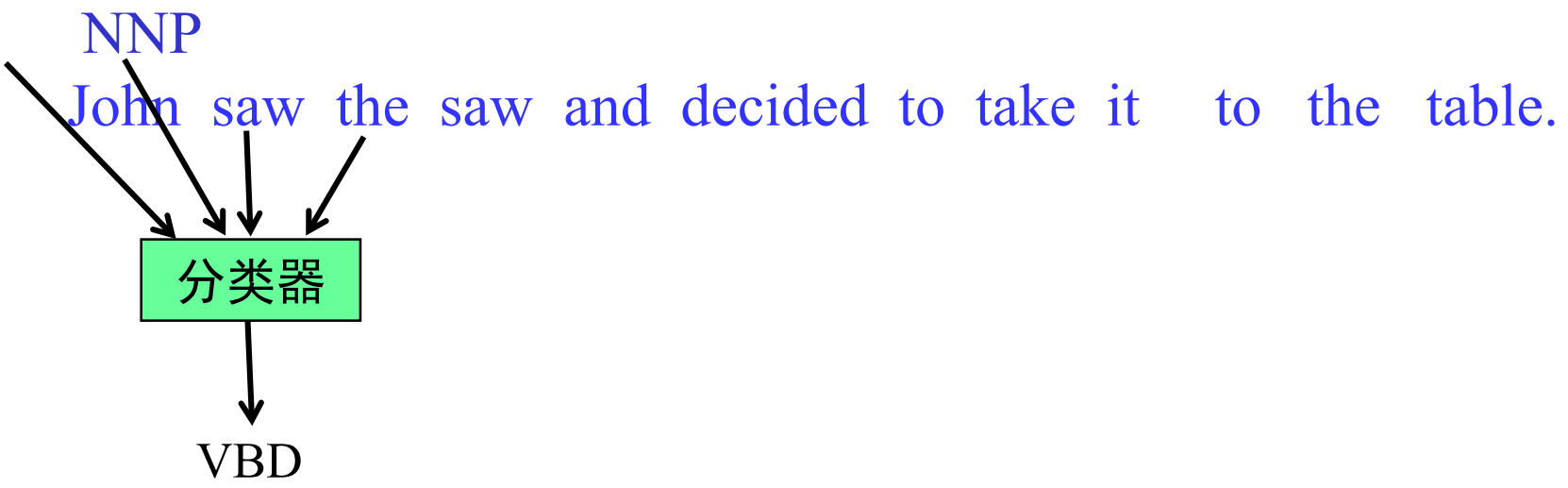
  - 简单分类效果不好，因为无法考虑周围的标签信息（分类信息）

前向分类(Forward Classification)

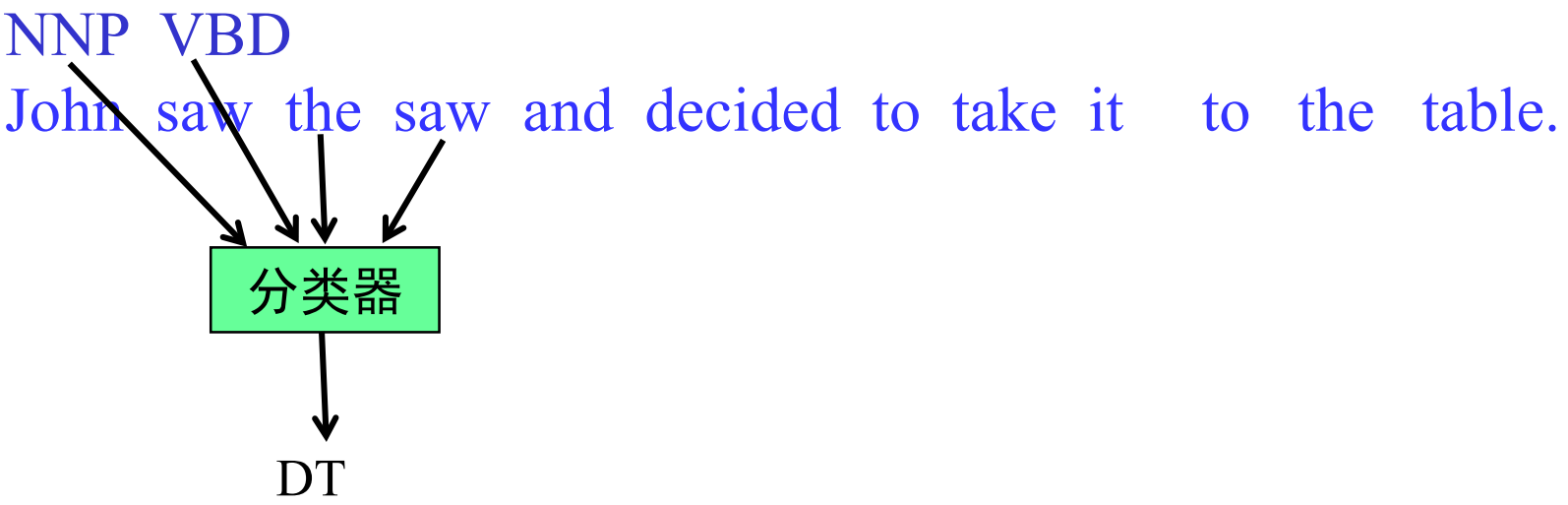


John saw the saw and decided to take it to the table.

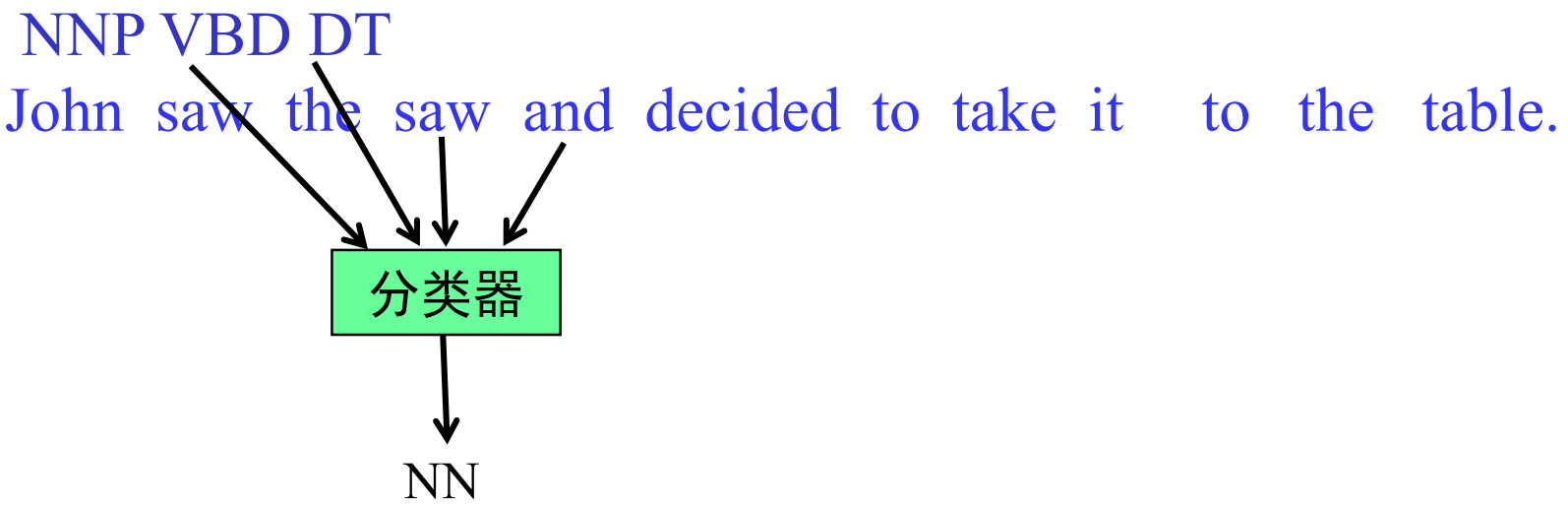
前向分类(Forward Classification)



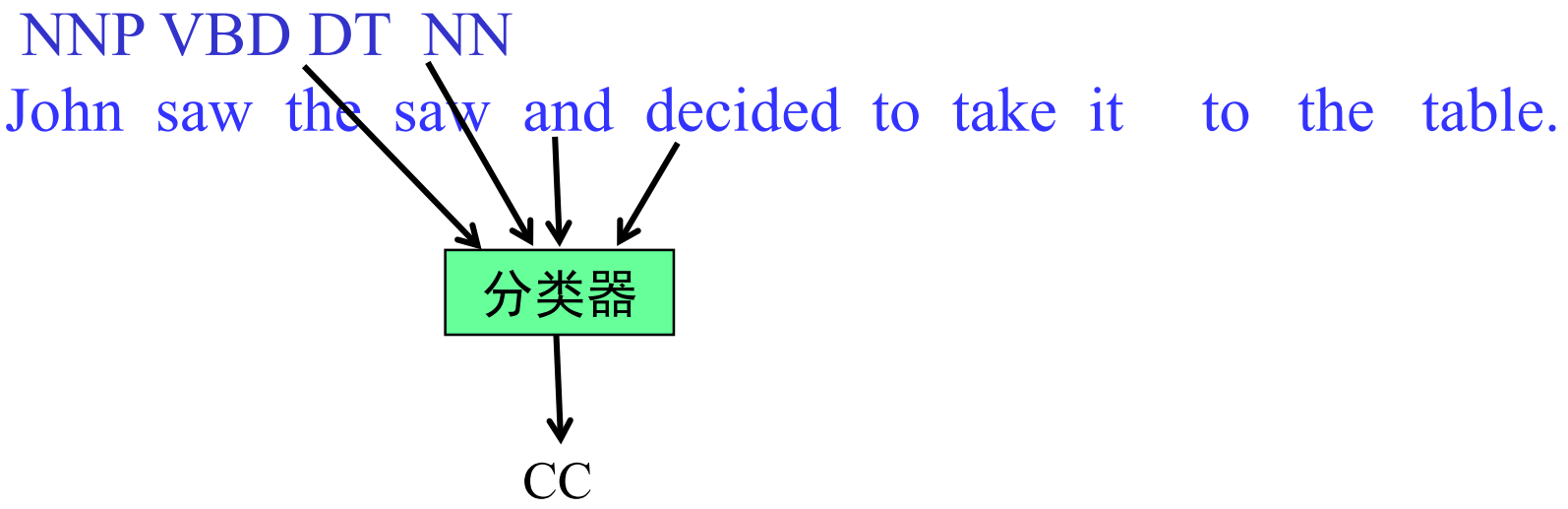
前向分类(Forward Classification)



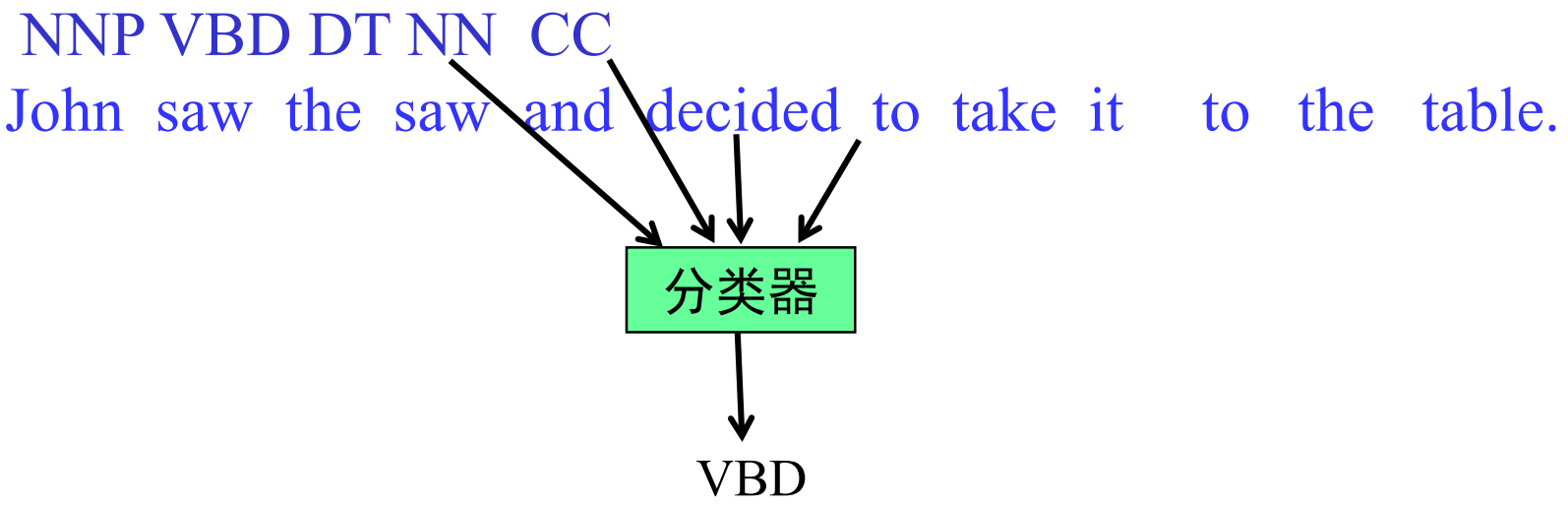
前向分类(Forward Classification)



前向分类(Forward Classification)



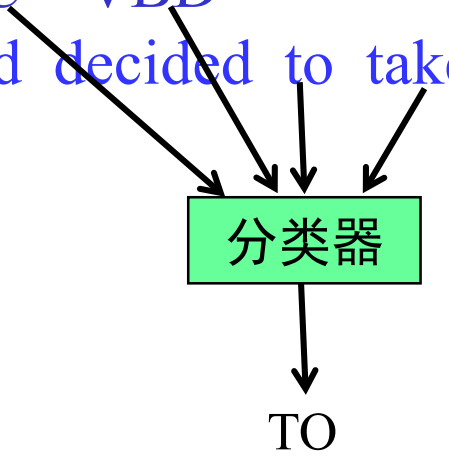
前向分类(Forward Classification)



前向分类(Forward Classification)

NNP VBD DT NN CC VBD

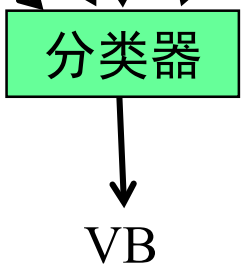
John saw the saw and decided to take it to the table.



前向分类(Forward Classification)

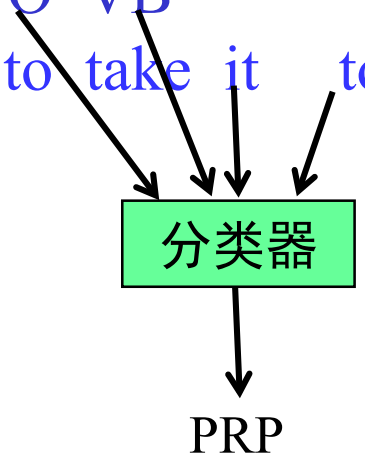
NNP VBD DT NN CC VBD TO

John saw the saw and decided to take it to the table.



前向分类(Forward Classification)

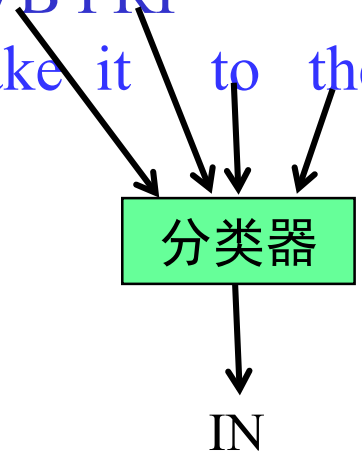
NNP VBD DT NN CC VBD TO VB
John saw the saw and decided to take it to the table.



前向分类(Forward Classification)

NNP VBD DT NN CC VBD TO VB PRP

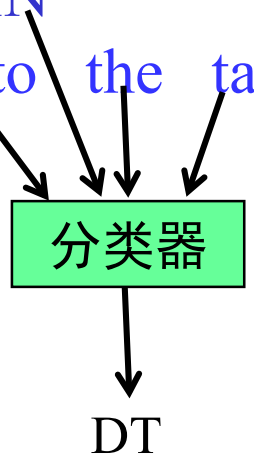
John saw the saw and decided to take it to the table.



前向分类(Forward Classification)

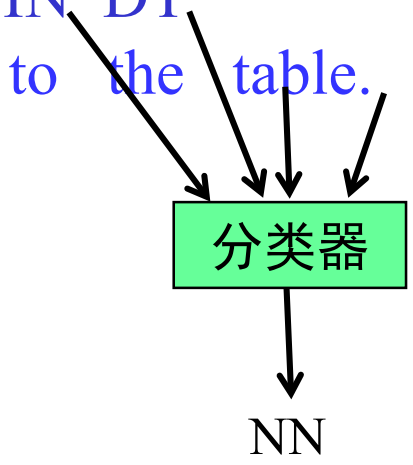
NNP VBD DT NN CC VBD TO VB PRP IN

John saw the saw and decided to take it to the table.



前向分类(Forward Classification)

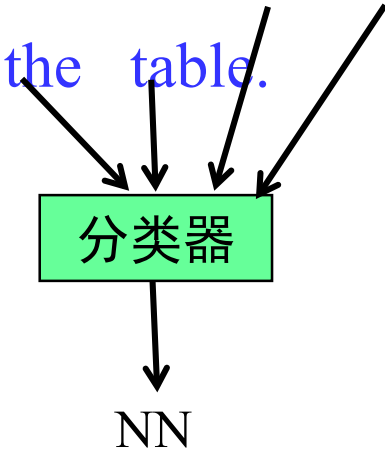
NNP VBD DT NN CC VBD TO VB PRP IN DT
John saw the saw and decided to take it to the table.



后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势

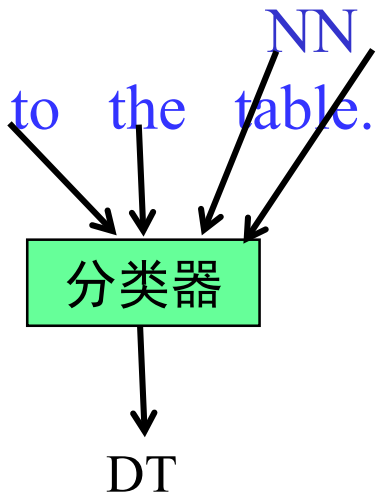
John saw the saw and decided to take it to the table.



后向分类(Backward Classification)

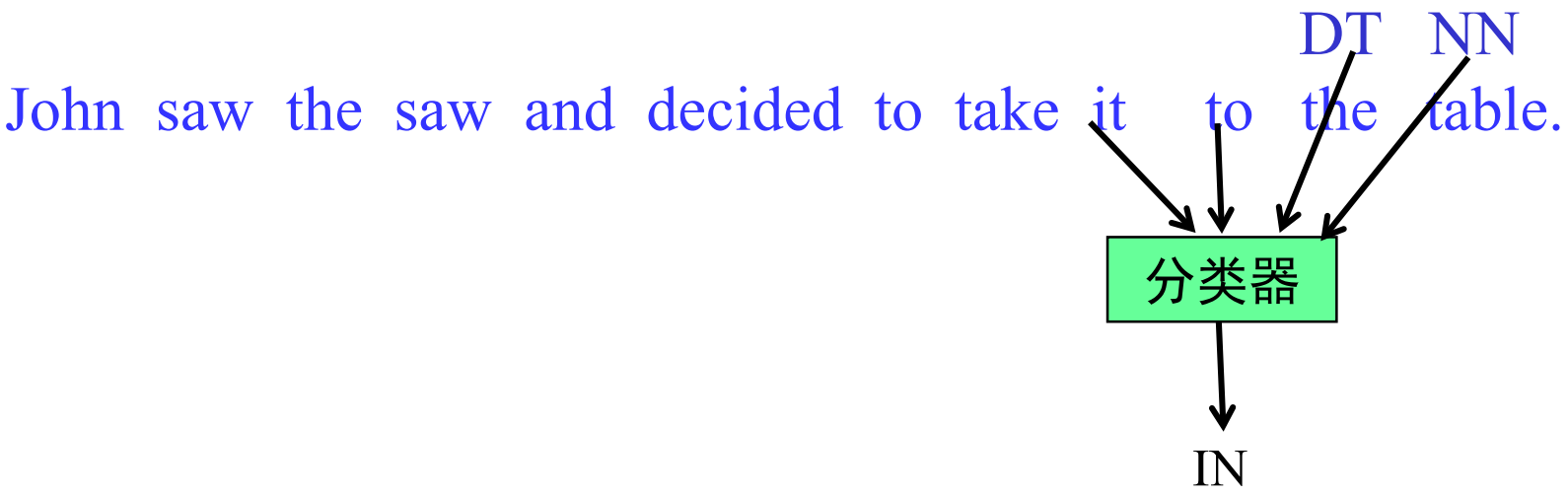
- 对“to”进行分类的时候，后向算法比前向算法有优势

John saw the saw and decided to take it to the table.



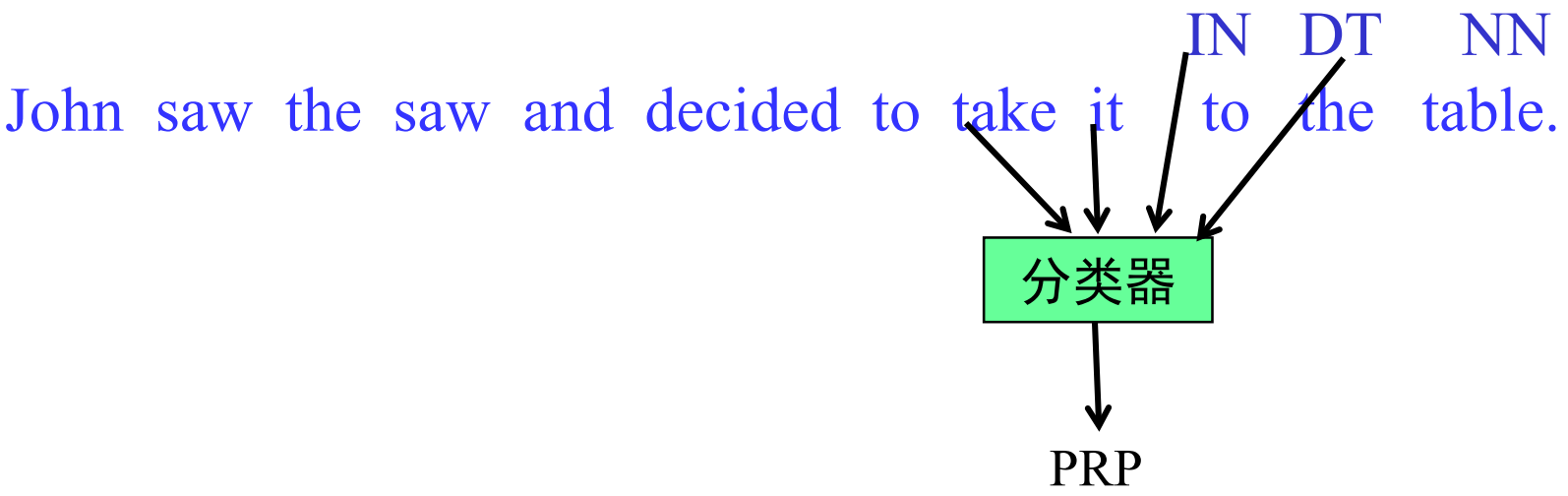
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



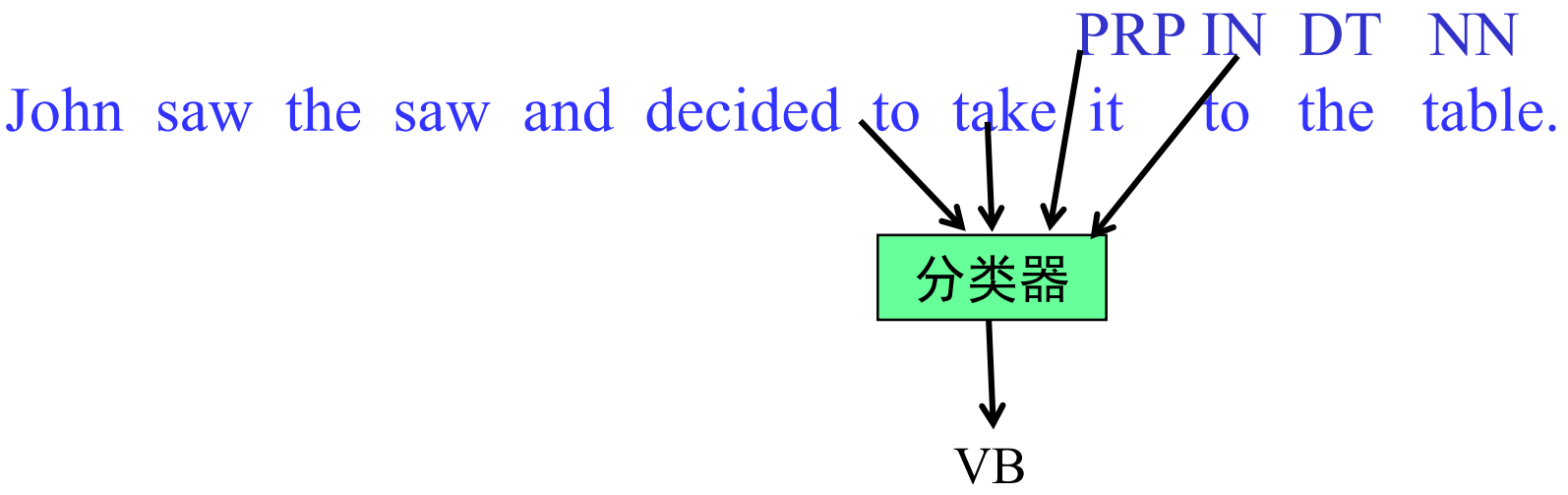
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



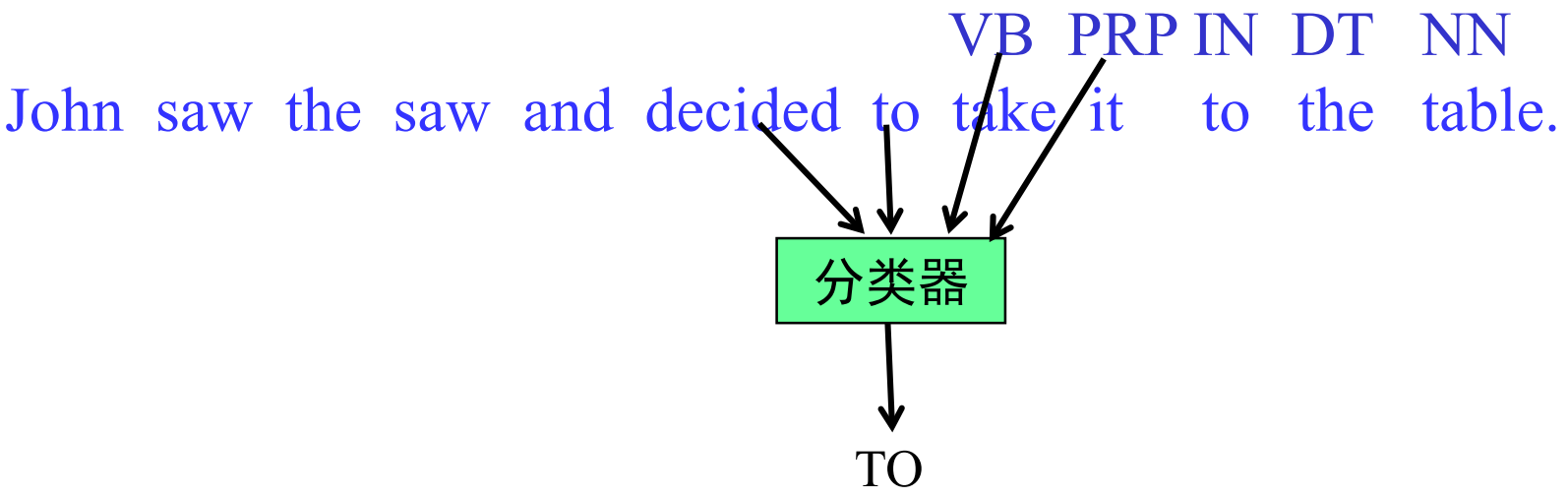
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



