

第八章 序列标注

- 1 序列标注问题
- 2 马尔可夫模型
- 3 隐马尔可夫模型
- 4 条件随机场 (自学)
- 5 命名实体识别
- 6 总结



1 序列标注问题

- **序列标注** (tagging) 指的是给定一个序列 $\mathbf{x} = x_1 x_2 \dots x_n$, 找出序列中每个元素对应标签 $\mathbf{y} = y_1 y_2 \dots y_n$ 的问题。
- 其中, y 所有可能的取值集合称为**标注集S** (tagset) 。

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

$$\mathbf{x}_i = (\mathbf{x}_{i1}, \mathbf{x}_{i2}, \dots, \mathbf{x}_{iT})$$

$$\mathbf{y}_i = (\mathbf{y}_{i1}, \mathbf{y}_{i2}, \dots, \mathbf{y}_{iT})$$

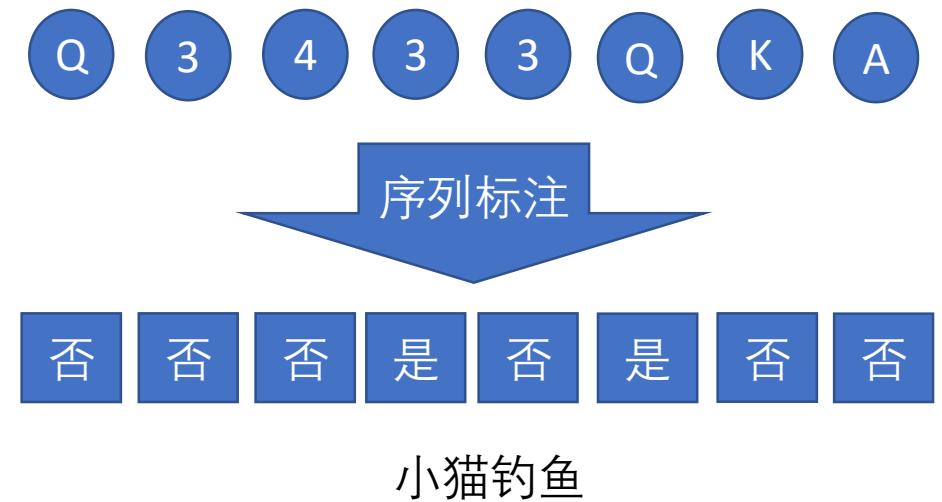
$$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\}$$

$$\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N\}$$



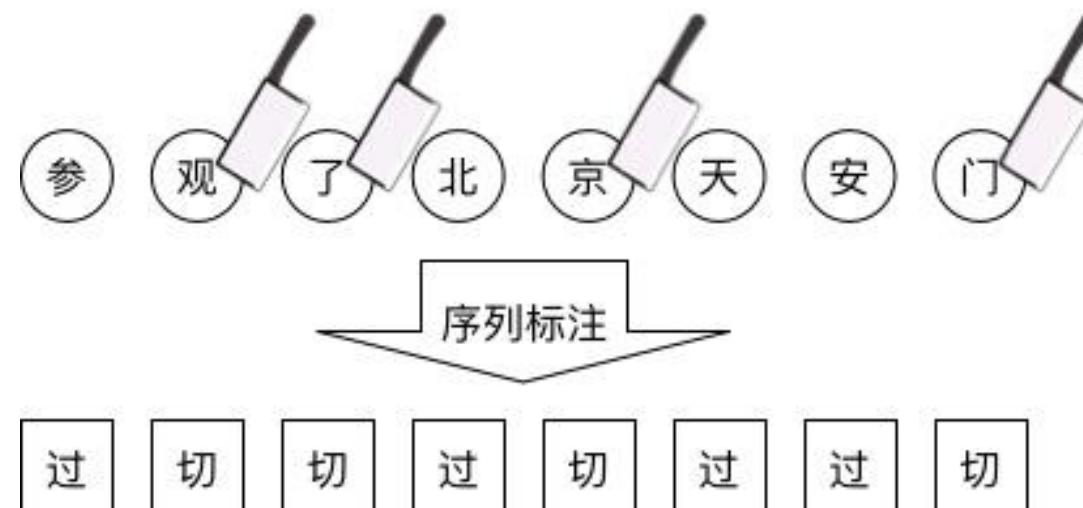
1 序列标注问题

- 许多应用任务都可以变换思路，转化为序列标注问题来解决。



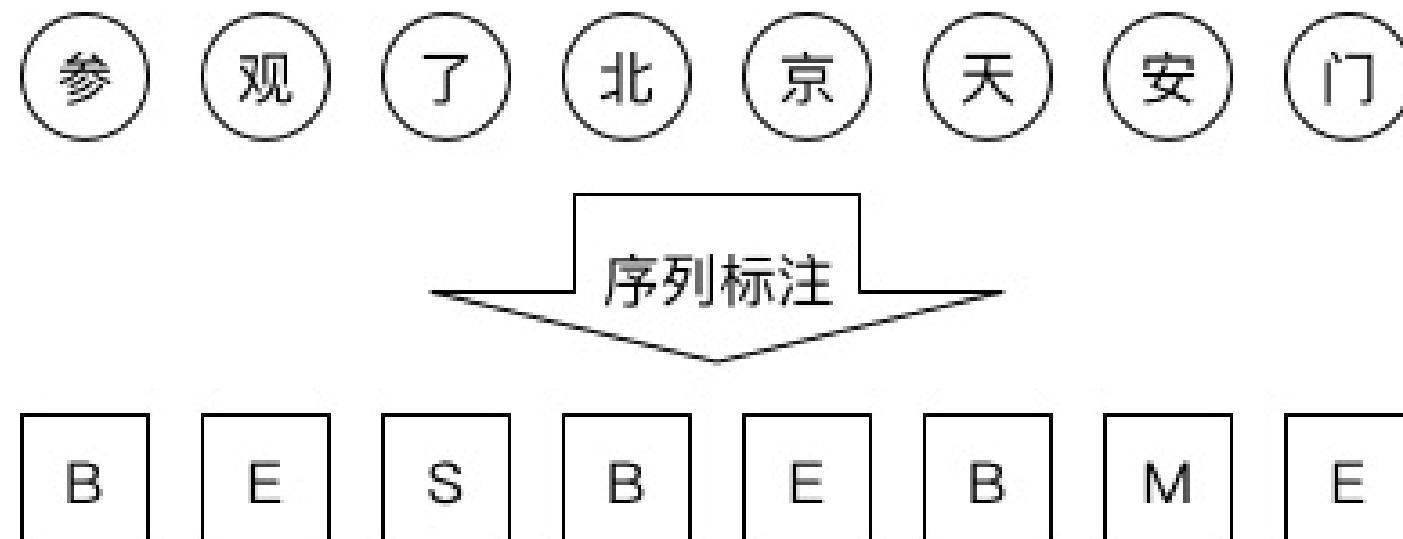
1. 1 序列标注与中文分词

- 中文分词转化为标注集为 {切, 过} 的序列标注问题



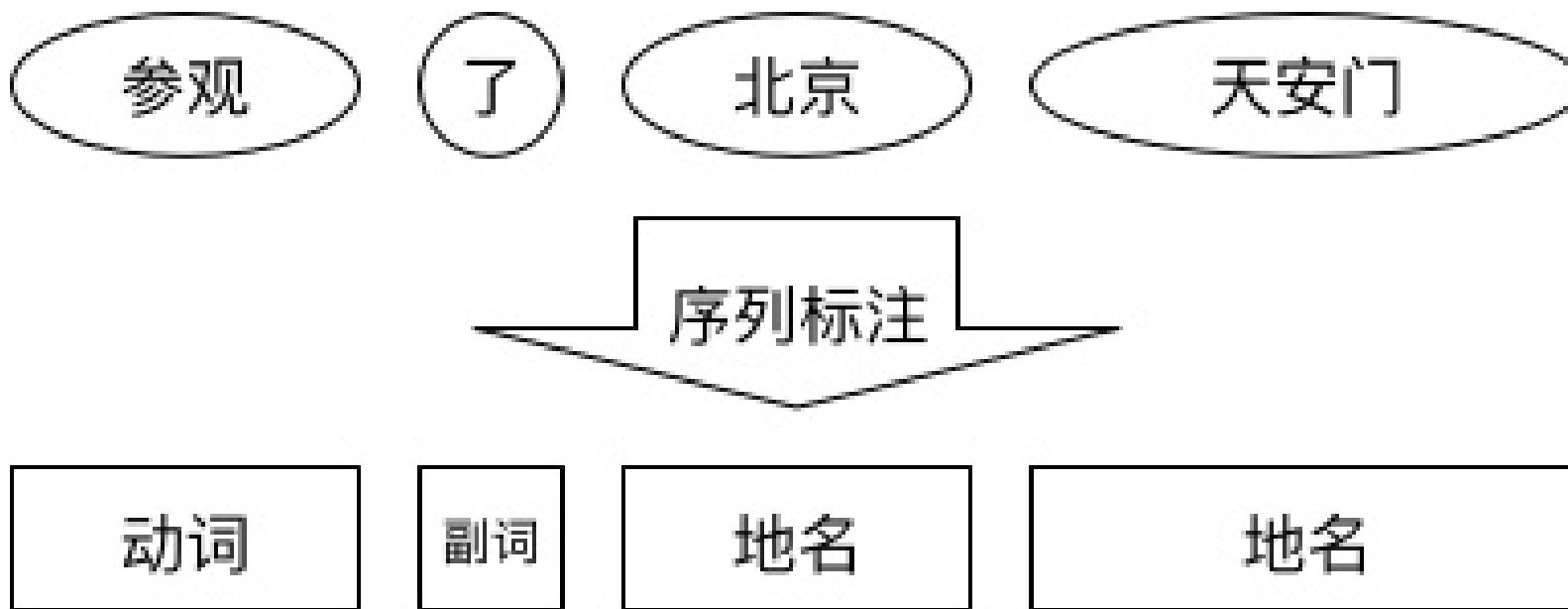
1. 1 序列标注与中文分词

- 人们提出 {B, M, E, S} 这种最流行的标注集
 - 汉字分别作为词语首尾（Begin、End）、词中（Middle）以及单字成词（Single）



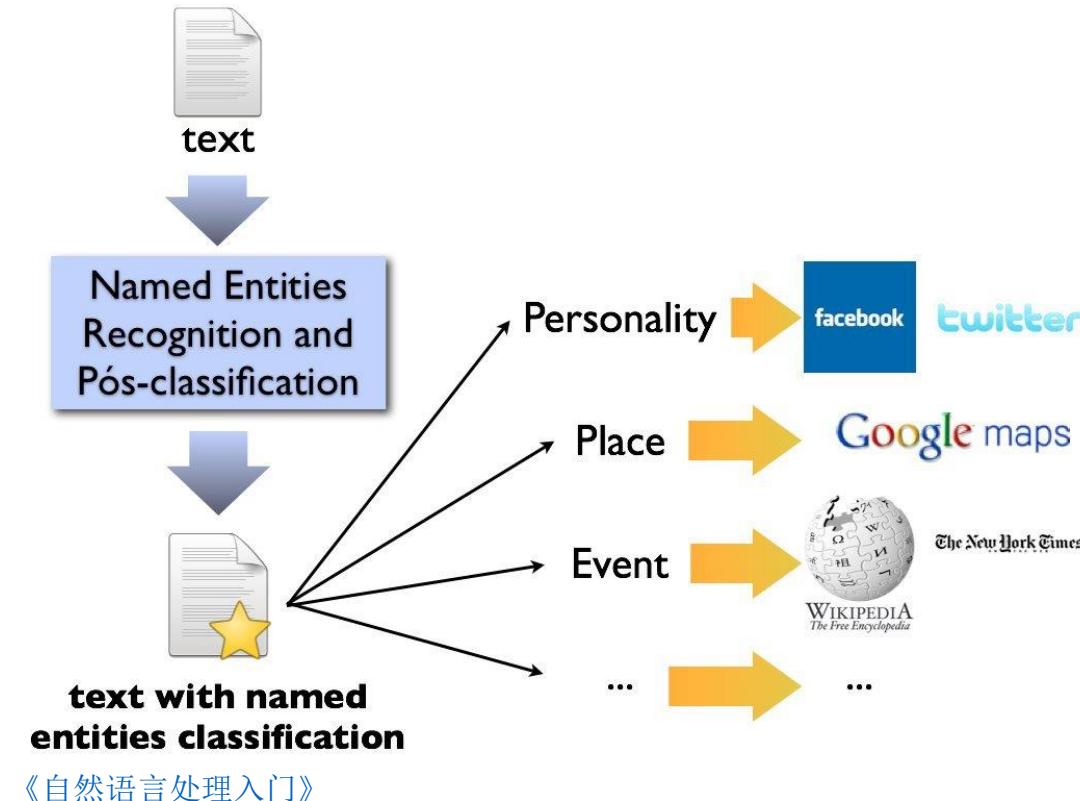
1.2 序列标注与词性标注

- 词性标注是个天然的序列标注问题。最著名的词性标注集当属863标注集和北大标注集。



1.3 序列标注与命名实体识别

- 文本中有一些描述实体的词汇，比如人名、地名、组织机构名、股票基金、医学术语等，称为命名实体（named entity）。
 - 数量无穷
 - 构词灵活
 - 类别模糊



