

法律声明

□ 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

■ 微信公众号：小象

■ 新浪微博：ChinaHadoop



隐马尔科夫模型



小象学院
ChinaHadoop.cn

邹博

主要内容

□ 隐马尔科夫模型

- 概率计算
- 参数估计
- 模型预测

□ 中文分词算法实践

- 思考：实践问题应该如何建模？

中文分词

前言 |

数据 |， | 数据 |， | 数据 |！ | 想 | 必在
等 | 媒介 | 的 | 持续 | 冲击 | 下 |， | 人们
的 | 洗礼 |。 | 现实 | 需求 | 推动 | 了 | 对
这些 | 数据 | 来 | 自于 | 社交 | 媒体 |、 |
“ | 物联网 |” |) |、 | 传感器 | 等 | 任何
大 | 多 | 数数 | 据 | 挖掘 | 的 | 宣传 | 着 |
数据 | 洪 | 水(| **data flood**) | 的 | 预言 |
数据 |， | 硬件 | 推销 | 人员 | 会进 | 一步
能够 | 满足 | 处理 | 速度 | 的 | 要求 |。 |
对 | 的 |， | 但 | 是 | 我们 | 值得 | 停下 |
务 | 进行 | 适当 | 的 | 再 | 认识 |。 |

近 | 年来 |， | 数据 | 挖掘 | 和 | 机器 | 学习 | 在 | 我们 | 周围 | 持续 | 火爆 |， | 各种 |
媒体 | 也 | 不断 | 推送 | 着 | 海量 | 的 | 数据 |。 | 仔细 | 观察 | 就 | 能 | 发现 |， | 实
际 | 应用 | 中 | 的 | 那些 | 机器 | 学习 | 算法 | 与 | 多 | 年前 | 并 | 没有 | 什么 | 两样 |
； | 它们 | 只 | 是 | 在 | 应用 | 的 | 数据 | 规模 | 上 | 有些 | 不同 |。 | 历数 | 一 | 下 |
产生 | 数据 | 的 | 组织 |， | 至少 | 在 | 我 | 看来 |， | 数目 | 其实 | 并 | 不 | 多 |。 |
无非 | 是 | Google |、 | Facebook |、 | Twitter |、 | NetFlix | 以及 | 其 | 他 | 为数 | 不
| 多 | 的 | 机构 | 在 | 使用 | 若 | 干学 | 习算法 | 和 | 工具 |， | 这些 | 算法 | 和 | 工具 |
使 | 得 | 他们 | 能够 | 对 | 数据 | 进行 | 测试 | 分析 |。 | 那么 |， | 真正 | 的 | 问题 |
是 |： | “ | 对于 | 其 | 他人 |， | 大数 | 据 | 框架 | 下 | 的 | 算法 | 和 | 工具 | 的 | 作用 |
是 | 什么 | 呢 |？ |” |

我承认 | 本书 | 将 | 多 | 次 | 提及 | 大 | 数据 | 和 | 机器 | 学习 | 之间 | 的 | 关系 |， |
这 | 是 | 我 | 无法 | 忽视 | 的 | 一个 | 客观 | 问题 |； | 但 | 是 | 它 | 只 | 是 | 一个 | 很
| 小 | 的 | 因素 |， | 终极 | 目标 | 是 | 如何 | 利用 | 可用 | 数据 | 获取 | 数据 | 的 | 本质 |

```
if __name__ == "__main__":
    pi, A, B = load_train()
    f = file("./text\\novel.txt")
    data = f.read()[3:].decode('utf-8')
    f.close()
    decode = viterbi(pi, A, B, data)
    segment(data, decode)
```

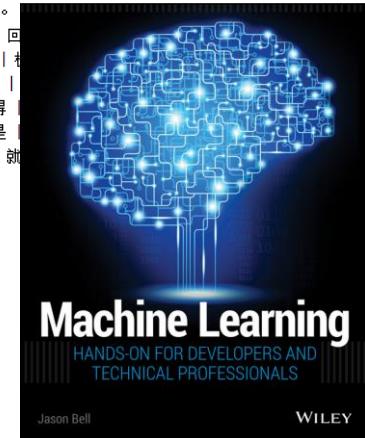
bug HMM

Connected to pydev debugger (build 139.1001)

我 | 与 | 地坛 |
史 | 铁生 |

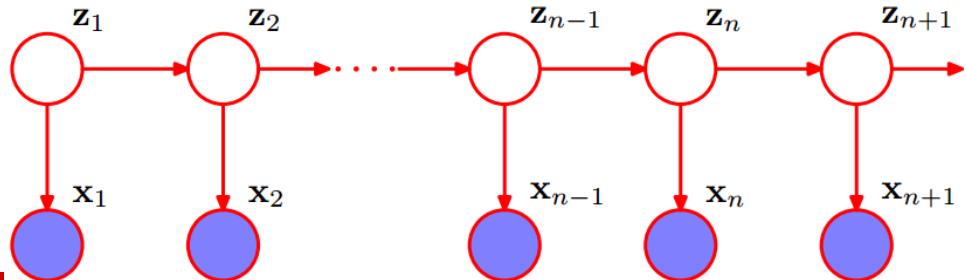
一 |
我 | 在 | 好 | 几 | 篇 | 小说 | 中 | 都 | 提到 | 过 | 一 | 座 | 废弃 | 的 | 古园 |， | 实际 | 就 | 是 | 地坛 |。 | 许多 | 年前 | 旅游业 | 还 | 没有 | 开
地坛 | 离 | 我家 | 很 | 近 |。 | 或者 | 说 | 我家 | 离 | 地坛 | 很 | 近 |。 | 总之 |， | 只好 | 认为 | 这 | 是 | 缘分 |。 | 地坛 | 在 | 我出 | 生前 |
它 | 等待 | 我 | 出生 |， | 然后 | 又 | 等待 | 我活 | 到 | 最狂 | 妄 | 的 | 年龄 | 上 | 忽地 | 残废 | 了 | 双腿 |。 | 四百 | 多 | 年里 |， | 它 | 一面 |
自 | 从 | 那个 | 下午 | 我 | 无意 | 中进 | 了 | 这 | 园子 |， | 就 | 再 | 没长 | 久 | 地离 | 开过 | 它 |。 | 我 | 一 | 下子 | 就 | 理解 | 了 | 它 | 的 |
两 | 条 | 腿 | 残废 | 后 | 的 | 最初 | 几年 |， | 我找 | 不 | 到 | 工作 |， | 找 | 不到 | 去路 |， | 忽然 | 间 | 几乎 | 什么 | 都 | 找 | 不到 | 了 |，
除 | 去 | 几 | 座 | 殿堂 | 我 | 无法 | 进去 |， | 除去 | 那 | 座 | 祭坛 | 我 | 不 | 能 | 上去 | 而 | 只能 | 从 | 各个 | 角度 | 张望 | 它 |， | 地坛 | 的 |
剩 | 下 | 的 | 就 | 是 | 是 | 怎样活 | 的 | 问题 | 了 |， | 这 | 却 | 不 | 是 | 在 | 某 | 一个 | 瞬间 | 就 | 能 | 完全 | 想透 | 的 |、 | 不 | 是 | 一次性 | 都

二 |
我 | 才 | 想到 |， | 当年 | 我 | 总是 | 独自 | 跑 | 到 | 地坛 | 去 |， | 曾 | 经给 | 母亲 | 出 | 了 | 一个 | 怎样 | 的 | 难题 |。 |
她 | 不 | 是 | 那种 | 光会 | 疼爱 | 儿子 | 而 | 不 | 懂得 | 理解 | 儿子 | 的 | 母亲 |。 |
有 | 一 | 回 | 我摇 | 车出 | 了 | 小院 |； | 想 | 起 | 一件 | 什么 | 事 | 又 | 返身 | 回 |
有 | 一次 | 与 | 一个 | 作家 | 朋友 | 聊天 |， | 我问 | 他学 | 写作 | 的 | 最 | 初动 | 情 |
在 | 我 | 的 | 头 | 一 | 篇 | 小 | 说 | 发表 | 的 | 时候 |， | 在 | 我 | 的 | 小 | 说 |
只 | 是 | 到 | 了 | 这 | 时候 |， | 纷纭 | 的 | 往事 | 才 | 在 | 我 | 前眼 | 幻现 | 得 |
摇 | 着 | 轮椅 | 在 | 园中 | 慢慢 | 走 |， | 又 | 是 | 雾霭 | 的 | 清晨 |， | 又 | 是 |
曾 | 有 | 过好 | 多 | 回 |， | 我 | 在 | 这园 | 子里 | 呆得 | 太久 | 了 |， | 母亲 | 就 |