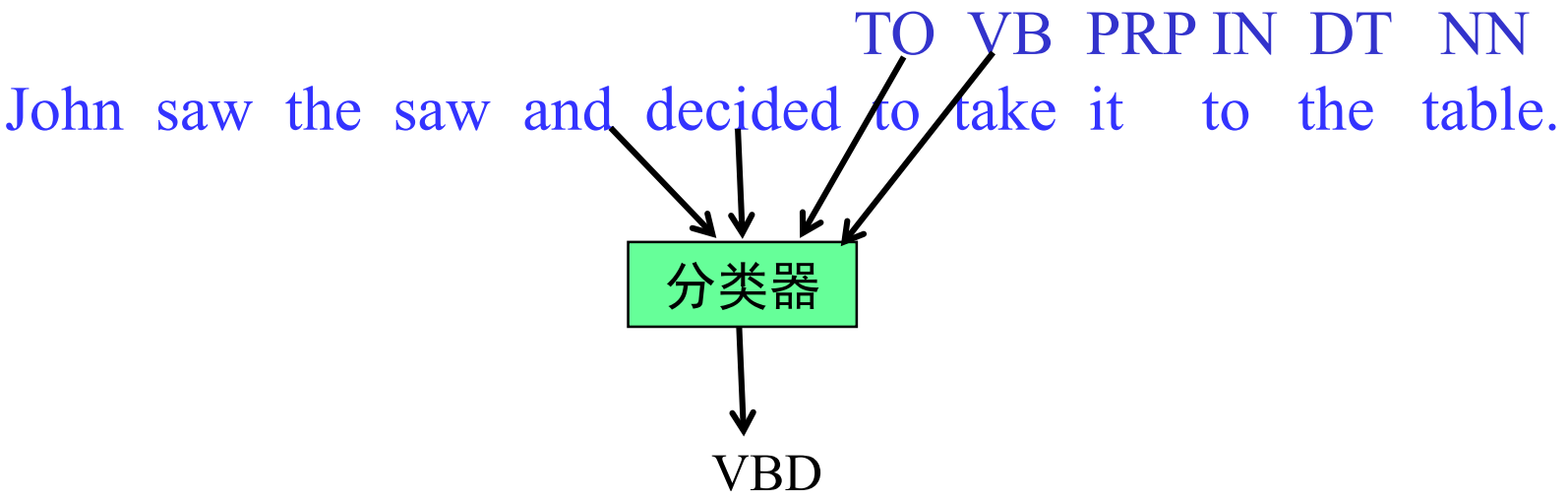
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



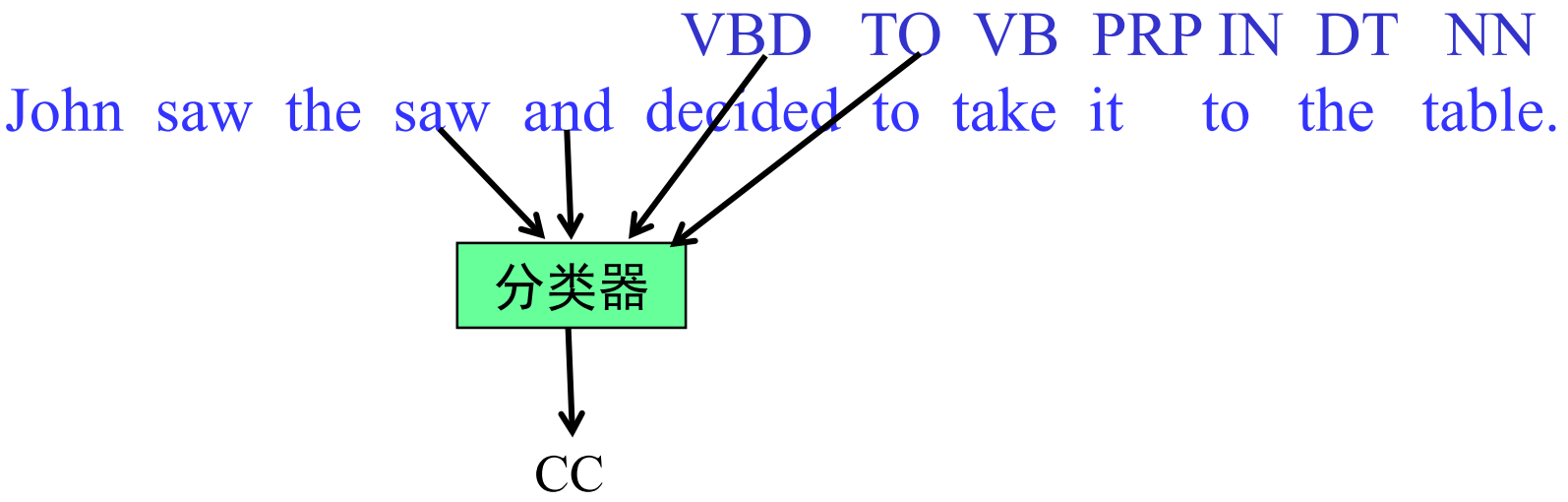
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



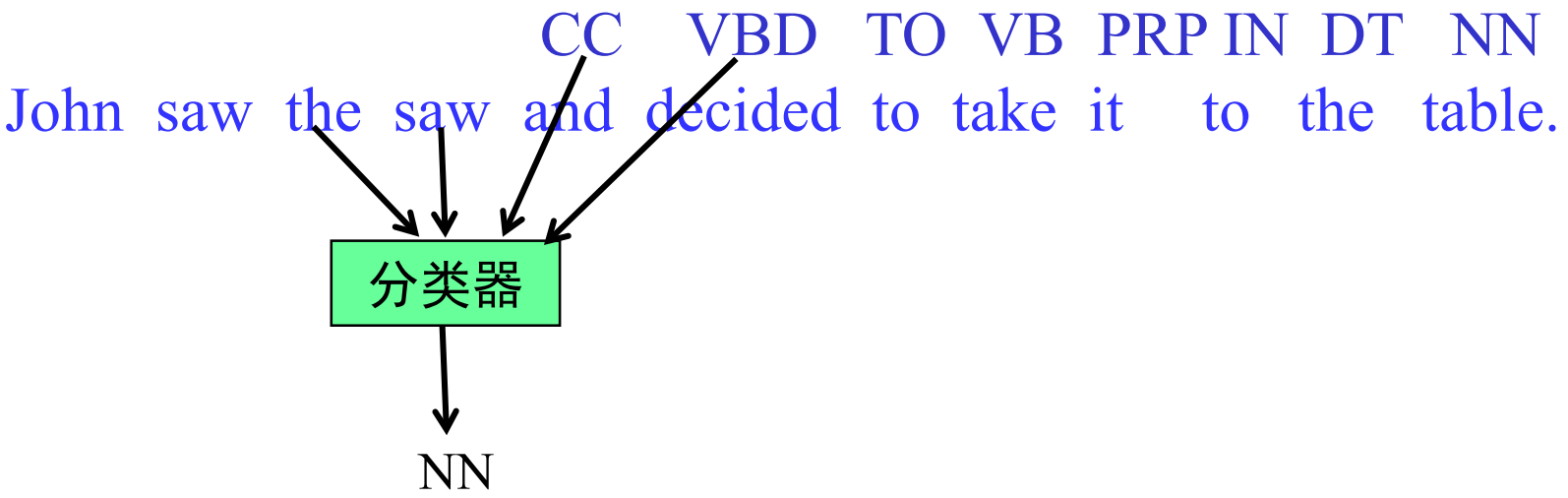
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



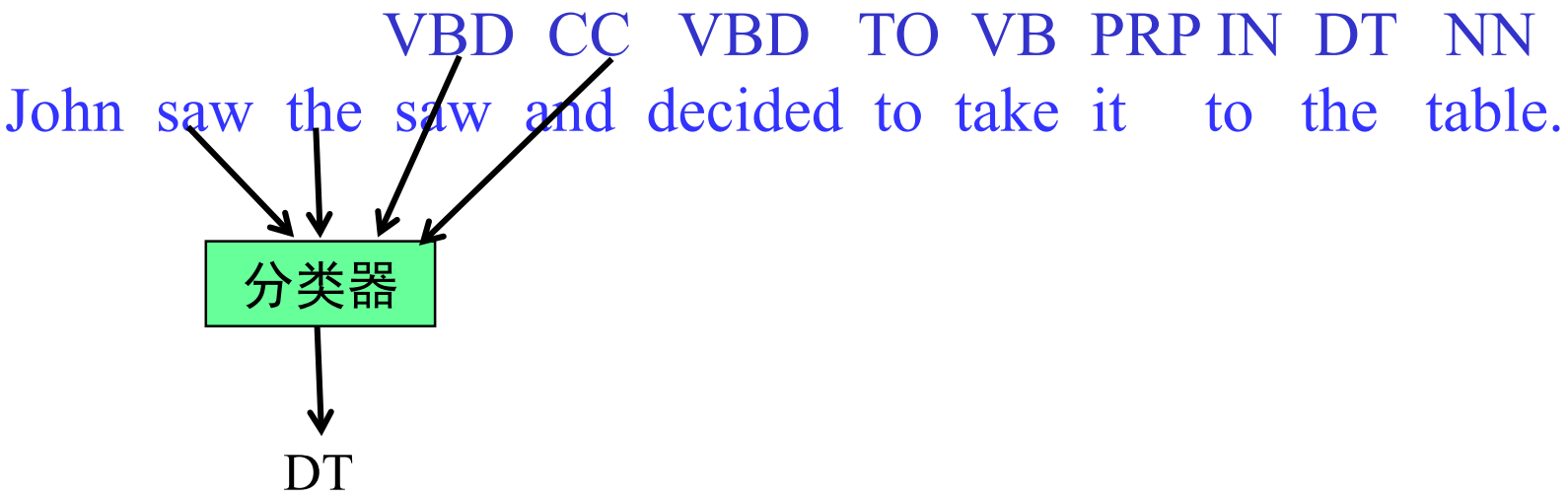
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



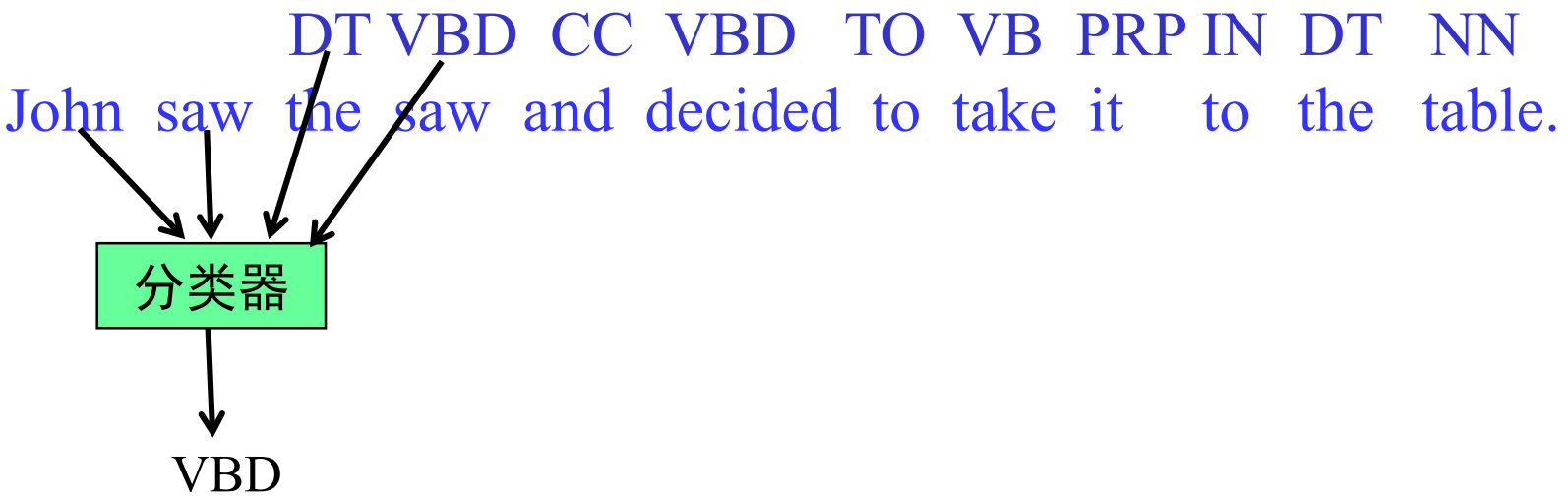
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



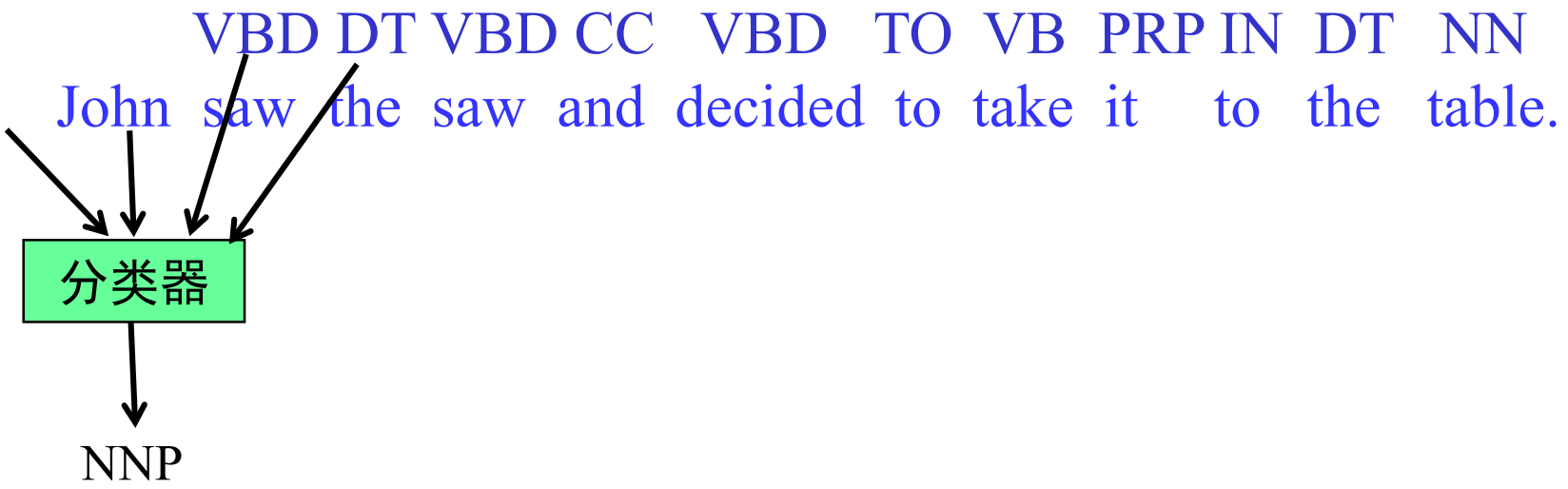
后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势



后向分类(Backward Classification)

- 对“to”进行分类的时候，后向算法比前向算法有优势

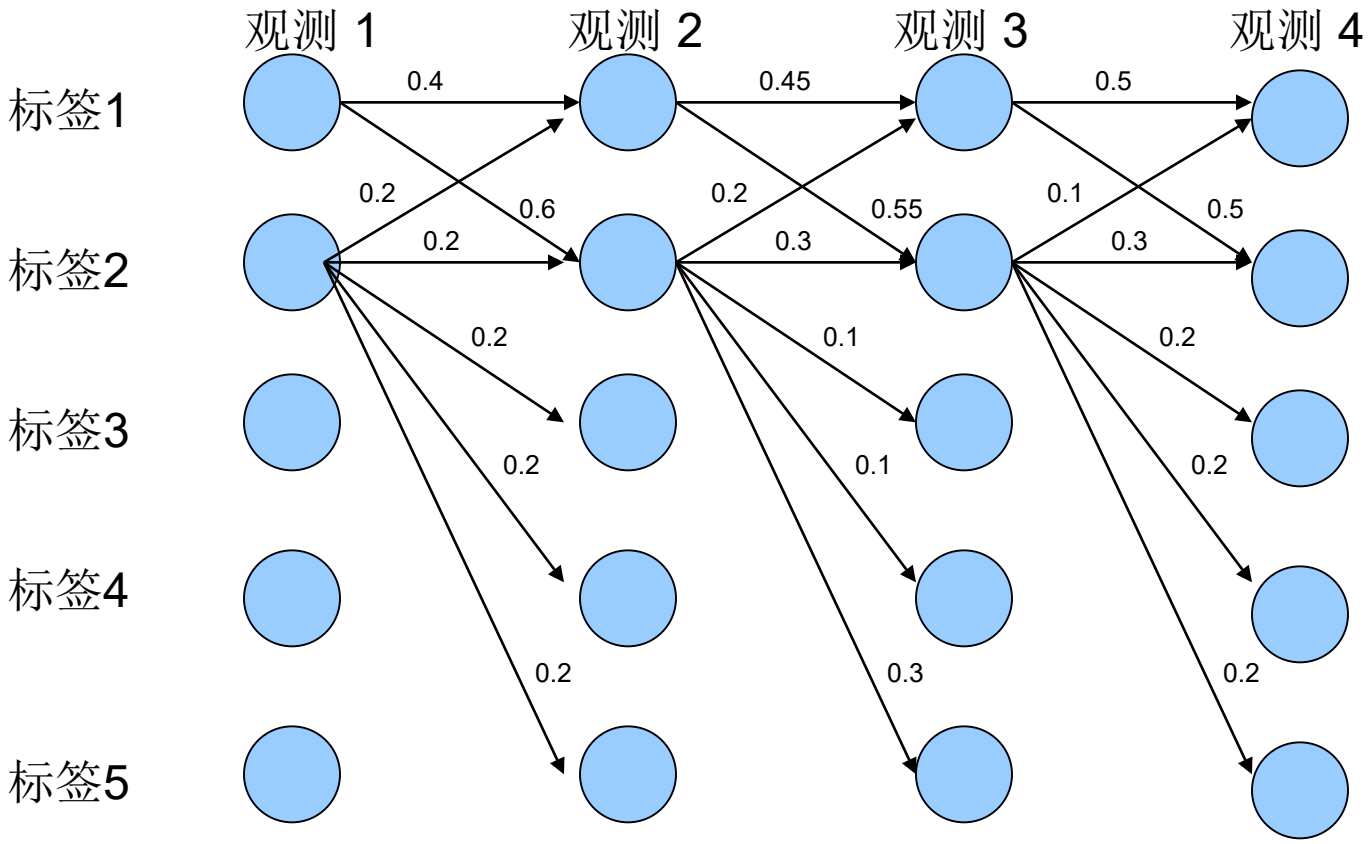


用改进的简单分类方法会怎样?

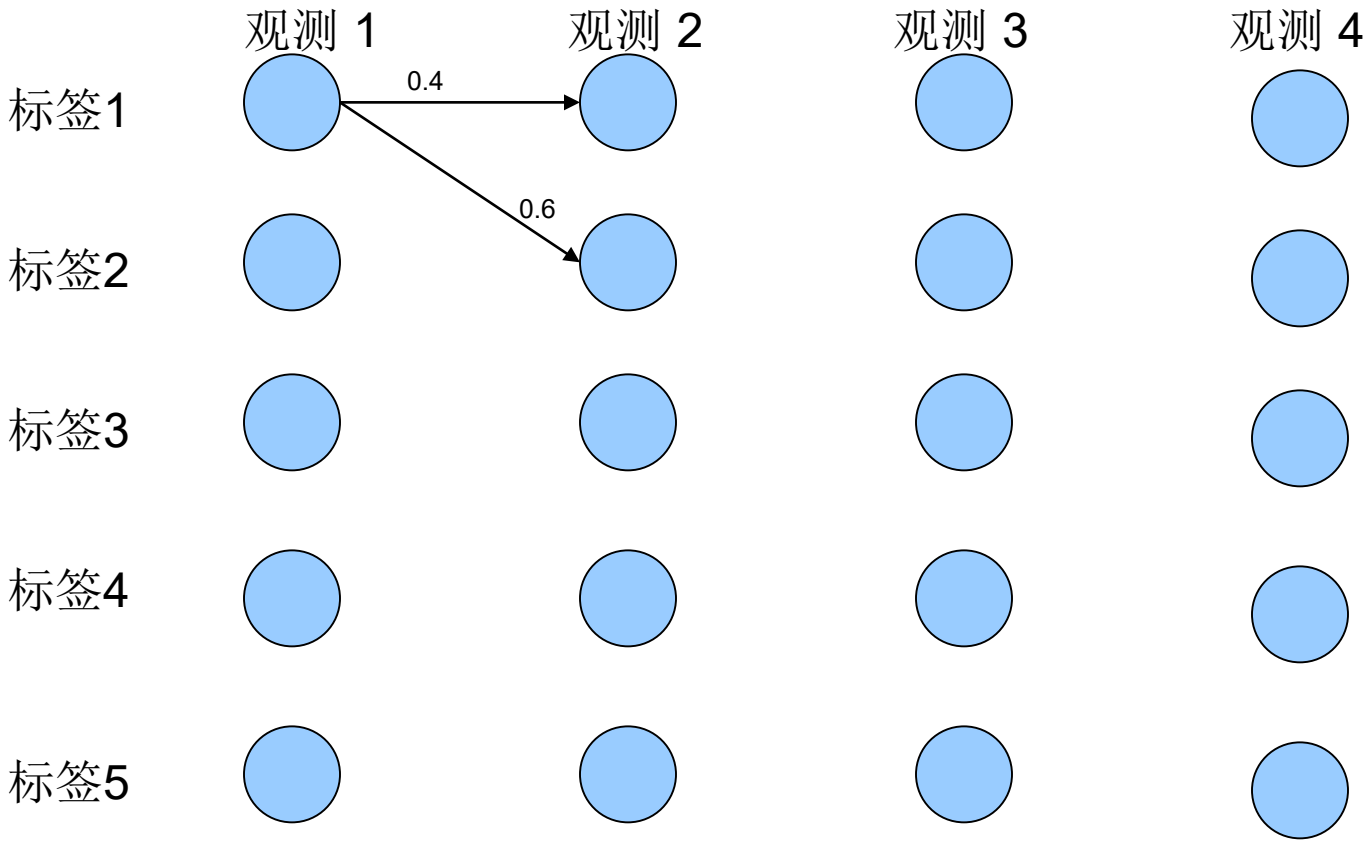
改进方法的问题

- 难以同时兼顾前向、后向的标签信息
- 每个决策仍然是局部最优的，难以统筹兼顾得到全局最优的决策
  - 全局最优的决策是指“同时”决定整个序列的标签
  - 局部最优的决策看似不错，但其实会有标注偏置问题 (label bias problem)

标注偏置问题(label bias problem)



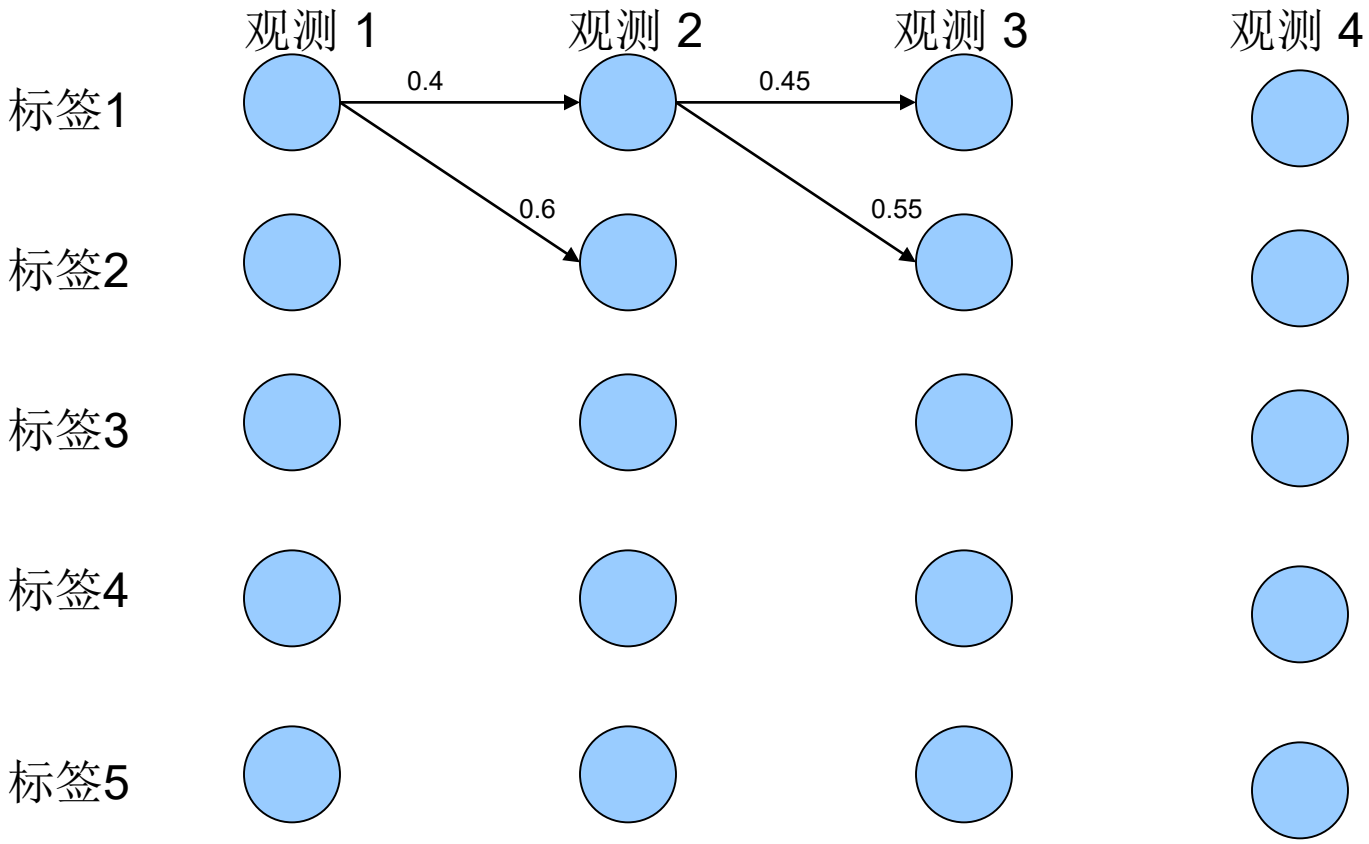
标注偏置问题(label bias problem)



从局部的概率转移情况可以看到:

- 标签1总是倾向于转移到标签2

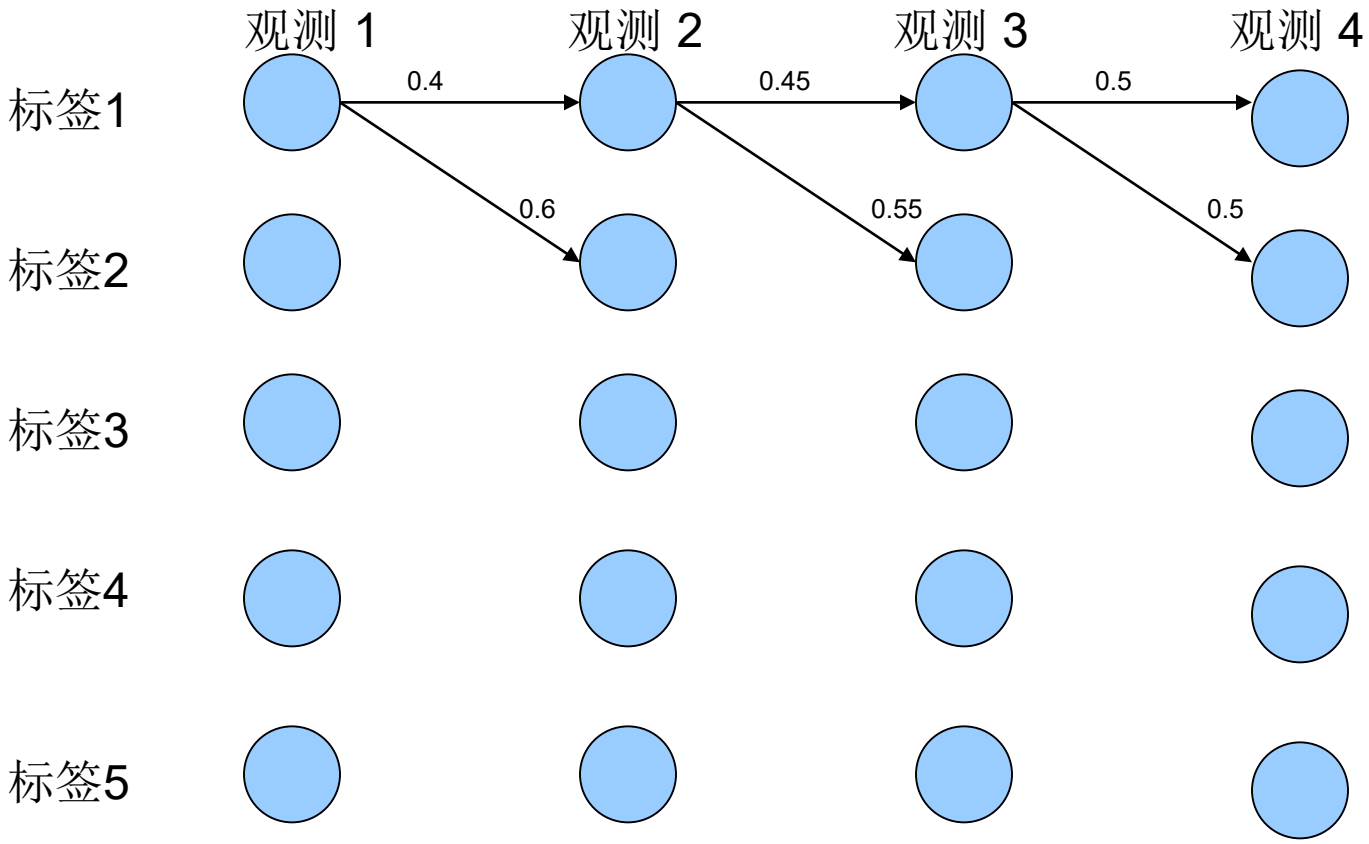
标注偏置问题(label bias problem)