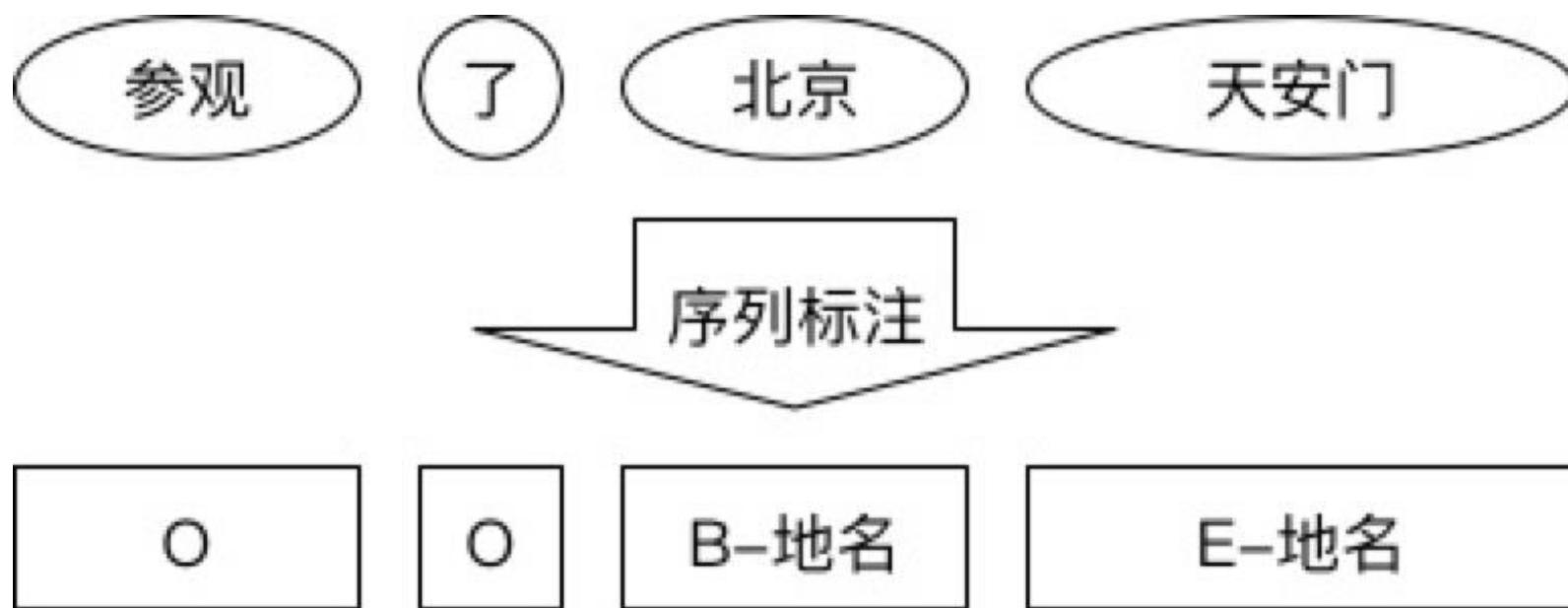
命名实体识别

- 识别出句子中命名实体的边界与类别的任务称为命名实体识别（Named Entity Recognition，NER）
 - 对于规则性较强的命名实体，比如邮箱、ISBN、商品编号，完全可以通过正则表达式处理。
 - 对于较短的命名实体，比如人名，完全可以通过分词确定边界，通过词性标注模块确定类别。
 - 在另一些语料库中（如PKU等），机构名这样的复合词是拆开的，此时就需要一个专门的命名实体识别模块了。



1.3 序列标注与命名实体识别

- 命名实体识别可以转换为一个序列标注问题，可以采用 {B, M, E, O, S} 标注集。
- 地名和机构名常常由多个单词组成（称为复合词）

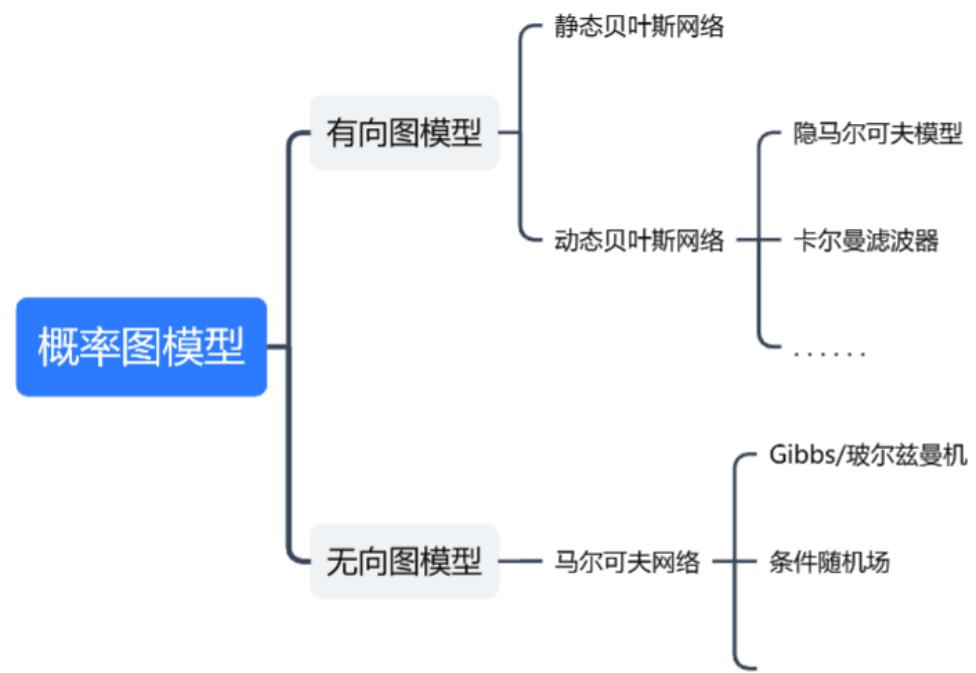


2 隐马尔可夫模型

- 隐马尔可夫模型是关于时序的概率图模型；
- 描述由一个~~隐藏~~的马尔可夫链随机生成不可~~观测~~的状态随机序列(state sequence)，再由各个状态生成一个观测而产生~~观测~~随机序列(observation sequence)的过程，序列的每一个位置又可以看作是一个时刻。

概率图模型

- 概率图模型（Probabilistic Graphical Model, PGM）是一种基于概率理论、使用图论方法来表示概率分布的模型。
- 图模型可以表示为： $G=(V, E)$ 。



Andrey Markov

- 中文名 马尔科夫
- 国 籍 俄国
- 出生地 梁赞
- 出生日期 1856年6月14日
- 逝世日期 1922年7月20日
- 主要成就 开创了[随机过程](#)这个新领域



马尔可夫模型几个重要的概念

- 马尔可夫性
- 马尔科夫过程
- 马尔科夫链
- 马尔可夫模型

马尔可夫性质

- 随机过程是随机变量的集合，其在随机变量的基础上引入时间的概念（可简单理解为随机变量是关于时间的函数）。
- 马尔可夫性质。当一个随机过程在给定现在状态及所有过去状态的情况下决定当前状态。

现在状态 $f(n - 1)$, 未来状态 $f(n)$



- 一般来说
- 马尔可夫

已知现在状态及所有过去状态，未来状态的条件概率分布仅依赖于现在状态。

具备记忆特质的。
态，而与过去无关。

现在决定未来



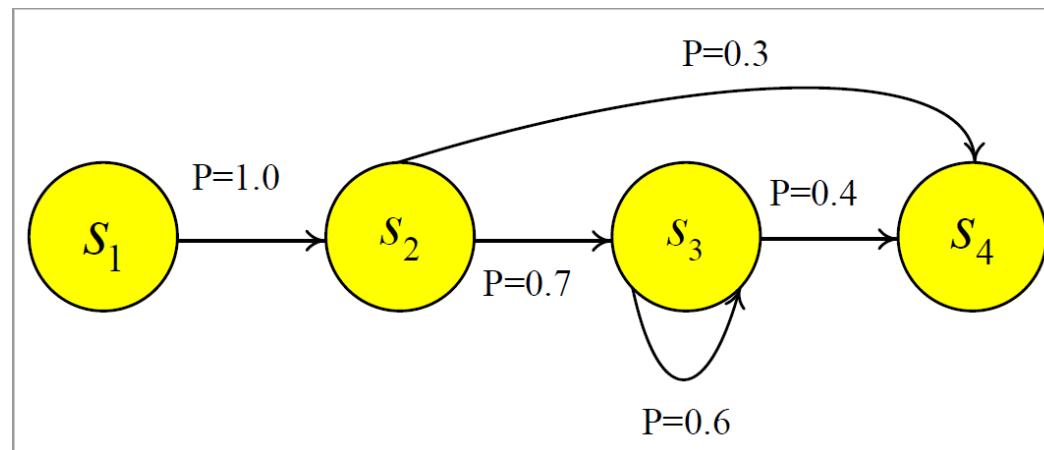
机器学习NLP小白的成长之路

马尔可夫过程和马尔可夫模型

- 马尔科夫过程是一类具有马尔可夫性的随机过程。马尔科夫过程是一个状态不断演变的过程，对其进行建模后称之为马尔科夫模型。
- 当一个马尔科夫过程中的每个状态的转移只依赖于之前的 n 个状态，对这个过程进行建模，称之为 n 阶马尔科夫模型，其中 n 是影响转移状态的数目。最简单的马尔科夫过程就是一阶过程，每一个状态的转移只依赖于其之前的那一个状态，对其建模得到一阶马尔可夫模型。
- 马尔可夫模型描述了一类重要的随机过程：对于一个随机变量序列，序列中各随机变量并不是相互独立的，每个随机变量的值可能会依赖于之前的序列状态。

马尔科夫链

- 具备离散状态的马尔可夫过程，通常被称为马尔可夫链。
- 给定时间线上有一串事件顺序发生，假设每个事件发生的概率只取决于前一个事件，那么这串事件构成的因果链被称作为马尔科夫链。
- 在NLP语境下，可以将事件具象为单词，于是一阶马尔可夫模型就具象为二元语言模型（2-gram LM）。



马尔可夫模型案例——天气预报

- 如果第一天是雨天，第二天还是雨天的概率是0.8，是晴天的概率是0.2；如果第一天是晴天，第二天还是晴天的概率是0.6，是雨天的概率是0.4。
- 问：如果第一天下雨了，第二天仍然是雨天的概率是多少？第十天是晴天的概率是多少？经过很长一段时间后雨天、晴天的概率分别是多少？

马尔可夫模型案例——天气预报

- 构建转移概率矩阵



雨天	晴天	
0.8	0.4	雨天
0.2	0.6	晴天

$$A = \begin{bmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{bmatrix}$$

马尔可夫模型案例——天气预报

- 假设初始状态第一天是雨天，记为 $\mathbf{p}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- 第二天仍然是雨天（记为P1）的概率为： $\mathbf{p}_1 = \mathbf{A} \cdot \mathbf{p}_0 = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$
- 第十天（记为P9）是晴天概率： $\mathbf{p}_9 = \mathbf{A}^9 \cdot \mathbf{p}_0 = \begin{bmatrix} 0.6668 \\ 0.3332 \end{bmatrix}$
- 经过很长一段时间后雨天、晴天的概率的递推公式： $\mathbf{p}_n = \mathbf{A}^n \cdot \mathbf{p}_0$

马尔可夫模型案例——天气预报

- 直接计算矩阵A的n次方是很难的，将A进行特征分解（谱分解），得到：

$$A = TDT^{-1}$$

$$T = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & 0 \\ 0 & 0.4^n \end{bmatrix}$$

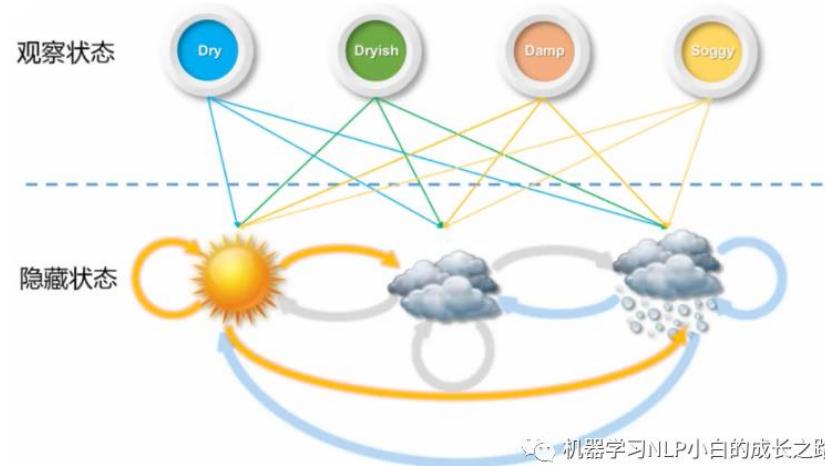
$$P_n = A^n \times P_0 = TD^n T^{-1} P_0$$

$$\begin{aligned} &= \frac{1}{3} \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0.4^n \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 + 0.4^n & 1 + 2 \times 0.4^n \\ 1 - 0.4^n & 1 + 2 \times 0.4^n \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{1}{3} \begin{bmatrix} 2 + 0.4^n \\ 1 - 0.4^n \end{bmatrix} \end{aligned}$$

- 当n趋于无穷即很长一段时间以后， $P_n = [0.67, 0.33]$ 。
- 额外发现：初始状态如果是 $P_0 = [0, 1]$ ，最后结果仍然是 $P_n = [0.67, 0.33]$ 。
- 这表明，马尔科夫过程与初始状态无关，跟转移矩阵有关。

马尔可夫模型的局限性

- 实际上，观察者无法直接获取天气状况的观测值。
- 生活在海边的世代渔民们发现，海藻的状态或许与天气的状态有密切的关系。可观测状态（海藻的干湿程度：dry, dryish, damp, soggy），和隐含状态（天气状况：sunny, cloudy, rainy）。
- 希望基于马尔可夫假设和海藻的状态设计出某种算法，实现在缺乏天气历史观测数据的情况下，对未来的天气情况进行预测。



3 隐马尔可夫模型

- 隐马尔可夫模型是由隐藏的马尔科夫链随机生成观测序列的过程，是一种经典的概率图模型。
- 隐马尔可夫模型（Hidden Markov Model, HMM）是描述两个时序序列联合分布 $p(x, y)$ 的概率模型
 - x 序列外界可见（外界指的是观测者），称为观测序列（observation sequence）
 - 观测 x 为单词
 - y 序列外界不可见，称为状态序列（state sequence）
 - 状态 y 为词性
- 人们也称状态为隐状态（hidden state），而称观测为显状态（visible state）。
- 隐马尔可夫模型之所以称为“马尔可夫模型”，是因为包含隐状态的随机过程满足马尔可夫性质，是马尔可夫过程的一种扩展。



3.1 从马尔可夫假设到隐马尔可夫模型

- 马尔可夫假设：每个事件的发生概率只取决于前一个事件
- 将满足该假设的连续多个事件串联在一起，就构成了马尔可夫链
- 隐马尔可夫模型的马尔可夫假设作用于状态序列
 - ①齐次马尔可夫假设：当前状态 y_t 仅仅依赖于前一个状态 y_{t-1} ，连续多个状态构成隐马尔可夫链 y

$$P(y_{t+1} | y_t, y_{t-1}, \dots, y_1, x_t, x_{t-1}, \dots, x_1) = P(y_{t+1} | y_t)$$