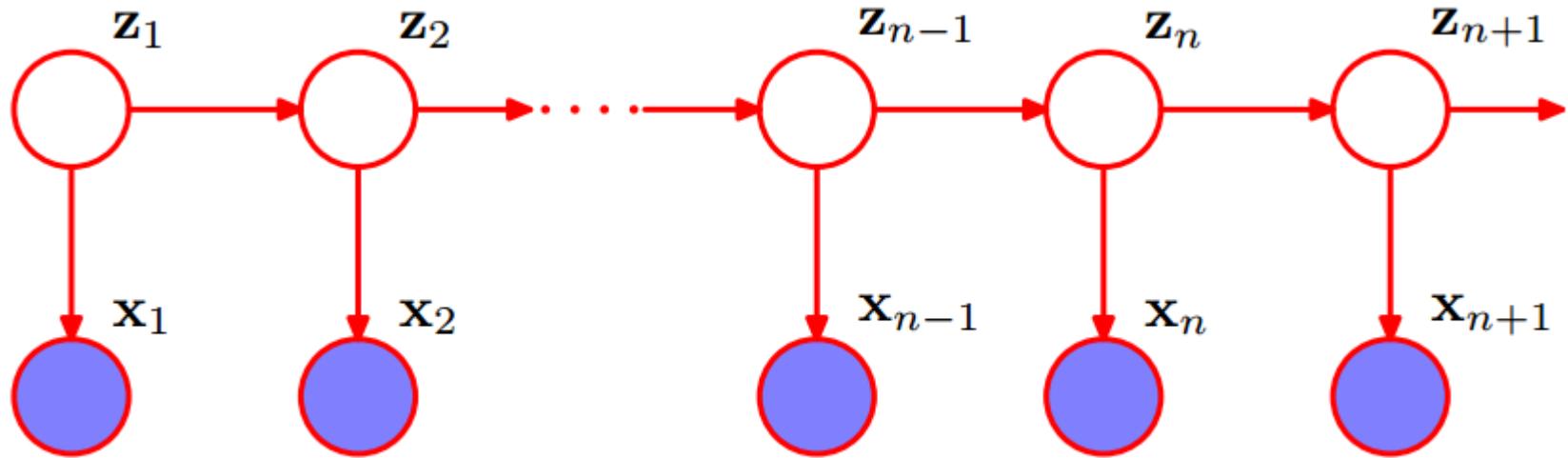
Jason Bell. *Machine Learning: Hands-On for Developers and Technical Professionals*. Wiley.2015

HMM定义



- 隐马尔科夫模型(HMM, Hidden Markov Model)可用标注问题，在语音识别、NLP、生物信息、模式识别等领域被实践证明是有效的算法。
- HMM是关于时序的概率模型，描述由一个隐藏的马尔科夫链生成不可观测的状态随机序列，再由各个状态生成观测随机序列的过程。
- 隐马尔科夫模型随机生成的状态随机序列，称为状态序列；每个状态生成一个观测，由此产生的观测随机序列，称为观测序列。
 - 序列的每个位置可看做是一个时刻。

隐马尔科夫模型的贝叶斯网络



□ 请思考：

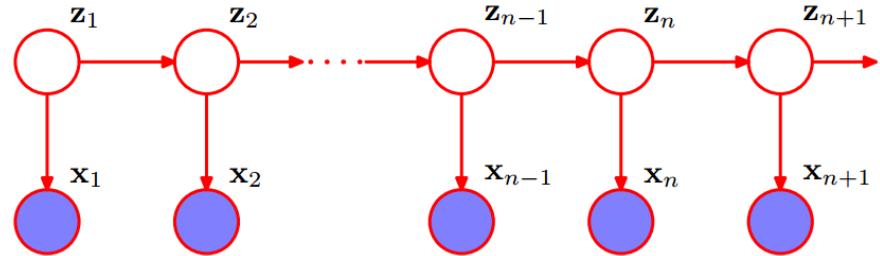
- 在 z_1 、 z_2 不可观察的前提下， x_1 和 z_2 独立吗？ x_1 和 x_2 独立吗？

HMM的确定

- HMM由初始概率分布 π 、状态转移概率分布A以及观测概率分布B确定。

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

HMM的参数



- Q是所有可能的状态的集合
 - N是可能的状态数
- V是所有可能的观测的集合
 - M是可能的观测数

$$Q = \{q_1, q_2, \dots, q_N\}$$

$$V = \{v_1, v_2, \dots, v_M\}$$

HMM的参数 $\lambda = (A, B, \pi)$

- I是长度为T的状态序列，O是对应的观测序列

$$I = \{i_1, i_2, \dots, i_T\} \quad O = \{o_1, o_2, \dots, o_T\}$$

- A是状态转移概率矩阵

$$A = [a_{ij}]_{N \times N}$$

- 其中 $a_{ij} = P(i_{t+1} = q_j | i_t = q_i)$

- a_{ij} 是在时刻t处于状态 q_i 的条件下时刻t+1转移到状态 q_j 的概率。

HMM的参数 $\lambda = (A, B, \pi)$

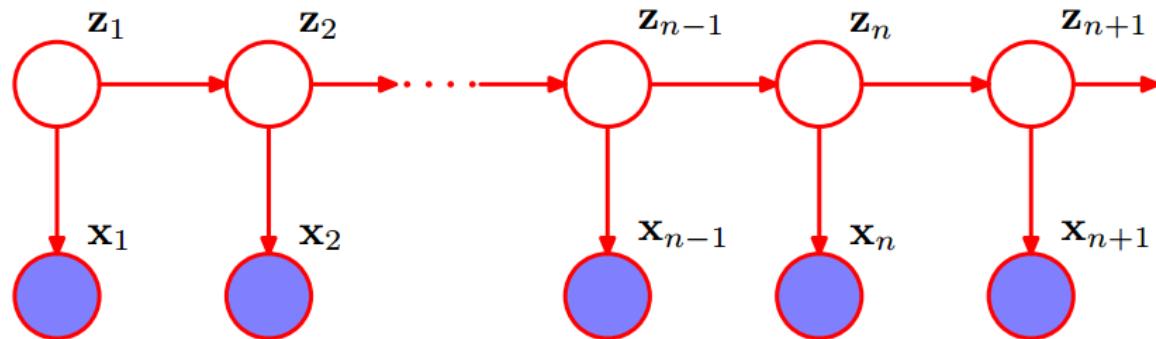
- B是观测概率矩阵 $B = [b_{ik}]_{N \times M}$
- 其中， $b_{ik} = P(O_t = v_k | i_t = q_i)$
 - b_{ik} 是在时刻t处于状态 q_i 的条件下生成观测 v_k 的概率。
- π 是初始状态概率向量： $\pi = (\pi_i)$
- 其中， $\pi_i = P(i_1 = q_i)$
 - π_i 是时刻t=1处于状态 q_i 的概率。

HMM的参数总结

□ HMM由初始概率分布 π (向量)、状态转移概率分布 A (矩阵)以及观测概率分布 B (矩阵)确定。 π 和 A 决定状态序列， B 决定观测序列。因此，HMM可以用三元符号表示，称为HMM的三要素：

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

HMM的两个基本性质



□ 齐次假设：

$$P(i_t | i_{t-1}, o_{t-1}, i_{t-2}, o_{t-2} \dots i_1, o_1) = P(i_t | i_{t-1})$$

□ 观测独立性假设：

$$P(o_t | i_T, o_T, i_{T-1}, o_{T-1} \dots i_1, o_1) = P(o_t | i_t)$$

HMM举例

- 假设有3个盒子，编号为1、2、3，每个盒子都装有红白两种颜色的小球，数目如下：

盒子号	1	2	3
红球数	5	4	7
白球数	5	6	3

- 按照下面的方法抽取小球，得到球颜色的观测序列：
 - 按照 $\pi=(0.2,0.4,0.4)$ 的概率选择1个盒子，从盒子随机抽出1个球，记录颜色后放回盒子；
 - 按照某条件概率(下页)选择新的盒子，重复上述过程；
 - 最终得到观测序列：“红红白白红”。

该示例的各个参数

- 状态集合: $Q=\{\text{盒子1}, \text{盒子2}, \text{盒子3}\}$
- 观测集合: $V=\{\text{红}, \text{白}\}$
- 状态序列和观测序列的长度 $T=5$
- 初始概率分布 π :
- 状态转移概率分布 A :
- 观测概率分布 B :

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

思考：

- 在给定参数 π 、A、B的前提下，得到观测序列“红红白白红”的概率是多少？

HMM的3个基本问题

- 概率计算问题：前向-后向算法——动态规划
 - 给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = \{o_1, o_2, \dots, o_T\}$ ，计算模型 λ 下观测序列 O 出现的概率 $P(O | \lambda)$
- 学习问题：Baum-Welch算法（状态未知）——EM
 - 已知观测序列 $O = \{o_1, o_2, \dots, o_T\}$ ，估计模型 $\lambda = (A, B, \pi)$ 的参数，使得在该模型下观测序列 $P(O | \lambda)$ 最大
- 预测问题：Viterbi算法——动态规划
 - 解码问题：已知模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = \{o_1, o_2, \dots, o_T\}$ 求给定观测序列条件概率 $P(I | O, \lambda)$ 最大的状态序列 I