从局部的概率转移情况可以看到:

- 标签1总是倾向于转移到标签2

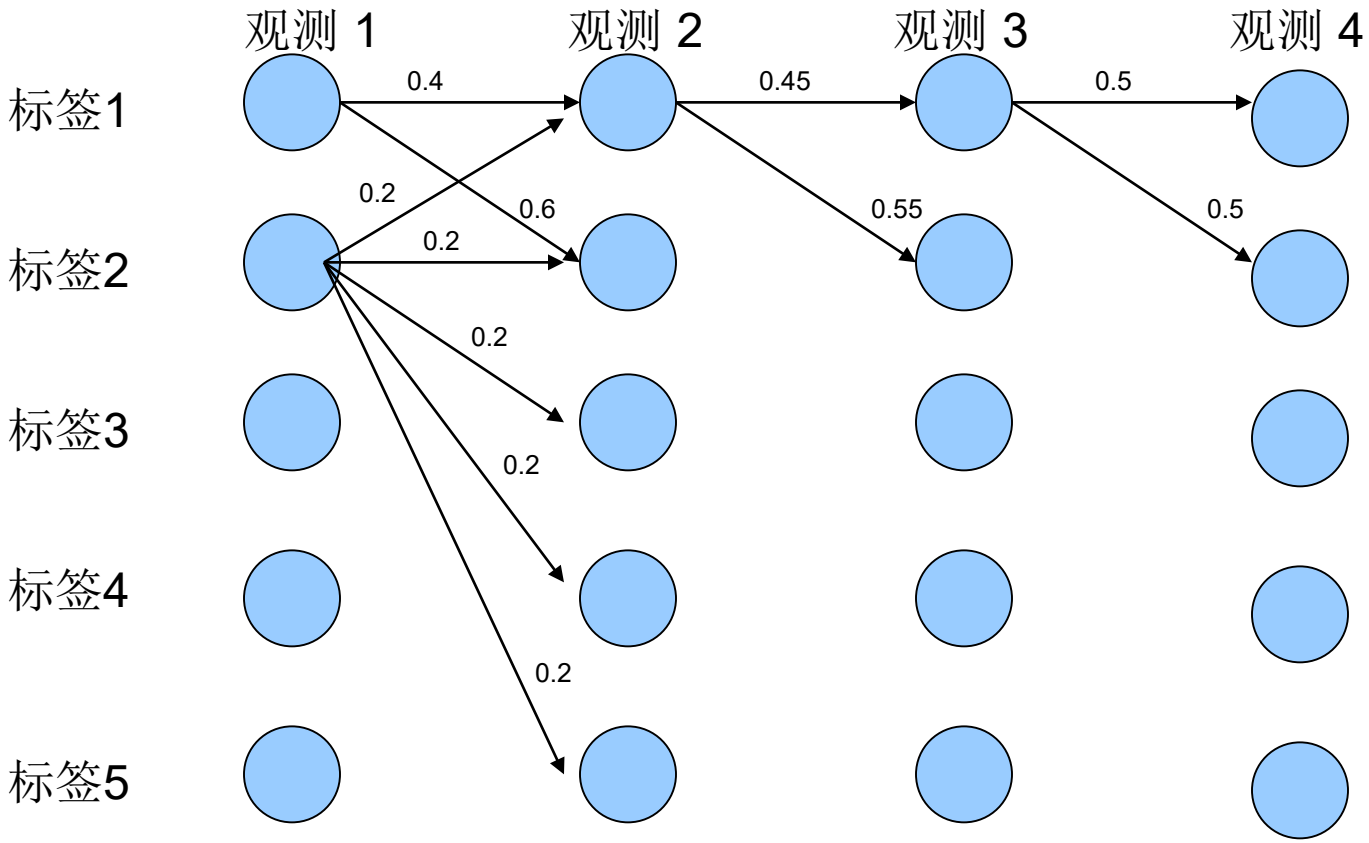
标注偏置问题(label bias problem)



从局部的概率转移情况可以看到:

- 标签1总是倾向于转移到标签2

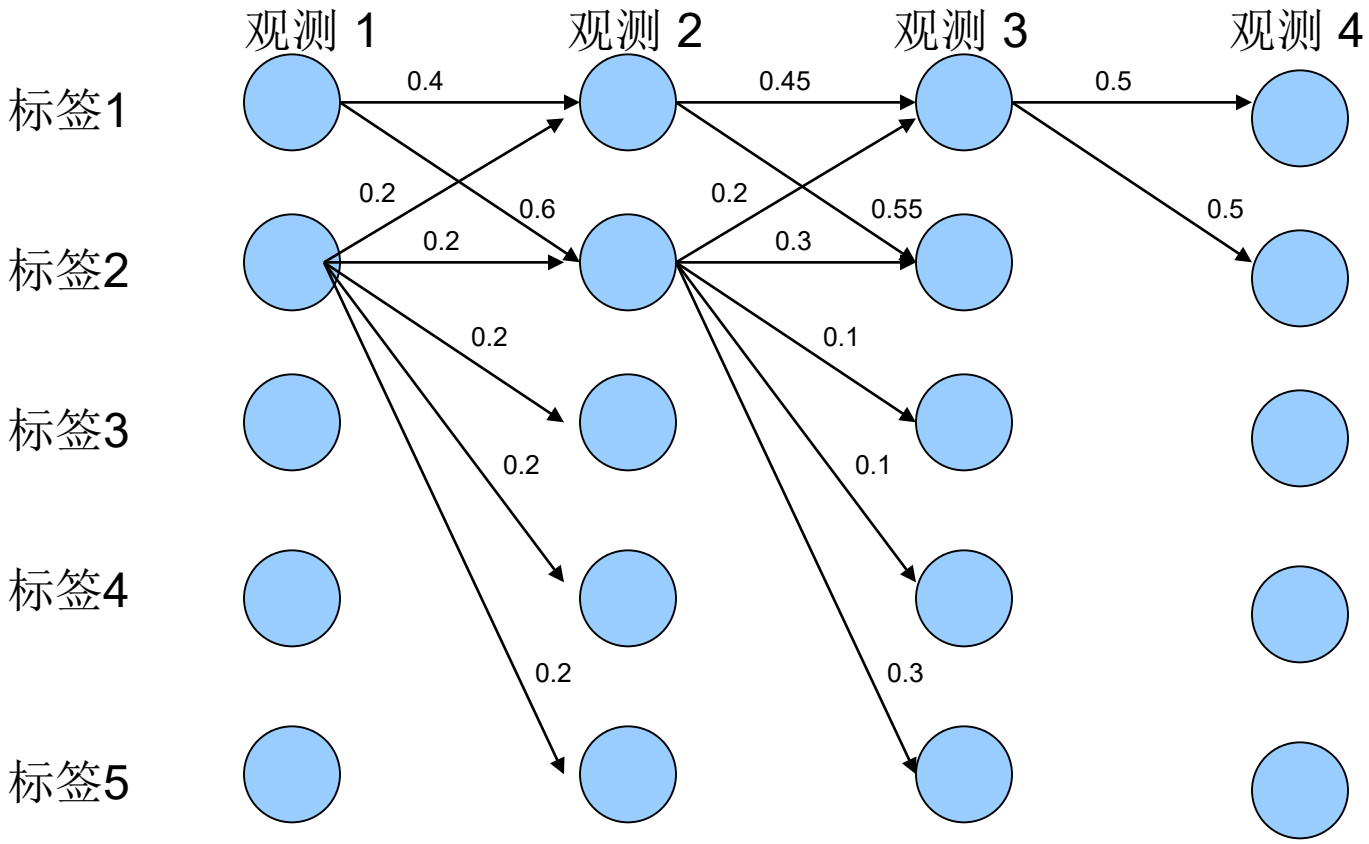
标注偏置问题(label bias problem)



从局部的概率转移情况可以看到:

- 标签1总是倾向于转移到标签2
- 标签2总是倾向于转移到标签2

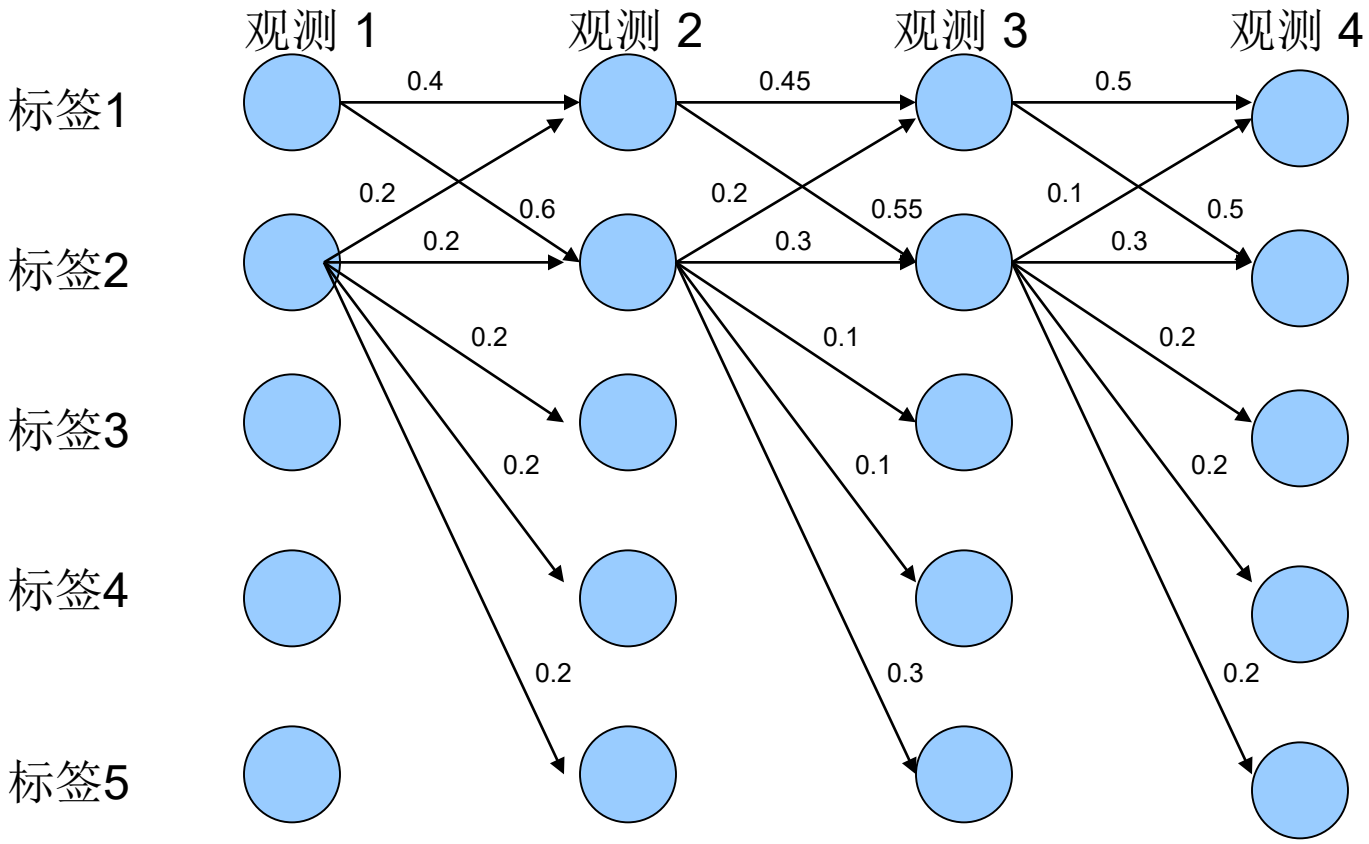
标注偏置问题(label bias problem)



从局部的概率转移情况可以看到:

- 标签1总是倾向于转移到标签2
- 标签2总是倾向于转移到标签2

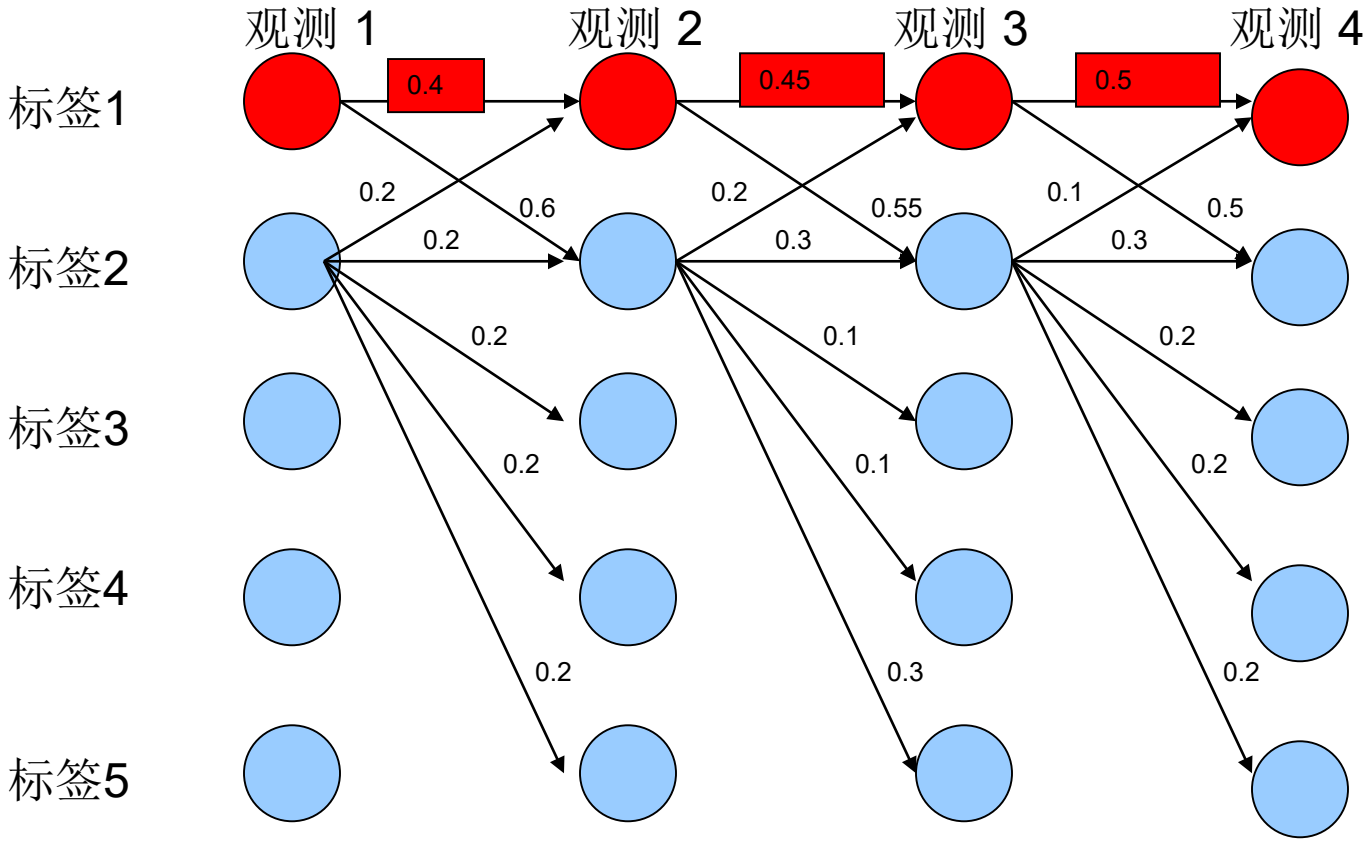
标注偏置问题(label bias problem)



从局部的概率转移情况可以看到:

- 标签1总是倾向于转移到标签2
- 标签2总是倾向于转移到标签2

标注偏置问题(label bias problem)



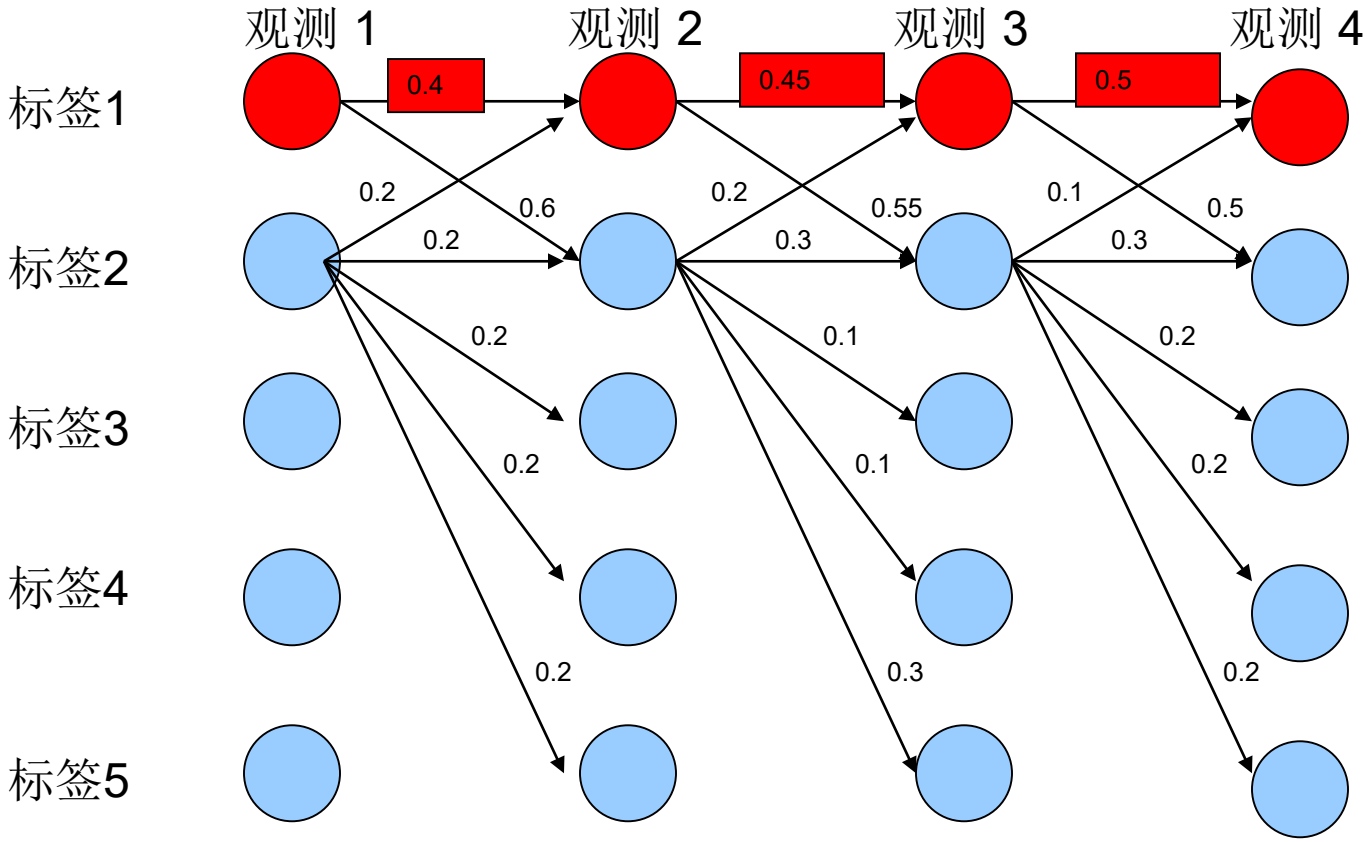
- **根据局部的概率转移情况，可得到局部最优的标签序列：**

  1 → 2 → 2 → 2 (0.6 x 0.3 x 0.3 = 0.054)

- **而实际上，全局最优的标签序列是:**

  1 → 1 → 1 → 1 (0.4 x 0.45 x 0.5 = 0.09)

标注偏置问题(label bias problem)



- **根据局部的概率转移情况，可得到局部最优的标签序列：**

  1 → 2 → 2 → 2 (0.[illegible]3 x 0.3 = 0.054)

- **而实际上，全局最优的标签序列是:**

  1 → 1 → 1 → 1 (0.[illegible]45 x 0.5 = 0.09)

(全局)序列标注模型

- **(全局)序列标注模型**
  - 能够对序列标注问题进行一个全局的建模，并确定一个全局最优的决策。从而解决局部最优导致的问题，例如标注偏置问题

常用的(全局)序列标注模型

- 隐马尔可夫模型 Hidden Markov Model（HMM)
- 结构化感知器 Structured Perceptron

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

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

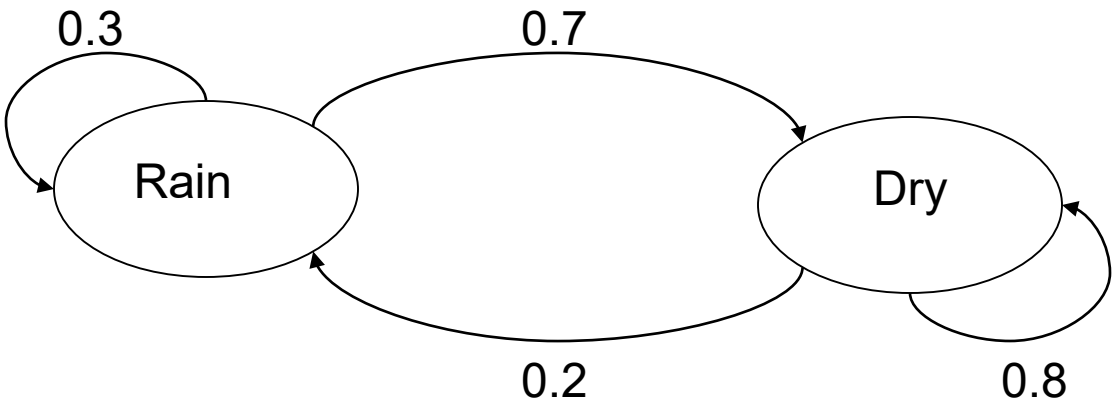
- 马尔科夫独立性假设(Markov assumption)：一个状态的概率只和之前的一个状态相关:

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

- 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

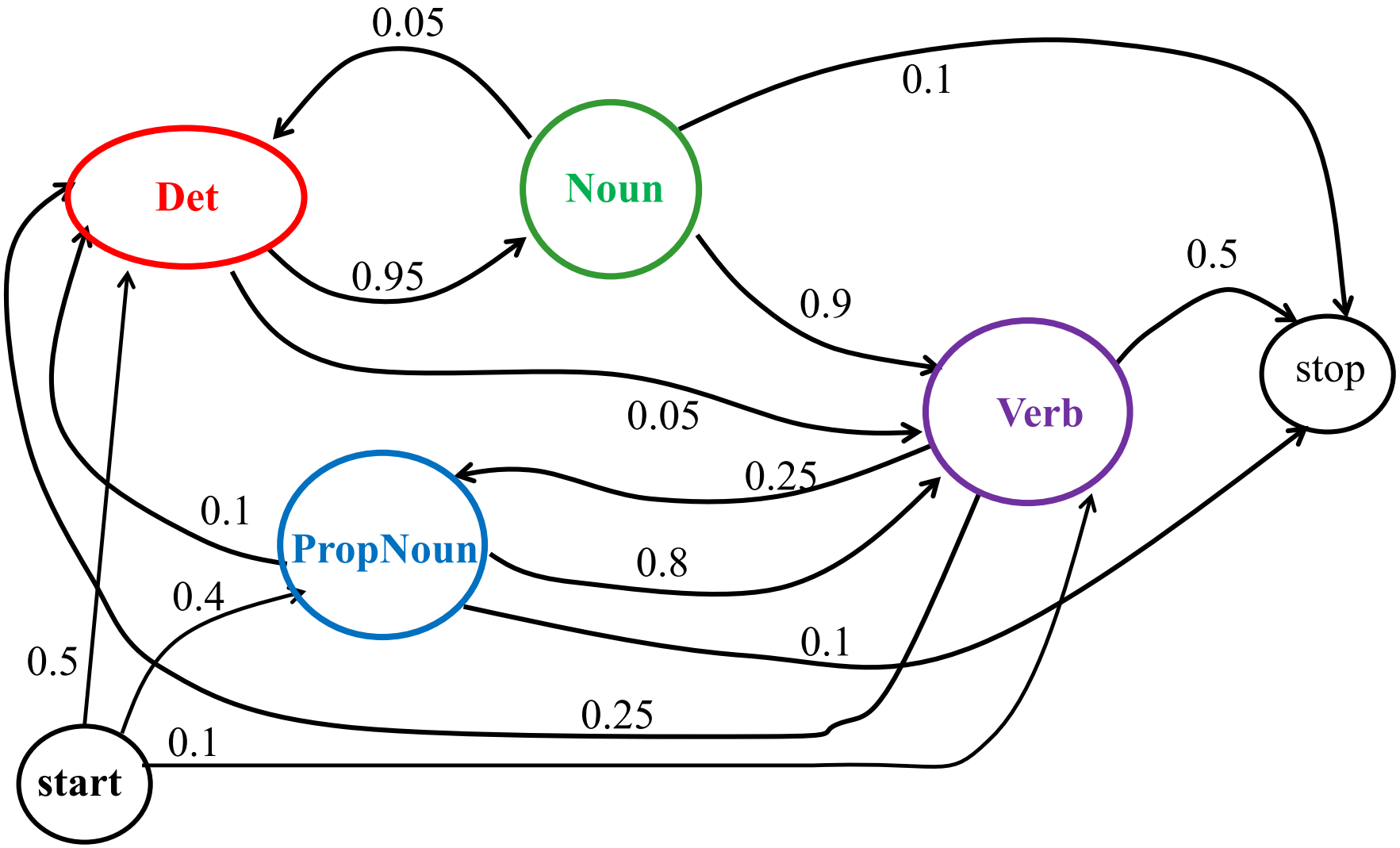
- 根据马尔科夫独立性假设，一个状态序列的概率可以这样计算：

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

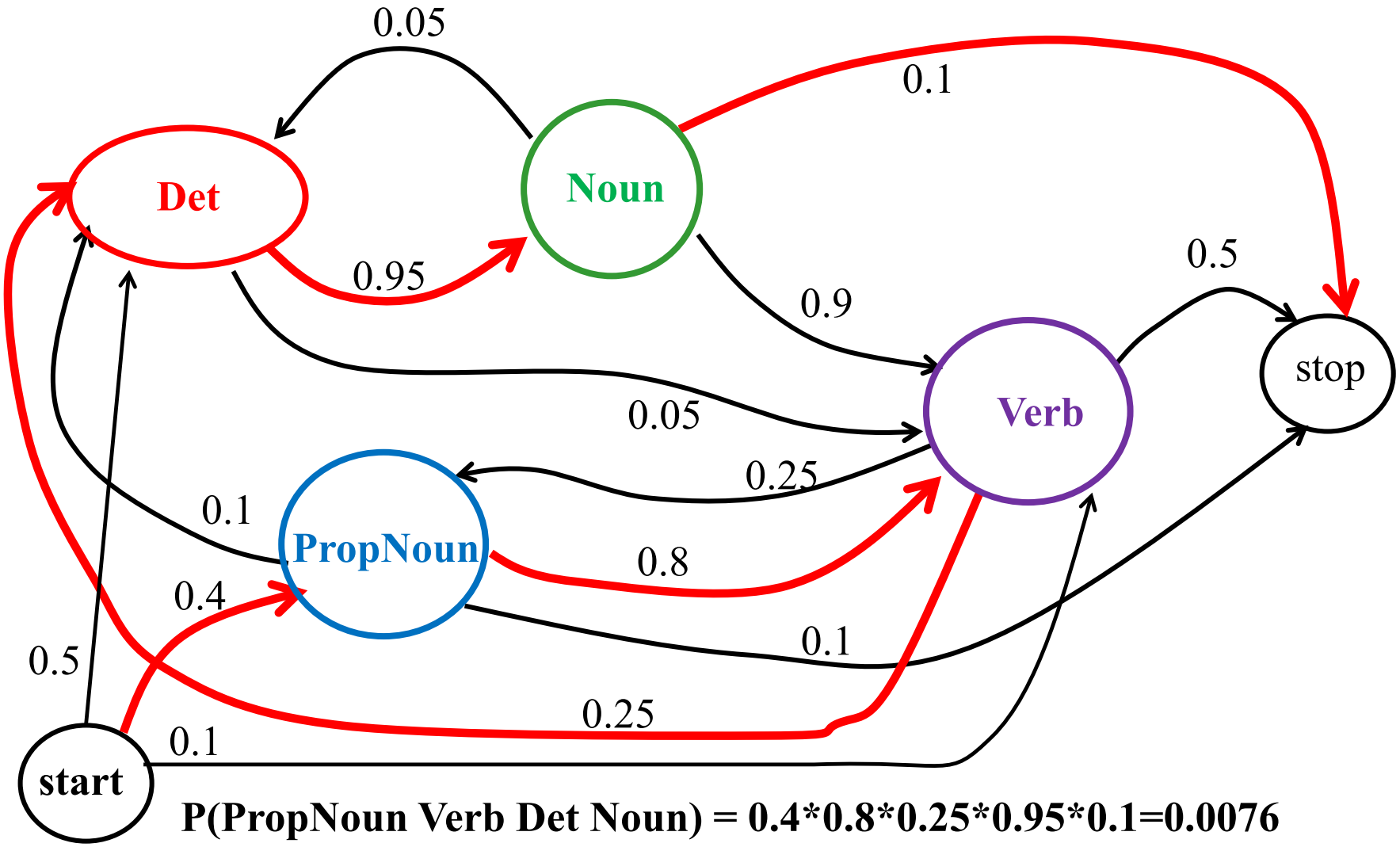
- 假设我们想计算这样的一个天气状态的序列：
{ ‘Dry’ ,’ Dry’ ,’ Rain’ ,Rain’ }