- ②观测独立假设：任意时刻的观测 x_t 只依赖于该时刻的状态 y_t ，与其他时刻的状态或观测独立无关

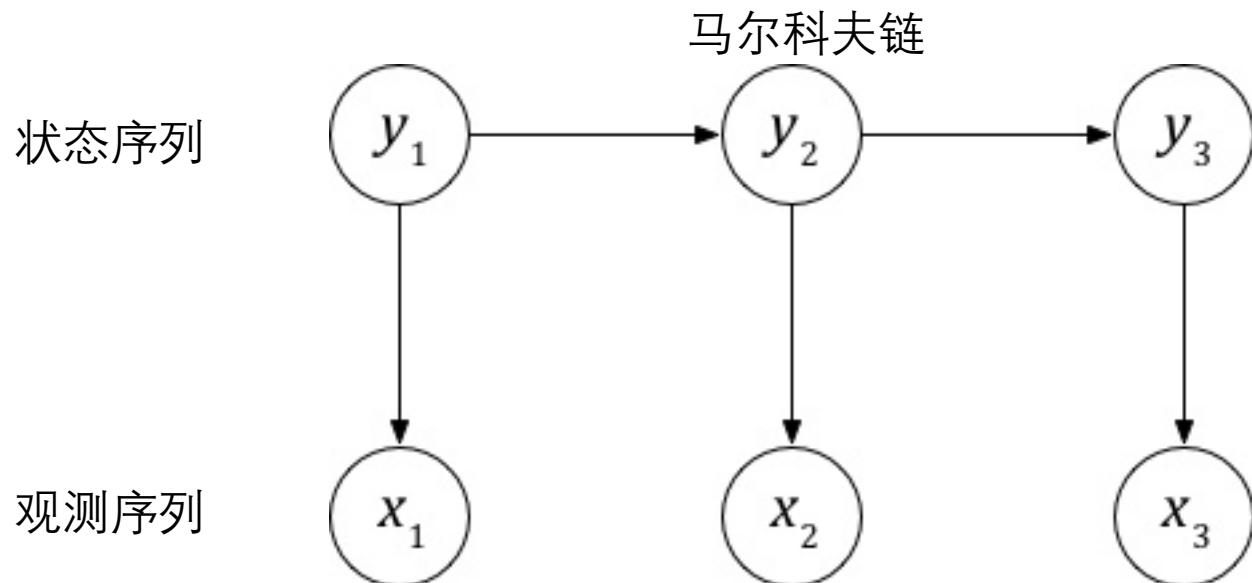
$$P(x_t | y_t, y_{t-1}, \dots, y_1, x_{t-1}, \dots, x_1) = P(x_t | y_t)$$



3.1 从马尔可夫假设到隐马尔可夫模型

- 用箭头表示事件的依赖关系（箭头终点是结果，依赖于起点的因缘）

图4-7 隐马尔可夫模型状态序列与观测序列的依赖关系



3. 2 隐马尔可夫模型的三要素

- 隐马尔可夫模型利用三个要素来模拟时序序列的发生过程
 - 初始状态概率向量 π
 - 状态转移概率矩阵 A
 - 发射概率矩阵（也称作观测概率矩阵） B
- 令 S 为所有可能的状态值的集合， V 是所有可能的观测值的集合，即： $S = \{s_1, s_2, \dots, s_N\}$ $V = \{v_1, v_2, \dots, v_M\}$
- 令 y 是长度为 T 的状态序列， x 是与之对应的观测序列，即：

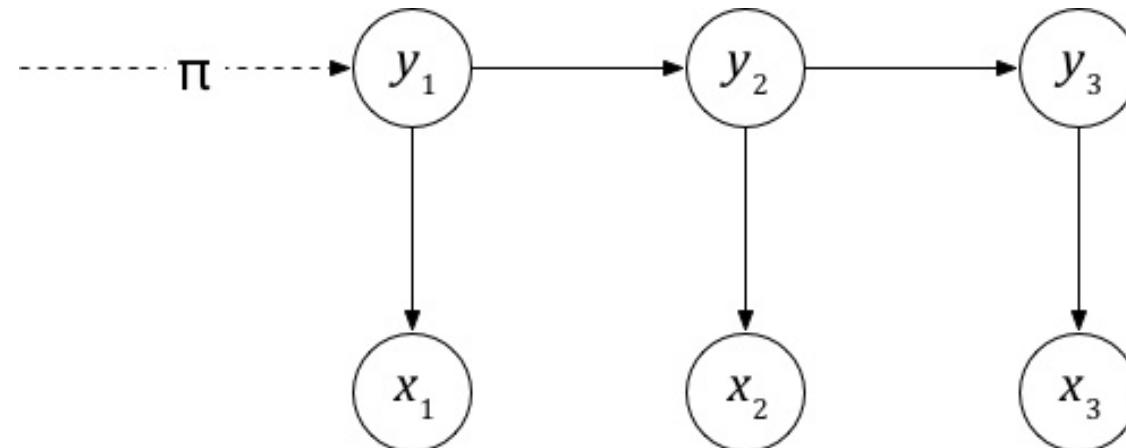
$$\begin{aligned}x &= (x_1, x_2, \dots, x_T) \\y &= (y_1, y_2, \dots, y_T)\end{aligned}\quad \text{一个样本}$$



初始状态概率向量

- 系统启动时进入的第一个状态 y_1 称为初始状态
 - 假设 y 有 N 种可能的取值，即 $y \in \{s_1, \dots, s_N\}$ ，那么 y_1 就是一个独立的离散型随机变量，由 $p(y_1 | \pi)$ 描述
 - 其中 $\pi = (\pi_1, \dots, \pi_N)^T$, $0 \leq \pi_i \leq 1$, $\sum_{i=1}^N \pi_i = 1$ 是概率分布的参数向量，称为初始状态概率向量

图4-8隐马尔可夫模型中的初始状态概率向量



初始状态概率向量

- $\pi = (\pi_1, \dots, \pi_N)^T, 0 \leq \pi_i \leq 1, \sum_{i=1}^N \pi_i = 1$ 是概率分布的参数向量，称为初始状态概率向量

$$p(y_1 = \text{B}) = 0.7$$

$$p(y_1 = \text{M}) = 0$$

$$p(y_1 = \text{E}) = 0$$

$$p(y_1 = \text{S}) = 0.3$$

此时隐马尔可夫模型的初始状态概率向量为 $\pi = [0.7, 0, 0, 0.3]$

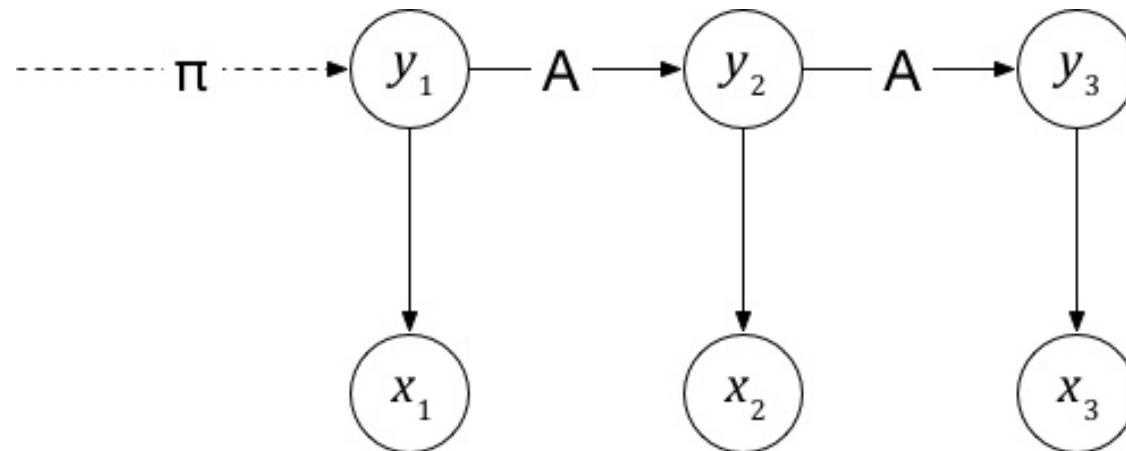


状态转移概率矩阵

- 从状态 s_i 到状态 s_j 的概率就构成了一个 $N \times N$ 的方阵，称为状态转移概率矩阵 A ：

$$A = [p(y_{t+1} = s_j \mid y_t = s_i)]_{N \times N}$$

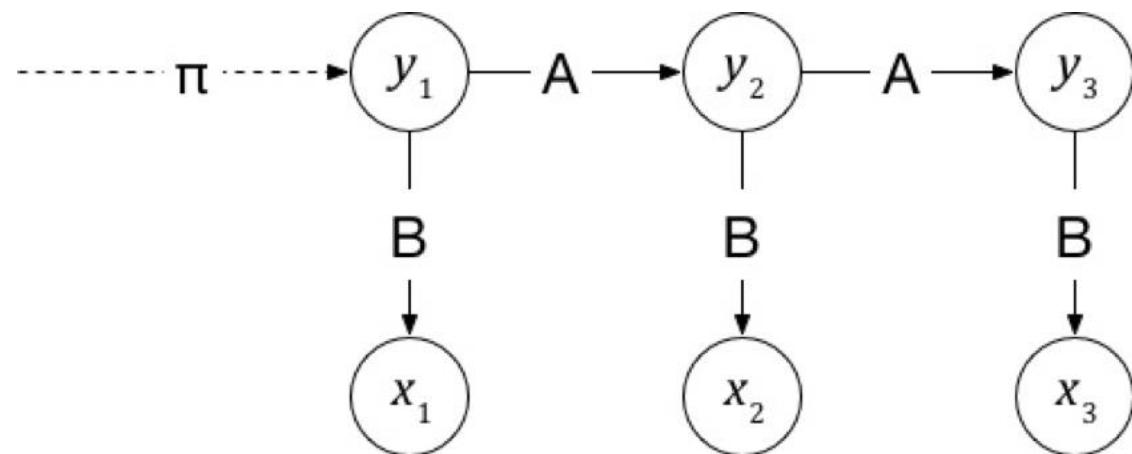
- 其中下标*i*、*j*分别表示状态的第*i*、*j*种取值，比如我们约定1表示标注集中的B，依序类推。



发射概率矩阵

- 给定每种 y , x 都是一个独立的离散型随机变量, 其参数对应一个向量
 - 这些参数向量构成了 $N \times M$ 的矩阵, 称为发射概率矩阵 B 。

$$B = [p(x_t = v_i \mid y_t = s_j)]_{N \times M}$$



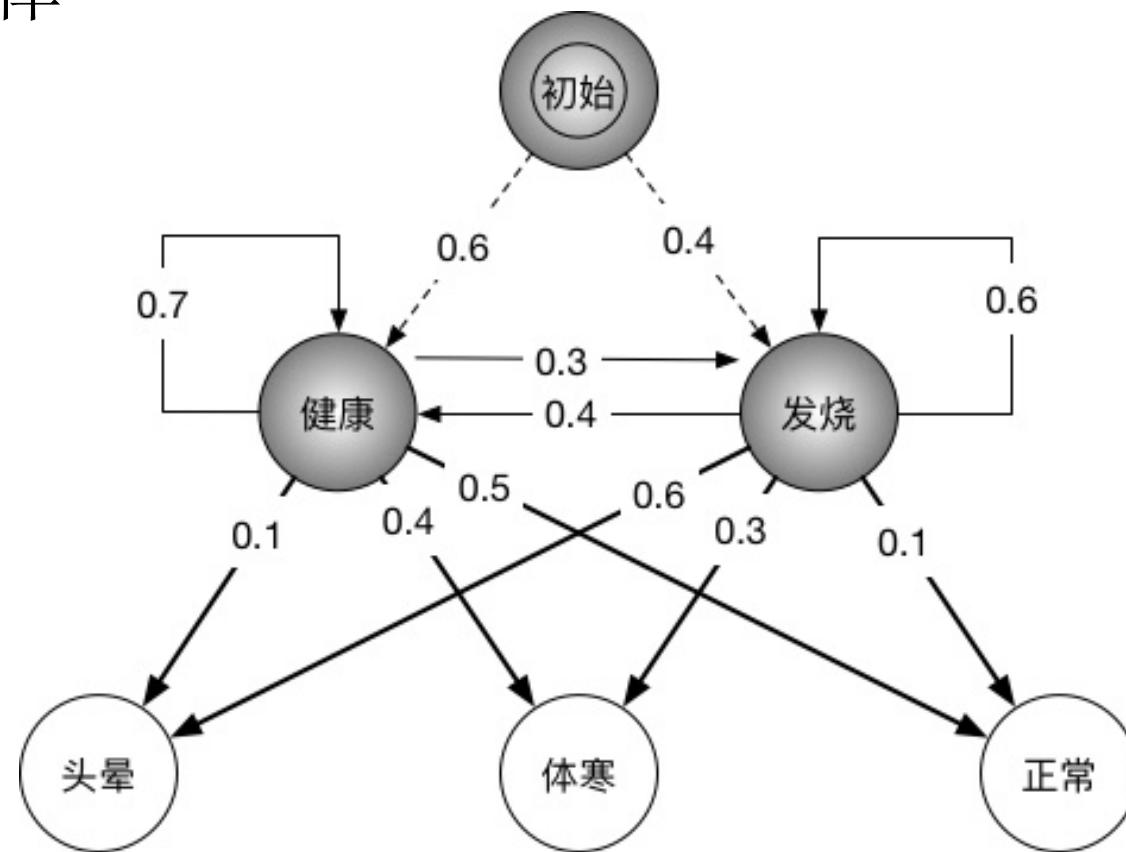
案例——医疗诊断

- 某医院招标开发“智能”医疗诊断系统，用来辅助感冒诊断。已知
 - ①来诊者只有两种状态：要么健康，要么发烧。
 - ②来诊者不确定自己到底是哪种状态，只能回答感觉头晕、体寒或正常。医院认为，
 - ③感冒这种病，只跟病人前一天的状态有关，并且，
 - ④当天的病情决定当天的身体感觉。有位来诊者的病历卡上完整地记录了最近T天的身体感受（头晕、体寒或正常），请预测这T天的身体状态（健康或发烧）