概率计算问题

- 直接算法
 - 暴力算法
- 前向算法
- 后向算法
 - 这二者是理解HMM的算法重点

直接计算法

- 按照概率公式，列举所有可能的长度为T的状态序列 $I = \{i_1, i_2, \dots, i_T\}$ ，求各个状态序列I与观测序列 $O = \{o_1, o_2, \dots, o_T\}$ 的联合概率 $P(O, I | \lambda)$ ，然后对所有可能的状态序列求和，从而得到 $P(O | \lambda)$

直接计算法

□ 状态序列 $I = \{i_1, i_2, \dots, i_T\}$ 的概率是：

$$P(I|\lambda) = \pi_{i_1} a_{i_1 i_2} a_{i_2 i_3} \cdots a_{i_{T-1} i_T}$$

□ 对固定的状态序列I，观测序列O的概率是：

$$P(O|I, \lambda) = b_{i_1 o_1} b_{i_2 o_2} \cdots b_{i_T o_T}$$

□ O和I同时出现的联合概率是：

$$P(O, I|\lambda) = P(O|I, \lambda)P(I|\lambda) = \pi_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} \cdots a_{i_{T-1} i_T} b_{i_T o_T}$$

$$P(I|\lambda) = \pi_{i_1} a_{i_1 i_2} a_{i_2 i_3} \cdots a_{i_{T-1} i_T}$$

直接计算法 $P(O|I, \lambda) = b_{i_1 o_1} b_{i_2 o_2} \cdots b_{i_T o_T}$

□ O和I同时出现的联合概率是：

$$\begin{aligned} P(O, I | \lambda) &= P(O|I, \lambda)P(I|\lambda) \\ &= \pi_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} \cdots a_{i_{T-1} i_T} b_{i_T o_T} \end{aligned}$$

□ 对所有可能的状态序列I求和，得到观测序列O的概率 $P(O|\lambda)$

$$\begin{aligned} P(O|\lambda) &= \sum_I P(O, I | \lambda) = \sum_I P(O|I, \lambda)P(I|\lambda) \\ &= \sum_{i_1, i_2, \dots, i_T} \pi_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} \cdots a_{i_{T-1} i_T} b_{i_T o_T} \end{aligned}$$

直接计算法分析

- 对于最终式

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

$$= \sum_{i_1, i_2, \dots, i_T} \pi_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} \cdots a_{i_{T-1} i_T} b_{i_T o_T}$$

- 分析：加和符号中有 $2T$ 个因子， I 的遍历个数为 N^T ，因此，时间复杂度为 $O(T N^T)$ ，复杂度过高。

借鉴算法的优化思想

□ 最长递增子序列

- 给定一个长度为N的数组，求该数组的一个最长的单调递增的子序列(不要求连续)。
- 数组：5, 6, 7, 1, 2, 8 的LIS：5, 6, 7, 8

□ 最大连续子数组

- 给定一个长度为N的数组，求该数组中连续的一段数组(子数组)，使得该子数组的和最大。
 - 数组：1, -2, 3, 10, -4, 7, 2, -5,
 - 最大子数组：3, 10, -4, 7, 2

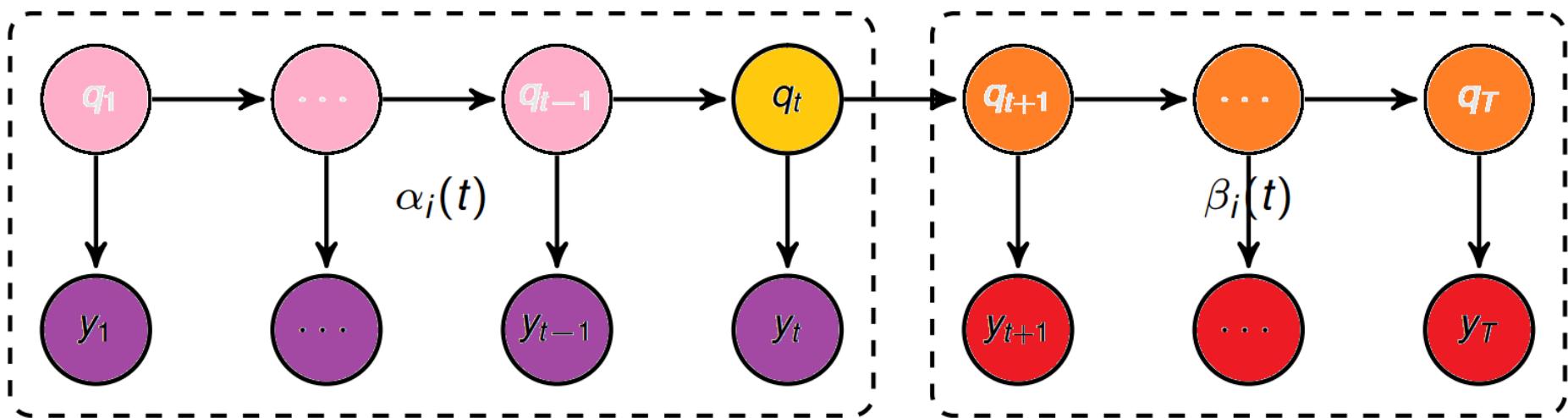
□ KMP中next数组的计算

模式串	a	b	a	a	b	c	a	b	a
next	-1	0	0	1	1	2	0	1	2

定义：前向概率-后向概率

$$\alpha_i(t) = p(y_1, y_2, \dots, y_t, q_t = i | \lambda)$$

$$\beta_i(t) = p(y_{t+1}, \dots, y_T | q_t = i, \lambda)$$



前向算法

- 定义：给定 λ ，定义到时刻t部分观测序列为 $o_1, o_2 \dots o_t$ 且状态为 q_i 的概率称为前向概率，记做：

$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, i_t = q_i | \lambda)$$

- 可以递推计算前向概率 $\alpha_t(i)$ 及观测序列概率 $P(O|\lambda)$

前向算法 $\alpha_t(i) = P(o_1, o_2, \dots, o_t, i_t = q_i | \lambda)$

□ 初值： $\alpha_1(i) = \pi_i b_{io_1}$

□ 递推： 对于 $t=1, 2, \dots, T-1$

$$\alpha_{t+1}(i) = \left(\sum_{j=1}^N \alpha_t(j) a_{ji} \right) b_{io_{t+1}}$$

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

前向算法 $\alpha_t(i) = P(o_1, o_2, \dots, o_t, i_t = q_i | \lambda)$

□ 思考：前向算法的时间复杂度是O(N²T)。

□ 为什么？

- 重点考察第二步：
- 递推：对于t=1,2...T-1

$$\alpha_{t+1}(i) = \left(\sum_{j=1}^N \alpha_t(j) a_{ji} \right) b_{io_{t+1}}$$

例：盒子球模型

- 考察盒子球模型，计算观测向量O=“红白 红”的出现概率。

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

解：盒子球模型

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

□ 计算初值

$$\alpha_1(1) = \pi_1 b_{1o_1} = 0.2 \times 0.5 = 0.1$$

$$\alpha_1(2) = \pi_2 b_{2o_1} = 0.4 \times 0.4 = 0.16$$

$$\alpha_1(3) = \pi_3 b_{3o_1} = 0.4 \times 0.7 = 0.28$$

解

$$A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \quad B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix} \quad \alpha_1(1) = \pi_1 b_{1o_1} = 0.2 \times 0.5 = 0.1 \\ \alpha_1(2) = \pi_2 b_{2o_1} = 0.4 \times 0.4 = 0.16 \\ \alpha_1(3) = \pi_3 b_{3o_1} = 0.4 \times 0.7 = 0.28$$

□ 递推

$$\begin{aligned}\alpha_2(1) &= \left(\sum_{j=1}^N \alpha_1(j) a_{j1} \right) b_{1o_2} \\ &= (0.1 \times 0.5 + 0.16 \times 0.3 + 0.28 \times 0.2) \times 0.5 \\ &= 0.077\end{aligned}$$

$$\begin{array}{ll}\alpha_2(2) = 0.1104 & \alpha_3(1) = 0.04187 \\ \alpha_2(3) = 0.0606 & \alpha_3(2) = 0.03551 \\ & \alpha_3(3) = 0.05284\end{array}$$

解：盒子球模型

□ 最终

$$P(O|\lambda) = \sum_{i=1}^3 \alpha_3(i)$$

$$\alpha_3(1) = 0.04187$$

$$\alpha_3(2) = 0.03551$$

$$\alpha_3(3) = 0.05284$$

$$= 0.04187 + 0.03551 + 0.05284$$

$$= 0.13022$$

后向算法

□ 定义：给定 λ ，定义到时刻t状态为 q_i 的前提下，从 $t+1$ 到 T 的部分观测序列为 $O_{t+1}, O_{t+2} \dots O_T$ 的概率为后向概率，记做：

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots, O_T | i_t = q_i, \lambda)$$

□ 可以递推计算后向概率 $\beta_t(i)$ 及观测序列概率 $P(O|\lambda)$

后向算法 $\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | i_t = q_i, \lambda)$