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 $A=\{a_{ij}\}$

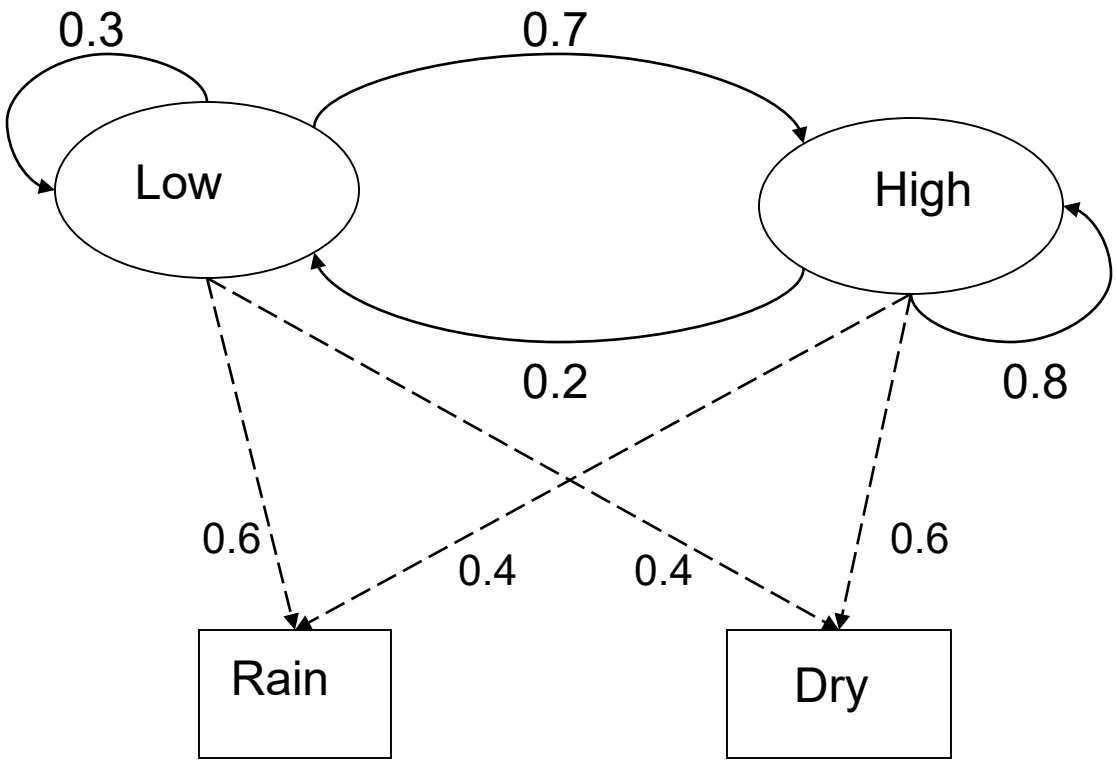
$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i) \qquad 1 \le i, j \le N \text{ and } i = 0, j = F$$

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

- 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 $B=\{b_j(k)\}$

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

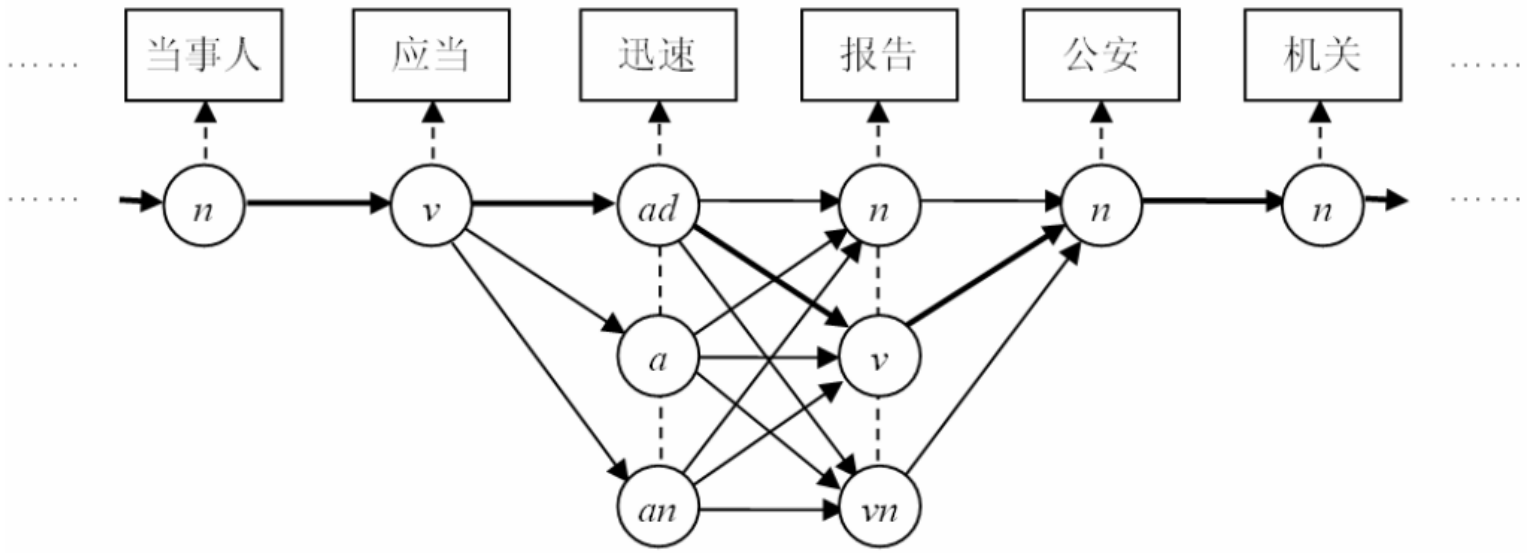
HMM举例



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

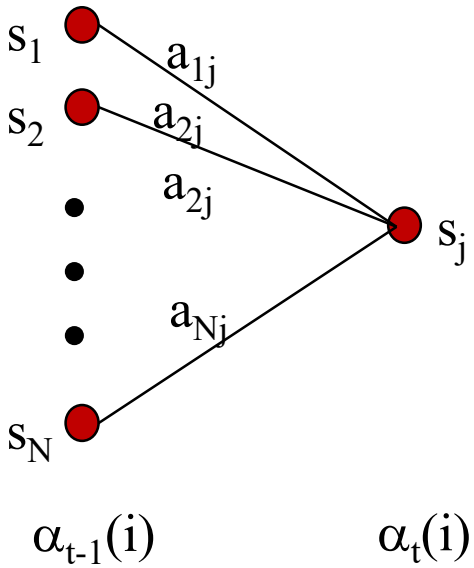
前向概率

- 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

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

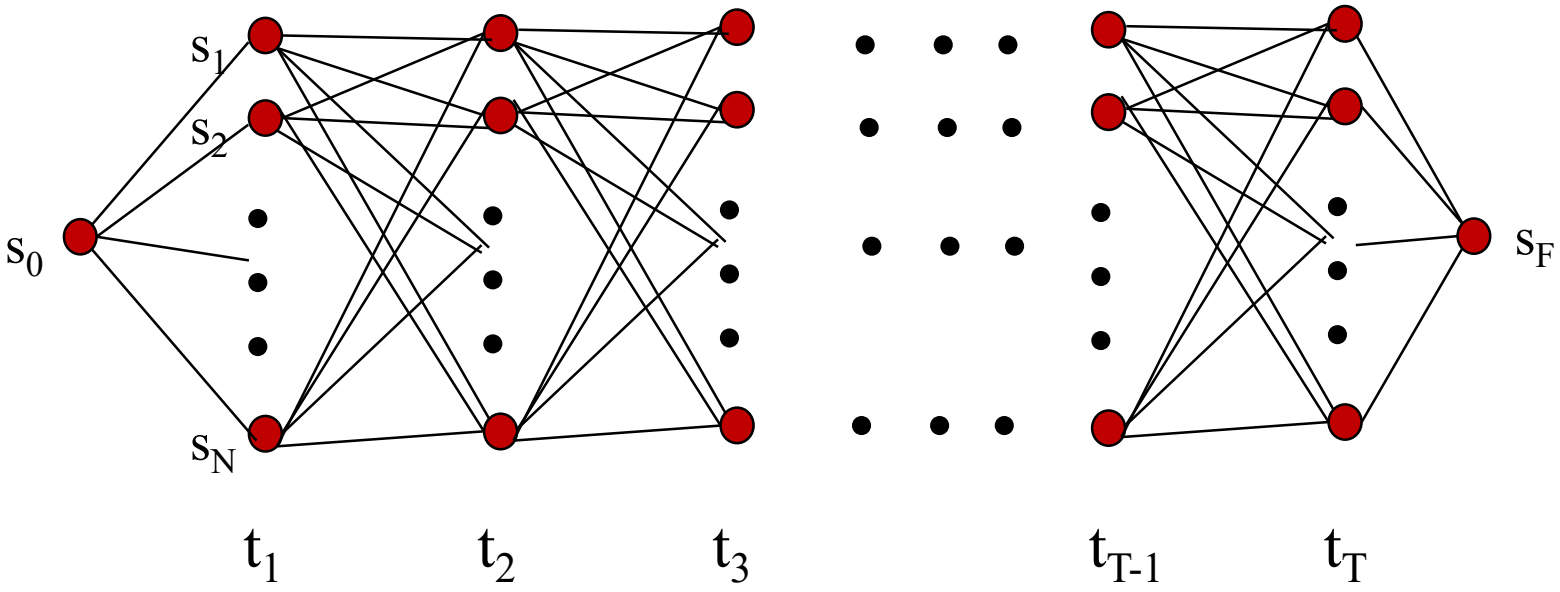
前向概率计算步骤

- 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(TN^2)

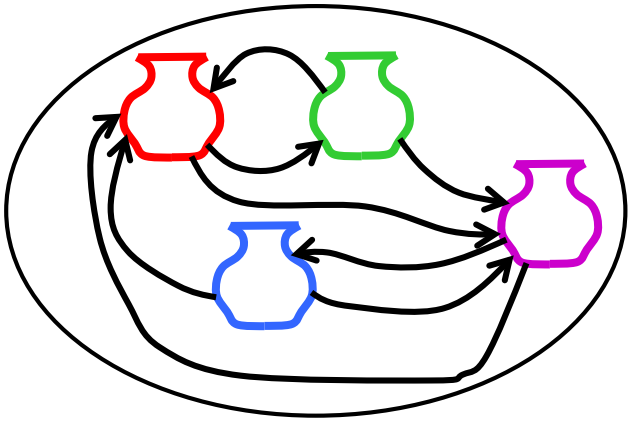
- 结束

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

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 $Q=q_1,q_2,...q_T$?

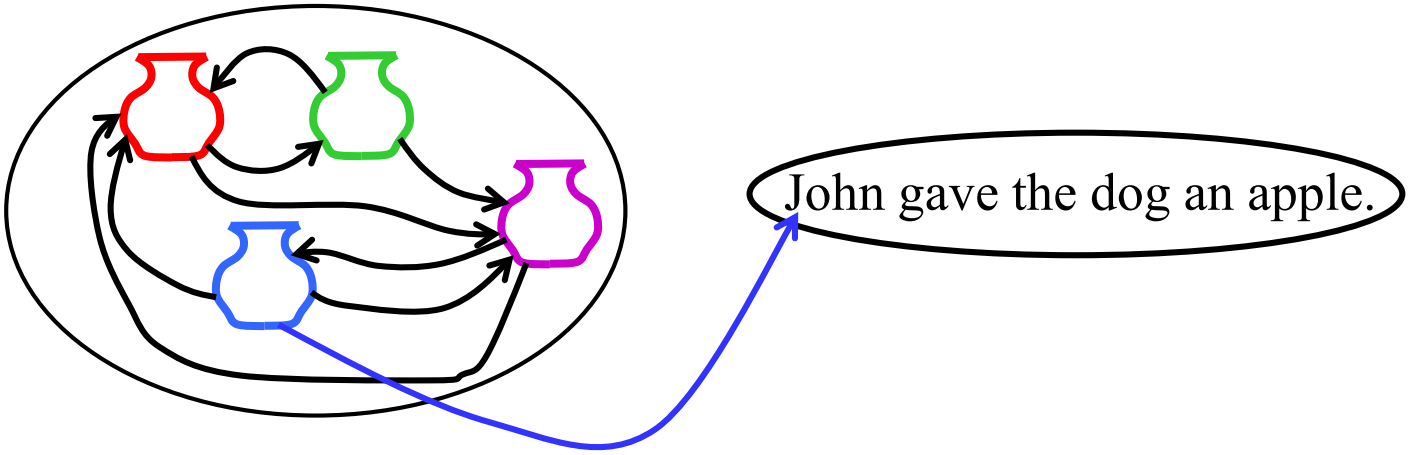
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



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 $Q=q_1,q_2,...q_T$?

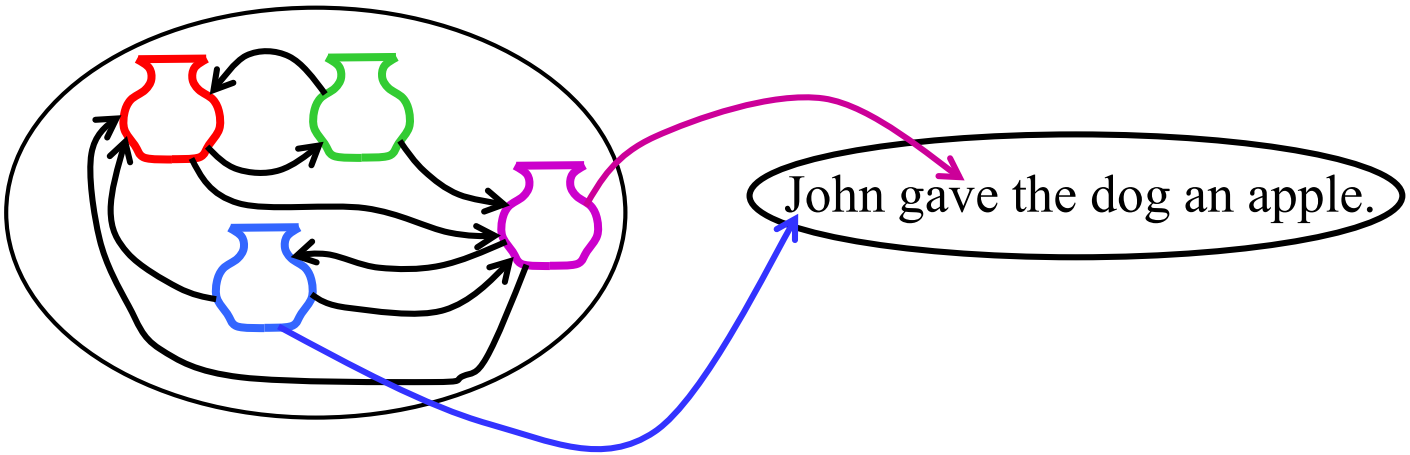
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



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 $Q=q_1,q_2,...q_T$?

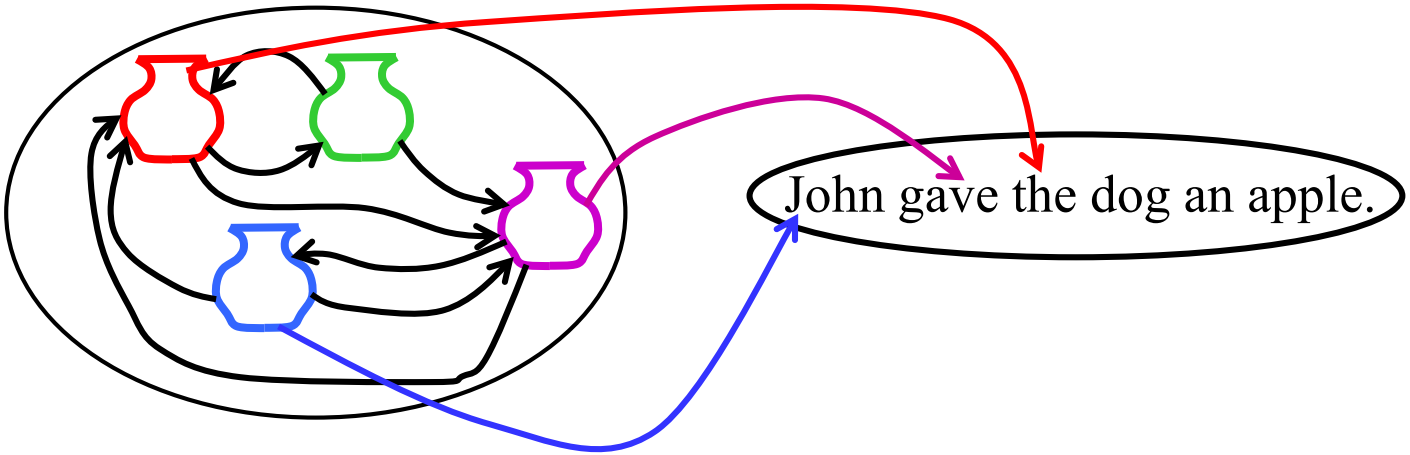
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



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 $Q=q_1,q_2,...q_T$?

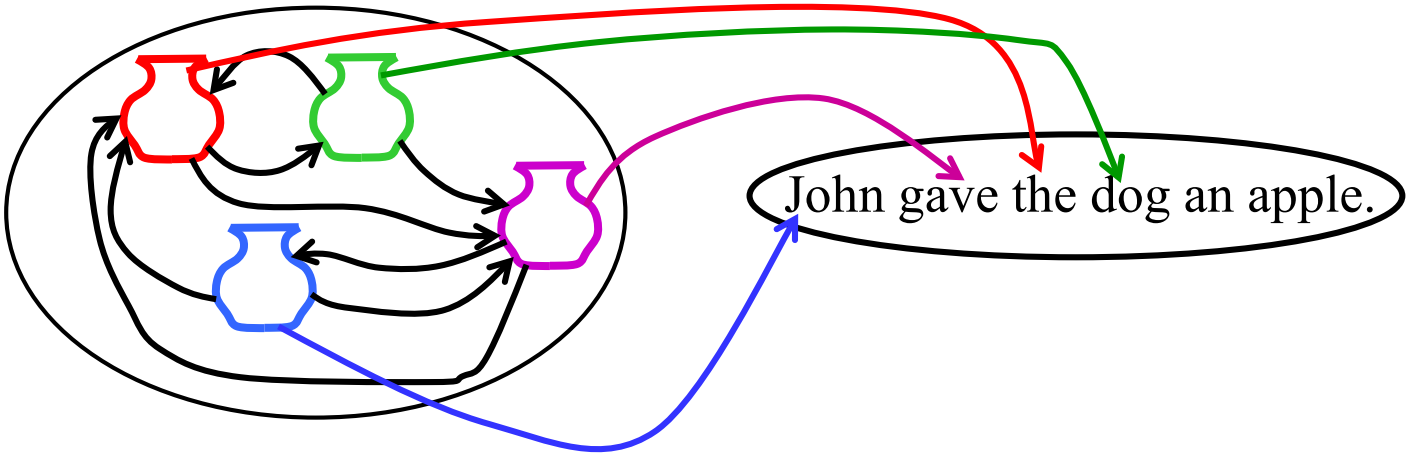
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



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 $Q=q_1,q_2,...q_T$?

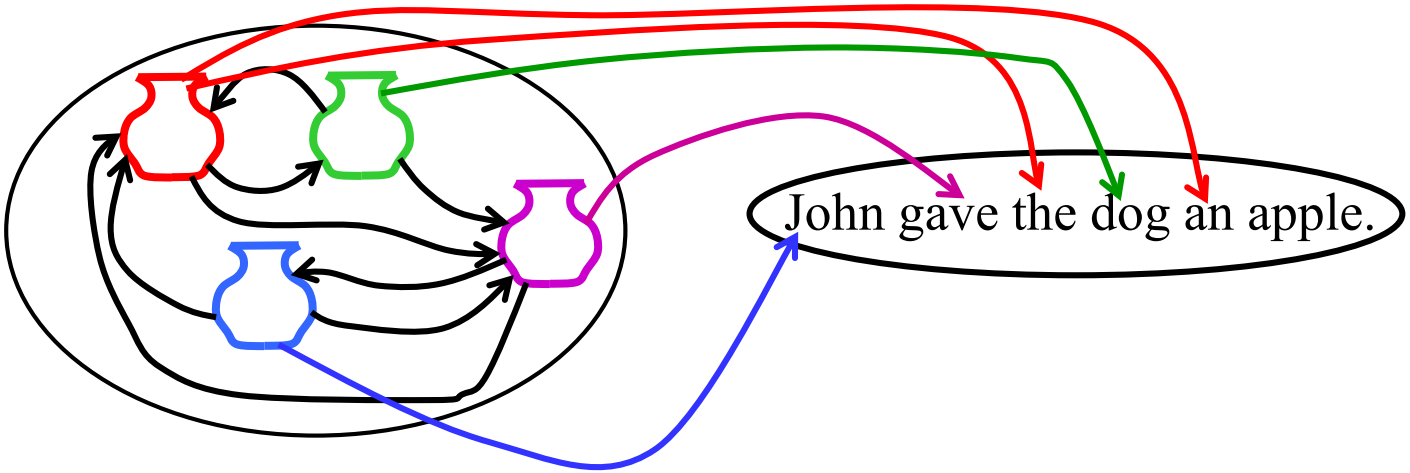
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



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 $Q=q_1,q_2,...q_T$?

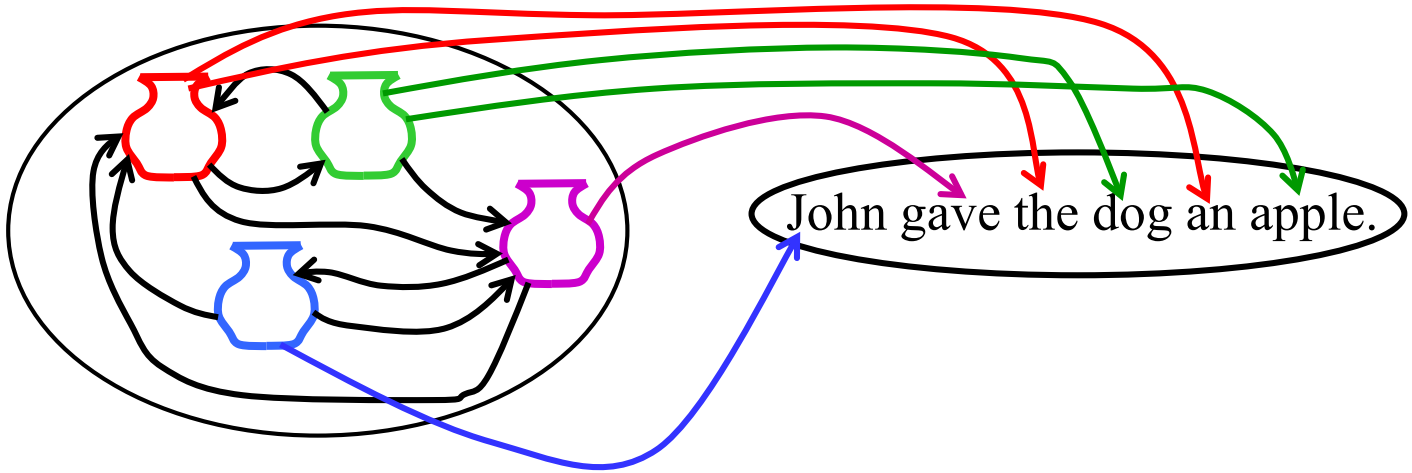
- 这个解码问题直接对应了序列标注问题！对于序列标注，假设每个状态对应一个标签，则解码问题在严谨概率计算框架下求解了“全局最优”的标签序列



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

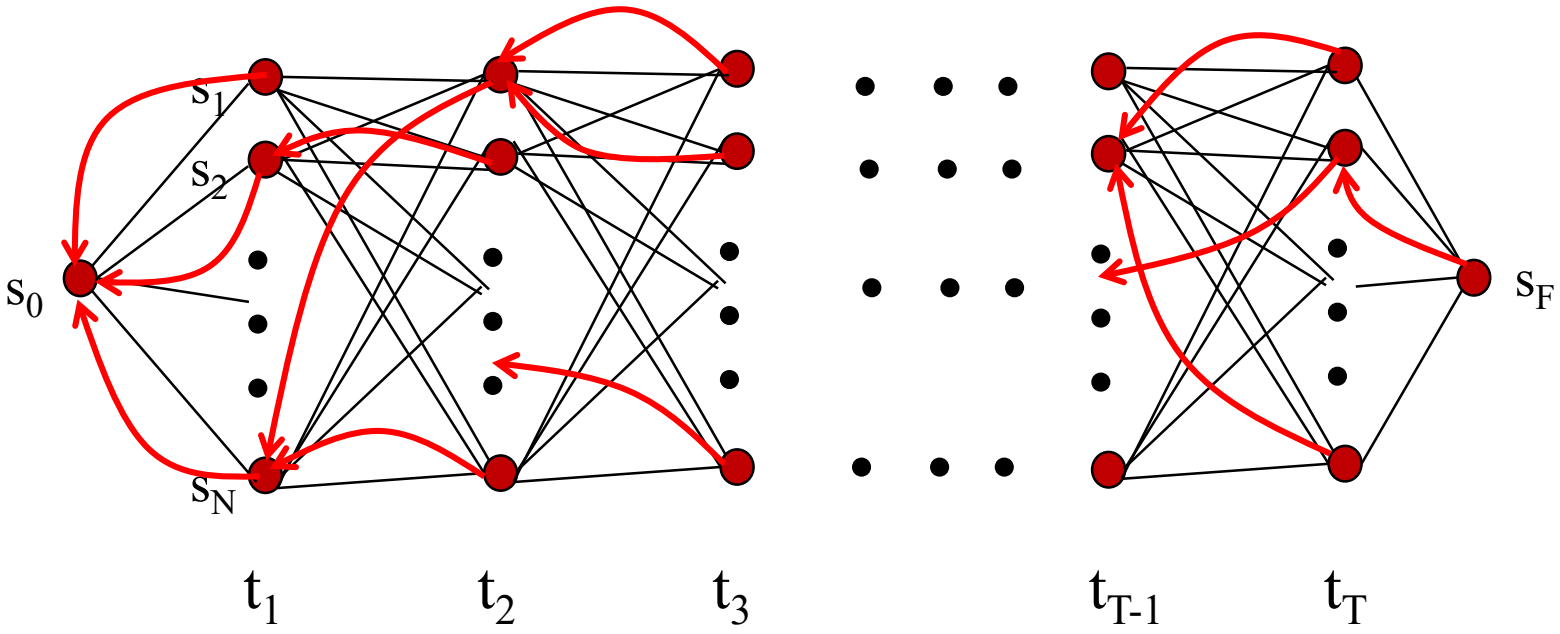
$$bt_t(j) = \underset{i=1}{\overset{N}{\operatorname{argmax}}}\, v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \quad 1 \le t \le T$$

