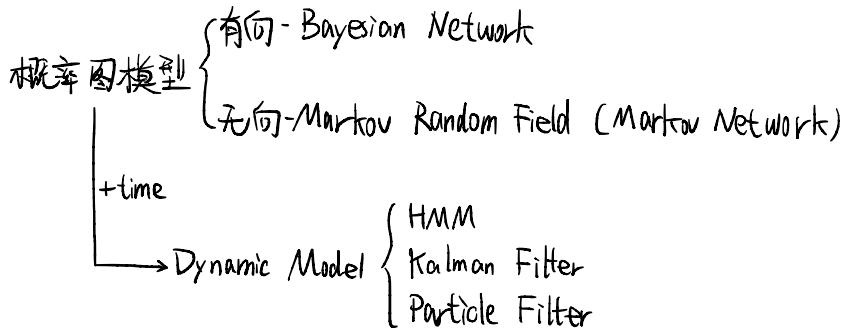
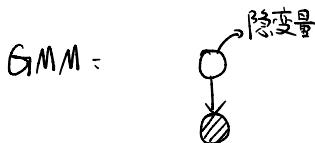


# 隐式马尔可夫模型 (HMM)

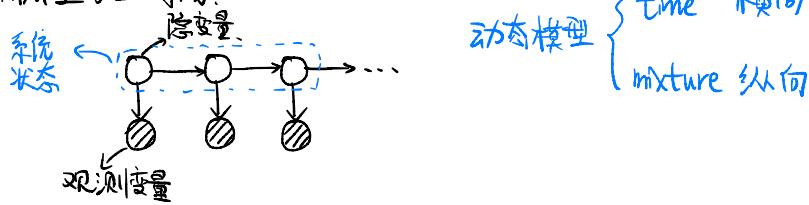
## 一、背景



Dynamic Model 的样本之间不是 iid 的。



如果在 GMM 上加上时间：

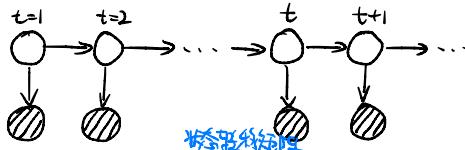


根据系统状态的连续与否, state → 高级 = HMM

连续 {

- { 线性 → Kalman Filter
- { 非线性 → Particle Filter

HMM



一个模型，两个假设，三个问题

i: 模型参数:  $\lambda = (\pi, A, B)$   
 ↓  
 初始概率分布      发射矩阵

取值集合

观测变量:  $o_1, o_2, \dots, o_t, \dots \rightarrow V = \{v_1, v_2, \dots, v_m\}$

状态变量:  $i_1, i_2, \dots, i_t, \dots \rightarrow Q = \{q_1, q_2, \dots, q_N\}$

对于本模型参数，其中。

$A = [a_{ij}]$ ,  $a_{ij} = P(i_{t+1} = q_i | i_t = q_j)$ , 即  $a_{ij}$  表示从状态  $q_j$  转移到状态  $q_i$  的概率。

$B = [b_j(k)]$ ,  $b_j(k) = P(o_t = v_k | i_t = q_j)$ , 即  $b_j(k)$  表示 t 时刻从状态  $q_j$  产生  $v_k$  的概率。

$\pi$  为初始概率分布,  $P(\pi = (p_1, p_2, \dots, p_N))$

ii: 两个假设:

① 齐次 Markov 假设 ② 观测独立假设

↓  
无后效性

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

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

iii: 三个问题

观测序列  $O = o_1, o_2, \dots, o_t$

① Evaluation: 已知  $\lambda$ ,  $P(O|\lambda)$  出现的概率, 常用算法: 前向后向

② Learning:  $\lambda$  如何求? ( $\lambda = \arg \max P(O|\lambda)$ )  $\rightarrow$  EM, Baum Welch

③ Decoding: 已知观测序列  $O$ , 状态序列  $I$  出现的最大概率? (后验概率)

$P(I = \arg \max I | O)$ , 维特比算法

状态序列  $I = i_1, i_2, \dots, i_t$

引申问题: { 预测:  $P(i_{t+1} | o_1, o_2, \dots, o_t)$   
 滤波:  $P(i_t | o_1, o_2, \dots, o_t)$

问题一. Evaluation: Given  $\lambda$ , 求  $P(O|\lambda)$

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

其中,  $P(I|\lambda) = P(i_1, i_2, \dots, i_T | \lambda) = \underbrace{P(i_T | i_1, i_2, \dots, i_{T-1}, \lambda)}_{\text{由齐次Markov性} \rightarrow} \cdot \underbrace{P(i_1, i_2, \dots, i_{T-1} | \lambda)}_{\substack{\text{递归是T-1级模型} \\ \text{分析因是T-1级模型}}}$

$$= a_{i_{T-1}, i_T} \cdot a_{i_{T-2}, i_{T-1}} \cdots a_{i_1, i_2} \cdot \pi(i_1)$$