- 初值： $\beta_T(i) = 1$
- 递推： 对于 $t=T-1, T-2, \dots, 1$

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

- 最终： $P(O|\lambda) = \sum_{i=1}^N \pi_i b_{io_1} \beta_1(i)$

后向算法的说明

- 为了计算在时刻t状态为 q_i 条件下时刻 $t+1$ 之后的观测序列为 $O_{t+1}, O_{t+2} \dots O_T$ 的后向概率 $\beta_t(i)$, 只需要考虑在时刻 $t+1$ 所有可能的N个状态 q_j 的转移概率(a_{ij} 项), 以及在此状态下的观测 O_{t+1} 的观测概率(b_{jot+1} 项), 然后考虑状态 q_j 之后的观测序列的后向概率 $\beta_{t+1}(j)$

$$\alpha_i(t) = p(y_1, y_2, \dots, y_t, q_t = i | \lambda)$$

$$\beta_i(t) = p(y_{t+1}, \dots, y_T | q_t = i, \lambda)$$

前后向关系

□ 根据定义：

$$P(i_t = q_i, O | \lambda)$$

$$= P(O | i_t = q_i, \lambda) P(i_t = q_i | \lambda)$$

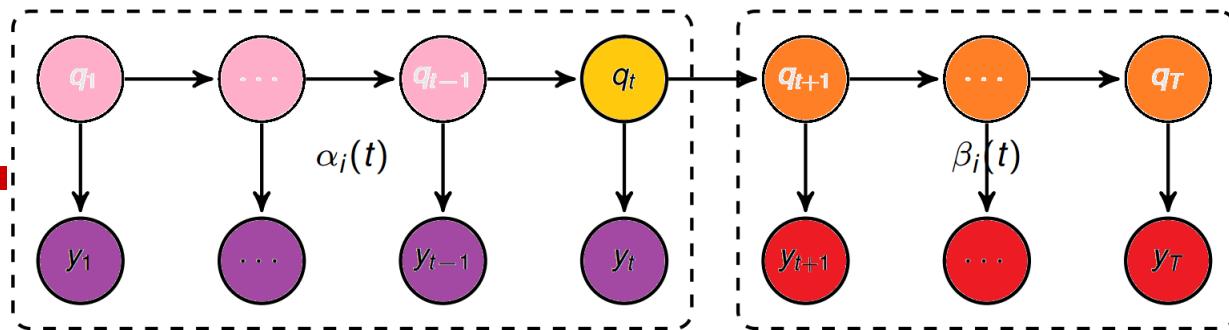
$$= P(o_1, \dots, o_t, o_{t+1}, \dots, o_T | i_t = q_i, \lambda) P(i_t = q_i | \lambda)$$

$$= P(o_1, \dots, o_t | i_t = q_i, \lambda) P(o_{t+1}, \dots, o_T | i_t = q_i, \lambda) P(i_t = q_i | \lambda)$$

$$= P(o_1, \dots, o_t, i_t = q_i | \lambda) P(o_{t+1}, \dots, o_T | i_t = q_i, \lambda)$$

$$= \alpha_t(i) \beta_t(i)$$

□ 思考：试计算 $\xi_t(i, j) = P(i_t = q_i, i_{t+1} = q_j | O, \lambda)$



单个状态的概率 $P(i_t = q_i, O | \lambda) = \alpha_t(i) \beta_t(i)$

- 求给定模型 λ 和观测 O , 在时刻t处于状态 q_i 的概率。
- 记: $\gamma_t(i) = P(i_t = q_i | O, \lambda)$

单个状态的概率

□ 根据前向后向概率的定义，

$$P(i_t = q_i, O | \lambda) = \alpha_t(i) \beta_t(i)$$

$$\gamma_t(i) = P(i_t = q_i | O, \lambda) = \frac{P(i_t = q_i, O | \lambda)}{P(O | \lambda)}$$

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(O | \lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)}$$

γ 的意义

- 在每个时刻t选择在该时刻最有可能出现的状态 i_t^* ，从而得到一个状态序列 $I^*=\{i_1^*, i_2^* \cdots i_T^*\}$ ，将它作为预测的结果。
- 给定模型和观测序列，时刻t处于状态 q_i 的概率为：

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)}$$

两个状态的联合概率

- 求给定模型 λ 和观测 O ，在时刻 t 处于状态 q_i 并且时刻 $t+1$ 处于状态 q_j 的概率。

$$\xi_t(i, j) = P(i_t = q_i, i_{t+1} = q_j | O, \lambda)$$

两个状态的联合概率

$$\begin{aligned}\xi_t(i, j) &= P(i_t = q_i, i_{t+1} = q_j | O, \lambda) \\ &= \frac{P(i_t = q_i, i_{t+1} = q_j, O | \lambda)}{P(O | \lambda)} \\ &= \frac{P(i_t = q_i, i_{t+1} = q_j, O | \lambda)}{\sum_{i=1}^N \sum_{j=1}^N P(i_t = q_i, i_{t+1} = q_j, O | \lambda)}\end{aligned}$$

$$P(i_t = q_i, i_{t+1} = q_j, O | \lambda) = \alpha_t(i) a_{ij} b_{j o_{t+1}} \beta_{t+1}(j)$$

期望

□ 在观测O下状态i出现的期望：

$$\sum_{t=1}^T \gamma_t(i)$$

□ 在观测O下状态i转移到状态j的期望：

$$\sum_{t=1}^{T-1} \xi_t(i, j)$$

□ 思考：在观测O下状态i转移的期望是多少？

学习算法

- 若训练数据包括观测序列和状态序列，则HMM的学习非常简单，是监督学习；
- 若训练数据只有观测序列，则HMM的学习需要使用EM算法，是非监督学习。

大数定理

- 假设已给定训练数据包含S个长度相同的观测序列和对应的状态序列 $\{(O_1, I_1), (O_2, I_2) \dots (O_s, I_s)\}$ ，那么，可以直接利用Bernoulli大数定理的结论“频率的极限是概率”，给出HMM的参数估计。

监督学习方法

□ 初始概率

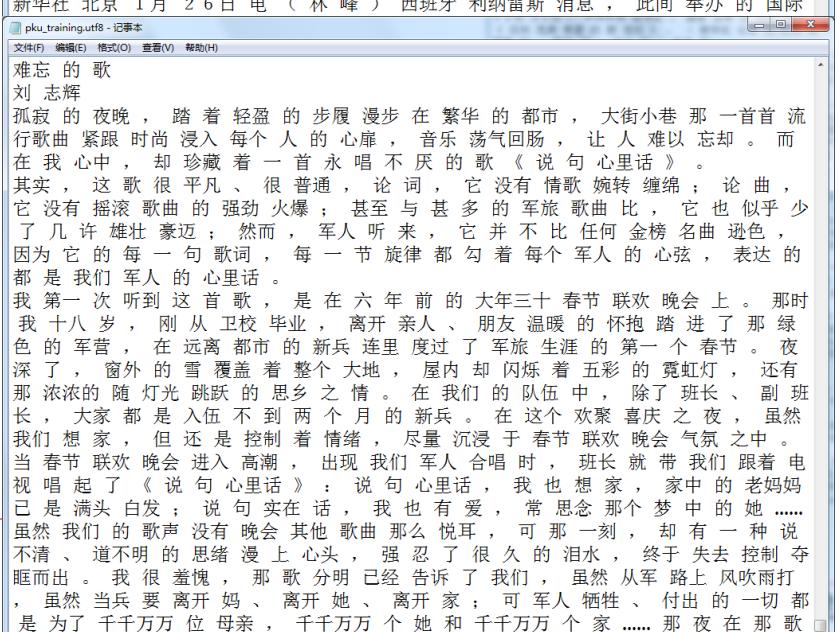
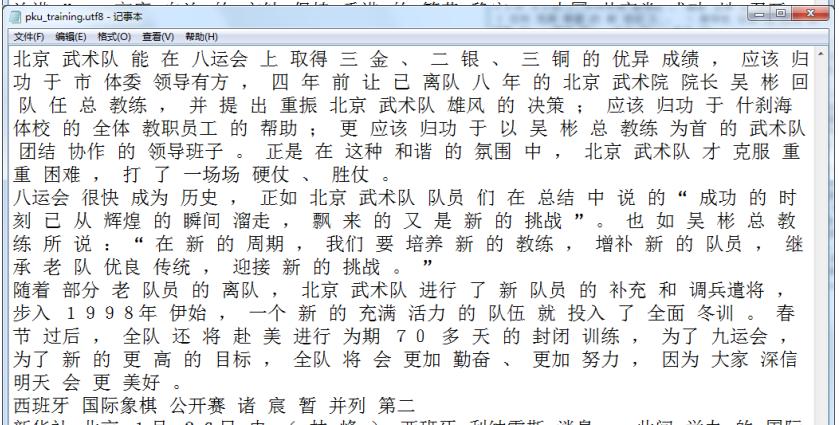
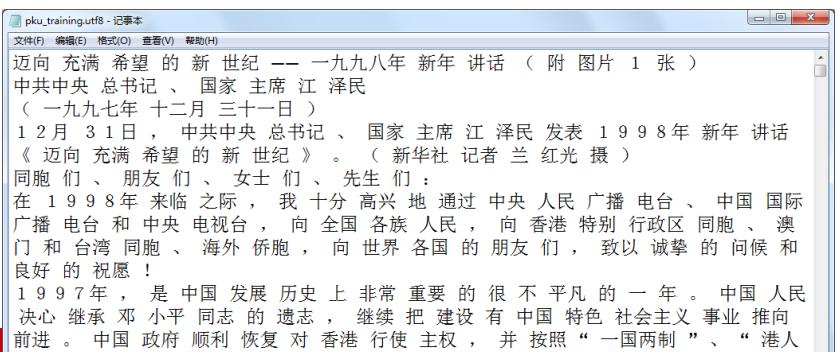
$$\hat{\pi}_i = \frac{|q_i|}{\sum_i |q_i|}$$