医疗诊断

- 感冒发病的规律



医疗诊断

```
states = ('Healthy', 'Fever')
start_probability = {'Healthy': 0.6, 'Fever': 0.4}
transition_probability = {
    'Healthy': {'Healthy': 0.7, 'Fever': 0.3},
    'Fever': {'Healthy': 0.4, 'Fever': 0.6},
}
emission_probability = {
    'Healthy': {'normal': 0.5, 'cold': 0.4, 'dizzy': 0.1},
    'Fever': {'normal': 0.1, 'cold': 0.3, 'dizzy': 0.6},
}
observations = ('normal', 'cold', 'dizzy')
```



案例——赌场风云



- 探子回报，某个赌场，有个大叔总赢钱，玩得一手好骰子，并且每次玩骰子时周围都有几个保镖站在身边。老板根据多年的经验，推测这位不善之客应该会偷换骰子大法。老板是个冷静的人，看这位大叔也不是善者，不想轻易得罪他，又不想让他坏了规矩。
- Questions:
 - 该大叔是不是在出千？
 - 如果是在出千，那么他用了几个作弊的骰子？还有当前是不是在用作弊的骰子？
 - 这几个作弊骰子出现各点的概率是多少？

赌场风云

- 假设这位大叔玩比大小游戏，有两个作弊骰子，分别用于掷出想要的大和小。

$$\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N\} = \{\text{正常骰子}, \text{作弊骰子1}, \text{作弊骰子2}\}$$

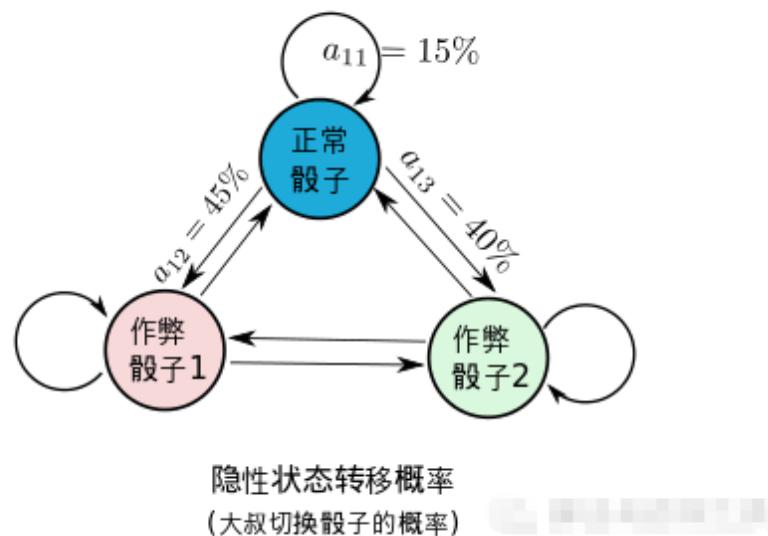
$$\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\} = \{1, 2, 3, 4, 5, 6\}$$

- 假设大叔每次采取随机方式选择骰子：

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_N)^T = (1/3, 1/3, 1/3)$$

赌场风云

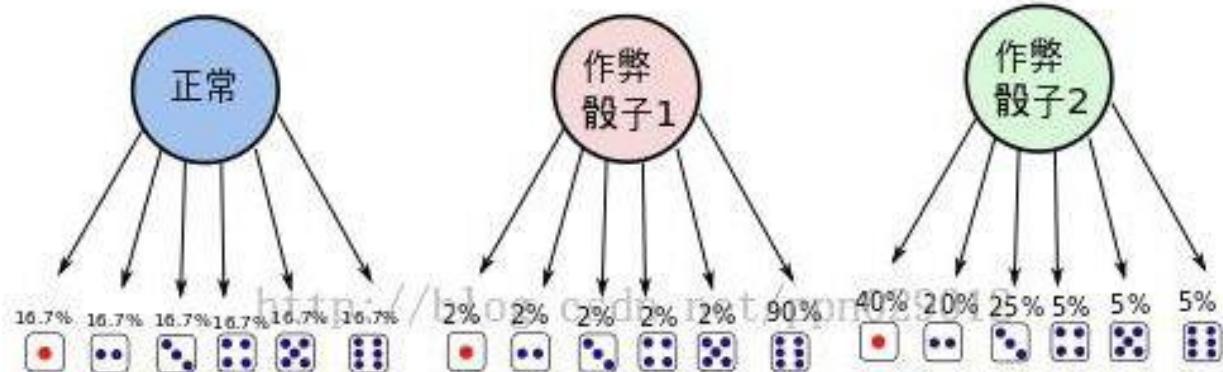
- 根据摄像头捕捉的多个样本，HMM 算出：



$$A = \begin{bmatrix} 0.15 & 0.45 & 0.4 \\ 0.25 & 0.35 & 0.4 \\ 0.10 & 0.55 & 0.35 \end{bmatrix}$$

赌场风云

- 骰子出现各点的概率分布：



隐性状态表现概率
(骰子点数分布概率)

$$B = \begin{bmatrix} 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.90 \\ 0.4 & 0.20 & 0.25 & 0.05 & 0.05 & 0.05 \end{bmatrix}$$

案例——创业基金

- 背景：丈夫想要创业，需要启动基金，但财政大权归妻子管。
- 思路：要在妻子心情好的时候，和妻子说创业的规划，成功率高。
- 假设妻子每天的活动包含工作、辅导孩子学习、买衣服、吃大餐、看电影、做美容，但每天只会选择一项。经过一个月的观察，丈夫把妻子这一个月的情况都记录下来。丈夫发现，妻子每天的心情和每天的活动有密切关联，于是想通过观测妻子每天选择的活动来猜测她的心情好坏。

$$\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N\} = \{\text{心情一般}, \text{ 心情超好}, \text{ 心情超差}\}$$

$$\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\} = \{\text{工作}, \text{辅导孩子学习}, \text{买衣服}, \text{吃大餐}, \text{看电影}, \text{做美容}\}$$

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_N)^T = (1/3, 1/3, 1/3)$$

3. 3 隐马尔科夫模型的三个经典问题

- 概率计算问题或评估 (Evaluation) 问题

给定: $\lambda = (A, B, \pi)$ $O = (o_1, o_2, \dots, o_T)$

计算: $P(O|\lambda)$

- 参数估计问题或学习 (Learning) 问题

已知: $O = (o_1, o_2, \dots, o_T)$

估计: $\lambda = (A, B, \pi)$, 使 $P(O|\lambda)$ 最大

- 序列标注问题或解码 (Decoding) 问题

已知: $\lambda = (A, B, \pi)$ $O = (o_1, o_2, \dots, o_T)$

求: 使 $P(I|O)$ 最大的状态序列 $I = (i_1, i_2, \dots, i_T)$

例：经典的盒子摸球模型

- 盒子: 1 2 3 4
- 红球: 5 3 6 8
- 白球: 5 7 4 2
- 转移规则:
 - 盒子1 下一个 盒子2
 - 盒子2或3 下一个 0.4 左, 0.6右
 - 盒子4 下一个 0.5 自身, 0.5盒子3
- 重复5次: 0={ 白, 红, 白, 红, 红}

例：经典的盒子摸球模型

- 状态集合: $Q=\{\text{盒子1}, \text{盒子2}, \text{盒子3}, \text{盒子4}\}$, $N=4$
- 观测集合: $V=\{\text{红球}, \text{白球}\}$ $M=2$
- 初始化概率分布:

$$\pi = (0.25, 0.25, 0.25, 0.25)^T$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.4 & 0 & 0.6 & 0 \\ 0 & 0.4 & 0 & 0.6 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \\ 0.6 & 0.4 \\ 0.8 & 0.2 \end{bmatrix}$$

观测序列的生成过程

算法 (观测序列的生成)

输入：隐马尔可夫模型 $\lambda = (A, B, \pi)$ ， 观测序列长度 T ；

输出：观测序列 $O = (o_1, o_2, \dots, o_T)$.

(1) 按照初始状态分布 π 产生状态 i_1

(2) 令 $t=1$

(3) 按照状态 i_t 的观测概率分布 $b_{i_t}(k)$ 生成 o_t

(4) 按照状态 i_t 的状态转移概率分布 $\{a_{i_t i_{t+1}}\}$ 产生状态 i_{t+1} ， $i_{t+1} = 1, 2, \dots, N$

(5) 令 $t=t+1$ ；如果 $t < T$ ，转步(3)；否则，终止

使用Numpy实现HMM

```
# 初始状态概率分布
pi = np.array([0.25, 0.25, 0.25, 0.25])
# 状态转移概率矩阵
A = np.array([
    [0, 1, 0, 0],
    [0.4, 0, 0.6, 0],
    [0, 0.4, 0, 0.6],
    [0, 0, 0.5, 0.5]])
# 观测概率矩阵
B = np.array([
    [0.5, 0.5],
    [0.3, 0.7],
    [0.6, 0.4],
    [0.2, 0.8]])
# 可能的状态数和观测数
N = 4
M = 2
# 创建HMM实例
hmm = HMM(N, M, pi, A, B)
# 生成观测序列
print(hmm.generate(5))
```

```
import numpy as np

### 定义HMM模型
class HMM:
    def __init__(self, N, M, pi=None, A=None, B=None):
        # 可能的状态数
        self.N = N
        # 可能的观测数
        self.M = M
        # 初始状态概率向量
        self.pi = pi
        # 状态转移概率矩阵
        self.A = A
        # 观测概率矩阵
        self.B = B

    # 根据给定的概率分布随机返回数据
    def rdistribution(self, dist):
        r = np.random.rand()
        for ix, p in enumerate(dist):
            if r < p:
                return ix
        r -= p

    # 生成HMM观测序列
    def generate(self, T):
        # 根据初始概率分布生成第一个状态
        i = self.rdistribution(self.pi)
        # 生成第一个观测数据
        o = self.rdistribution(self.B[i])
        observed_data = [o]
        # 遍历生成剩下的状态和观测数据
        for _ in range(T-1):
            i = self.rdistribution(self.A[i])
            o = self.rdistribution(self.B[i])
            observed_data.append(o)
        return observed_data
```

3.3.1 概率计算问题

- 概率计算问题：在给定模型 $\lambda = (\pi, A, B)$ 和观测序列 $O = \{o_1, o_2, \dots, o_T\}$ ，计算观测序列出现的概率 $P(O|\lambda)$ 。
- 解决这个问题，目的是检测观察到的结果和已知模型是否吻合，评估模型的有效性。这也是为何概率计算问题也叫做评估问题的缘由。
- 计算方法：
 - 直接计算
 - 前向/后向算法（使用动态规划思想，递归解法）

数学推导

$$p(O|\lambda) = \sum_I p(I, O|\lambda) = \sum_I p(O|I, \lambda)p(I|\lambda)$$

$$p(I|\lambda) = p(i_1, i_2, \dots, i_t|\lambda) = p(i_t|i_1, i_2, \dots, i_{t-1}, \lambda)p(i_1, i_2, \dots, i_{t-1}|\lambda)$$

根据齐次 Markov 假设：

$$p(i_t|i_1, i_2, \dots, i_{t-1}, \lambda) = p(i_t|i_{t-1}) = a_{i_{t-1}i_t}$$

所以：

$$p(I|\lambda) = \pi_1 \prod_{t=2}^T a_{i_{t-1}i_t}$$

又由于：

$$p(O|I, \lambda) = \prod_{t=1}^T b_{i_t}(o_t)$$

于是：

$$p(O|\lambda) = \sum_I \pi_{i_1} \prod_{t=2}^T a_{i_{t-1}i_t} \prod_{t=1}^T b_{i_t}(o_t)$$

- 直接计算，计算量非常大，时间复杂度达到

$$\mathcal{O}(\mathbf{T}\mathbf{N}^T)$$

前向算法

下面，记 $\alpha_t(i) = p(o_1, o_2, \dots, o_t, i_t = q_i | \lambda)$ ，所以， $\alpha_T(i) = p(O, i_T = q_i | \lambda)$ 。我们看到：

$$p(O|\lambda) = \sum_{i=1}^N p(O, i_T = q_i | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

$$\alpha_1(i) = p(o_1, i_1 = q_i | \lambda) = p(o_1 | i_1 = q_i, \lambda) p(i_1 = q_i, \lambda) = \pi_i b_i(o_1)$$

对 $\alpha_{t+1}(j)$ ：

$$\begin{aligned} \alpha_{t+1}(j) &= p(o_1, o_2, \dots, o_{t+1}, i_{t+1} = q_j | \lambda) \\ &= \sum_{i=1}^N p(o_1, o_2, \dots, o_{t+1}, i_{t+1} = q_j, i_t = q_i | \lambda) \\ &= \sum_{i=1}^N p(o_{t+1} | o_1, o_2, \dots, i_{t+1} = q_j, i_t = q_i, \lambda) p(o_1, \dots, o_t, i_t = q_i, i_{t+1} = q_j, \lambda) \end{aligned}$$

利用观测独立假设：

$$\begin{aligned} \alpha_{t+1}(j) &= \sum_{i=1}^N p(o_{t+1} | i_{t+1} = q_j) p(o_1, \dots, o_t, i_t = q_i, i_{t+1} = q_j | \lambda) \\ &= \sum_{i=1}^N p(o_{t+1} | i_{t+1} = q_j) p(i_{t+1} = q_j | o_1, \dots, o_t, i_t = q_i, \lambda) p(o_1, \dots, o_t, i_t = q_i | \lambda) \\ &= \sum_{i=1}^N b_j(o_{t+1}) a_{ij} \alpha_t(i) \end{aligned}$$

上面利用了齐次 Markov 假设得到了一个递推公式，这个算法叫做前向算法。

前向算法过程

- 1、初始化

$$\alpha_1(i) = p(o_1, i_1 = q_i | \lambda) = p(o_1 | i_1 = q_i, \lambda) p(i_1 = q_i, \lambda) = \pi_i b_i(o_1)$$

- 2、推导

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

- 3、求和

$$p(O|\lambda) = \sum_{i=1}^N p(O, i_T = q_i | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

```
### 前向算法计算条件概率
def prob_calc(0):
    """
    输入:
    0: 观测序列
    输出:
    alpha.sum(): 条件概率
    """

    # 初值
    alpha = pi * B[:, 0[0]]
    # 递推
    for o in 0[1:]:
        alpha_next = np.empty(4)
        for j in range(4):
            alpha_next[j] = np.sum(A[:, j] * alpha * B[j, o])
        alpha = alpha_next
    return alpha.sum()

# 给定观测
0 = [1, 0, 1, 0, 0]
print(prob_calc(0))
```