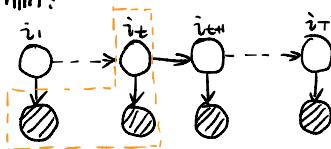
$$= \pi(i_1) \prod_{t=2}^T a_{i_{t-1}, i_t}$$

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

$$\therefore P(O|\lambda) = \sum_I \pi(i_1) \prod_{t=2}^T a_{i_{t-1}, i_t} \cdot \prod_{t=1}^T b_{i_t}(o_t) = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_T} \pi(i_1) \prod_{t=2}^T a_{i_{t-1}, i_t} \cdot \prod_{t=1}^T b_{i_t}(o_t)$$

对于每一个字符和符号, 每一个  $i$  都有  $N$  种选择, 共有  $T$  个  $i$ , 因此时间复杂度为  $O(N^T)$ , 计算量很大, 因此需要一种算法来计算。

Forward Algorithm:



$$\text{记 } \alpha_t(i) = P(o_1, \dots, o_t, i_t = q_i | \lambda)$$

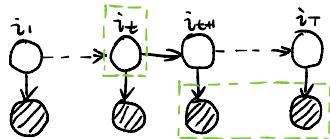
$$\alpha_T(i) = P(O, i_T = q_i | \lambda)$$

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

$$\begin{aligned} \alpha_{t+1}(j) &= P(O_1, O_2, \dots, O_{t+1}, i_{t+1} = q_j | \lambda) \\ &= \sum_{i=1}^N P(O_1, O_2, \dots, O_{t+1}, i_{t+1} = q_j, i_t = q_i | \lambda) \\ &= \sum_{i=1}^N P(O_{t+1} | i_{t+1} = q_j) \cdot P(O_1, \dots, O_t, i_t = q_i, i_{t+1} = q_j | \lambda) \\ &= \sum_{i=1}^N P(O_{t+1} | i_{t+1} = q_j) \cdot P(i_{t+1} = q_j | O_1, \dots, O_t, i_t = q_i, \lambda) \cdot P(O_1, \dots, O_t, i_t = q_i | \lambda) \\ &= \sum_{i=1}^N P(O_{t+1} | i_{t+1} = q_j) \cdot P(i_{t+1} = q_j | i_t = q_i) \cdot \boxed{\alpha_t(i)} \end{aligned}$$

因此, 我们可以一步一步求出  $\alpha$ , 进而由  $P(O|\lambda) = \sum_{i=1}^N \alpha_T(i)$  求出  $P(O|\lambda)$

## Backward Algorithm:



$$\text{记: } \beta_t(i) = P(o_{t+1}, \dots, o_T | i_t = q_i, \lambda)$$

⋮

$$\beta_1(i) = P(o_2, \dots, o_T | i_1 = q_i, \lambda)$$

$$\begin{aligned} P(o|\lambda) &= P(o_1, \dots, o_T | \lambda) \\ &= \sum_{i=1}^N P(o_1, \dots, o_T, i=q_i | \lambda) \\ &= \sum_{i=1}^N P(o_1, \dots, o_T | i=q_i, \lambda) \underbrace{P(i=q_i | \lambda)}_{\pi_i} \\ &= \sum_{i=1}^N P(o_1 | o_2, \dots, o_T, i=q_i, \lambda) \cdot P(o_2, \dots, o_T | i=q_i, \lambda) \cdot \pi_i \\ &= \sum_{i=1}^N P(o_1 | i=q_i, \lambda) \cdot \beta_1(i) \cdot \pi_i \\ &= \sum_{i=1}^N b_i(o_1) \cdot \pi_i \cdot \beta_1(i) \end{aligned}$$

$$\pi = (\pi_1, \pi_2, \dots, \pi_N), \sum_{i=1}^N \pi_i = 1$$

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

因此，我们可以从  $\beta_T$  开始，逆向计算  $\beta_{T-1}, \dots, \beta_1$ ，从而求出  $P(o|\lambda)$

## 问题二. Learning

由于  $P(O|I)$  过于复杂，直接使用 MLE 无法求出，因此使用 EM 算法。

$$\text{EM: } \theta^{(t+1)} = \arg \max_{\theta} \int \log P(X, Z | \theta) P(Z | X, \theta^{(t)}) dZ$$

$X$ : 观测数据  
 $\downarrow$   
 $Z$ : 隐变量  
 $\downarrow$   
 $\theta$ : 参数  
 $\downarrow$

$$\lambda^{(t)} = (\pi^{(t)}, A^{(t)}, B^{(t)})$$

$$\text{则 } \lambda^{(t+1)} = \arg \max_{\lambda} \sum_I \log P(O, I | \lambda) P(I | O, \lambda^{(t)})$$