□ 转移概率

$$\hat{a}_{ij} = \frac{|q_{ij}|}{\sum_{j=1}^N |q_{ij}|}$$

□ 观测概率

$$\hat{b}_{ik} = \frac{|s_{ik}|}{\sum_{k=1}^M |s_{ik}|}$$



Baum-Welch算法

- 若训练数据只有观测序列，则HMM的学习需要使用EM算法，是非监督学习。

附： EM算法整体框架

Repeat until convergence {

(E-step) For each i , set

$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)}; \theta).$$

(M-step) Set

$$\theta := \arg \max_{\theta} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

}

Baum-Welch算法

- 所有观测数据写成 $O = (o_1, o_2 \dots o_T)$, 所有隐数据写成 $I = (i_1, i_2 \dots i_T)$, 完全数据是 $(O, I) = (o_1, o_2 \dots o_T, i_1, i_2 \dots i_T)$, 完全数据的对数似然函数是 $\ln P(O, I | \lambda)$
- 假设 $\bar{\lambda}$ 是HMM参数的当前估计值, λ 为待求的参数。

$$Q(\lambda, \bar{\lambda}) = \sum_I (\ln P(O, I | \lambda)) P(I | O, \bar{\lambda})$$

$$\begin{aligned} &= \sum_I \ln P(O, I | \lambda) \frac{P(O, I | \bar{\lambda})}{P(O, \bar{\lambda})} \\ &\propto \sum_I \ln P(O, I | \lambda) P(O, I | \bar{\lambda}) \end{aligned}$$

EM过程

□ 根据 $P(O, I | \lambda) = P(O | I, \lambda)P(I | \lambda)$

$$= \pi_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} \cdots a_{i_{T-1} i_T} b_{i_T o_T}$$

□ 函数可写成 $Q(\lambda, \bar{\lambda}) = \sum_I \ln P(O, I | \lambda)P(O, I | \bar{\lambda})$

$$= \sum_I \ln \pi_{i_1} P(O, I | \bar{\lambda})$$

$$+ \sum_I \left(\sum_{t=1}^{T-1} \ln a_{i_t i_{t+1}} \right) P(O, I | \bar{\lambda})$$

$$+ \sum_I \left(\sum_{t=1}^T \ln b_{i_t o_t} \right) P(O, I | \bar{\lambda})$$

极大化

- 极大化Q，求得参数A,B, π
- 由于该三个参数分别位于三个项中，可分别极大化
- 注意到 π_i 满足加和为1，利用拉格朗日乘子法，得到：

$$\sum_I \ln \pi_{i_1} P(O, I | \bar{\lambda}) = \sum_{i=1}^N \ln \pi_{i_1} P(O, i_1 = i | \bar{\lambda})$$

$$\sum_{i=1}^N \ln \pi_i P(O, i_1 = i | \bar{\lambda}) + \gamma \left(\sum_{i=1}^N \pi_i - 1 \right)$$

初始状态概率 $\sum_{i=1}^N \ln \pi_i P(O, i_1 = i | \bar{\lambda}) + \gamma \left(\sum_{i=1}^N \pi_i - 1 \right)$

□ 对上式相对于 π_i 求偏导，得到：

$$P(O, i_1 = i | \bar{\lambda}) + \gamma \pi_i = 0$$

□ 对 i 求和，得到：

$$\gamma = -P(O | \bar{\lambda})$$

□ 从而得到初始状态概率：

$$\pi_i = \frac{P(O, i_1 = i | \bar{\lambda})}{P(O | \bar{\lambda})} = \frac{P(O, i_1 = i | \bar{\lambda})}{\sum_{i=1}^N P(O, i_1 = i | \bar{\lambda})} = \gamma_1(i)$$

转移概率和观测概率

□ 第二项可写成：

$$\sum_I \left(\sum_{t=1}^{T-1} \ln a_{i_t i_{t+1}} \right) P(O, I | \bar{\lambda}) = \sum_{i=1}^N \sum_{j=1}^N \sum_{t=1}^{T-1} \ln a_{ij} P(O, i_t = i, i_{t+1} = j | \bar{\lambda})$$

□ 仍然使用拉格朗日乘子法，得到

$$a_{ij} = \frac{\sum_{t=1}^{T-1} P(O, i_t = i, i_{t+1} = j | \bar{\lambda})}{\sum_{t=1}^{T-1} P(O, i_t = i | \bar{\lambda})} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

□ 同理，得到： $b_{ik} = \frac{\sum_{t=1}^T P(O, i_t = i | \bar{\lambda}) I(o_t = v_k)}{\sum_{t=1}^T P(O, i_t = i | \bar{\lambda})} = \frac{\sum_{t=1, o_t = v_k}^T \gamma_t(i)}{\sum_{t=1}^T \gamma_t(i)}$

预测算法

- 近似算法
- Viterbi 算法

预测的近似算法

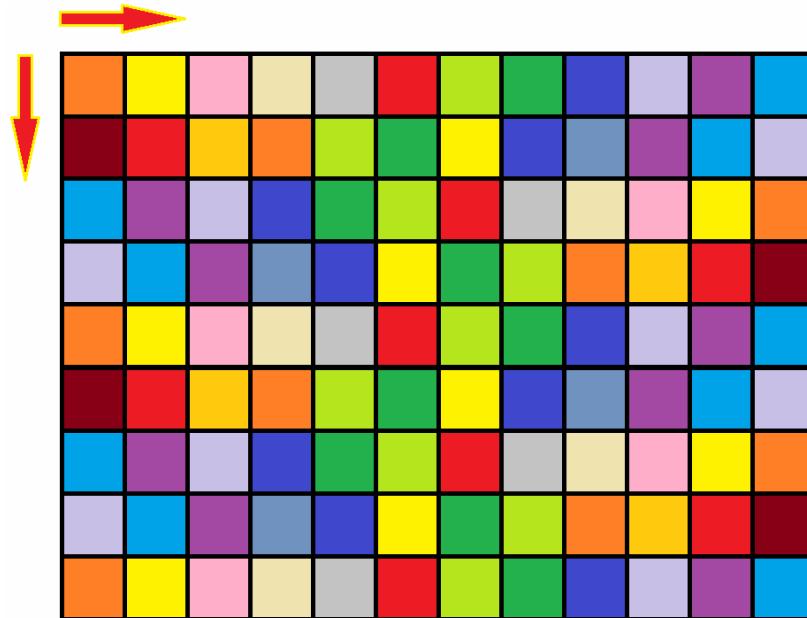
- 在每个时刻t选择在该时刻最有可能出现的状态 i_t^* ，从而得到一个状态序列 $I^*=\{i_1^*, i_2^* \cdots i_T^*\}$ ，将它作为预测的结果。
- 给定模型和观测序列，时刻t处于状态 q_i 的概率为：

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)}$$

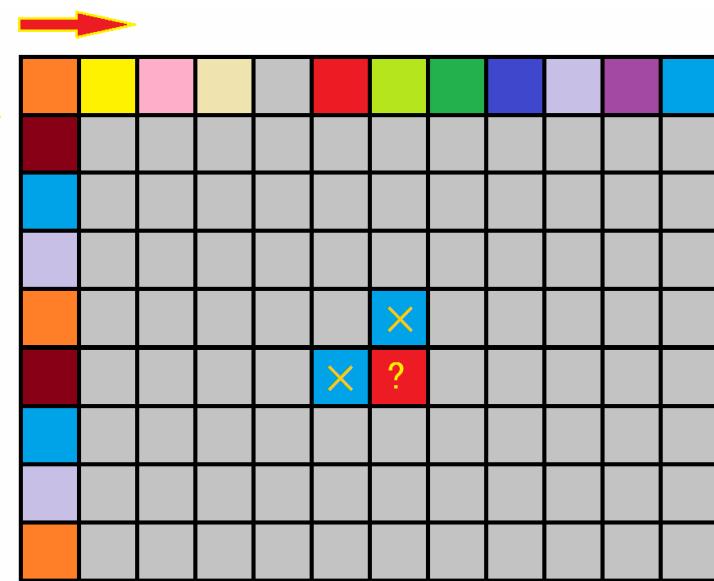
- 选择概率最大的i作为最有可能的状态
 - 会出现此状态在实际中可能不会发生的情况