- 算法结束

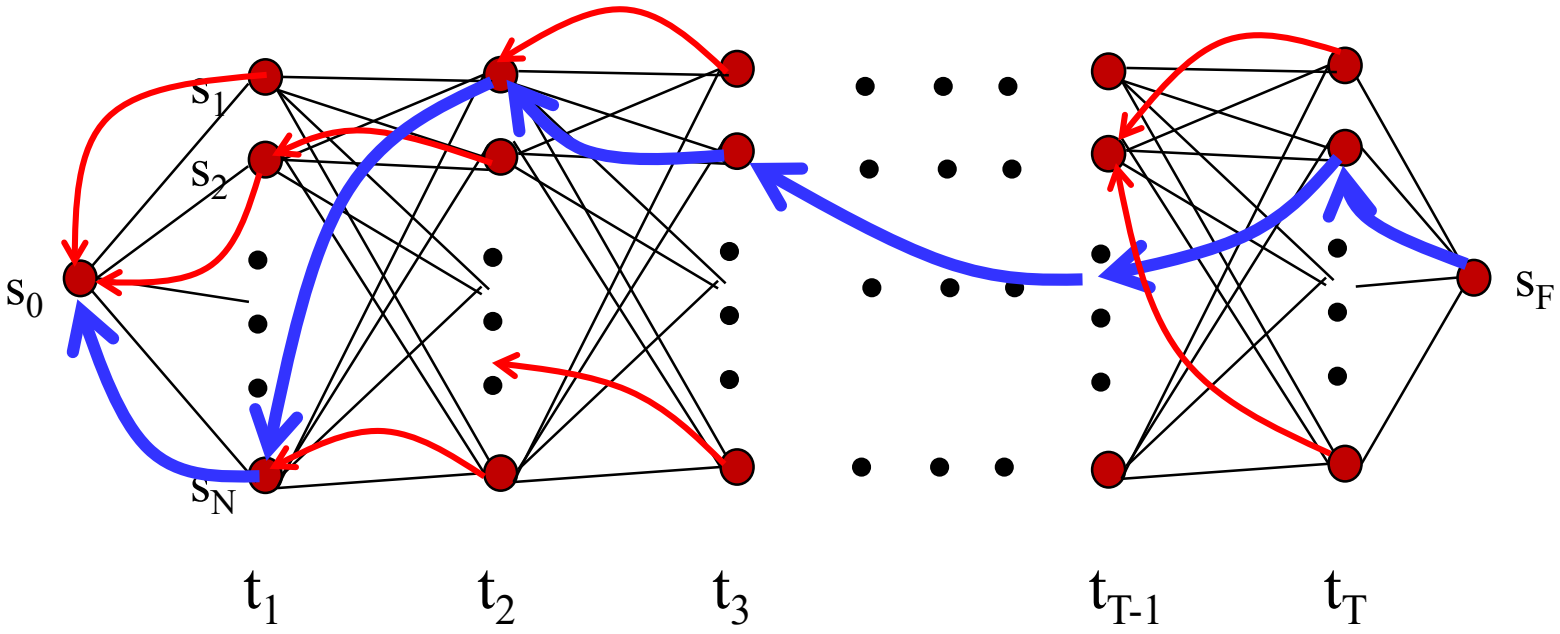
$$q_T^* = bt_{T+1}(s_F) = \underset{i=1}{\overset{N}{\operatorname{argmax}}}\, v_T(i) a_{iF}$$

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



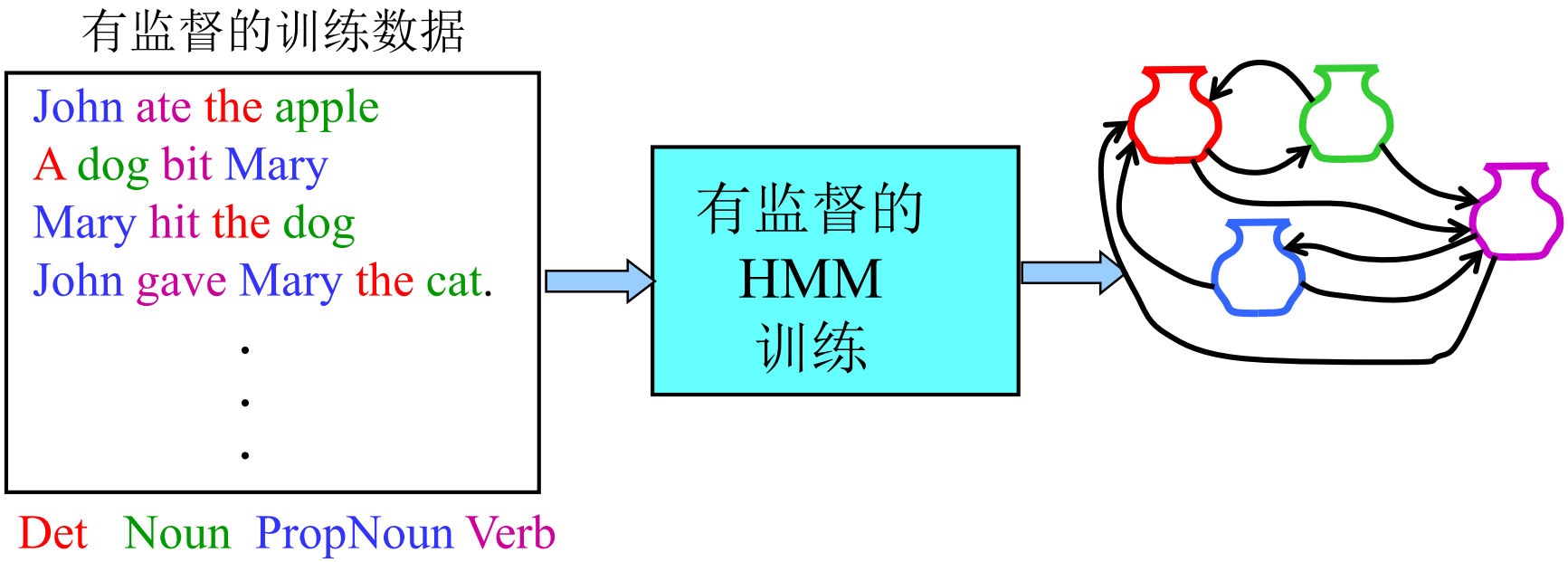
维特比算法：回溯



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

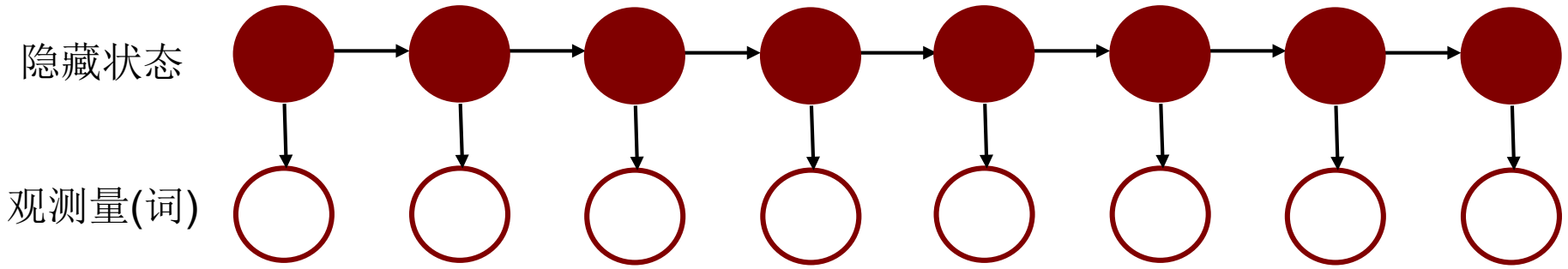
HMM优缺点

- 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

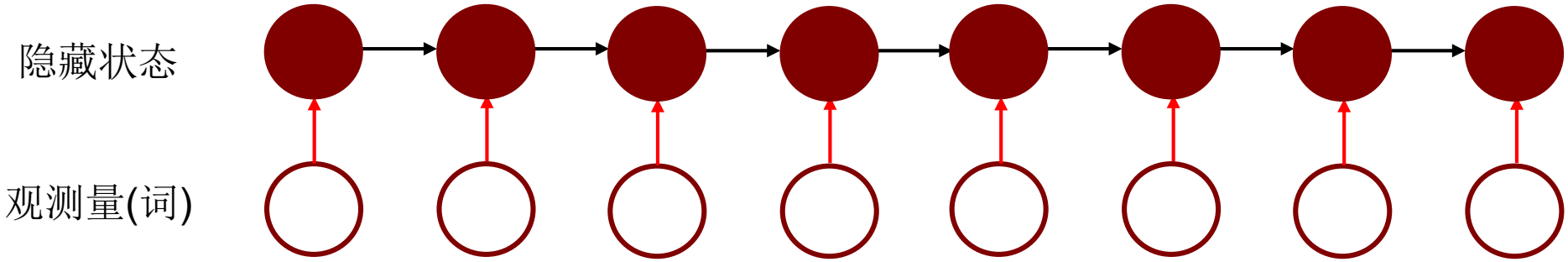
- 缺点:
  - HMM是一个**生成模型(generative model)**，解码过程需要建模 (x,y) 的联合概率分布，以及生成概率。
    → 对于序列标注问题来说，这相当于绕弯路



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

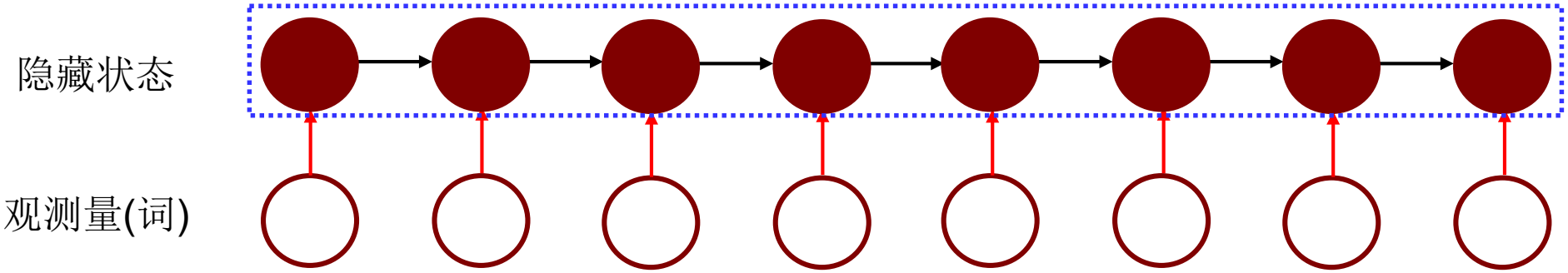
判别模型 Discriminative Models



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)**

- **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
    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

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

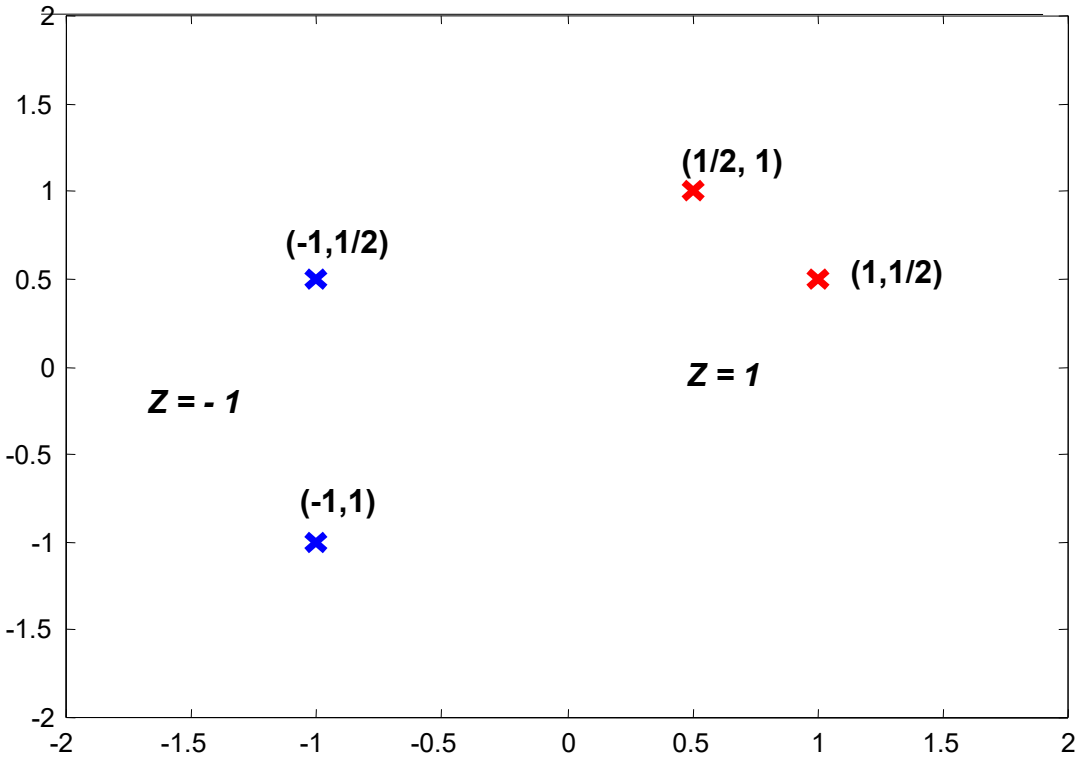
具体算法

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* $i = 1 \ldots d$

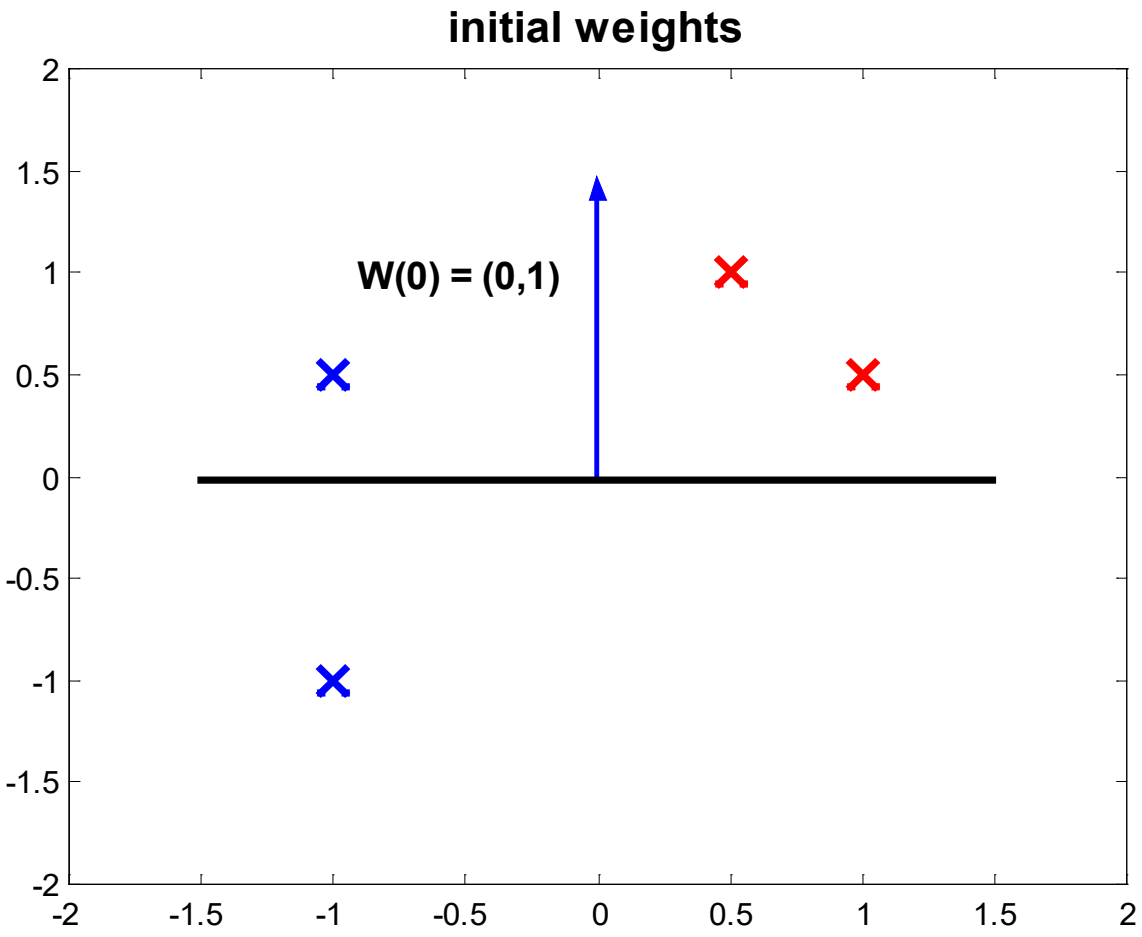
- 感知器模型是基于简单的加减法！
  - 优点一：非常容易实现
  - 优点二：而且实际效果好

举例说明：为什么简单的加减法可以实现线性分类

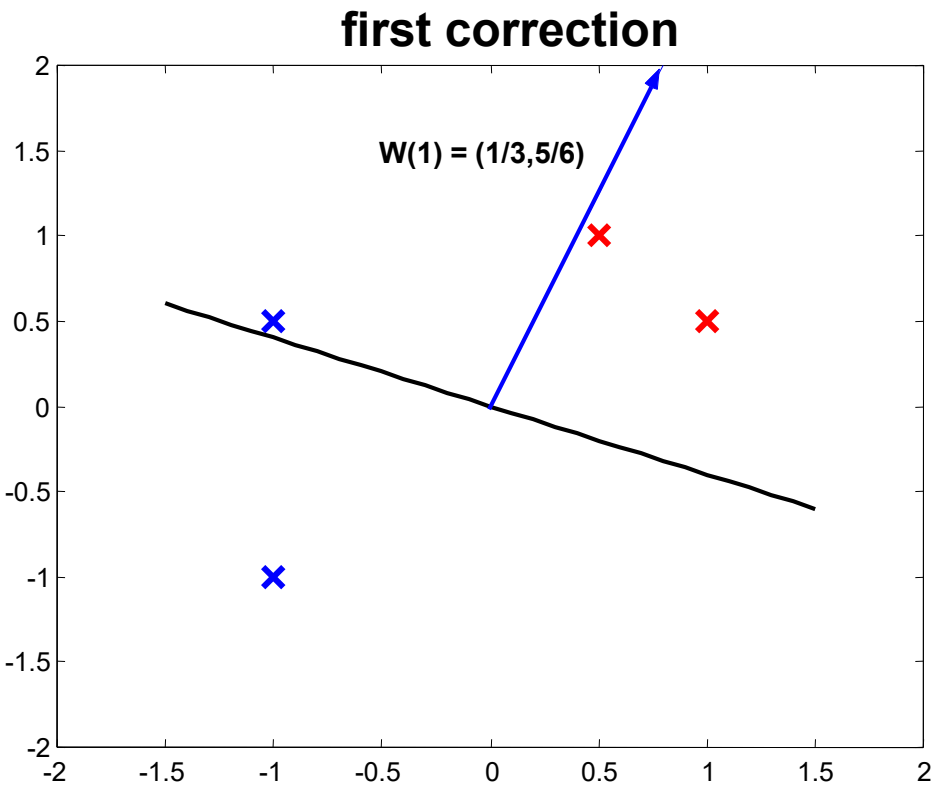
4个线性可分(linearly separable)的数据点



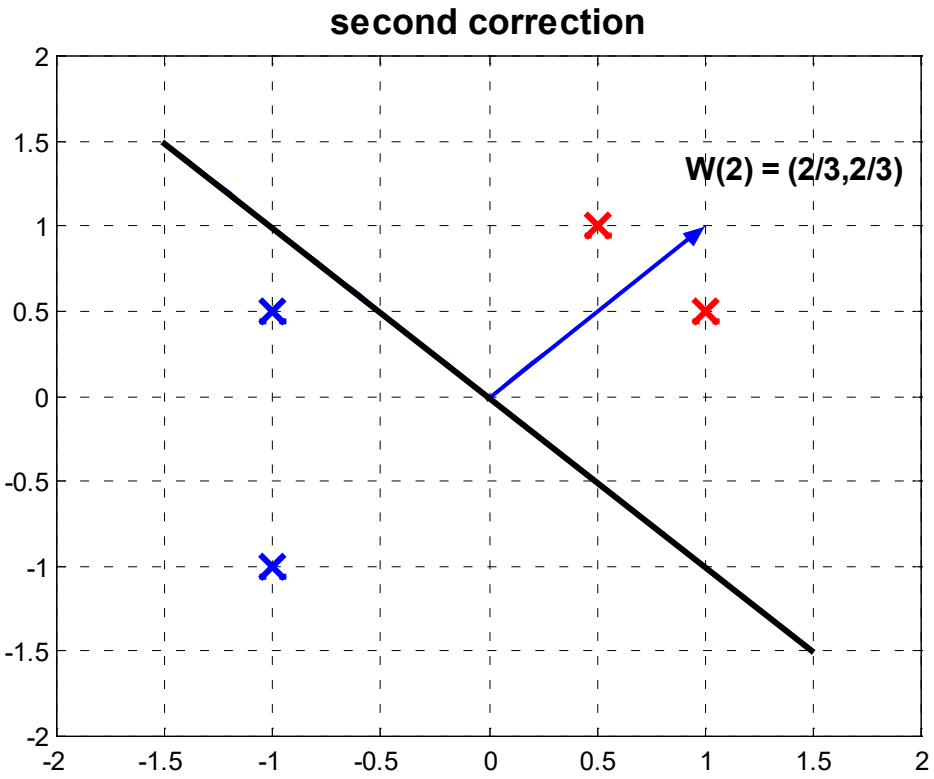
举例说明：为什么简单的加减法可以实现线性分类



举例说明：为什么简单的加减法可以实现线性分类



举例说明：为什么简单的加减法可以实现线性分类



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

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

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

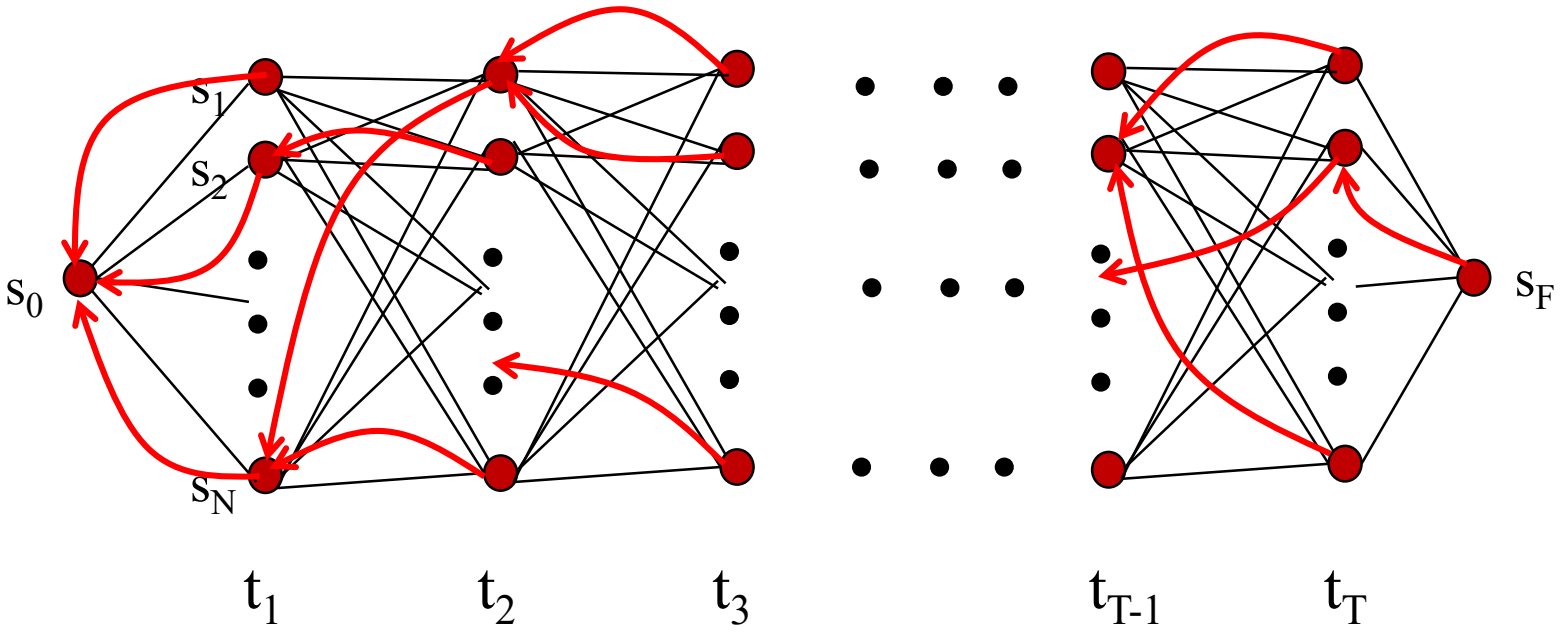
$$bt_t(j) = \underset{i=1}{\overset{N}{\operatorname{argmax}}}\, v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \quad 1 \le t \le T$$

- 算法结束

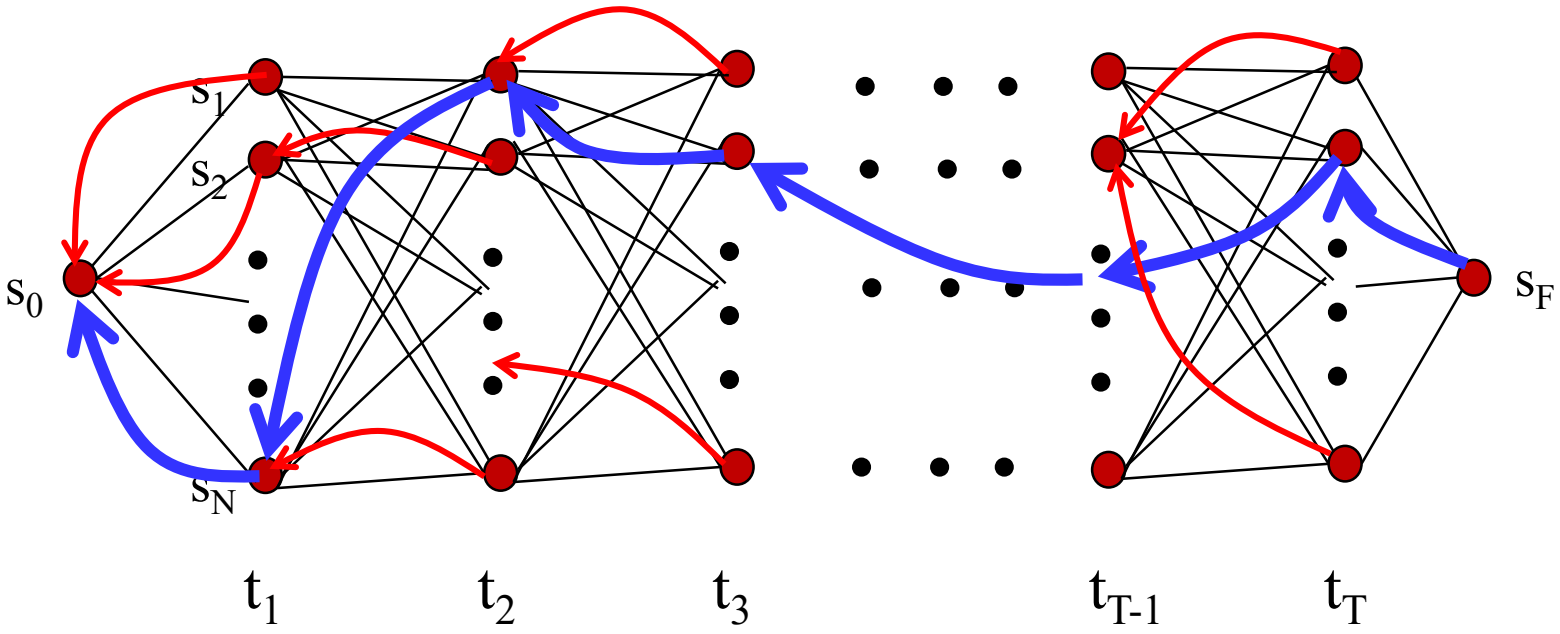
$$q_T^* = bt_{T+1}(s_F) = \underset{i=1}{\overset{N}{\operatorname{argmax}}}\, v_T(i) a_{iF}$$