$\frac{P(I, O | \lambda^{(t)})}{P(O | \lambda^{(t)})}$  常数值

$$\lambda^{(t+1)} = (\pi^{(t+1)}, A^{(t+1)}, B^{(t+1)})$$

$$= \arg \max_{\lambda} \sum_I \log P(O, I | \lambda) P(I, O | \lambda^{(t)})$$

$$\text{全 Q}(\lambda, \lambda^{(t)}) = \sum_I \log P(O, I) \lambda^{(t)} P(I, O | \lambda^{(t)})$$

$$Q(\lambda, \lambda^{(t)}) = \sum_I \left[ \log \pi_{i_1} + \sum_{t=1}^T \log a_{i_{t-1}, i_t} + \sum_{t=1}^T \log b_{i_t}(O_t) \right] \cdot P(I, O | \lambda^{(t)})$$

以计算  $\pi^{(t)}$  为例：

$$\begin{aligned} \pi^{(t+1)} &= \arg \max_{\pi} Q(\lambda, \lambda^{(t)}) \\ &= \arg \max_{\pi} \sum_I [\log \pi_{i_1} \cdot P(I, O | \lambda^{(t)})] \\ &= \arg \max_{\pi} \sum_{i_1} \cdots \sum_{i_T} [\log \pi_{i_1} \cdot P(O, i_1, \dots, i_T | \lambda^{(t)})] \\ &= \arg \max_{\pi} \sum_{i_1} [\log \pi_{i_1} \cdot P(O, i_1 | \lambda^{(t)})] \\ &= \arg \max_{\pi} \sum_{i_1=1}^N [\log \pi_{i_1} \cdot P(O, i_1 = q_i | \lambda^{(t)})] \quad (\text{s.t. } \sum_{i_1} \pi_{i_1} = 1) \end{aligned}$$

由拉格朗日乘子法：

$$L(\pi, \gamma) = \sum_{i_1=1}^N [\log \pi_{i_1} \cdot P(O, i_1 = q_i | \lambda^{(t)})] + \gamma (\sum_{i_1=1}^N \pi_{i_1} - 1)$$

$$\text{令 } \frac{\partial L}{\partial \pi_{i_1}} = \frac{1}{\pi_{i_1}} \cdot P(O, i_1 = q_i | \lambda^{(t)}) + \gamma = 0 \quad \dots \textcircled{1}$$

$$\mathbb{E} [P(O, i_1 = q_i | \lambda^{(t)}) + \gamma] = 0$$

$$\text{则 } \sum_{i_1=1}^N [P(O, i_1 = q_i | \lambda^{(t)}) + \gamma] = 0$$

$$P(O | \lambda^{(t)}) + \gamma = 0$$

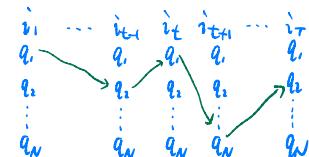
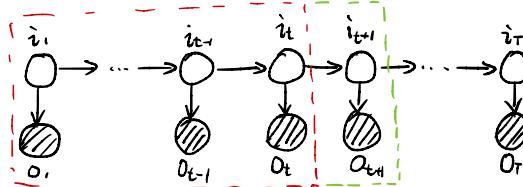
$$\therefore \gamma = -P(O | \lambda^{(t)})$$

$$\text{将 } \gamma \text{ 代入式 } \textcircled{1} \text{, 得 } \pi_{i_1}^{(t+1)} = \frac{P(O, i_1 = q_i | \lambda^{(t)})}{P(O | \lambda^{(t)})}$$

$$\therefore \pi^{(t+1)} = (\pi_1^{(t+1)}, \pi_2^{(t+1)}, \dots, \pi_N^{(t+1)})$$

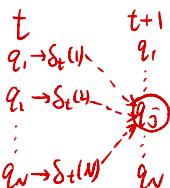
### 问题三, Decoding

Viterbi:



设  $\delta_t(i) = \max_{i_1, \dots, i_{t-1}} P(o_1, o_2, \dots, o_t, i_1, \dots, i_t, i_t = q_i)$  表示 t 时刻时到达状态  $q_i$  的最大概率。

$$\text{则 } \delta_{t+1}(j) = \max_{i_1, \dots, i_t} P(o_1, o_2, \dots, o_{t+1}, i_1, \dots, i_t, i_{t+1} = q_j) \\ = \max_{1 \leq i \leq N} \delta_t(i) \cdot a_{ij} \cdot b_j(o_{t+1})$$



因此,我们可以计算  $\delta_1, \delta_2$  直到  $\delta_T$ ,  $\delta_T$  就是我们要求得的那个概率值。

若想记录路径,可以令  $\pi_{t+1}(i) = \arg \max_{1 \leq i \leq N} \delta_t(i) \cdot a_{ij}$ , 即记录使得  $t+1$  时刻到达  $q_i$  的  $t$  时刻所在的状态  $i$ ,  $\pi_{t+1}(i)$  返回的就是使这个概率最大的状态,所以每一步就可以记录一个状态.