算法：走棋盘/格子取数

□ 给定 $m \times n$ 的矩阵，每个位置是一个非负整数，从左上角开始，每次只能朝右和下走，走到右下角，求总和最小的路径。



问题分析

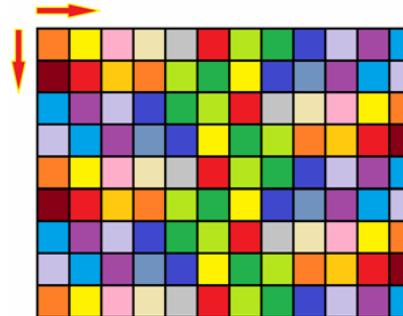


- 走的方向决定了同一个格子不会经过两次。
 - 若当前位于(x,y)处，它来自于哪些格子呢？
 - $dp[0,0]=a[0,0]$ / 第一行(列)累积
 - $dp[x,y] = \min(dp[x-1,y]+a[x,y], dp[x,y-1]+a[x,y])$
 - 即： $dp[x,y] = \min(dp[x-1,y], dp[x,y-1]) + a[x,y]$
 - 思考：若将上述问题改成“求从左上到右下的最大路径”呢？

Viterbi算法

- Viterbi算法实际是用动态规划解HMM预测问题，用DP求概率最大的路径(最优路径)，这是一条路径对应一个状态序列。
- 定义变量 $\delta_t(i)$: 在时刻t状态为i的所有路径中，概率的最大值。

给定 $m \times n$ 的矩阵，每个位置是一个非负整数，从左上角开始，每次只能朝右和下走，走到右下角，求总和最小的路径。



Viterbi 算法

□ 定义：

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

□ 递推：

$$\delta_1(i) = \pi_i b_{io_1}$$

$$\delta_{t+1}(i) = \max_{i_1, i_2, \dots, i_t} P(i_{t+1} = i, i_t, \dots, i_1, o_{t+1}, \dots, o_1 | \lambda)$$

$$= \max_{1 \leq j \leq N} (\delta_t(j) a_{ji}) b_{io_{t+1}}$$

□ 终止：

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

例

- 考察盒子球模型，观测向量 $O=“\text{红白红}”$ ，
试求最优状态序列。

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

解：观测向量O=“红白红”

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

- 初始化：
- 在t=1时，对于每一个状态i，求状态为i观测到o₁=红的概率，记此概率为δ₁(t)
$$\delta_1(i) = \pi_i b_{io_1} = \pi_i b_{i\text{红}}$$
- 求得δ₁(1)=0.1
- δ₁(2)=0.16
- δ₁(3)=0.28

解：观测向量O=“红白红” $A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$ $B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}$

□ 在t=2时，对每个状态i，求在t=1时状态为j观测为红并且在t=2时状态为i观测为白的路径的最大概率，记概率为 $\delta_2(t)$ ，则：

$$\delta_{t+1}(i) = \max_{1 \leq j \leq 3} (\delta_t(j) a_{ji}) b_{io_2} = \max_{1 \leq j \leq 3} (\delta_t(j) a_{ji}) b_{i\text{白}}$$

□ 求得

$$\begin{aligned} \delta_2(1) &= \max_{1 \leq j \leq 3} (\delta_1(j) a_{j1}) b_{i\text{白}} \\ &= \max \{0.10 \times 0.5, 0.16 \times 0.3, 0.28 \times 0.2\} \times 0.5 = 0.028 \end{aligned}$$

□ 同理：

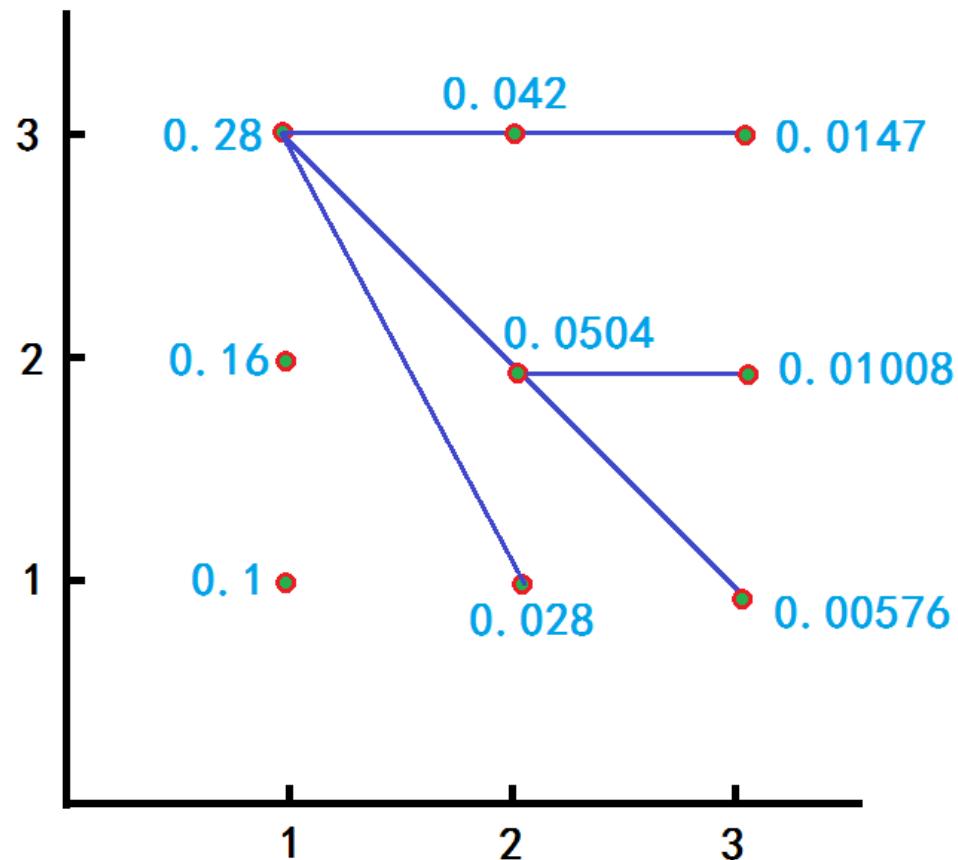
- $\delta_2(2)=0.0504$
- $\delta_2(3)=0.042$

解： 观测向量O=“红白红”

- 同理，求得
- $\delta_3(1)=0.00756$
- $\delta_3(2)=0.01008$
- $\delta_3(3)=0.0147$

- 从而，最大是 $\delta_3(3)=0.0147$ ，根据每一步的最大，得到序列是(3,3,3)

求最优路径图解



Baum-Welch：主函数

```
def baum_welch(pi, A, B):
    f = file("./\text\1.txt")
    sentence = f.read()[3: ].decode('utf-8') # 跳过文件头
    f.close()
    T = len(sentence) # 观测序列
    alpha = [[0 for i in range(4)] for t in range(T)]
    beta = [[0 for i in range(4)] for t in range(T)]
    gamma = [[0 for i in range(4)] for t in range(T)]
    ksi = [[[0 for j in range(4)] for i in range(4)] for t in range(T-1)]
    for time in range(100):
        calc_alpha(pi, A, B, sentence, alpha)      # alpha(t,i): 给定Lambda, 在时刻t的状态为i
        calc_beta(pi, A, B, sentence, beta)         # beta(t,i): 给定Lambda和时刻t的状态i, 观察序列
        calc_gamma(alpha, beta, gamma)              # gamma(t,i): 给定Lambda和O, 在时刻t状态位i
        calc_ksi(alpha, beta, A, B, sentence, ksi)   # ksi(t,i,j): 给定Lambda和O, 在时刻t状态位i, 观察序列j
        bw(pi, A, B, alpha, beta, gamma, ksi, sentence) #baum_welch算法
```

前向-后向

```
def calc_alpha(pi, A, B, o, alpha):
    for i in range(4):
        alpha[0][i] = pi[i] + B[i][ord(o[0])]
    T = len(o)
    temp = [0 for i in range(4)]
    del i
    for t in range(1, T):
        for i in range(4):
            for j in range(4):
                temp[j] = (alpha[t-1][j] + A[j][i])
            alpha[t][i] = log_sum(temp)
            alpha[t][i] += B[i][ord(o[t])]

def calc_beta(pi, A, B, o, beta):
    T = len(o)
    for i in range(4):
        beta[T-1][i] = 1
    temp = [0 for i in range(4)]
    del i
    for t in range(T-2, -1, -1):
        for i in range(4):
            beta[t][i] = 0
            for j in range(4):
                temp[j] = A[i][j] + B[j][ord(o[t+1])] + beta[t+1][j]
            beta[t][i] += log_sum(temp)
```