后向算法

对于这个 $\beta_t(\mathbf{i})$:

还有一种算法叫做后向算法, 定义 $\beta_t(i) = p(o_{t+1}, o_{t+2}, \dots, o_T | i_t = q_i, \lambda)$:

$$\begin{aligned} p(O|\lambda) &= p(o_1, \dots, o_T | \lambda) \\ &= \sum_{i=1}^N p(o_1, o_2, \dots, o_T, i_1 = q_i | \lambda) \\ &= \sum_{i=1}^N p(o_1, o_2, \dots, o_T | i_1 = q_i, \lambda) \pi_i \\ &= \sum_{i=1}^N p(o_1 | o_2, \dots, o_T, i_1 = q_i, \lambda) p(o_2, \dots, o_T | i_1 = q_i, \lambda) \pi_i \\ &= \sum_{i=1}^N b_i(o_1) \pi_i \beta_1(i) \end{aligned}$$

$$\beta_{T-1}(\mathbf{i}) = p(o_T | i_T = q_i, \lambda) = b_i(o_T)$$

$$\beta_T(\mathbf{i}) = 1$$

$$\begin{aligned} \beta_t(i) &= p(o_{t+1}, \dots, o_T | i_t = q_i) \\ &= \sum_{j=1}^N p(o_{t+1}, o_{t+2}, \dots, o_T, i_{t+1} = q_j | i_t = q_i) \\ &= \sum_{j=1}^N p(o_{t+1}, \dots, o_T | i_{t+1} = q_j, i_t = q_i) p(i_{t+1} = q_j | i_t = q_i) \\ &= \sum_{j=1}^N p(o_{t+1}, \dots, o_T | i_{t+1} = q_j) a_{ij} \\ &= \sum_{j=1}^N p(o_{t+1} | o_{t+2}, \dots, o_T, i_{t+1} = q_j) p(o_{t+2}, \dots, o_T | i_{t+1} = q_j) a_{ij} \\ &= \sum_{j=1}^N b_j(o_{t+1}) a_{ij} \beta_{t+1}(j) \end{aligned}$$

于是后向地得到了第一项。

后向算法过程

- 1、初始化

$$\beta_T(i) = 1$$

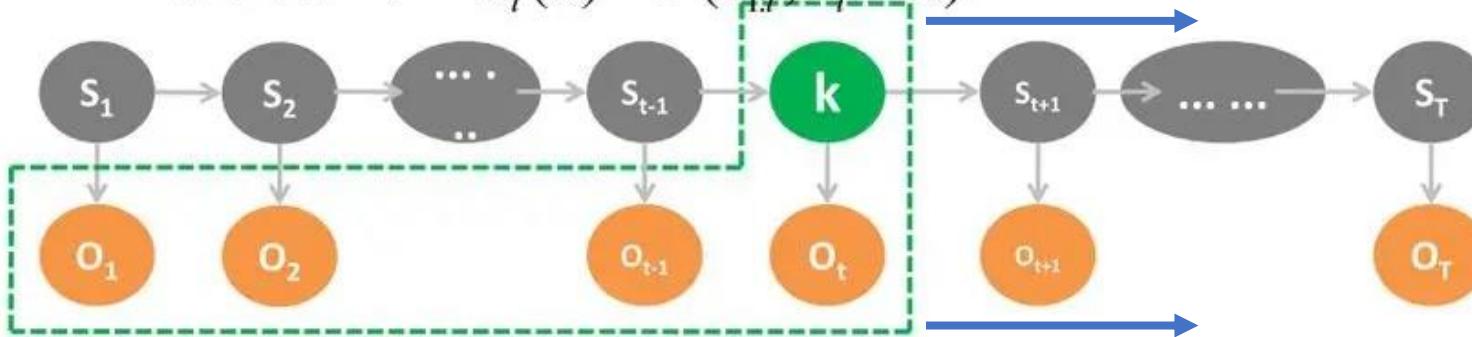
- 2、推导

$$\beta_t(i) = \sum_{j=1}^N b_j(o_{t+1}) a_{ij} \beta_{t+1}(j)$$

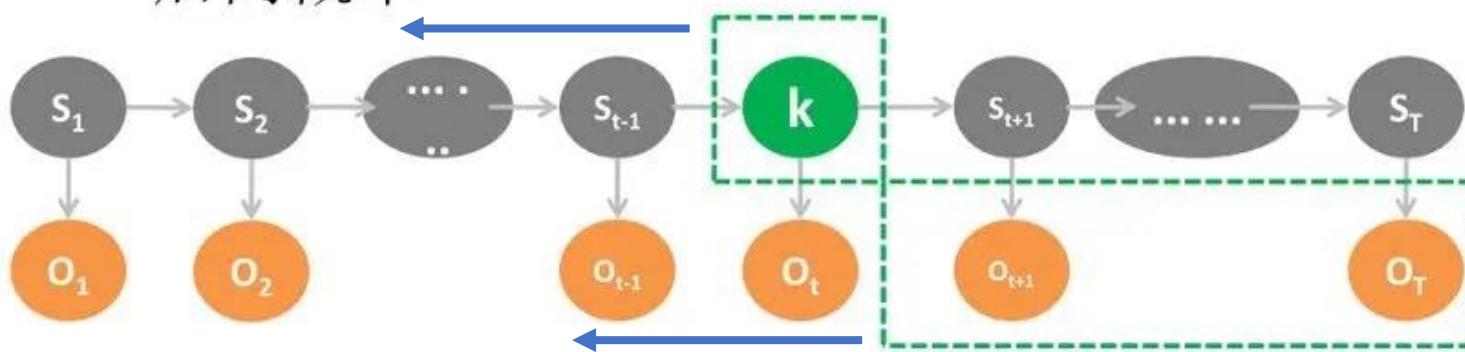
- 3、求和

$$p(O|\lambda) = \sum_{i=1}^N b_i(o_1) \pi_i \beta_1(i)$$

• 前向概率 $\alpha_t(k) = P(o_{1:t}, S_t = k)$

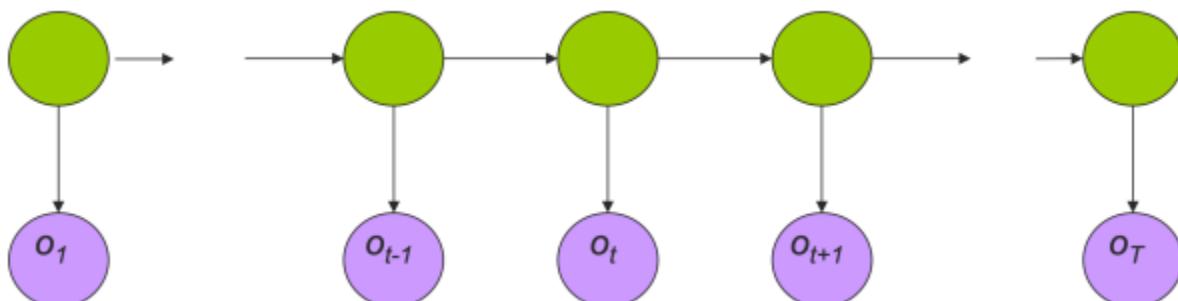


• 后向概率 $\beta_t(k) = P(o_{t+1:T} | S_t = k)$



3.3.2 参数估计问题

- 参数估计问题：给定 $\mathcal{O} = \{o_1, o_2, \dots, o_T\}$ ，如何调整模型的参数 λ 使得产生这一序列的概率 $P(\mathcal{O}|\lambda)$ 最大？
- 解题思路：根据训练数据是包括观测数据和对应的状态序列还是只有观测序列，可以分为有监督学习和无监督学习。
- 解决方法：
 - 有监督学习：极大似然估计法 MLE
 - 无监督学习：基于EM思想的Baum-Welch算法



方案一：极大似然估计法 MLE

- 假设训练数据为: $D = \{(\mathbf{O}_1, \mathbf{I}_1), (\mathbf{O}_2, \mathbf{I}_2), \dots, (\mathbf{O}_n, \mathbf{I}_n)\}$

$$\mathbf{O}_i = (\mathbf{o}_{i1}, \mathbf{o}_{i2}, \dots, \mathbf{o}_{iT})$$

$$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\}$$



$$\lambda = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$$

$$\mathbf{I}_i = (\mathbf{i}_{i1}, \mathbf{i}_{i2}, \dots, \mathbf{i}_{iT})$$

$$\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N\}$$

- 1. 转移概率 a_{ij} 的估计

设样本中时刻 t 处于状态 i 时刻 $t + 1$ 处于状态 j 的频数为 A_{ij} ，那么状态转移概率 a_{ij} 的估计是

$$\hat{a}_{ij} = \frac{A_{ij}}{\sum_{j=1}^N A_{ij}}, \quad i = 1, 2, \dots, N; \quad j = 1, 2, \dots, N$$

方案一：极大似然估计法 MLE

- 2. 观测概率 $b_j(k)$ 的估计

设样本中状态为 j 并观测为 k 的频数是 B_{jk} ，那么状态为 j 观测为 k 的概率 $b_j(k)$ 的估计是

$$\hat{b}_j(k) = \frac{B_{jk}}{\sum_{k=1}^M B_{jk}}, \quad j = 1, 2, \dots, N; k = 1, 2, \dots, M$$

- 3. 初始状态概率 π_i 的估计

设训练数据中初始状态为 q_i 的样本数为 $|Q_i|$ ，那么初始状态概率 π_i 的估计是

$$\hat{\pi}_i = \frac{|Q_i|}{n}$$

方案二：基于EM思想的Baum-Welch算法

- 假设训练数据为：

$$D = \{\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_n\}$$

$$\mathbf{O}_i = (\mathbf{o}_{i1}, \mathbf{o}_{i2}, \dots, \mathbf{o}_{iT})$$

$$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\}$$

$$\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N\}$$



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

- 假设观测数据样本 \mathbf{o}_i 的状态序列数据为： $\mathbf{I}_i = (\mathbf{i}_{i1}, \mathbf{i}_{i2}, \dots, \mathbf{i}_{iT})$

$$\mathbf{P}(\mathbf{O}|\lambda) = \sum_{\mathbf{I}} \mathbf{P}(\mathbf{I}, \mathbf{O}|\lambda) = \sum_{\mathbf{I}} \mathbf{P}(\mathbf{O}|\mathbf{I}, \lambda) \mathbf{P}(\mathbf{I}|\lambda)$$

- 该公式的参数学习可以由 EM 算法实现求解

EM算法

- EM(Expectation Maximum, 期望最大化)是一种迭代算法，用于对含有隐变量的概率模型的极大似然估计或极大后验估计，其中概率模型依赖于无法观测的隐变量。
- EM 算法每次迭代包含两个步骤：
 - E步是计算期望，利用对隐变量的现有估计值，计算其最大似然估计值。
 - M步是最大化，通过最大化在E步上求得的最大似然函数值来计算参数的值。M步上找到的参数估计值被用于下一个E步计算中，这个过程不断交替执行。
- 模型参数的每一次迭代，含有隐变量的概率模型的似然函数都会增大，当似然函数不再增加或增加的值小于设置的阈值时，迭代结束。

从MLE到EM

- **问题描述:** 现在我们学校想调研学生的身高分布。我们先假设所有学生的身高服从正态分布 $N(\mu, \sigma^2)$ ，注意，使用MLE的前提是要先假设已知数据分布，但不知道这个分布的均值 μ 和方差 σ^2 。
- 学生太多，挨个统计不靠谱，那么就需要用到概率统计中的抽样思想，根据抽样的样本来估算总体。假设随机抽样200个同学的身高数据，要求根据这200个人的身高数据来估计均值 μ 和方差 σ^2

$$X = x_1, x_2, \dots, x_N \quad p(x|\theta) \text{ 服从高斯分布 } N(\mu, \sigma^2) \quad \text{每个样本独立同分布}$$

从MLE到EM

- 问题一：抽到这200人的概率是多少？

似然函数
$$L(\theta) = L(x_1, x_2, \dots, x_n; \theta) = \prod_{i=1}^n p(x_i | \theta), \quad \theta \in \Theta$$

- 对 L 取对数，将其变成连加形式，称为对数似然函数：

$$H(\theta) = \ln L(\theta) = \ln \prod_{i=1}^n p(x_i | \theta) = \sum_{i=1}^n \ln p(x_i | \theta)$$

从MLE到EM

- 问题二：学校那么多学生，怎么样才能恰好抽到这 200 个人？
- 意味着在整个学校中，这 200 个人（的身高）出现的概率最大，也就是其对应的似然函数 $L(\theta)$ 极大，即：

$$\hat{\theta} = \operatorname{argmax}_{\theta} L(\theta)$$

- $\hat{\theta}$ 就是 θ 的极大似然估计量，即为我们所求的值。
- 问题三：如何求似然函数的极大值？

求 $L(\theta)$ 对所有参数的偏导数，然后让这些偏导数为 0，假设有 n 个参数，就有 n 个方程组成的方程组，那么方程组的解就是似然函数的极值点了，从而得到对应的 θ 了。

求极大似然函数极大值的一般步骤

- (1) 写出似然函数;
- (2) 对似然函数取对数，并整理;
- (3) 求导数，令导数为 0，得到似然方程;
- (4) 解似然方程，得到的参数。

从MLE到EM

- **问题描述：**实际上，男生和女生的身高服从两种不同的正态分布，即男生身高服从于 $N(\mu_1, \sigma_1^2)$ ，女生身高服从于 $N(\mu_2, \sigma_2^2)$ 。现在随机抽样200个同学的身高数据，但问题是不知道每次抽样的同学是男生还是女生，所以直接使用MLE来求参数的估计值不可行。