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

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

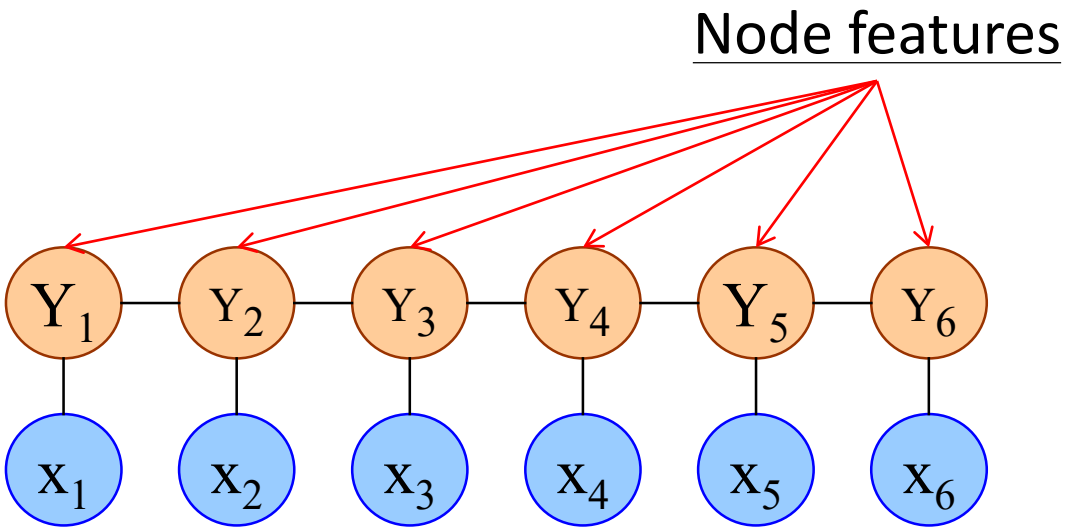
结构化感知器模型

- 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

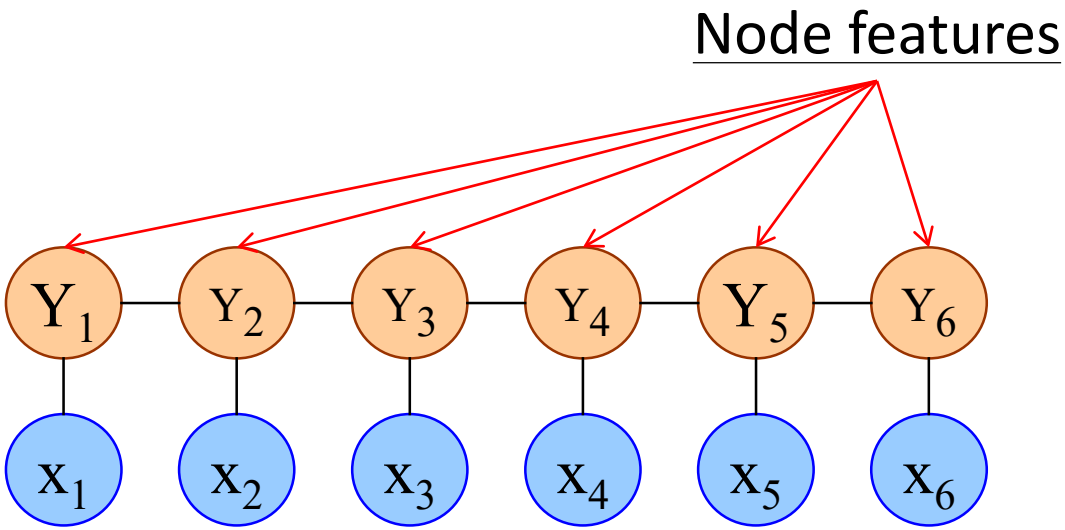
怎么提取结构相关特征(Feature)



节点特征

Node feature

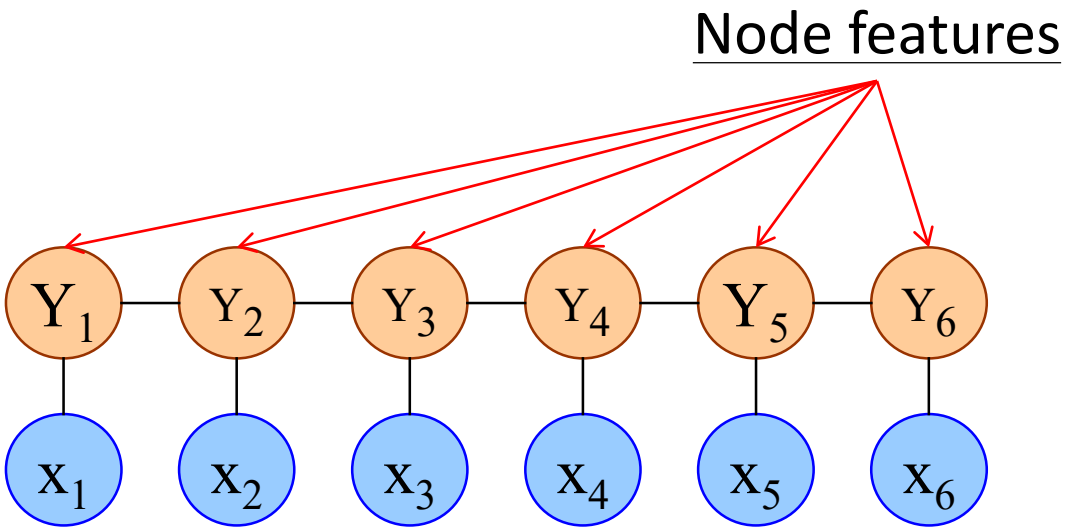
怎么提取结构相关特征(Feature)



节点特征

Node feature

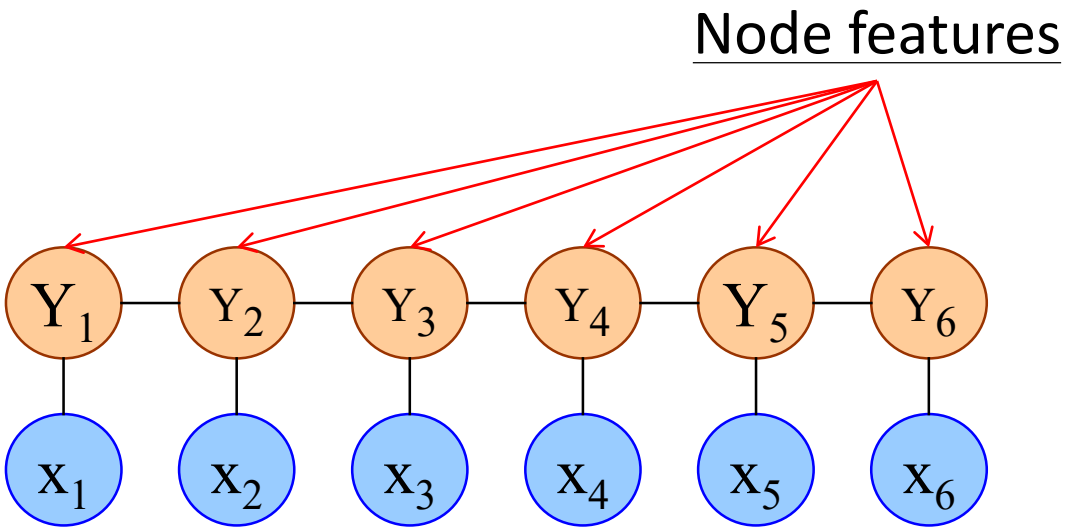
怎么提取结构相关特征(Feature)



节点特征

Node feature

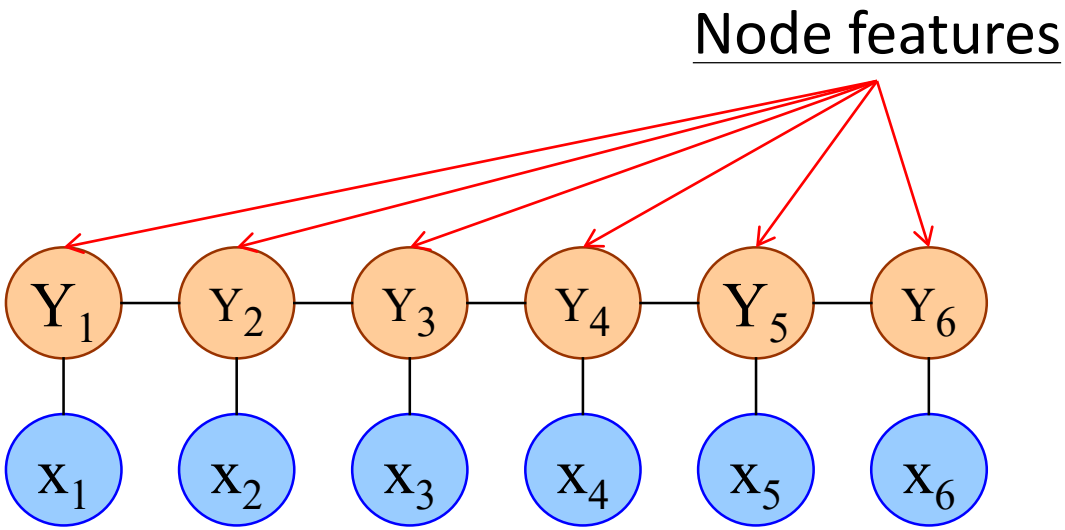
怎么提取结构相关特征(Feature)



节点特征

Node feature

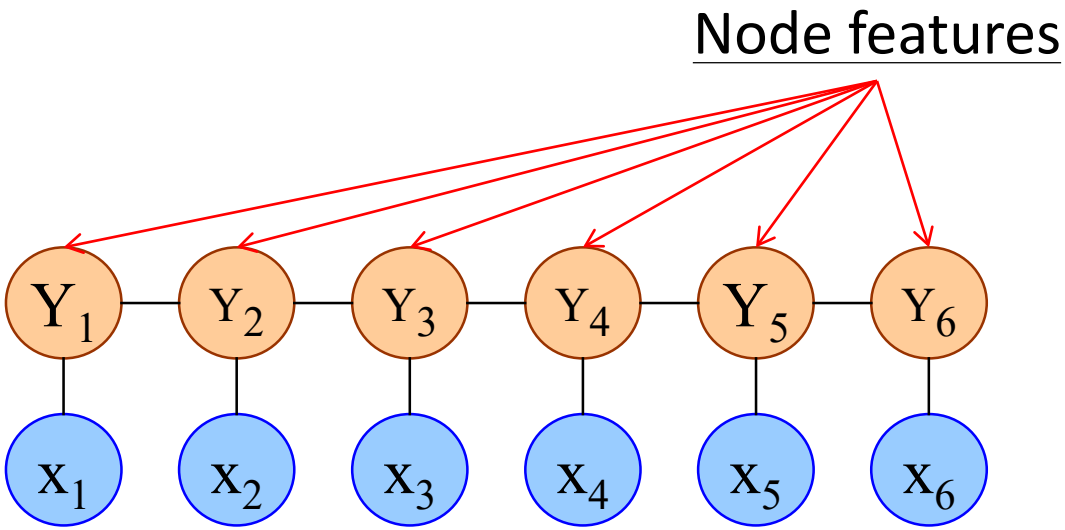
怎么提取结构相关特征(Feature)



节点特征

Node feature

怎么提取结构相关特征(Feature)

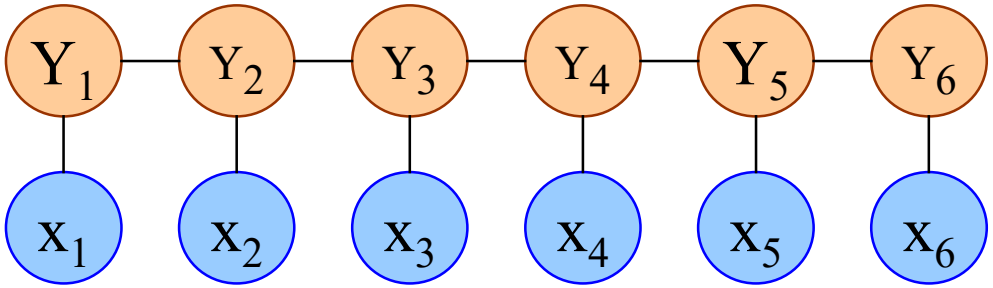


节点特征

Node feature

怎么提取结构相关特征(Feature)

Edge feature

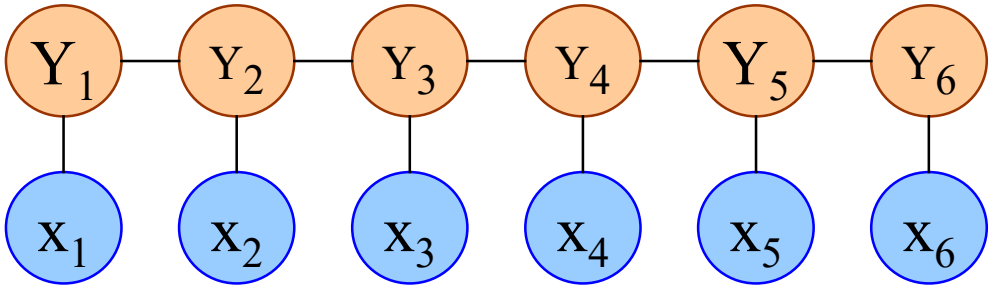


边特征

Edge feature

怎么提取结构相关特征(Feature)

Edge feature

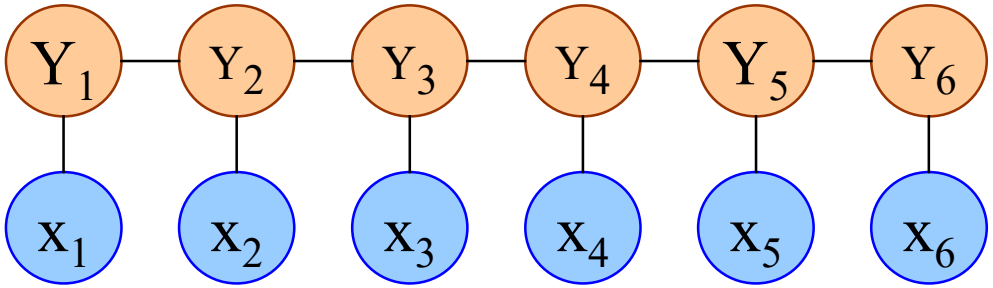


边特征

Edge feature

怎么提取结构相关特征(Feature)

Edge feature

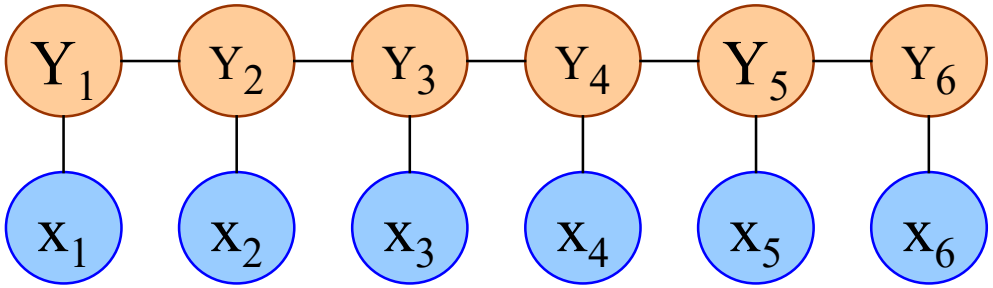


边特征

Edge feature

怎么提取结构相关特征(Feature)

Edge feature

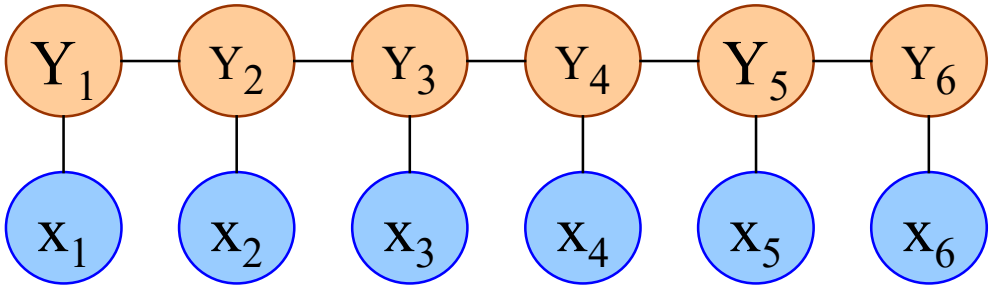


边特征

Edge feature

怎么提取结构相关特征(Feature)

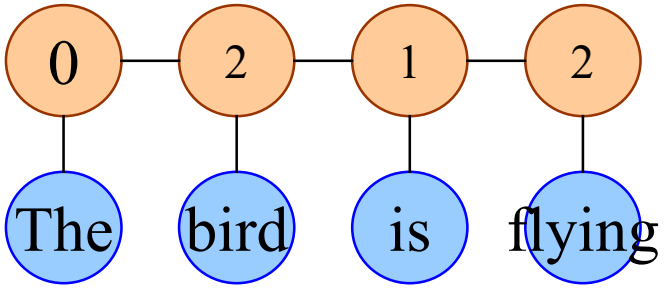
Edge feature



边特征

Edge feature

特征提取举例



假设我们采用如下特征模板：

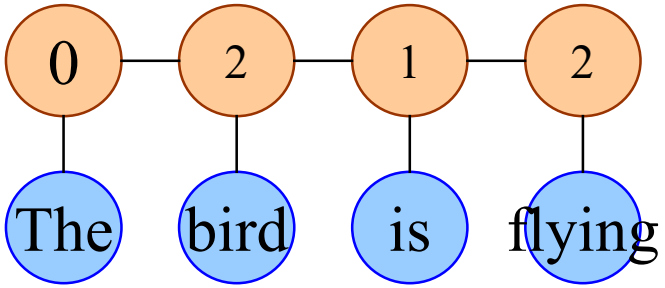
1，节点特征：

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:

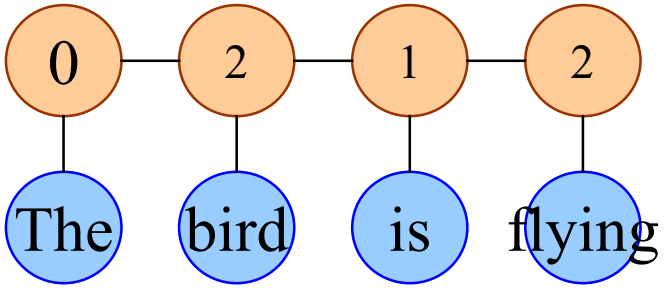
the_0, bird_2, is_1, flying_2

*_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:

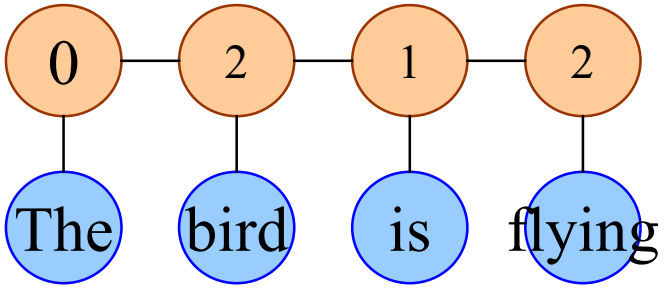
2, 4, 5, 7

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

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

大纲

- **序列标注问题**
- **传统方法**
  - 简单分类
  - 标签偏置问题
  - HMM、结构化感知器
- **深度学习方法**
  - 词向量
  - 全连接神经网络
  - 循环神经网络

动机

- **如何表示一个词的含义?**

- **Webster dictionary中meaning的定义**
  - The idea that is represented by a word, phrase, etc.
  - The idea that a person wants to express by using words, signs, etc.
  - The idea that is expressed in a work of writing, art, etc.

- **传统的NLP方法：词是离散表示的**

动机

- **在传统的NLP方法中，词是原子性的符号**

hotel, conference, walk

- **在运算中一般表示为一个向量，一般称为one-hot表示**
  - 只有一个1，很多很多0

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

- **问题：维度爆炸**
  - 口语：2万
  - PTB：5万
  - 词表：50万，
  - Google 1T：1300万

动机

- **在传统的NLP方法中，词是原子性的符号**

hotel, conference, walk

- **在运算中一般表示为一个向量，一般称为one-hot表示**
  - 只有一个1，很多很多0

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

- **问题：没有含义！**
  - 没有含义信息，词与词的计算是没有意义的

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

动机

- 改进：使用共现矩阵(co-occurrence matrix)
- 示例语料
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

动机

- **改进：共现矩阵(co-occurrence matrix)**
- **问题**
  - 随词典大小增大而增大
  - 高维度：需要大量存储空间
  - 后续的分类模型会有数据稀疏问题(data sparsity)
- **模型不强健(robust)**
- **如何解决这个问题?**

解决方案

- **词向量：使用低维向量来表示一个词**
- **用固定大小且较少的维度来存储“大多数”重要的信息**
  - 通常在25-1000维
- **向量是密集的**
- **如何缩减维度?**
  - 不同的方法会产生不同类别的词向量
  - word2vec，CBOW，GloVe等

Word2Vec：思路

- **主要思想**
  - 预测每个词的周围词
  - 而不是直接计数共现信息

- **Word2Vec和GloVe的思路很相似**
  - Glove: Global Vectors for Word Representation
    - Pennington et al. 2014 and Levy and Goldberg 2014
  - 更快
  - 易于合入新的句子/文章或增大词典

Word2Vec：细节

- **预测每个词周围窗口为m内的词**

- **目标函数:**
  - 给定一个词，最大化 大小为m内的窗口内的 词的概率

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$

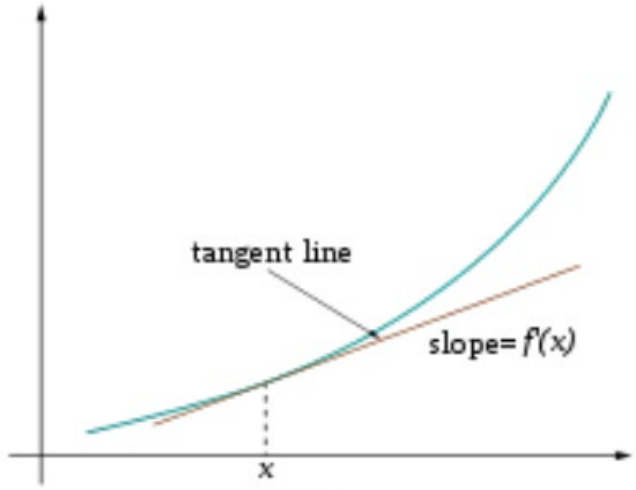
  - θ是我们要优化的参数

Word2Vec：参数更新

- **我们需要最优化我们的目标/代价函数**
  - 对于该问题要最小化
  - 最小化→梯度下降法
- **例子**

Refresher with trivial example: (from Wikipedia)
Find a local minimum of the function
$f(x)=x^4-3x^3+2$, with derivative $f'(x)=4x^3-9x^2$.

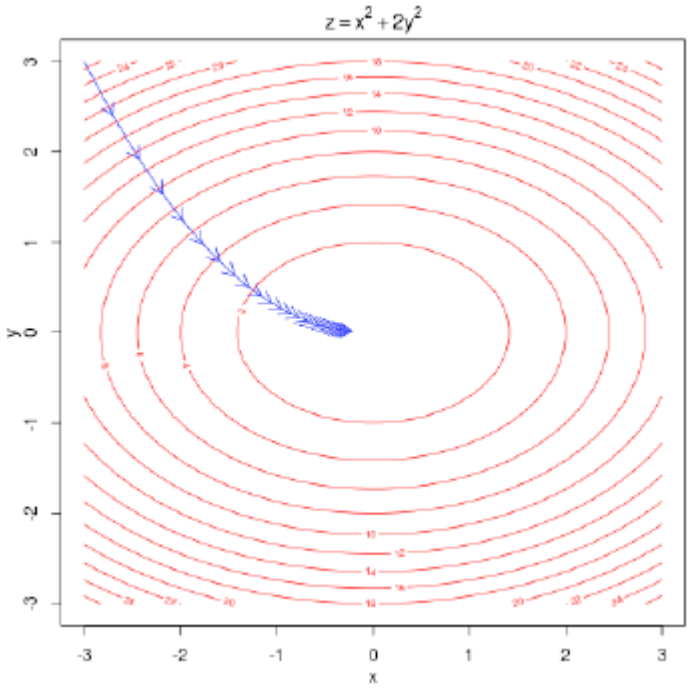
```
x_old = 0
x_new = 6 # The algorithm starts at x=6
eps = 0.01 # step size
precision = 0.00001

def f_derivative(x):
    return 4 * x**3 - 9 * x**2

while abs(x_new - x_old) > precision:
    x_old = x_new
    x_new = x_old - eps * f_derivative(x_old)

print("Local minimum occurs at", x_new)
```



Word2Vec：参数更新

- 梯度下降法的直观表示

- 对于一个简单的两个参数的凸函数
- 轮廓线表示目标函数的不同值



词向量：特点

- **词向量非常善于编码相似性**
  - 相似性检验可以直接通过在词向量空间中进行向量相减完成
  - 结果也不错 $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$
- **语法**
  - 对于动词和形容词变化也类似
- **语义(Semeval 2012 task 2)**

$$x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$$

$$x_{king} - x_{man} \approx x_{queen} - x_{woman}$$

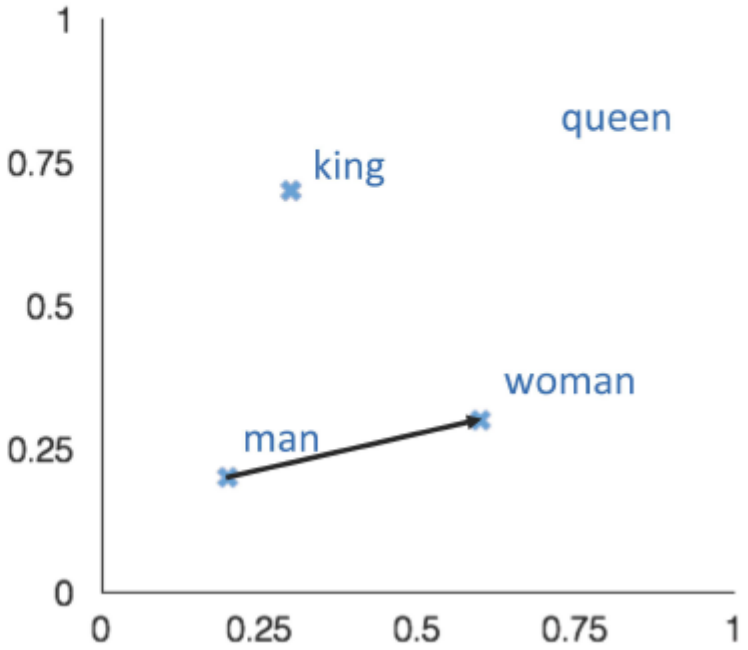
- 词向量间有线性关系[Mikolov et al., 2014]

a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



GloVe：距离frog最近的词

- 距离相近的词向量有相似的语义[Pennington et al., 2014]