## 四. 隐式马尔可夫模型小结

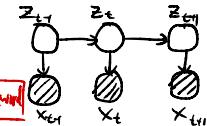
HMM (mixture + time) 混合+序列。

在 HMM 中，只对隐变量有要求，要求其为离散变量，不对观测变量有要求。

HMM 是一种动态模型 (Dynamic Model)，也叫状态空间模型 (state space Model)。  
对于这种模型，与其它概率图模型或贝叶斯法一样，都是要解决两个问题：

即 { Learning: Baum Welch (EM) 对于时间序列的状态空间模型，Inference 可分为：  
Inference.

Inference	$\begin{cases} \text{decoding: } P(z_1, \dots, z_T   x_1, \dots, x_T) & \boxed{\text{Viterbi}} \\ \text{st. evidence: } P(x   \theta) = P(x_1, \dots, x_T   \theta) & \boxed{\text{Forward / Backward}} \\ \text{filtering: } P(z_t   x_1, x_2, \dots, x_t) \rightarrow \text{online} & \boxed{\text{Forward}} \\ \text{smoothing: } P(z_t   x_1, x_2, \dots, x_T) \rightarrow \text{offline} & \boxed{\text{Forward-Backward Algorithm}} \\ \text{prediction: } \begin{cases} P(z_{t+1}, z_{t+2}   x_1, \dots, x_t) \\ P(x_{t+1}, x_{t+2}   x_1, \dots, x_t) \end{cases} \end{cases}$
-----------	--



$$\text{Filtering: } P(z_t | x_1, \dots, x_t) = \frac{P(z_t, x_1, \dots, x_t)}{P(x_1, \dots, x_t)} = \frac{P(x_1, \dots, x_t, z_t)}{\sum_z P(x_1, \dots, x_t, z)} \propto P(x_1, \dots, x_t, z_t) = \alpha_t$$

即 Filtering 问题用的方法是 Forward 算法。

$$\text{Smoothing: } P(z_t | x_1, \dots, x_T) = \frac{P(x_1, \dots, x_T, z_t)}{P(x_1, \dots, x_T)} = \frac{P(x_1, \dots, x_T, z_t)}{\sum_z P(x_1, \dots, x_T, z)}$$

$$\begin{aligned} P(x_1, \dots, x_T, z_t) &= P(x_1, \dots, x_t, x_{t+1}, \dots, x_T, z_t) \\ &= \underbrace{P(x_{t+1}, \dots, x_T | x_1, \dots, x_t, z_t)}_{\beta_t} \cdot \underbrace{P(x_1, \dots, x_t, z_t)}_{\alpha_t} \end{aligned}$$

借用 D-划分，设  $x_1, \dots, x_t$  为 A， $z_t$  为 B， $x_{t+1}, \dots, x_T$  为 C，则由 A 到 C 的路径为：

$A \xrightarrow[B]{z_t} z_{t+1} \rightarrow C$ ，因此，在给定 B 的情况下， $A \perp C | B$ 。

$$\therefore P(x_{t+1}, \dots, x_T | x_1, \dots, x_t, z_t) = P(x_{t+1}, \dots, x_T | z_t) = \beta_t$$

$$\text{因此 } P(x_1, \dots, x_T, z_t) = \beta_t \cdot \alpha_t. \quad \boxed{P(z_t | x_1, \dots, x_T) \propto P(x_1, \dots, x_T, z_t) = \alpha_t \cdot \beta_t}$$

(注意，有时文献中称 Baum Welch 为 Forward-Backward 算法，这里不准确的，原因是 Baum Welch 算法在实现过程中使用了 Forward-Backward 算法)

$$\text{Prediction: } P(z_{t+1} | x_1, \dots, x_t) = \sum_{z_t} P(z_{t+1}, z_t | x_1, \dots, x_t) = \sum_{z_t} \underbrace{P(z_{t+1} | z_t, x_1, \dots, x_t)}_{P(z_{t+1} | z_t)} \cdot \underbrace{P(z_t | x_1, \dots, x_t)}_{\text{filtering}}$$

$$P(x_{t+1} | x_1, \dots, x_t) = \sum_{z_{t+1}} P(x_{t+1}, z_{t+1} | x_1, \dots, x_t) = \sum_{z_{t+1}} \underbrace{P(x_{t+1} | z_{t+1}, x_1, \dots, x_t)}_{P(x_{t+1} | z_{t+1})} \cdot \underbrace{P(z_{t+1} | x_1, \dots, x_t)}_{P(z_{t+1} | z_t)}$$