EM迭代

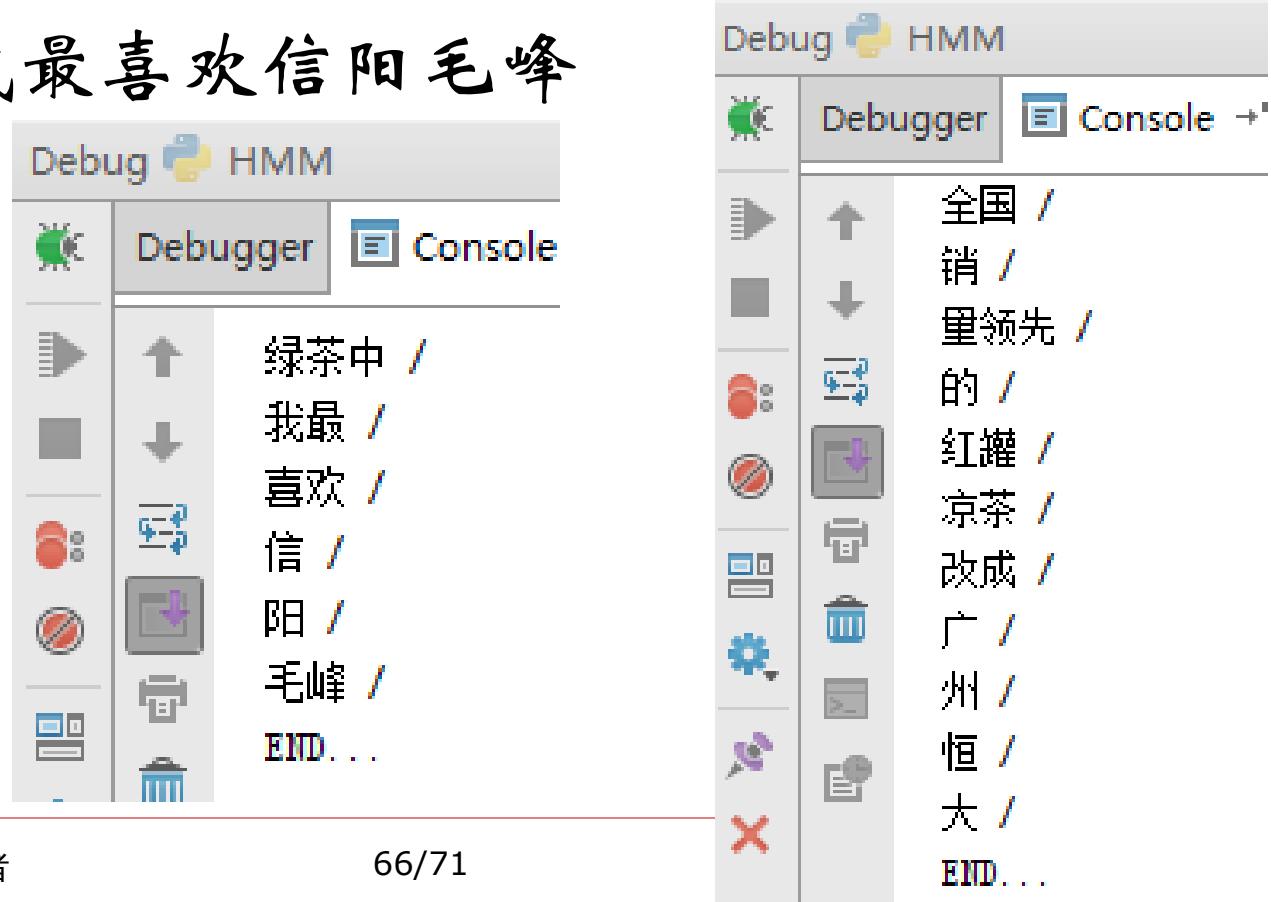
```
def bw(pi, A, B, alpha, beta, gamma, ksi, o):
    T = len(alpha)
    for i in range(4):
        pi[i] = gamma[0][i]
    s1 = [0 for x in range(T-1)]
    s2 = [0 for x in range(T-1)]
    for i in range(4):
        for j in range(4):
            for t in range(T-1):
                s1[t] = ksi[t][i][j]
                s2[t] = gamma[t][i]
            A[i][j] = log_sum(s1) - log_sum(s2)
    s1 = [0 for x in range(T)]
    s2 = [0 for x in range(T)]
    for i in range(4):
        for k in range(65536):
            valid = 0
            for t in range(T):
                if ord(o[t]) == k:
                    s1[valid] = gamma[t][i]
                    valid += 1
                s2[t] = gamma[t][i]
            if valid == 0:
                B[i][k] = infinite
            else:
                B[i][k] = log_sum(s1[:valid]) - log_sum(s2)
```

Viterbi

```
def viterbi(pi, A, B, o):
    T = len(o)    # 观测序列
    delta = [[0 for i in range(4)] for t in range(T)]
    pre = [[0 for i in range(4)] for t in range(T)]  # 前一个状态
    for i in range(4):
        delta[0][i] = pi[i] + B[i][ord(o[0])]
    for t in range(1, T):
        for i in range(4):
            delta[t][i] = delta[t-1][0] + A[0][i]
            for j in range(1,4):
                vj = delta[t-1][j] + A[j][i]
                if delta[t][i] < vj:
                    delta[t][i] = vj
                    pre[t][i] = j
            delta[t][i] += B[i][ord(o[t])]
    decode = [-1 for t in range(T)]  # 解码: 回溯查找最大路径
    q = 0
    for i in range(1, 4):
        if delta[T-1][i] > delta[T-1][q]:
            q = i
    decode[T-1] = q
    for t in range(T-2, -1, -1):
        q = pre[t+1][q]
        decode[t] = q
    return decode
```

Baum-Welch算法的结果

- 全国销量领先的红罐凉茶改成广州恒大
- 绿茶中我最喜欢信阳毛峰



HMM与中文分词

前言 |

数据 |， | 数据 |， | 数据 |！ | 想 | 必在 | 新闻 |、 | 报刊 |等 | 媒介 | 的 | 持续 | 冲击 | 下 |， | 人们 | 无法 | 摆脱 | 大的 | 洗礼 |。 | 现实 | 需求 | 推动 | 了 | 对 | 数据 | 的 | 学习 | 这些 | 数据 | 来 | 自于 | 社交 | 媒体 |、 | 智能 | 手机 |、 | “ | 物联网 |” |) |、 | 传感器 | 等 | 任何 | 可以 | 产生 | 数大 | 多 | 数数 | 据 | 挖掘 | 的 | 宣传 | 着 | 重于 | 数据 | 规模 | 数据 | 洪 | 水(| **data flood**) | 的 | 预言 | 告诉 | 人们 | 我们 | 无法 | 实时 | 处理 | 这些 | 数据 |， | 硬件 | 推销 | 人员 | 会进 | 一步 | 卖 | 给 | 我们 | 需要 | 的 | 服务 |， | 以期 | 能够 | 满足 | 处理 | 速度 | 的 | 要求 |。 | 从 | 某种 | 程度 | 上来 | 说 |， | 他们 | 是 | 对 | 的 |， | 但 | 是 | 我们 | 值得 | 停下 | 来 | 思考 | 片刻 |， | 并 | 对 | 手边 | 的 | 任务 | 进行 | 适当 | 的 | 再 | 认识 |。 |

近 | 年来 |， | 数据 | 挖掘 | 和 | 机器 | 学习 | 在 | 我们 | 周围 | 持续 | 火爆 |， | 各种 | 媒体 | 也 | 不断 | 推送 | 着 | 海量 | 的 | 数据 |。 | 仔细 | 观察 | 就 | 能 | 发现 |， | 实际 | 应用 | 中 | 的 | 那些 | 机器 | 学习 | 算法 | 与 | 多 | 年前 | 并 | 没有 | 什么 | 两样 |； | 它们 | 只 | 是 | 在 | 应用 | 的 | 数据 | 规模 | 上 | 有些 | 不同 |。 | 历数 | 一 | 下 | 产生 | 数据 | 的 | 组织 |， | 至少 | 在 | 我 | 看来 |， | 数目 | 其实 | 并 | 不 | 多 |。 | 无非 | 是 | Google |、 | Facebook |、 | Twitter |、 | NetFlix | 以及 | 其 | 他 | 为数 | 不 | 多 | 的 | 机构 | 在 | 使用 | 若 | 干学 | 习算法 | 和 | 工具 |， | 这些 | 算法 | 和 | 工具 | 使 | 得 | 他们 | 能够 | 对 | 数据 | 进行 | 测试 | 分析 |。 | 那么 |， | 真正 | 的 | 问题 | 是 |： | “ | 对于 | 其 | 他人 |， | 大数 | 据 | 框架 | 下 | 的 | 算法 | 和 | 工具 | 的 | 作用 | 是 | 什么 | 呢 |？ |” |