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

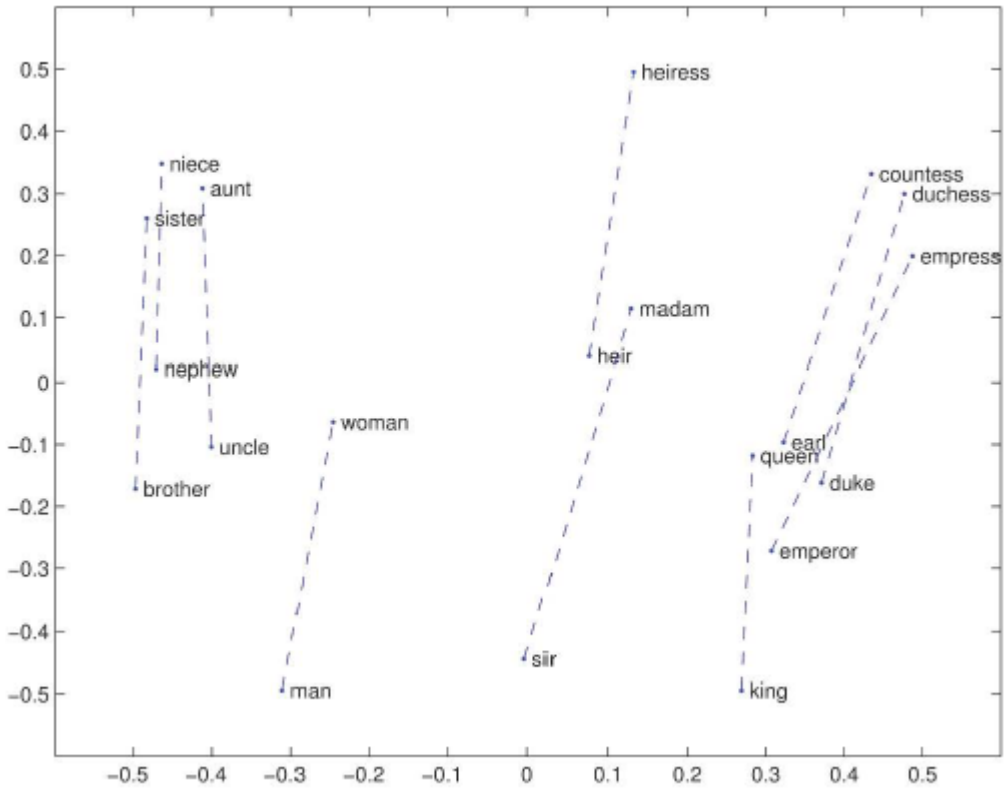
leptodactylidae

rana

eleutherodactylus

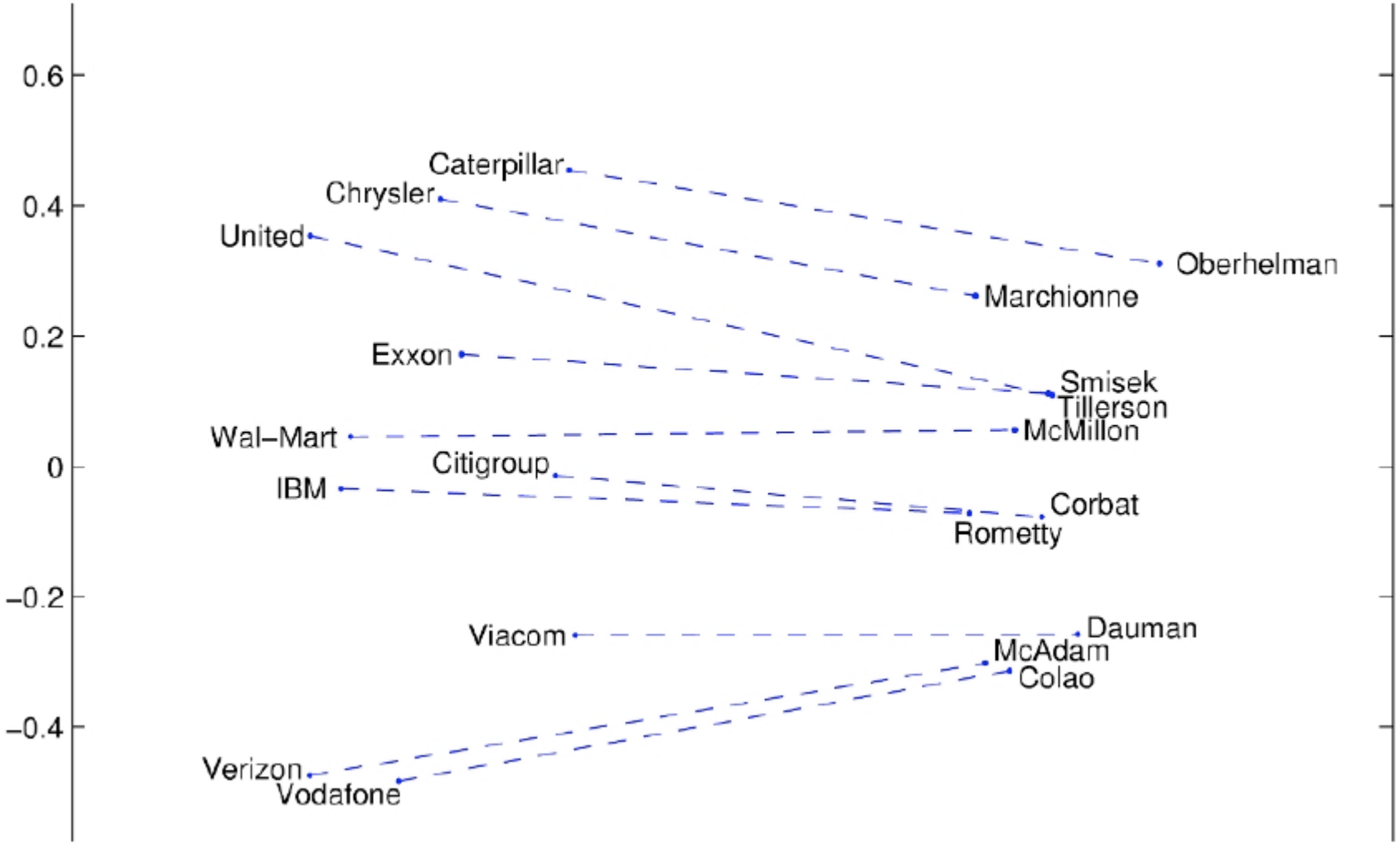
GloVe可视化

- GloVe线性关系：亲属[Pennington et al., 2014]



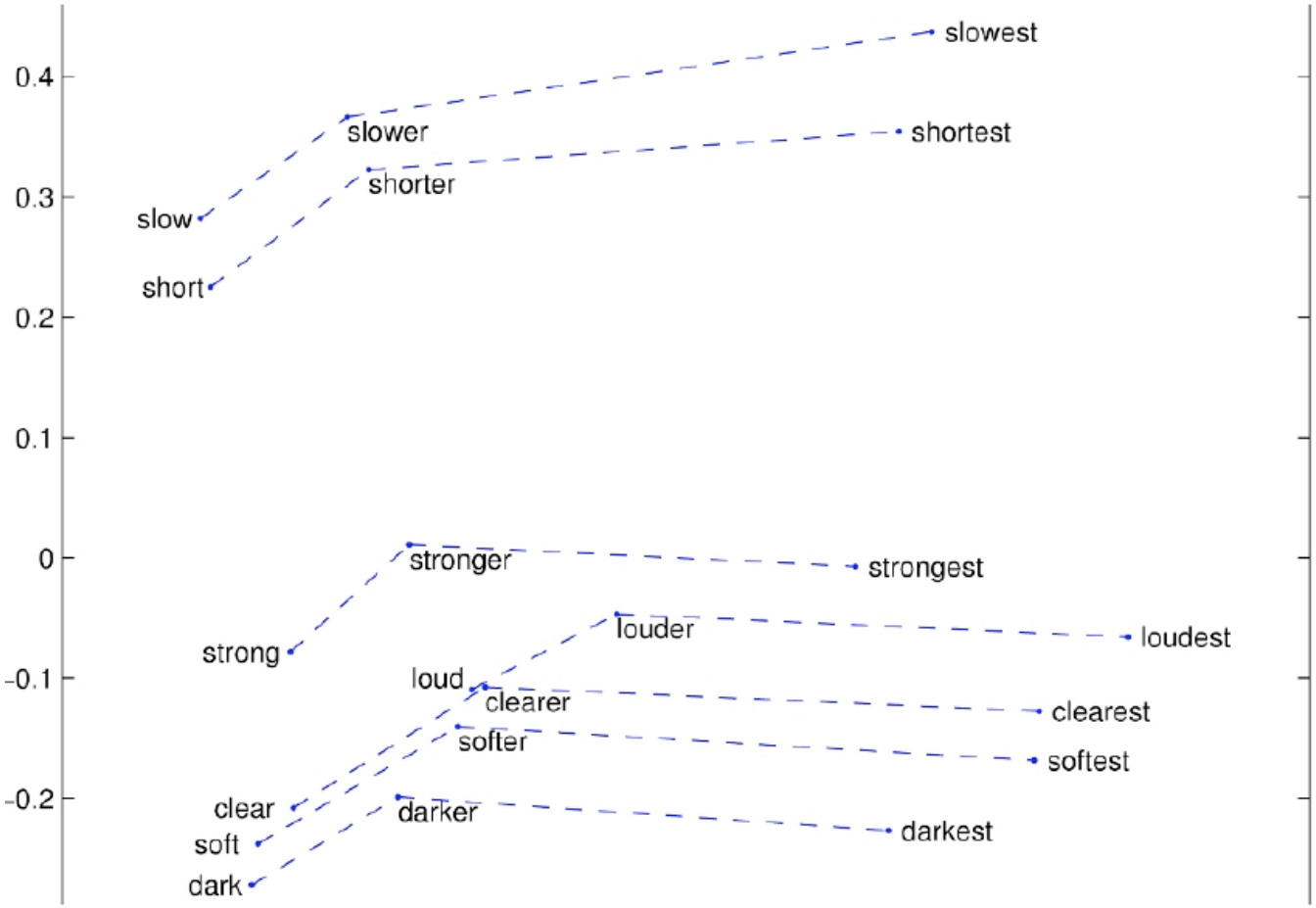
GloVe可视化

- GloVe线性关系：公司与CEO[Pennington et al., 2014]



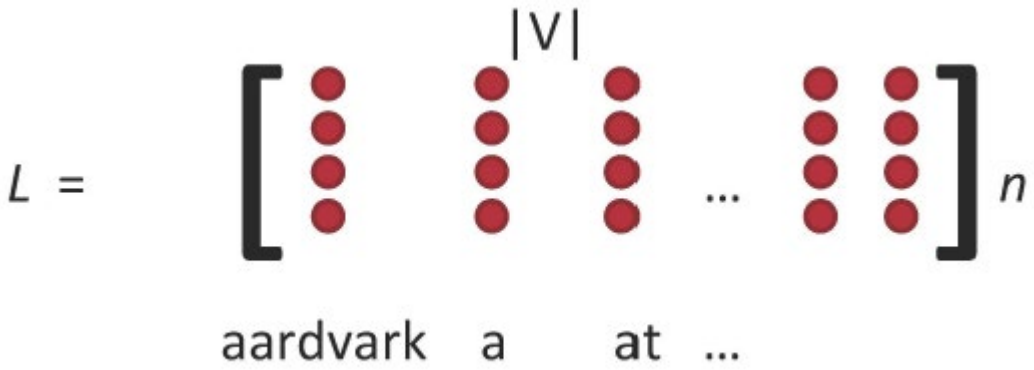
GloVe可视化

- GloVe线性关系：形容词-比较级-最高级[Pennington et al., 2014]



词向量的用途

- **代替原有的one-hot作为模型的输入**
  - 可以在大语料上预训练，获得有用的信息，对模型有益
- **embedding matrix**



$$L \in \mathbb{R}^{n \times |V|}$$

- **也叫look-up table**
  - 概念上，通过L左乘一个One-hot向量e(of length |V|)就可以得到一个词向量

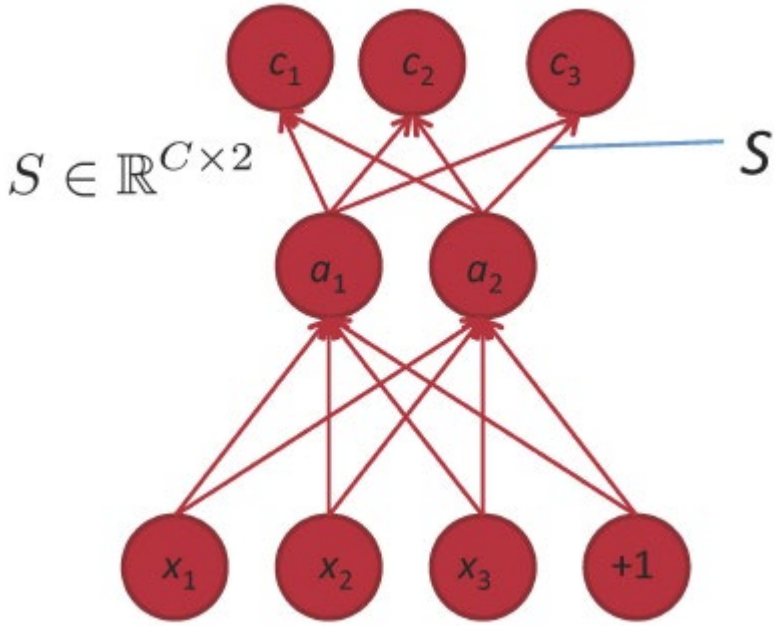
$$x = Le$$

词向量的用途

- **优势：**
  - 可以通过反向传播在训练过程中更新
  - 可以通过神经网络传播任何信息到词向量

$$P(c \mid d, \lambda) = \frac{e^{\lambda^T f(c,d)}}{\sum_{c'} e^{\lambda^T f(c',d)}}$$

$$p(c|x) = \frac{\exp(S_c.a)}{\sum_{c'} \exp(S_{c'}.a)}$$



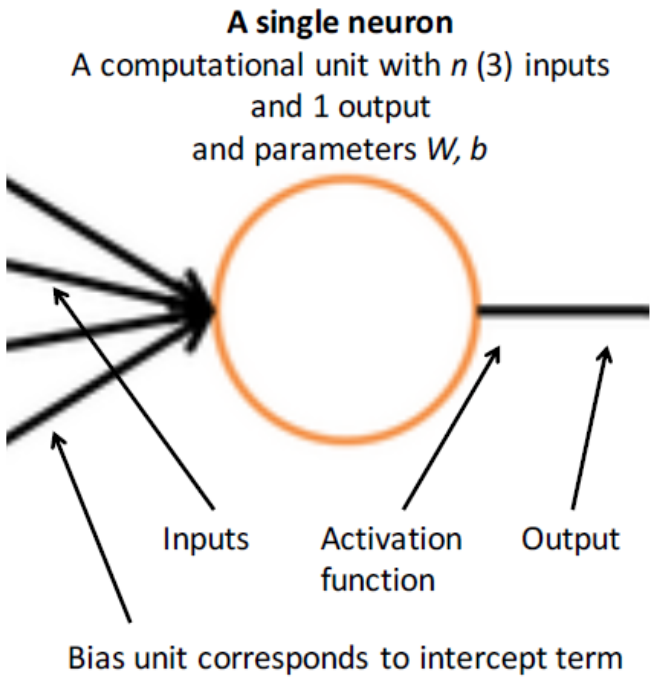
大纲

- **序列标注问题**
- **传统方法**
  - 简单分类
  - 标签偏置问题
  - HMM、结构化感知器
- **深度学习方法**
  - 词向量
  - 全连接神经网络 
  - 循环神经网络

简介神经网络

- **神经网络有一套自己的术语**
  - 正如SVMs

- **如果你理解Logistic regression模型的原理**
  - 你**已经理解**一个神经网络中的神经元的操作



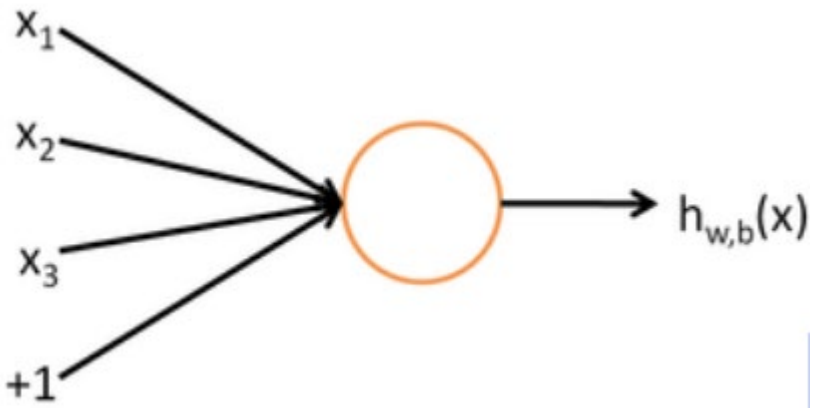
神经元：一个logistic regression单元

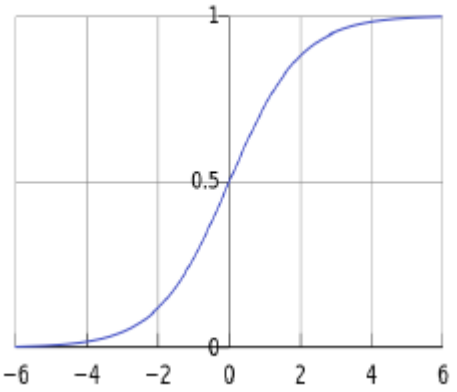
- **神经元：**
  - 一个logistic regression单元
  - 一个非线性激活函数

$$h_{w,b}(x) = f(w^{\mathsf{T}}x + b)$$

b: We can have an "always on" feature, which gives a class prior, or separate it out, as a bias term

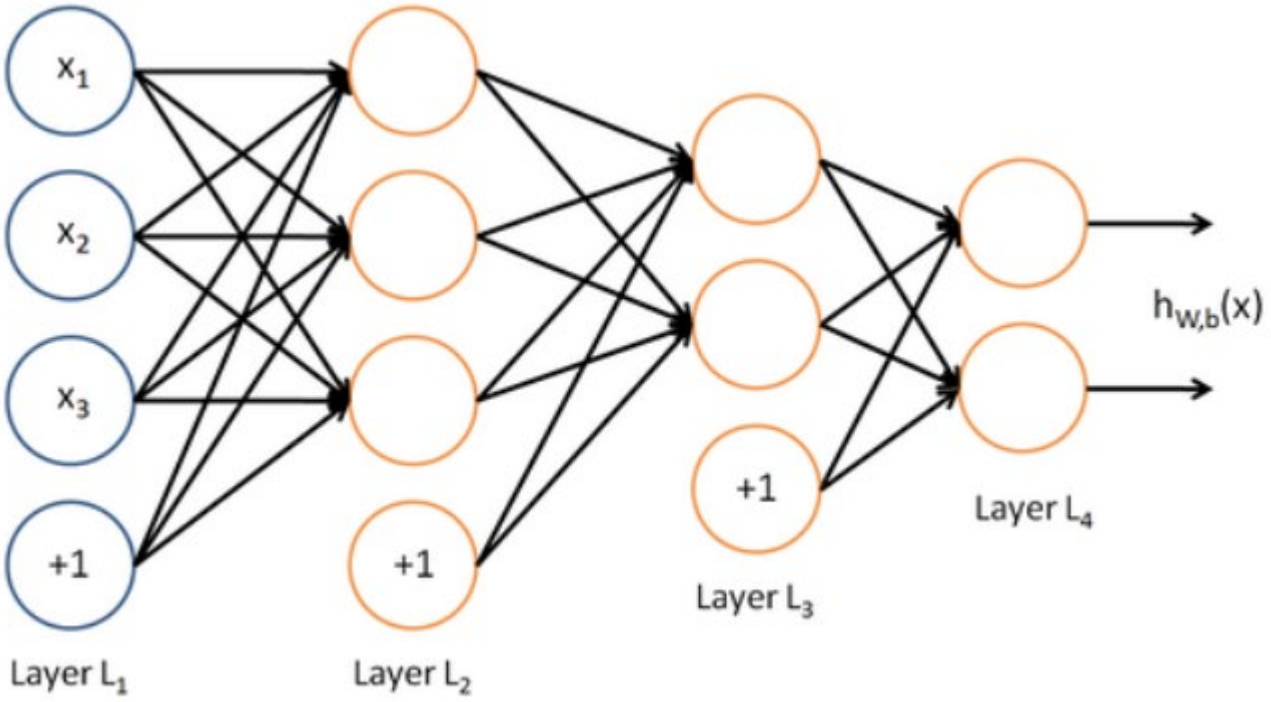
$$f(z) = \frac{1}{1+e^{-z}}$$





w, *b* are the parameters of this neuron i.e., this logistic regression model

- 多叠加几次，我们就有了一个多层神经网络



神经网络：矩阵表示

We have

$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$

$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$

etc.

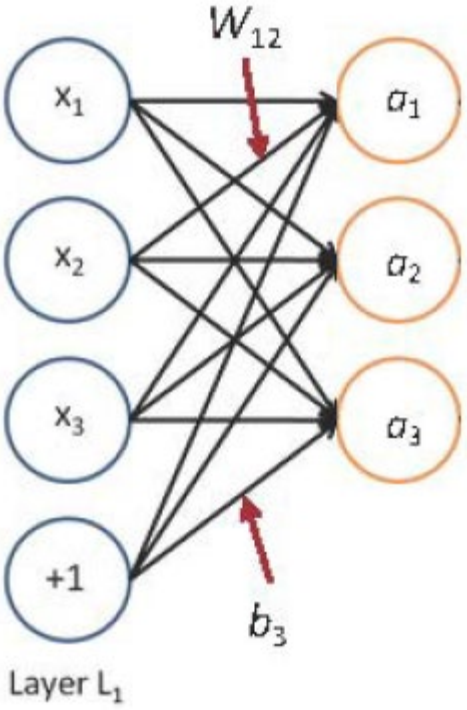
In matrix notation

$$z = Wx + b$$

$$a = f(z)$$

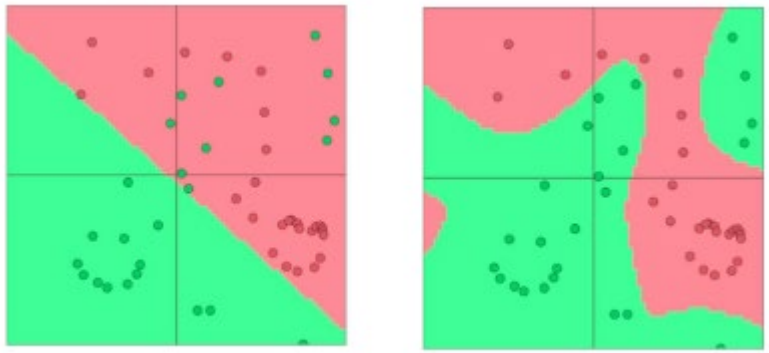
where f is applied element-wise:

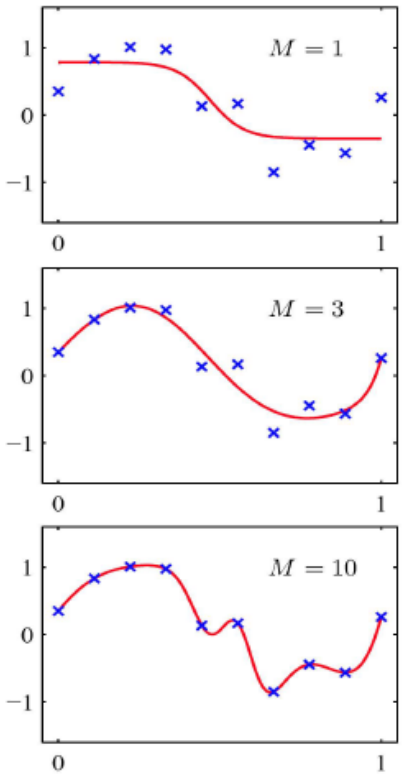
$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$



神经网络：非线性

- **例子：函数拟合，如回归或分类**
  - 没有非线性，深度神经网络只能进行线性变换
  - 更多的层数可以直接由一个单一的线性变换完成
  - 层数更多，他们可以拟合更复杂的函数





- **一层是一个线性层和非线性的组合：前馈计算**

$$z = Wx + b$$
$$a = f(z)$$

- **这样的一层有很多名字**
  - 全连接层、前馈层
- **这样的网络也有很多名字**
  - 前馈神经网络、全连接网络、多层感知器
- **神经元最后的激活值a可以进一步用来计算**
  - 例如，softmax概率或者一个未归一化的分数

$$score(x) = U^T a \in \mathbb{R}$$

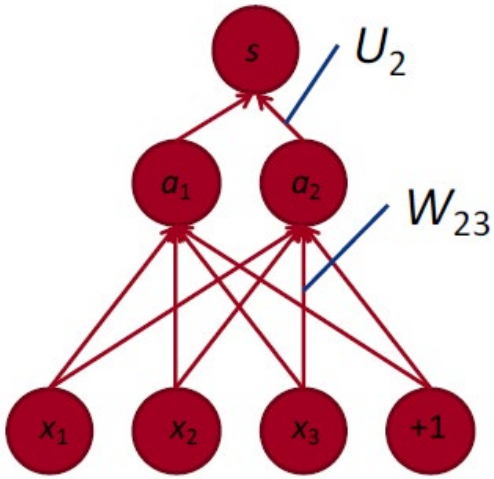
反向传播(Backpropagation)