从MLE到EM

- EM的解题思路：引入隐变量（男生、女生）
 - 先设定男生和女生的身高分布参数(初始值)，例如男生的身高分布为 $N(\mu_1 = 172, \sigma_1^2 = 5^2)$ ，女生的身高分布为 $N(\mu_2 = 162, \sigma_2^2 = 5^2)$ ，当然肯定不准。
 - 然后计算出每个人更可能属于男生的还是女生的正态分布（例如，一个人的身高是180，那他极大可能是男生身高），这步称为 Expectation；
 - 按上面的方法将这 200 个人分为男生和女生两部分，就可以根据MLE分别对男生和女生的身高分布参数进行估计，这步称为 Maximization；
 - 接着，当更新这两个分布时，每一个学生属于女生还是男生的概率又变了，那么就再需要调整E步；
 -如此往复，直到参数基本不再发生变化或满足结束条件为止。

Jensen不等式

- 设是定义在实数域上的函数 f , 如果对于任意的实数, 都有二阶导数满足:

$$f'' \geq 0$$

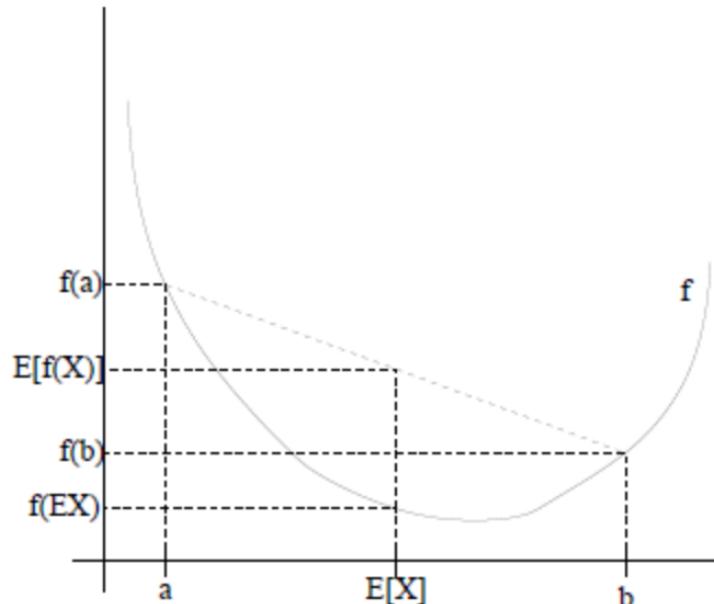
- 那么 f 是**凸函数**。

如下图, 如果函数 f 是凸函数, x 是随机变量, 有 0.5 的概率是 a , 有 0.5 的概率是 b , x 的期望值就是 a 和 b 的中值了那么:

$$E[f(x)] \geq f(E(x))$$

其中, $E[f(x)] = 0.5f(a) + 0.5f(b)$, $f(E(x)) = f(0.5a + 0.5b)$, 这里 a 和 b 的权值为 0.5, $f(a)$ 与 a 的权值相等, $f(b)$ 与 b 的权值相等。

特别地, 如果函数 f 是严格凸函数, 当且仅当: $p(x = E(x)) = 1$ (即随机变量是常量) 时等号成立。



注: 若函数 f 是凹函数, Jensen不等式符号相反。

EM算法推导

对于 m 个相互独立的样本 $x = (x^{(1)}, x^{(2)}, \dots, x^{(m)})$ ，对应的隐含数据 $z = (z^{(1)}, z^{(2)}, \dots, z^{(m)})$ ，此时 (x, z) 即为完全数据，样本的模型参数为 θ ，则观察数据 $x^{(i)}$ 的概率为 $P(x^{(i)} | \theta)$ ，完全数据 $(x^{(i)}, z^{(i)})$ 的似然函数为 $P(x^{(i)}, z^{(i)} | \theta)$ 。

假如没有隐含变量 z ，我们仅需要找到合适的 θ 极大化对数似然函数即可：

$$\theta = \arg \max_{\theta} L(\theta) = \arg \max_{\theta} \sum_{i=1}^m \log P(x^{(i)} | \theta)$$

增加隐含变量 z 之后，我们的目标变成了找到合适的 θ 和 z 让对数似然函数极大：

$$\theta, z = \arg \max_{\theta, z} L(\theta, z) = \arg \max_{\theta, z} \sum_{i=1}^m \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)} | \theta)$$

对参数就偏导，“和的对数”的偏导很难求解。

也就是说 $\frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})}$ 为第 i 个样本, $Q_i(z^{(i)})$ 为第 i 个样本对应的权重, 那么:

EM算法推导

$$E(\log \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})}) = \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})}$$

形式上很像期望的计算公式

$$\begin{aligned} \sum_{i=1}^m \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}|\theta) &= \sum_{i=1}^m \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})} \\ \text{和的对数} \rightarrow \text{对数的和} &\geq \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})} \end{aligned}$$

上面第(1)式引入了一个未知的新的分布 $Q_i(z^{(i)})$, 满足:

关于隐变量的概率分布 $\sum_z Q_i(z) = 1, 0 \leq Q_i(z) \leq 1$



思考: 等号何时成立?

其中:

第(2)式用到了 Jensen 不等式 (对数函数是凹函数):

$$E(y) = \sum_i \lambda_i y_i, \lambda_i \geq 0, \sum_i \lambda_i = 1$$

$$\log(E(y)) \geq E(\log(y)) \quad y_i = \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})}$$

《自然》 $\lambda_i = Q_i(z^{(i)})$

新事件y
这就是E步中的
Expectation的来历!

EM算法推导

由 Jensen 不等式可知，等式成立的条件是随机变量是常数，则有：

$$\frac{P(x^{(i)}, z^{(i)} | \theta)}{Q_i(z^{(i)})} = c$$

其中 c 为常数，对于任意 i ，我们得到：

$$P(x^{(i)}, z^{(i)} | \theta) = c Q_i(z^{(i)})$$

方程两边同时累加和：

$$\sum_z P(x^{(i)}, z^{(i)} | \theta) = c \sum_z Q_i(z^{(i)})$$

由于 $\sum_z Q_i(z^{(i)}) = 1$ 。从上面两式，我们可以得到：

其中：

边缘概率公式： $P(x^{(i)} | \theta) = \sum_z P(x^{(i)}, z^{(i)} | \theta)$

条件概率公式： $\frac{P(x^{(i)}, z^{(i)} | \theta)}{P(x^{(i)} | \theta)} = P(z^{(i)} | x^{(i)}, \theta)$

$$\sum_z P(x^{(i)}, z^{(i)} | \theta) = c$$

$$Q_i(z^{(i)}) = \frac{P(x^{(i)}, z^{(i)} | \theta)}{c} = \frac{P(x^{(i)}, z^{(i)} | \theta)}{\sum_z P(x^{(i)}, z^{(i)} | \theta)} = \frac{P(x^{(i)}, z^{(i)} | \theta)}{P(x^{(i)} | \theta)}$$
$$= P(z^{(i)} | x^{(i)}, \theta)$$

从上式可以发现 $Q(z)$ 是已知样本和模型参数下的隐变量分布。

《自然语言处理入门》

EM算法推导

- E步固定模型参数 θ ， 预测隐变量 z $Q_i(z^{(i)}) := P(z^{(i)} | x^{(i)}, \theta)$

通过 $\arg \max_{\mathbf{z}} \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)} | \theta)}{Q_i(z^{(i)})}$

- M步固定隐变量 z ， 估计 θ

去掉上式中常数的部分 $Q_i(z^{(i)})$ ， 则我们需要极大化的对数似然下界为：

$$\arg \max_{\theta} \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log P(x^{(i)}, z^{(i)} | \theta)$$

EM算法流程

现在我们总结下EM算法的流程。

输入：观察数据 $x = (x^{(1)}, x^{(2)}, \dots, x^{(m)})$, 联合分布 $p(x, z|\theta)$, 条件分布 $p(z|x, \theta)$, 极大迭代次数 J 。

1) 随机初始化模型参数 θ 的初值 θ^0

2) for j from 1 to J:

- E步：计算联合分布的条件概率期望： Jensen不等式取等号时， $Q(z)$ 是已知样本和参数下的隐变量分布

$$Q_i(z^{(i)}) := P(z^{(i)}|x^{(i)}, \theta)$$

- M步：极大化 $L(\theta)$,得到 θ :

$$\theta := \arg \max_{\theta} \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log P(x^{(i)}, z^{(i)} | \theta)$$

- 重复E、M步骤直到 θ 收敛

输出：模型参数 θ

EM算法的收敛性思考

- EM算法的流程并不复杂，但是还有两个问题需要我们思考：
- 1) EM算法能保证收敛吗？
- 2) EM算法如果收敛，那么能保证收敛到全局极大值吗？

首先我们来看第一个问题, EM 算法的收敛性。要证明 EM 算法收敛, 则我们需要证明我们的对数似然函数的值在迭代的过程中一直在增大。即:

EM算法收敛吗?

$$\sum_{i=1}^m \log P(x^{(i)} | \theta^{j+1}) \geq \sum_{i=1}^m \log P(x^{(i)} | \theta^j)$$

由于:

$$L(\theta, \theta^j) = \sum_{i=1}^m \sum_{z^{(i)}} P(z^{(i)} | x^{(i)}, \theta^j) \log P(x^{(i)}, z^{(i)} | \theta)$$

令:

$$H(\theta, \theta^j) = \sum_{i=1}^m \sum_{z^{(i)}} P(z^{(i)} | x^{(i)}, \theta^j) \log P(z^{(i)} | x^{(i)}, \theta)$$

上两式相减得到:

$$\sum_{i=1}^m \log P(x^{(i)} | \theta) = L(\theta, \theta^j) - H(\theta, \theta^j)$$

在上式中分别取 θ 为 θ^j 和 θ^{j+1} , 并相减得到:

$$\begin{aligned} \sum_{i=1}^m \log P(x^{(i)} | \theta^{j+1}) - \sum_{i=1}^m \log P(x^{(i)} | \theta^j) &= [L(\theta^{j+1}, \theta^j) - L(\theta^j, \theta^j)] \\ &\quad - [H(\theta^{j+1}, \theta^j) - H(\theta^j, \theta^j)] \end{aligned}$$

EM算法收敛吗？

由于 θ^{j+1} 使得 $L(\theta, \theta^j)$ 极大，因此有：

$$L(\theta^{j+1}, \theta^j) - L(\theta^j, \theta^j) \geq 0$$

而对于第二部分，我们有：

$$H(\theta^{j+1}, \theta^j) - H(\theta^j, \theta^j) = \sum_{i=1}^m \sum_{z^{(i)}} P(z^{(i)} | x^{(i)}, \theta^j) \log \frac{P(z^{(i)} | x^{(i)}, \theta^{j+1})}{P(z^{(i)} | x^{(i)}, \theta^j)} \quad (3)$$

$$\leq \sum_{i=1}^m \log \left(\sum_{z^{(i)}} P(z^{(i)} | x^{(i)}, \theta^j) \frac{P(z^{(i)} | x^{(i)}, \theta^{j+1})}{P(z^{(i)} | x^{(i)}, \theta^j)} \right) \quad (4)$$

$$= \sum_{i=1}^m \log \left(\sum_{z^{(i)}} P(z^{(i)} | x^{(i)}, \theta^{j+1}) \right) = 0 \quad (5)$$

其中第 (4) 式用到了Jensen不等式，只不过和第二节的使用相反而已，第 (5) 式用到了概率分布累积为1的性质。

至此，我们得到了： $\sum_{i=1}^m \log P(x^{(i)} | \theta^{j+1}) - \sum_{i=1}^m \log P(x^{(i)} | \theta^j) \geq 0$ ，证明了EM算法的收敛性。

EM能收敛到全局极大值吗？

从上面的推导可以看出，EM 算法可以保证收敛到一个稳定点，但是却不能保证收敛到全局的极大值点，因此它是局部最优的算法，当然，如果我们的优化目标 $L(\theta, \theta^j)$ 是凸的，则EM算法可以保证收敛到全局极大值，这点和梯度下降法这样的迭代算法相同。至此我们也回答了上面提到的第二个问题。

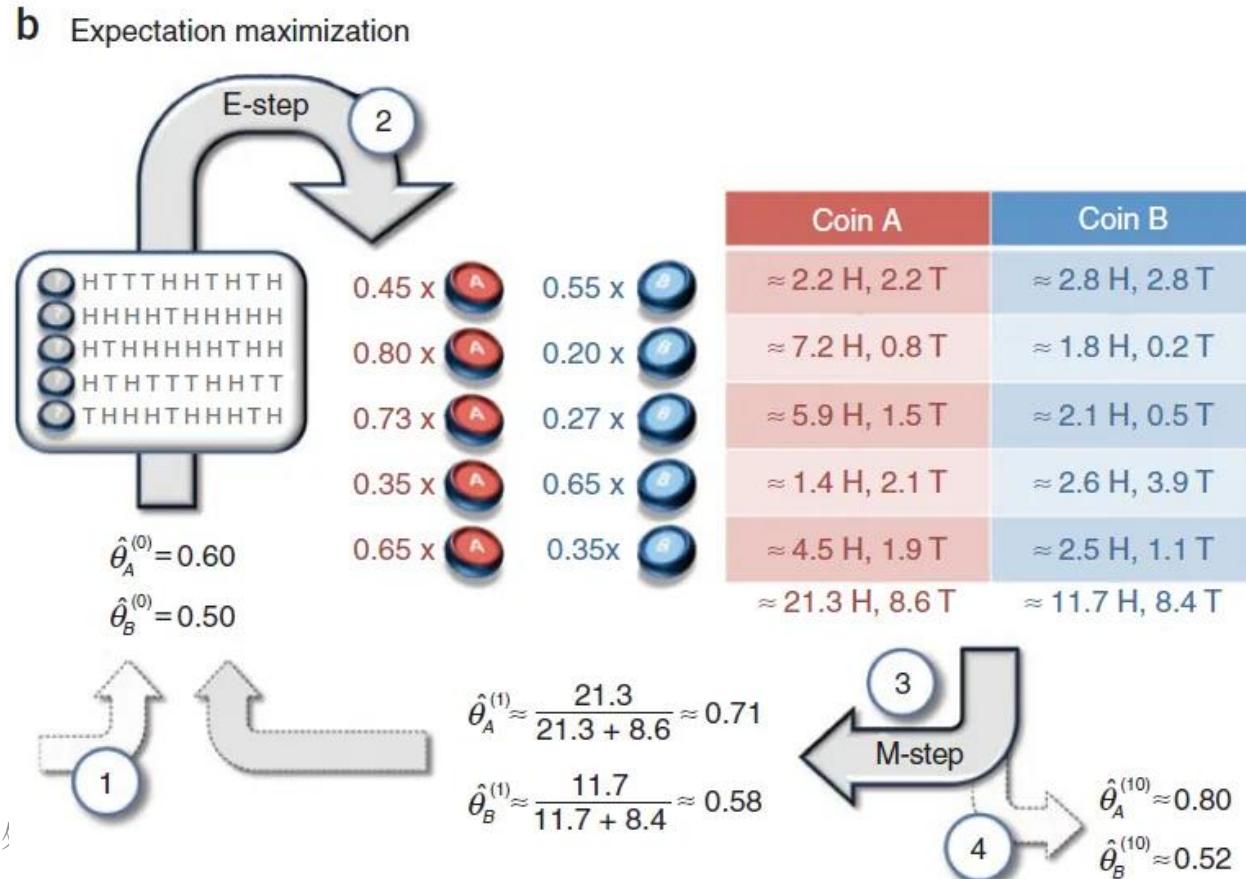
EM案例——抛硬币

- 两枚硬币A和B，如果知道每次抛的是A还是B，可以直接估计两枚硬币的正面概率（见图a）。
- 如果不知道每次抛的是A还是B，只观测到5轮，每轮循环10次，共计50次投币的结果，这时就没法直接估计A和B的正面概率，需要使用EM算法来解决（见图b）。