我承认 | 本书 | 将 | 多 | 次 | 提及 | 大 | 数据 | 和 | 机器 | 学习 | 之间 | 的 | 关系 |， | 这 | 是 | 我 | 无法 | 忽视 | 的 | 一个 | 客观 | 问题 |； | 但 | 是 | 它 | 只 | 是 | 一个 | 很 | 小 | 的 | 因素 |， | 终极 | 目标 | 是 | 如何 | 利用 | 可用 | 数据 | 获取 | 数据 | 的 | 本质 |

```
if __name__ == "__main__":
    pi, A, B = load_train()
    f = file("./text\\novel.txt")
    data = f.read()[3:].decode('utf-8')
    f.close()
    decode = viterbi(pi, A, B, data)
    segment(data, decode)
```

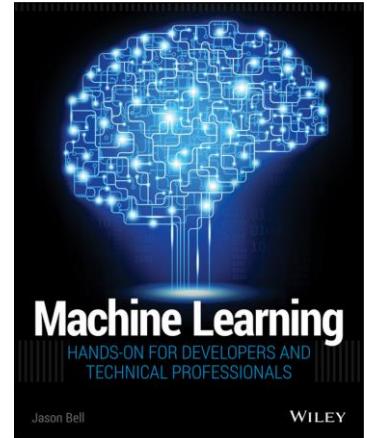
bug HMM

Connected to pydev debugger (build 139.1001)

我 | 与 | 地坛 |
史 | 铁生 |
— |

我 | 在 | 好 | 几 | 篇 | 小说 | 中 | 都 | 提到 | 过 | 一 | 座 | 废弃 | 的 | 古园 |， | 实际 | 就 | 是 | 地坛 |。 | 许多 | 年前 | 旅游业 | 还 | 没有 | 开 | 地坛 | 离 | 我家 | 很 | 近 |。 | 或者 | 说 | 我家 | 离 | 地坛 | 很 | 近 |。 | 总之 |， | 只好 | 认为 | 这 | 是 | 缘分 |。 | 地坛 | 在 | 我出 | 生前 | 它 | 等待 | 我 | 出生 |， | 然后 | 又 | 等待 | 我活 | 到 | 最狂 | 奔 | 的 | 年龄 | 上 | 忽地 | 疾病 | 了 | 双腿 |。 | 四 | 多 | 年里 |， | 它 | 一 | 面 | 自 | 从 | 那个 | 下午 | 我 | 无意 | 中进 | 了 | 这 | 园子 |， | 就 | 再 | 没长 | 久 | 地离 | 开过 | 它 |。 | 我 | 一 | 下子 | 就 | 理解 | 了 | 它 | 的 | 两 | 条 | 髋 | 疾病 | 后 | 的 | 最初 | 几年 |， | 我找 | 不 | 到 | 工作 |， | 找 | 不 | 到 | 去处 |， | 忽然 | 间 | 几乎 | 什么 | 都 | 找 | 不 | 到 | 了 |， | 除 | 去 | 几 | 座 | 廊堂 | 我 | 无法 | 进去 |， | 除去 | 那 | 座 | 祭坛 | 我 | 不 | 能 | 上去 | 而 | 只能 | 从 | 各个 | 角度 | 张望 | 它 |， | 地坛 | 的 | 剩 | 下 | 的 | 就 | 是 | 怎样活 | 的 | 问题 | 了 |， | 这 | 却 | 不 | 是 | 在 | 某 | 一个 | 瞬间 | 就 | 能 | 完全 | 想透 | 的 |、 | 不 | 是 | 一次性 | 都

我 | 才 | 想到 |， | 当年 | 我 | 总是 | 独自 | 跑 | 到 | 地坛 | 去 |， | 曾 | 给 | 母亲 | 出 | 了 | 一个 | 怎样 | 的 | 难题 |。 | 她 | 不 | 是 | 那种 | 光会 | 疼爱 | 儿子 | 而 | 不 | 懂得 | 理解 | 儿子 | 的 | 母亲 |。 | 她 | 知道 | 我心 | 里 | 的 | 苦闷 |， | 知道 | 不 | 该 | 阻止 | 有 | 一 | 回 | 我摇 | 车出 | 了 | 小院 |； | 想 | 起 | 一件 | 什么 | 事 | 又 | 逝身 | 回来 |， | 看见 | 母亲 | 仍站 | 在 | 原地 |， | 还 | 是 | 送 | 我 | 有 | 一次 | 与 | 一个 | 作家 | 朋友 | 聊天 |， | 我问 | 他学 | 写作 | 的 | 最 | 初动 | 机 | 是 | 什么 |？ | 他想 | 了 | 一会 | 说 |： | “ | 为 | 我 | 母亲 |。 | 在 | 我 | 的 | 头 | 一 | 篇 | 小 | 说 | 发表 | 的 | 时候 |， | 在 | 我 | 的 | 小 | 说 | 第 | 一 | 次 | 获奖 | 的 | 那些 | 日子 | 里 |， | 我真 | 是 | 多么 | 只 | 到 | 了 | 这 | 时候 |， | 纷纭 | 的 | 往事 | 在 | 我 | 前 | 现 | 得 | 清晰 |， | 母亲 | 的 | 苦难 | 与 | 伟大 | 才 | 在 | 我 | 心 | 振 | 起 | 轮椅 | 在 | 园中 | 慢慢 | 走 |， | 又 | 是 | 露雾 | 的 | 清晨 |， | 又 | 是 | 晴阳 | 高 | 悬 | 的 | 白昼 |， | 我 | 只想 | 着 | 一件 | 事 |； | 曾 | 有 | 过好 | 多 | 回 |， | 我 | 在 | 这园 | 里 | 待得 | 太久 | 了 |， | 母亲 | 就 | 来找 | 我 |。 | 她 | 来找 | 我 | 又 | 不想 | 让 | 我 | 觉得 |



Jason Bell. *Machine Learning: Hands-On for Developers and Technical Professionals*. Wiley.2014

总结

- 马尔科夫模型可以用来统一解释贪心法和动态规划。
- HMM解决标注问题，在语音识别、NLP、生物信息、模式识别等领域被广泛使用。
 - 思考：可否用深度学习代替HMM？
 - 思考：如果观测状态是连续值，可否将多项分布改成高斯分布或者混合高斯分布？
- 在一定意义上，数据比算法更重要。
- 加强算法模型和实践问题的相互转换能力。

参考文献

- 李航, 统计学习方法, 清华大学出版社, 2012
- Christopher M. Bishop. Pattern Recognition and Machine Learning Chapter 10. Springer-Verlag, 2006
- Radiner L, Juang B. *An introduction of hidden markov Models*. IEEE ASSP Magazine, 1986
- Lawrence R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. Proceedings of the IEEE 77.2, pp. 257-286, 1989
- Jeff A. Bilmes. *A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models*. 1998.
- https://en.wikipedia.org/wiki/Hidden_Markov_model

我们在这里

- <http://wenda.ChinaHadoop.cn>
 - 视频/课程/社区
- 微博
 - @ChinaHadoop
 - @邹博_机器学习
- 微信公众号
 - 小象
 - 大数据分析挖掘

The screenshot shows a web browser displaying the '发现' (Discover) section of the wenda.chinahadoop.cn website. The URL in the address bar is wenda.chinahadoop.cn/explore/. The page has a blue header with the site logo '小象问答' and various navigation links like '搜索标题、用户' (Search title, user), '全站内容搜索' (Full site search), '提问' (Ask question), '首页' (Home), '动态' (Activity), '发现' (Discover) (which is circled in red), '话题' (Topic), and '通知' (Notification). Below the header is a navigation bar with categories: 全部 (All), 招聘求职 (Recruitment), 机器学习 (Machine Learning), 大数据平台技术 (Big Data Platform Technology), DCon, 大数据行业应用 (Big Data Industry Application), NoSQL数据库 (NoSQL Database), 数据科学 (Data Science), and 江湖救急 (Emergency Rescue). The main content area displays a list of discovered topics, each with a thumbnail, title, and details like replies and views. To the right of the list are two columns: '专题' (Topics) and '热门话题' (Hot Topics). The '专题' column lists categories with counts: 招聘求职 (193), 大数据行业应用 (200), 数据科学 (55), 系统与编程 (179), and 云计算技术 (224). The '热门话题' column lists specific topics with counts: 机器学习 (193), spark (200), 算法 (55), linux (179), 和 hbase (224). At the bottom right is a '热门用户' (Hot User) section with user profiles.

专题	贡献
招聘求职	193 个问题, 86 人关注
大数据行业应用	200 个问题, 91 人关注
数据科学	55 个问题, 65 人关注
系统与编程	179 个问题, 47 人关注
云计算技术	224 个问题, 62 人关注

热门话题	更多 >
机器学习	193 个问题, 86 人关注
spark	200 个问题, 91 人关注
算法	55 个问题, 65 人关注
linux	179 个问题, 47 人关注
hbase	224 个问题, 62 人关注

热门用户	更多 >
gongfu	17 个问题, 0 次赞同
Hagrid	55 个问题, 3 次赞同
yanglei	55 个问题, 12 次赞同
天热不喝水	48 个问题, 0 次赞同
hiveman	19 个问题, 1 次赞同

感谢大家！

恳请大家批评指正！