- **我们计算了所有变量的梯度**
  - U, W, b
- **这基本就是反向传播**
  - 算导数、用chain-rule!
- **剩下的窍门**
  - 算低层梯度时复用高层梯度

$$s = U^T f(Wx + b)$$

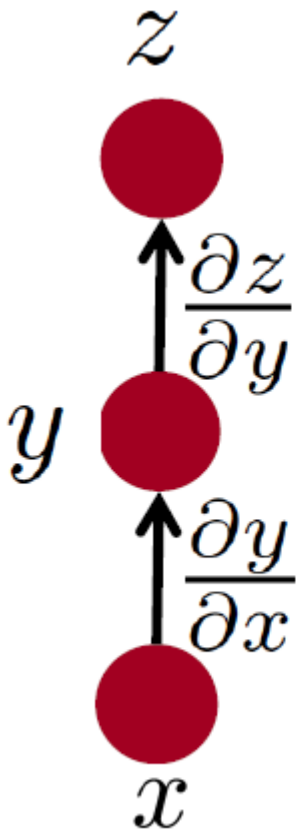
$$s_c = U^T f(Wx_c + b)$$

- **例子**
  - 模型最后一个导数：x中的词向量



反向传播：另一种解释

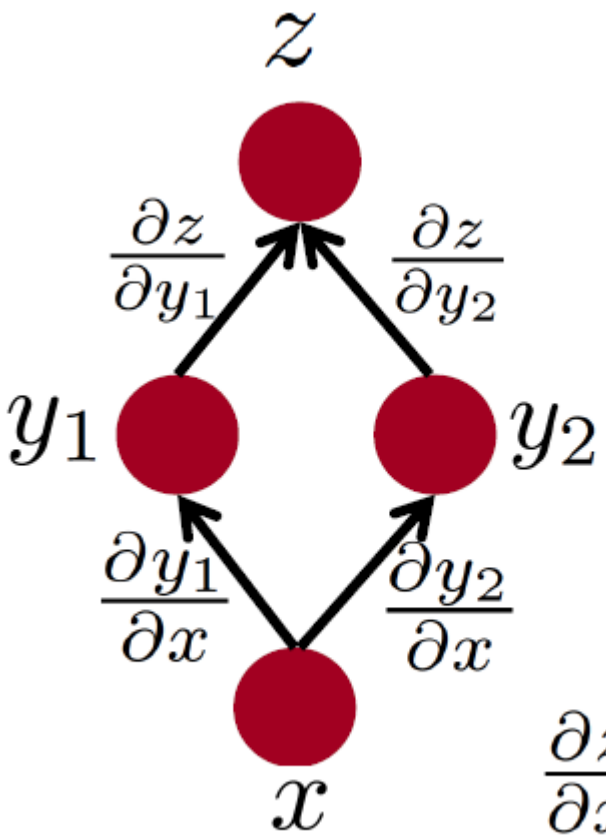
- 简单路径的链式法则



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

反向传播：另一种解释

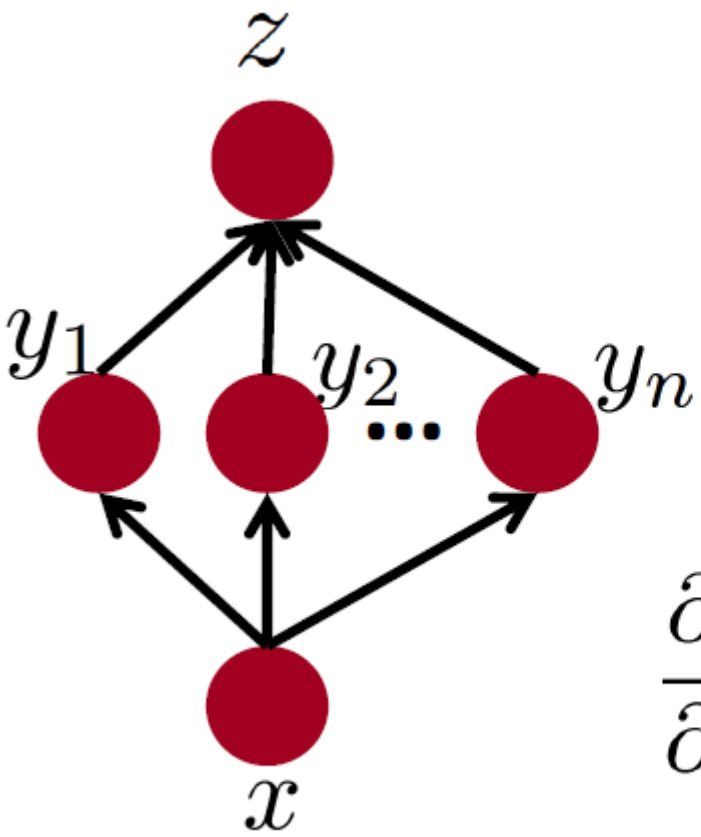
- 两条路径的链式法则



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

反向传播：另一种解释

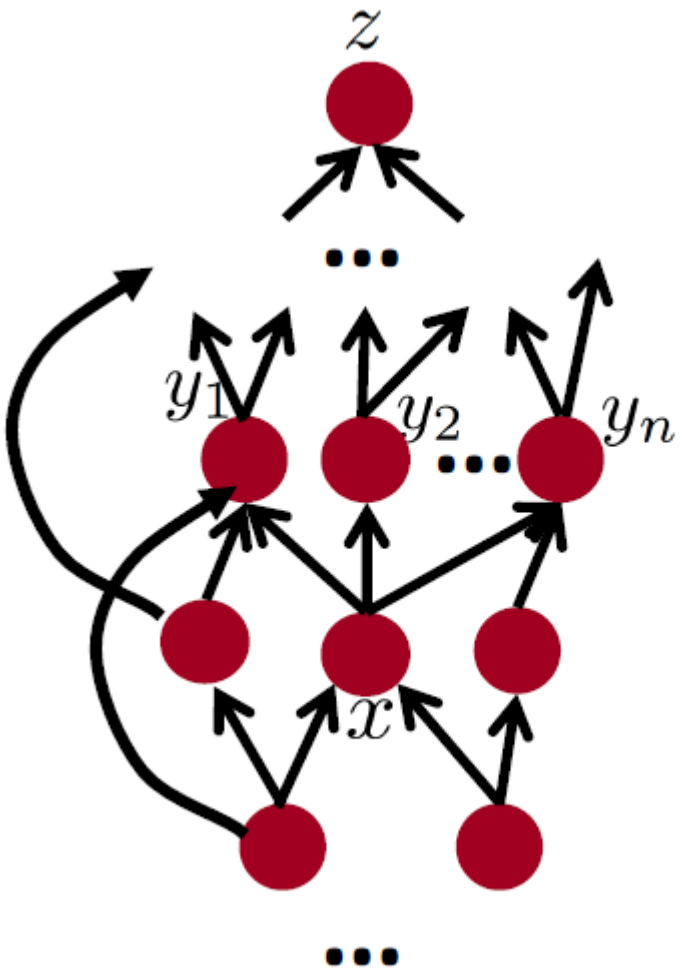
- 多条路径的链式法则



$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

反向传播：另一种解释

- 计算流图中的链式法则

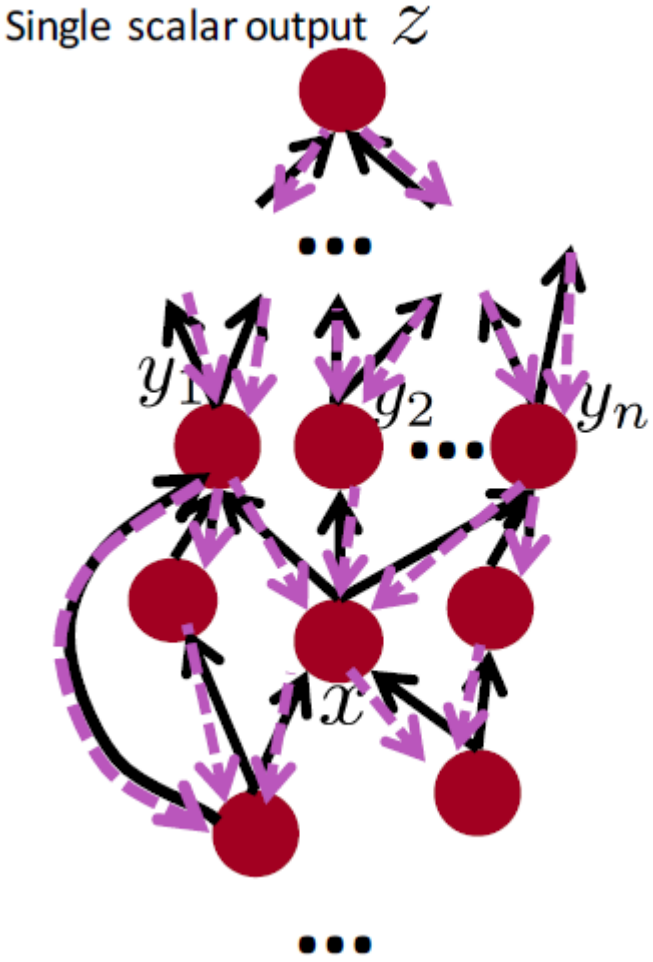


$\{y_1, y_2, \ldots y_n\}$ = successors of x

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

反向传播：另一种解释

- 多层神经网络的反向传播



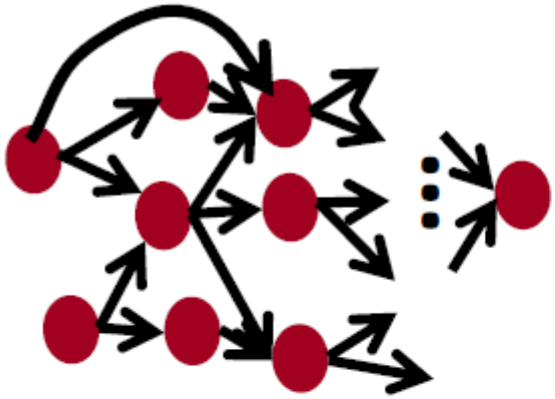
- 前向传播
  - 按照拓扑序计算每个节点
- 反向传播
  - 最终的梯度为1
  - 按照拓扑序的反序计算每个节点的梯度

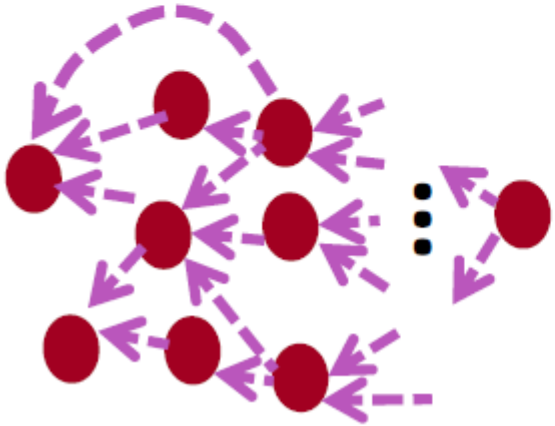
$\{y_1, y_2, \ldots y_n\}$ = successors of x

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

反向传播：另一种解释

- 自动求导





- The gradient computation can

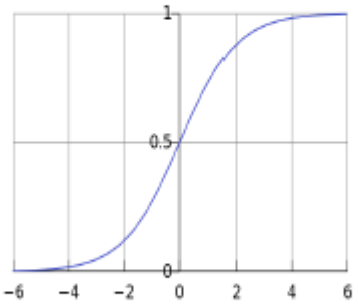
- 根据前向的计算，可以自动推断对应的梯度计算
- 每个节点规定
  - 如何根据输入求得输出
  - 如何根据输出的梯度求得输入的梯度
  - 对于原型设计而言是简单且迅速的

提示与技巧：非线性函数

- **最早使用的非线性函数**

logistic ("sigmoi

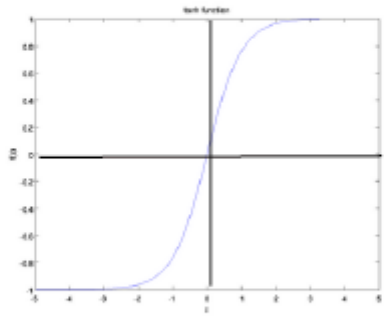
$$f(z) = \frac{1}{1 + \exp(-}$$



$$f'(z) = f(z)(1 - .$$

tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$



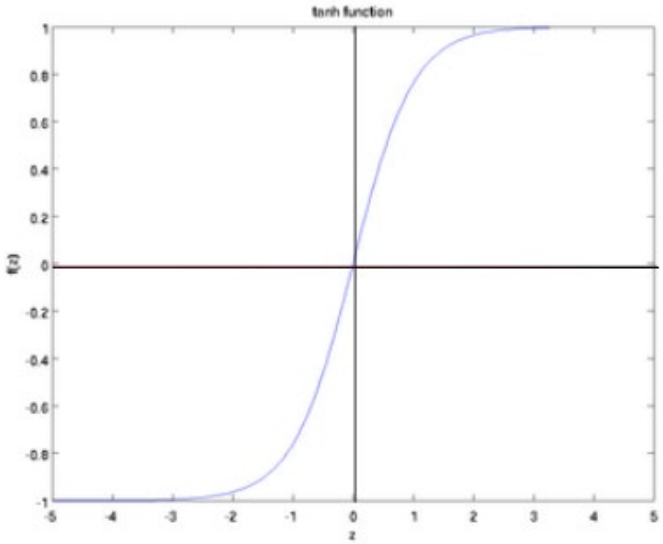
$$f'(z) = 1 - f(z)^2$$

tanh is just a rescaled and shifted sigmoid

$$\tanh(z) = 2\text{logistic}(2z) - 1$$

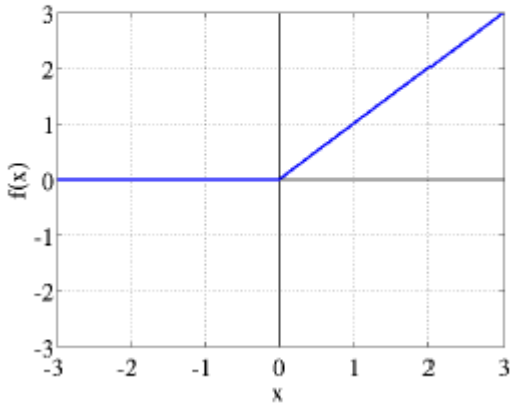
提示与技巧：非线性函数

- **tanh vs. sigmoid**
  - 初始化：接近0
  - 实际中更快收敛
  - 漂亮的导数
    - $f'(z) = 1 - \tanh^2(z)$



- **目前最广泛：rectifier**
  - 使用rectifier的神经元一般称为Rectified Linear Unit (ReLU)
  - 极低的计算量
  - 实际中在视觉领域效果更好

$$\mathrm{rect}(z) = \max(z, 0)$$



提示与技巧：参数优化

- **参数优化**
  - 梯度下降：每次更新使用所有样例的总梯度
  - 随机梯度下降：每次只使用一个或几个样例
- **一般的更新规则：** $\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J_t(\theta)$
  - J是当前样例的损失/代价函数， θ是参数， α是学习率
- **普通梯度下降法是全批量方法**
  - 非常慢，勿用。使用2阶的批量方法，如L-BFGS。
- **随机梯度下降法(SGD) vs 全批量梯度下降法(GD)**
  - 在大规模数据上，SGD通常胜过所有GD方法
  - 在一些较小数据上，L-BFGS或Conjugate Gradients更好

提示与技巧：参数优化

- **SGD的改进：Momentum**
  - 增加一部分之前的更新量到当前更新
  - 当梯度一直指向同样的方向时，这会加快到达极小点的步伐
  - 当动量很大时，减小学习率

- **更新规则：**

$$v = \mu v - \alpha \nabla_\theta J_t(\theta)$$

$$\theta^{new} = \theta^{old} + v$$

  - v初始为0，一般μ取0.9
  - 动量过几轮后通常会增大(0.5->0.99)

- **直观理解Momentum**
  - 增加部分(momentum misnomer)
  - 参数修改会形成在持续梯度方向的速度

- **简单凸函数的优化过程**

without momentum

with momentum:

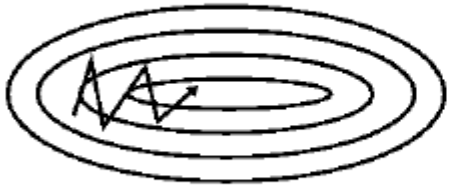


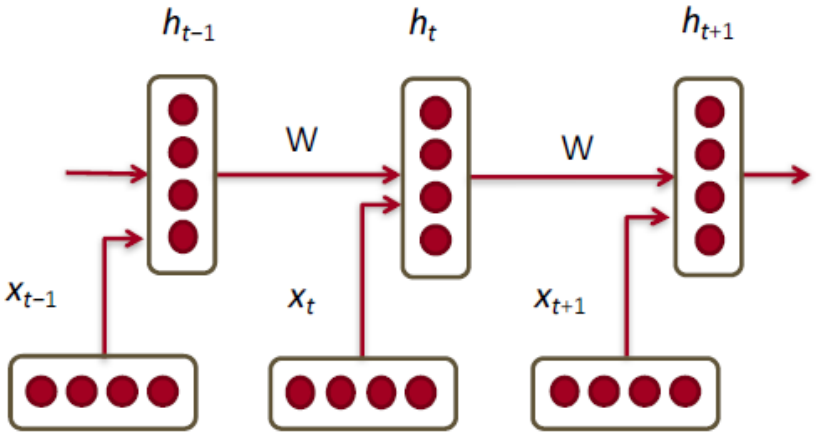
Figure from https://www.willamette.edu/~gorr/classes/cs449/momrate.html

大纲

- **序列标注问题**
- **传统方法**
  - 简单分类
  - 标签偏置问题
  - HMM、结构化感知器
- **深度学习方法**
  - 词向量
  - 全连接神经网络
  - 循环神经网络 

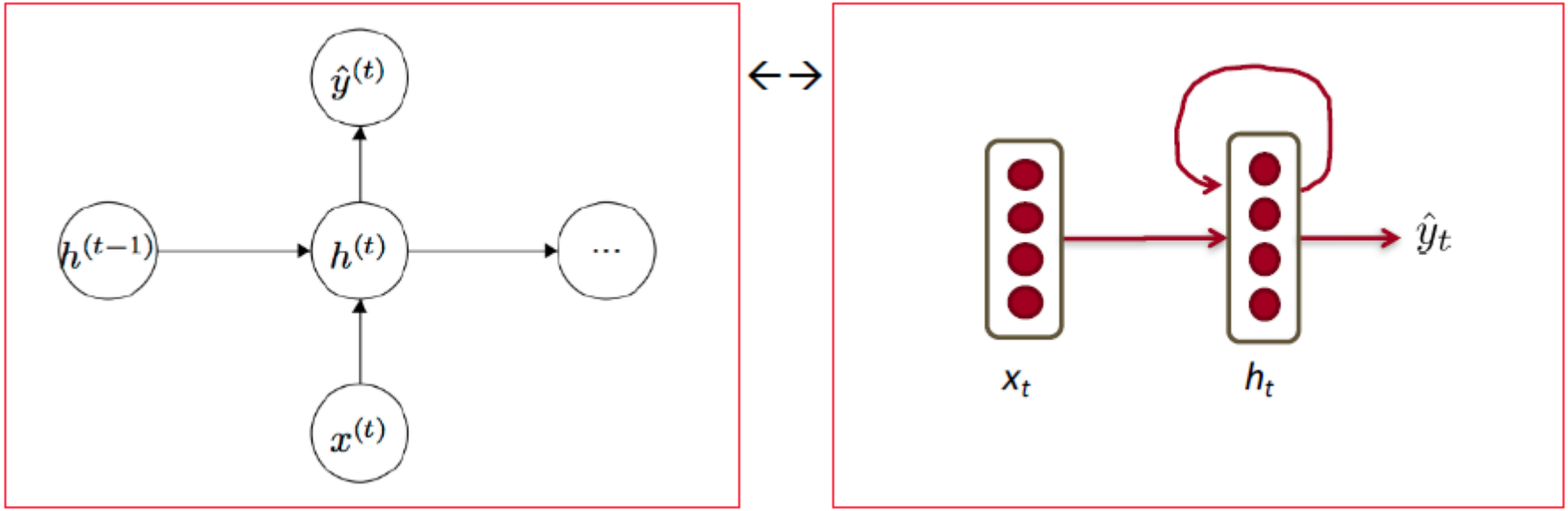
循环神经网络

- **循环神经网络 (recurrent neural network)**
  - 循环：每个时间点使用**相同的参数**
  - 输入增加之前的网络输出
    - 让神经网络以之前所有的词为条件



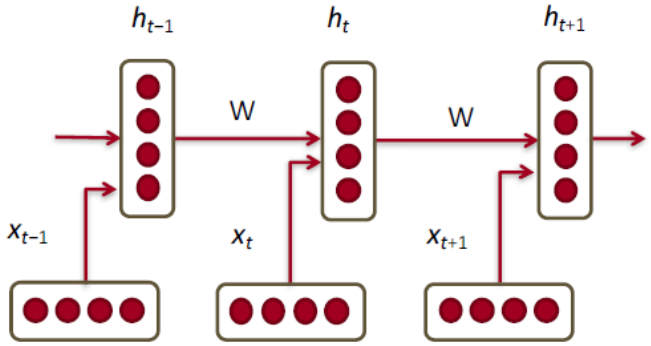
- **一个词的计算包含之前所有词的信息！**

- 利用RNN进行分类
  - 给定一个词向量序列$x_1, \dots, x_{t-1}, x_t, x_{t+1}, x_T$
  - 在每个时刻t计算
    - $h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$
    - $\hat{y}_t = softmax(W^S h_t)$
    - $\hat{P}(x_{t+1} = v_j | x_t, \dots, x_1) = \widehat{y_{t,j}}$



循环神经网络

- **利用RNN进行分类**
  - $h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$
  - $\hat{y}_t = softmax(W^S h_t)$
  - $\hat{P}(x_{t+1} = v_j | x_t, \dots, x_1) = \widehat{y_{t,j}}$



- **参数含义**
  - 每个时刻使用相同的W
  - x_t是t时刻时的输入对应的词向量
  - h_t是t时刻时的隐层表示
    - 需要使用h_0，可以初始化为0向量，也可以作为参数学习
  - $\hat{y} \in \mathbb{R}^{|V|}$ 是一个词典中词的概率分布

序列标注任务

- **把词标注为**
  - 词类(Part-of-speech Tagging)
  - 命名实体(Named Entity Recognition)
  - 实体级别的情感（上下文中）(Sentiment Analysis)
  - 表述是否含有某种观点 (Opinion Mining)

- **下面以Opinion Mining为例**
  - Example application and slides from paper *Opinion Mining with Deep Recurrent Nets* by Irsoy and Cardie 2014

序列标注任务

- **目标：把每个词分类为**
  - *直接主观表述：direct subjective expressions* (DSEs)
  - *间接主观表述：expressive subjective expressions* (ESEs)

- **DSE: Explicit mentions of private states or speech events expressing private states**
  - 直接说了观点
- **ESE: Expressions that indicate sentiment, emotion, etc. without explicitly conveying them**
  - 间接提及了观点

序列标注任务

- 使用BIO标签

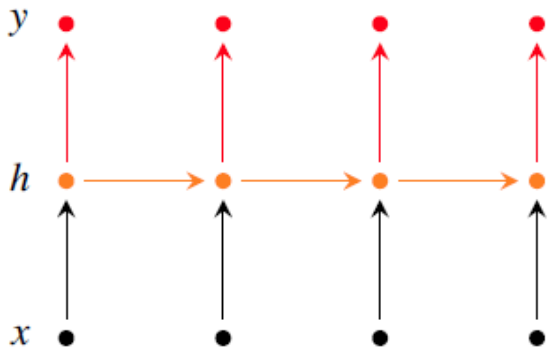
The committee, [as usual]$_{ESE}$, [has refused to make any statements]$_{DSE}$.

The	committee	,	as	usual	,	has
O	O	O	B_ESE	I_ESE	O	B_DSE

refused	to	make	any	statements	.
I_DSE	I_DSE	I_DSE	I_DSE	I_DSE	O

序列标注任务

- **模型：循环神经网络**

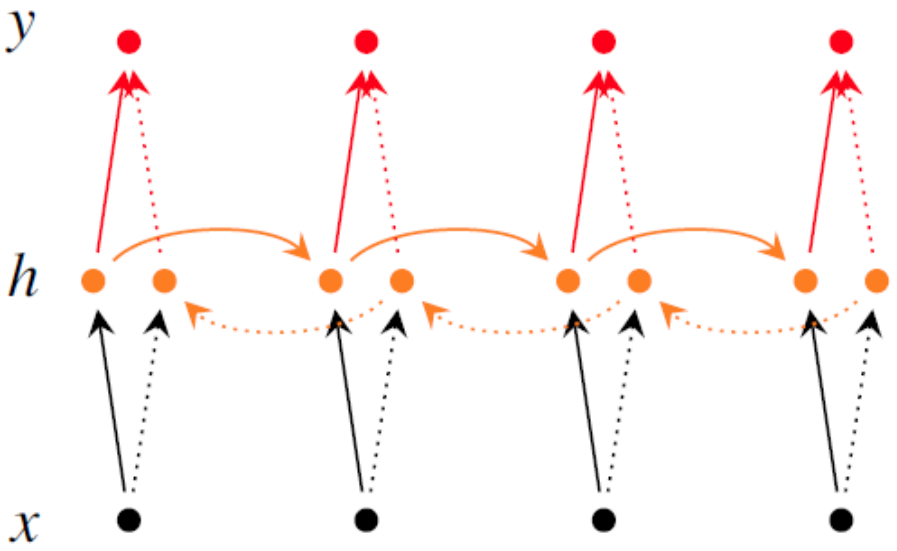


$$h_t = f(Wx_t + Vh_{t-1} + b)$$
$$y_t = g(Uh_t + c)$$

- ***x*是输入的词向量**
- ***y*表示输出标签(B, I or O)**
  - g = softmax
- **h是隐层表示**
  - 由过去的表示和当前的词得到
  - 记录句子到此时刻的信息

序列标注任务

- 问题：对于分类任务，之前和之后的词可能都有用
- 方法：双向循环神经网络(Bidirectional RNN)



$$\vec{h}_t = f(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b})$$

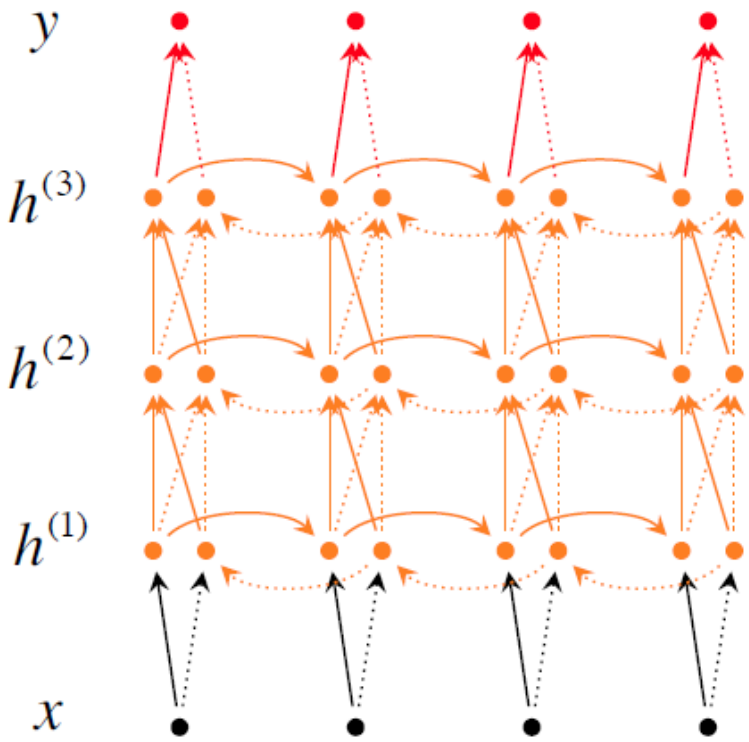
$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\vec{h}_t; \overleftarrow{h}_t] + c)$$

$h = [\vec{h}; \overleftarrow{h}]$ now represents (summarizes) the past and future around a single token.

序列标注任务

- 深度双向循环神经网络(Deep Bidirectional RNN)



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Each memory layer passes an intermediate sequential representation to the next.

序列标注任务

- **数据：**
  - MPQA 1.2 corpus (Wiebe et al., 2005)
  - 包含535篇新闻，共11111个句子
  - 人工标注了短语级别的DSE和ESE

- **评价: F1**

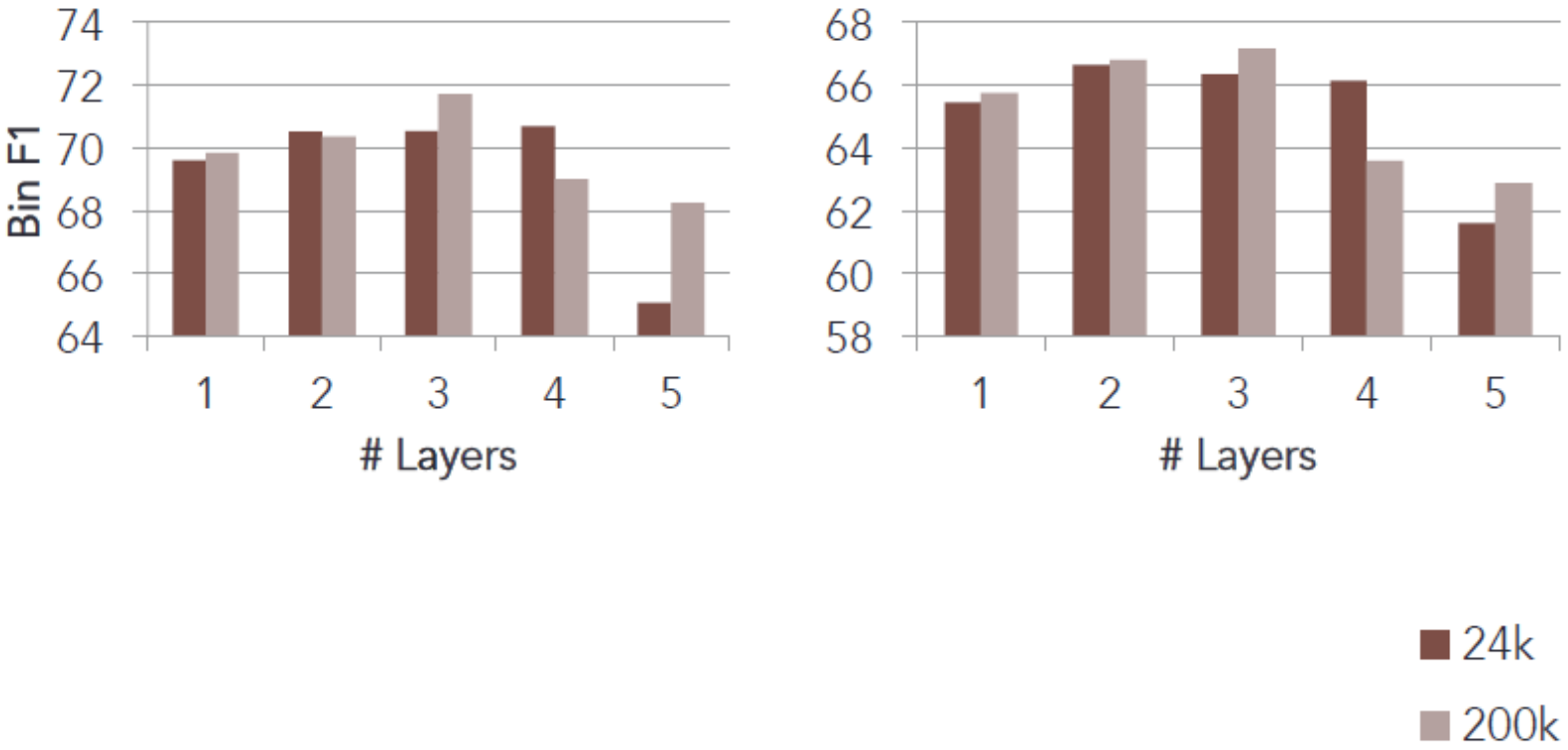
$$\text{precision} = \frac{tp}{tp + fp}$$

$$\text{recall} = \frac{tp}{tp + fn}$$

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

序列标注任务

- 结果


Bin F1
74
72
70
68
66
64
1
2
3
4
5
Layers
68
66
64
62
60
58
1
2
3
4
5
Layers
24k
200k

THANKS!