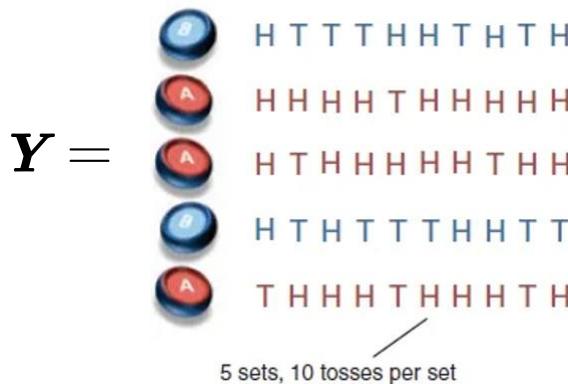
a Maximum likelihood

Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	
24 H, 6 T	9 H, 11 T

$$\hat{\theta}_A = \frac{24}{24 + 6} = 0.80$$

$$\hat{\theta}_B = \frac{9}{9 + 11} = 0.45$$


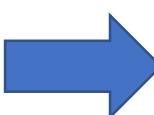
抛硬币 Case a



已知每个实验选择的是硬币A 还是硬币 B， 重点是如何计算输出的概率分布，这其实也是极大似然求导所得。

$$\begin{aligned} \underset{\theta}{\operatorname{argmax}} \log P(Y|\theta) &= \log((\theta_B^5(1-\theta_B)^5)(\theta_A^9(1-\theta_A))(\theta_A^8(1-\theta_A)^2)(\theta_B^4(1-\theta_B)^6)(\theta_A^7(1-\theta_A)^3)) \\ &= \log[(\theta_A^{24}(1-\theta_A)^6)(\theta_B^9(1-\theta_B)^{11})] \end{aligned}$$

上面这个式子求导之后发现，5 次实验中A正面向上的次数再除以总次数作为即为 $\hat{\theta}_A$ ， 5次实验中B正面向上的次数再除以总次数作为即为 $\hat{\theta}_B$ ， 即：

似然方程组 $\left\{ \begin{array}{l} \frac{24}{\theta_A} - \frac{6}{1-\theta_A} = 0 \\ \frac{9}{\theta_B} - \frac{11}{1-\theta_B} = 0 \end{array} \right.$  $\hat{\theta}_A = \frac{24}{24+6} = 0.80$

$$\hat{\theta}_B = \frac{9}{9+11} = 0.45$$

抛硬币 Case b

- 给定观测序列: $\mathbf{Y} =$

每一轮实验之间相互独立

H T T T H H T H T H
H H H H T H H H H H
H T H H H H H T H H
H T H T T T H H T T
T H H H T H H H T H
5 sets, 10 tosses per set

, 求解硬币A、B掷正面的概率

$$\boldsymbol{\theta} = \{\boldsymbol{\theta}_A, \boldsymbol{\theta}_B\}$$

- 假设隐变量为 $\mathbf{z} = \{\mathbf{A}, \mathbf{B}\}$

抛硬币 Case b

- E步

E步：初始化 $\hat{\theta}_A^{(0)} = 0.60$ 和 $\hat{\theta}_B^{(0)} = 0.50$ ，计算每个实验中选择的硬币是 A 和 B 的概率，例如第一个实验中选择 A 的概率为：

$$\begin{aligned} P(z = A|y_1, \theta) &= \frac{P(z = A, y_1|\theta)}{P(z = A, y_1|\theta) + P(z = B, y_1|\theta)} = \frac{(0.6)^5 * (0.4)^5}{(0.6)^5 * (0.4)^5 + (0.5)^{10}} \\ &= 0.45 \end{aligned}$$

$$P(z = B|y_1, \theta) = 1 - P(z = A|y_1, \theta) = 0.55$$

计算出每个实验为硬币 A 和硬币 B 的概率，然后进行加权求和。

抛硬币 Case b

- M步

M步: 求出似然函数下界 $Q(\theta, \theta^i)$, y_j 代表第 j 次实验正面朝上的个数, μ_j 代表第 j 次实验选择硬币 A 的概率, $1 - \mu_j$ 代表第 j 次实验选择硬币 B 的概率。

$$\begin{aligned} Q(\theta, \theta^i) &= \sum_{j=1}^5 \sum_z P(z|y_j, \theta^i) \log P(y_j, z|\theta) \\ &= \sum_{j=1}^5 \mu_j \log(\theta_A^{y_j} (1 - \theta_A)^{10-y_j}) + (1 - \mu_j) \log(\theta_B^{y_j} (1 - \theta_B)^{10-y_j}) \end{aligned}$$

针对 L 函数求导来对参数求导, 例如对 θ_A 求导:

$$\begin{aligned} \frac{\partial Q}{\partial \theta_A} &= \mu_1 \left(\frac{y_1}{\theta_A} - \frac{10 - y_1}{1 - \theta_A} \right) + \cdots + \mu_5 \left(\frac{y_5}{\theta_A} - \frac{10 - y_5}{1 - \theta_A} \right) = \mu_1 \left(\frac{y_1 - 10\theta_A}{\theta_A(1 - \theta_A)} \right) + \cdots \\ &\quad + \mu_5 \left(\frac{y_5 - 10\theta_A}{\theta_A(1 - \theta_A)} \right) \\ &= \frac{\sum_{j=1}^5 \mu_j y_j - \sum_{j=1}^5 10\mu_j \theta_A}{\theta_A(1 - \theta_A)} \end{aligned}$$

当然, 基于 Case a 我们也可以用一种更简单的方法求得:

$$\hat{\theta}_A^{(1)} = \frac{21.3}{21.3 + 8.6} = 0.71$$

$$\hat{\theta}_B^{(1)} = \frac{11.7}{11.7 + 8.4} = 0.58$$

求导等于 0 之后就可得到图中的第一次迭代之后的参数值:

$$\hat{\theta}_A^{(1)} = 0.71$$

$$\hat{\theta}_B^{(1)} = 0.58$$

抛硬币 Case b

第二轮迭代：基于第一轮EM计算好的 $\hat{\theta}_A^{(1)}, \hat{\theta}_B^{(1)}$ ，进行第二轮 EM，计算每个实验中选择的硬币是 A 和 B 的概率 (E步)，然后在计算M步，如此继续迭代……迭代十步之后

$$\hat{\theta}_A^{(10)} = 0.8, \hat{\theta}_B^{(10)} = 0.52$$

EM算法的应用

如果我们从算法思想的角度来思考EM算法，我们可以发现我们的算法里已知的是观察数据，未知的是隐含数据和模型参数，在E步，我们所做的事情是固定模型参数的值，优化隐含数据的分布，而在M步，我们所做的事情是固定隐含数据分布，优化模型参数的值。EM的应用包括：

- 支持向量机的SMO算法
- 混合高斯模型
- K-means
- 隐马尔可夫模型

3.3.3 序列标注问题

- 概率计算问题：在给定模型 $\lambda = (\pi, A, B)$ 和观测序列 $O = \{o_1, o_2, \dots, o_T\}$ ，求给定观测序列条件概率 $P(I|O)$ 最大的状态序列 $I = (i_1, i_2, \dots, i_T)$ 。
- 解题思路：将序列标注问题的求解目标，即隐状态序列的最大概率，对应为一种最优路径求解，需要随着时间步动态地搜索最优路径。
- 解决方法：
 - 维特比算法 Viterbi
 - 一种基于动态规划求解最优路径的算法
 - 最优路径的特性是：如果路径是最优的，那么路径上节点 i_t^* 到终点 i_T^* 的部分路径也是最优的。

介绍两个变量 δ 和 Ψ

δ 是到时刻 t 时 状态 i 的所有单个路径 (i_1, i_2, \dots, i_t) 概率最大值，记作

$$\delta_t(i) = \max_{i_1, i_2, \dots, i_{t-1}} P(i_{t=i}, i_{t-1}, \dots, i_1, o_t, \dots, o_1 | \lambda), \quad i = 1, 2, \dots, N$$

| 给定模型参数下，得到从 时刻1 到 时刻 t 的路径（状态序列）的概率最大值

递推公式为：

$$\delta_{t+1}(i) = P(i_{t+1=i}, i_t, \dots, i_1, o_{t+1}, \dots, o_1 | \lambda)$$

$$= \arg \max_{1 \leq j \leq N} [\delta_t(j) a_{ji}] b_i(o_{t+1}), \quad i = 1, 2, \dots, N; t = 1, 2, \dots, T - 1$$

| $\delta_{t+1}(i)$ ： $t + 1$ 时刻，状态 i 的概率最大值

| j ： t 时刻（上一时刻）的状态

Ψ 是在时刻 t 状态 i 中所有单个路径 (i_1, i_2, \dots, i_t) 概率最大的路径的第 $t - 1$ 个结点，记作

$$\Psi_t(i) = \arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}], \quad i = 1, 2, \dots, N$$

| Ψ 就是上一时刻的 δ 对应的结点

算法 (维特比算法)

输入：模型 $\lambda = (A, B, \pi)$ 和观测 $O = (o_1, o_2, \dots, o_T)$;

输出：最优路径 $I^* = (i_1^*, i_2^*, \dots, i_T^*)$ 。

(1) 初始化

$$\delta_1(i) = \pi_i b_i(o_1), \quad i = 1, 2, \dots, N$$

$$\Psi_1(i) = 0, \quad i = 1, 2, \dots, N$$

(2) 递推。对 $t = 2, 3, \dots, T$

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}] b_i(o_t), \quad i = 1, 2, \dots, N$$

$$\Psi_t(i) = \arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}], \quad i = 1, 2, \dots, N$$

(3) 终止

$$P^* = \max_{1 \leq i \leq N} \delta_T(i)$$

$$i_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$$

盒子摸球案例

例

模型 $\lambda = (A, B, \pi)$,

$$A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}, \quad \pi = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

已知观测序列 $O = (\text{红}, \text{白}, \text{红})$, 试求最优状态序列, 即最优路径 $I^* = (i_1^*, i_2^*, i_3^*)$ 。

(1) 初始化。 $t = 1$ 时, 对每一个状态 i , $i = 1, 2, 3$, 求状态 i 观测 o_1 为红的概率, 记为 $\delta_1(i)$

$$\delta_1(i) = \pi_i b_i(o_1) = \pi_i b_i(\text{红})$$

$$\delta_1(1) = 0.2 * 0.5 = 0.1, \delta_1(2) = 0.4 * 0.4 = 0.16, \delta_1(3) = 0.4 * 0.7 = 0.28$$

$$\Psi_1(i) = 0, i = 1, 2, 3$$

(2) $t = 2$, 对每个状态 $i, i = 1, 2, 3$, 求在 $t=1$ 时状态为 j 观测为红, 在 $t = 2$ 时状态为 i 观测 o_2 为白的路径的最大概率, 记为 $\delta_2(i)$

| 感觉把 δ 的意思讲明白了

$$\delta_2(i) = \max_{1 \leq j \leq 3} [\delta_1(j) a_{ji} b_i(o_2)]$$

对每个状态 i , $i = 1, 2, 3$, 记录概率最大路径的前一个状态 j

$$\Psi_2(i) = \arg \max_{1 \leq j \leq 3} [\delta_1(j) a_{ji}], i = 1, 2, 3$$

状态1:

$$\begin{aligned}\delta_2(1) &= \max_{1 \leq j \leq 3} [\delta_1(j)a_{j1}]b_1(o_2) = \max_j [\delta_1(1)a_{11}, \delta_1(2)a_{21}, \delta_1(3)a_{31}]b_1(o_2) \\ &= \max_j [0.1 * 0.5, 0.16 * 0.3, 0.28 * 0.2] * 0.5 = \max_j [0.05, 0.048, 0.056] * 0.5 \\ &= 0.056 * 0.5 = 0.028\end{aligned}$$

$$\Psi_2(1) = 3$$

状态2:

$$\begin{aligned}\delta_2(2) &= \max_{1 \leq j \leq 3} [\delta_1(j)a_{j2}]b_2(o_2) = \max_j [\delta_1(1)a_{12}, \delta_1(2)a_{22}, \delta_1(3)a_{32}]b_2(o_2) \\ &= \max_j [0.1 * 0.2, 0.16 * 0.5, 0.28 * 0.3] * 0.6 = \max_j [0.02, 0.08, 0.084] * 0.6 \\ &= 0.084 * 0.6 = 0.0504\end{aligned}$$

$$\Psi_2(2) = 3$$

状态3:

$$\begin{aligned}\delta_2(3) &= \max_{1 \leq j \leq 3} [\delta_1(j)a_{j3}]b_3(o_2) = \max_j [\delta_1(1)a_{13}, \delta_1(2)a_{23}, \delta_1(3)a_{33}]b_3(o_2) \\ &= \max_j [0.1 * 0.3, 0.16 * 0.2, 0.28 * 0.5] * 0.3 = \max_j [0.03, 0.032, 0.14] * 0.3 = 0.14 \\ &\quad * 0.3 = 0.042\end{aligned}$$

$$\Psi_2(3) = 3$$

(3) t = 3 ,

$$\delta_3(i) = \max_{1 \leq j \leq 3} [\delta_2(j)a_{ji}b_i(o_3)]$$

状态1:

$$\begin{aligned}\delta_3(1) &= \max_{1 \leq j \leq 3} [\delta_2(j)a_{j1}]b_1(o_3) = \max_j [\delta_2(1)a_{11}, \delta_2(2)a_{21}, \delta_2(3)a_{31}]b_1(o_3) \\ &= \max_j [0.028 * 0.5, 0.0504 * 0.3, 0.042 * 0.2] * 0.5 = \max_j [0.014, 0.01512, 0.0084] * 0.5 \\ &= 0.1512 * 0.5 = 0.00756\end{aligned}$$

$$\Psi_3(1) = 2$$

状态2:

$$\begin{aligned}\delta_3(2) &= \max_{1 \leq j \leq 3} [\delta_2(j)a_{j2}]b_2(o_3) = \max_j [\delta_2(1)a_{12}, \delta_2(2)a_{22}, \delta_2(3)a_{32}]b_2(o_3) \\ &= \max_j [0.028 * 0.2, 0.0504 * 0.5, 0.042 * 0.3] * 0.4 \\ &= \max_j [0.0056, 0.0252, 0.0126] * 0.4 = 0.0252 * 0.4 = 0.01008\end{aligned}$$

$$\Psi_3(2) = 2$$

状态3：

$$\begin{aligned}\delta_3(3) &= \max_{1 \leq j \leq 3} [\delta_2(j)a_{j3}]b_3(o_3) = \max_j [\delta_2(1)a_{13}, \delta_2(2)a_{23}, \delta_2(3)a_{33}]b_3(o_3) \\ &= \max_j [0.028 * 0.3, 0.0504 * 0.2, 0.042 * 0.5] * 0.7 \\ &= \max_j [0.0084, 0.01008, 0.021] * 0.7 = 0.021 * 0.7 = 0.0147\end{aligned}$$

$$\Psi_3(3) = 3$$

(3) P^* 表示最优路径的概率

$$P^* = \max_{1 \leq i \leq 3} \delta_3(i) = 0.0147$$

最优路径的终点 $i_3^* = \arg \max_i [\delta_3(i)] = 3$

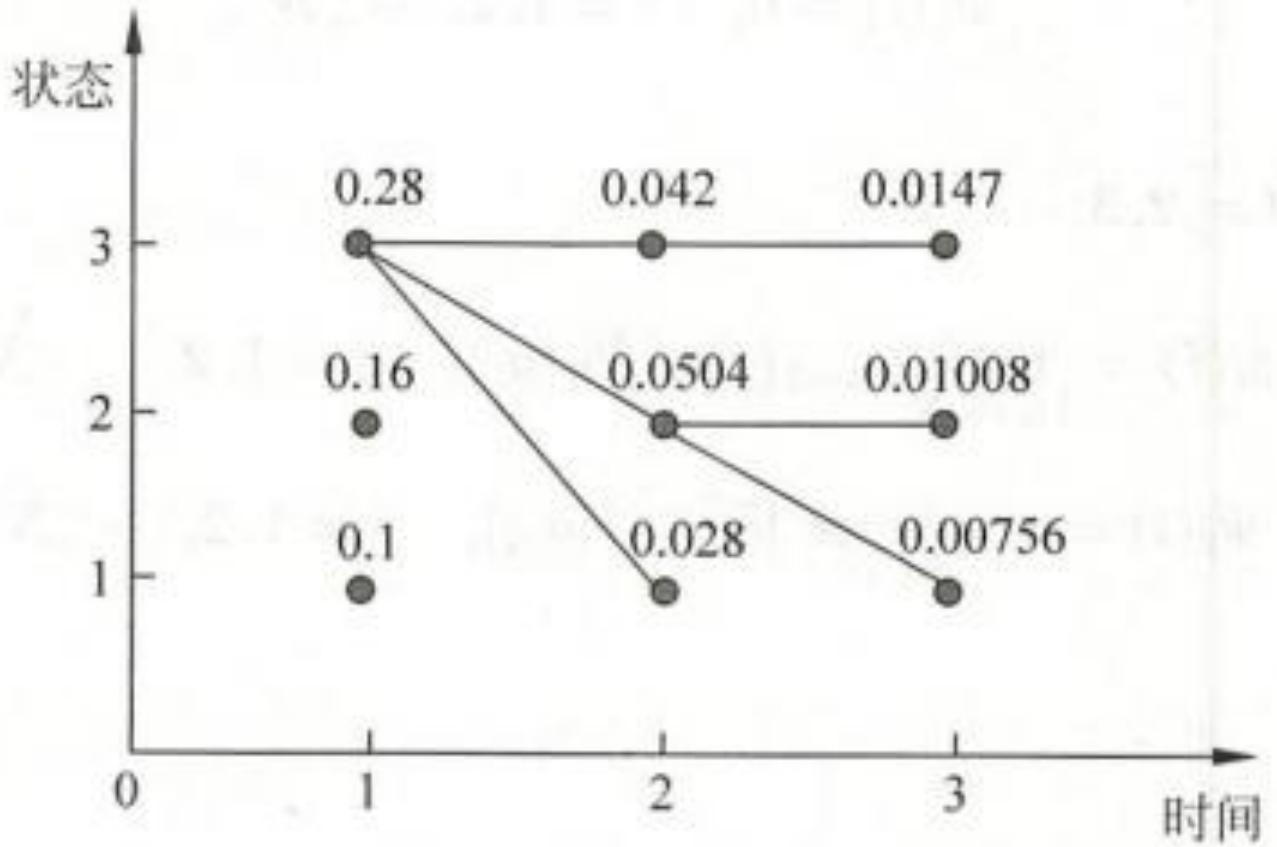
(4) 反推最优路径 i_2^*, i_1^* ：

在 $t=2$ 时, $i_2^* = \Psi_3(i_3^*) = \Psi_3(3) = 3$

在 $t=1$ 时, $i_1^* = \Psi_2(i_2^*) = \Psi_2(3) = 3$

所以, 最优状态序列 $I^* = (3, 3, 3)$

《自然语言处理入门》



求最优路径

知乎 @Nash

```
### 序列标注问题和维特比算法
def viterbi_decode(0):
    """
    输入:
    0: 观测序列
    输出:
    path: 最优隐状态路径
    """

    # 序列长度和初始观测
    T, o = len(0), 0[0]
    # 初始化delta变量
    delta = pi * B[:, o]
    # 初始化varphi变量
    varphi = np.zeros((T, 4), dtype=int)
    path = [0] * T
    # 递推
    for i in range(1, T):
        delta = delta.reshape(-1, 1)
        tmp = delta * A
        varphi[i, :] = np.argmax(tmp, axis=0)
        delta = np.max(tmp, axis=0) * B[:, 0[i]]
    # 终止
    path[-1] = np.argmax(delta)
    # 回溯最优路径
    for i in range(T-1, 0, -1):
        path[i-1] = varphi[i, path[i]]
    return path

# 给定观测序列
0 = [1, 0, 1, 1, 0]
# 输出最可能的隐状态序列
print(viterbi_decode(0))
```

[0, 1, 2, 3, 3]

4 条件随机场 CRF

- 自学



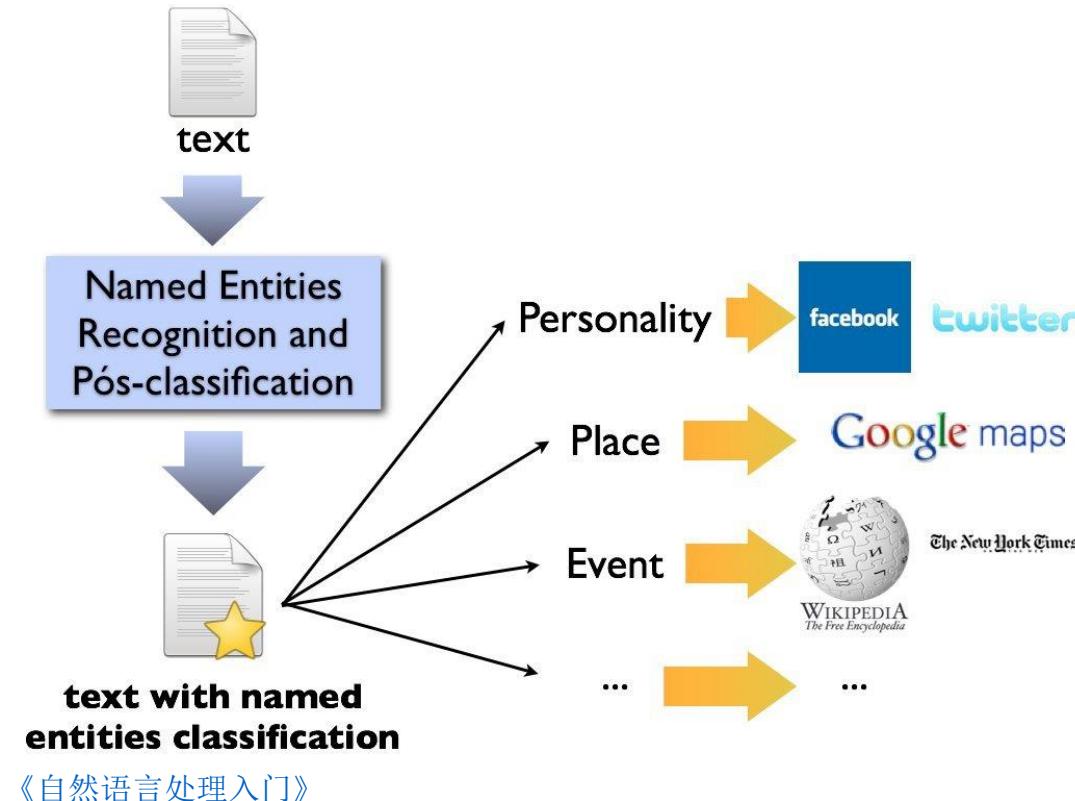
5 命名实体识别

- 识别出句子中命名实体的边界与类别的任务称为命名实体识别（Named Entity Recognition, NER）
 - 对于规则性较强的命名实体，比如邮箱、ISBN、商品编号，完全可以通过正则表达式处理。
 - 对于较短的命名实体，比如人名，完全可以通过分词确定边界，通过词性标注模块确定类别。
 - 在另一些语料库中（如PKU等），机构名这样的复合词是拆开的，此时就需要一个专门的命名实体识别模块了。



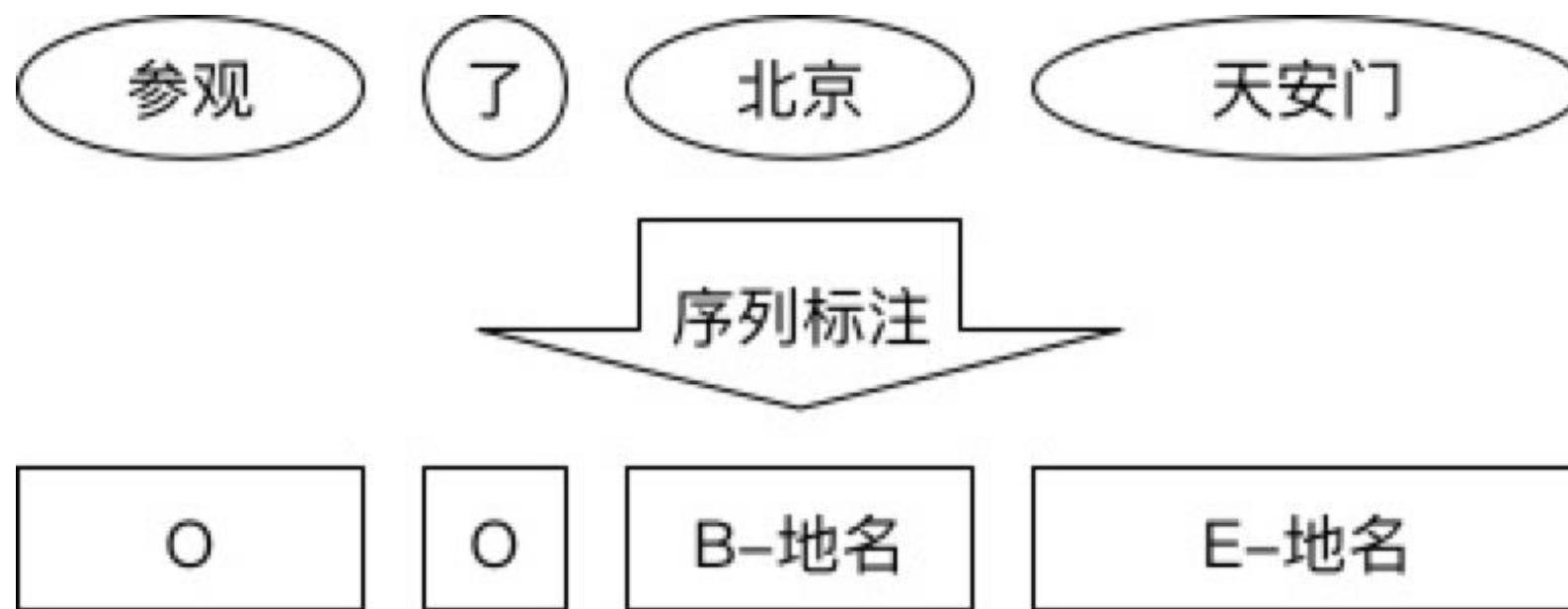
命名实体

- 文本中有一些描述实体的词汇，比如人名、地名、组织机构名、股票基金、医学术语等，称为命名实体（named entity）。
 - 数量无穷
 - 构词灵活
 - 类别模糊



序列标注与命名实体识别

- 命名实体识别可以转换为一个序列标注问题，可以采用 {B, M, E, O, S} 标注集。
- 地名和机构名常常由多个单词组成（称为复合词）





同学们，HMM 命名实体识别模型如何训练呢？

6 总结

- 隐马尔可夫模型的要点：
 - 2种状态：隐状态、观测状态
 - 2个假设：齐次马尔可夫假设、观测独立性假设
 - 3要素：初始概率向量、状态转移概率矩阵、观测概率矩阵
 - 3个经典问题：概率计算问题、参数估计问题、序列标注问题
- 隐马尔可夫模型作为入门模型，比较容易上手，同时也是许多高级模型的基础

