隐马尔可夫模型
Hidden Markov Model

① 背景

② 前向-后向算法

③ Baum-Welch算法

④ Viterbi算法

⑤ 小结

背景(Background)

复习一下之前的知识.

频率派 ⟶ 统计机器学习 ⟶ 优化问题
- ① model: $f(w)=w^Tx+b$
- ② strategy: loss function
- ③ algorithm: GD, SGD, 牛顿等

贝叶斯派 ⟶ 概率图模型 ⟶ 求后验及其期望 ⟶ 积分问题
- 数值积分
- MCMC

$\boxed{\text{HMM与概率图模型的关系:}}$

概率图
- 有向: Bayes Network
- 无向: Markov Random Field (Markov Network)

+time (即数据是顺序数据)

⟶ State Space Model
- HMM — 潜变量离散
- LDS (Kalman Filter) — 潜变量连续且线性
- Particle Filter — 潜变量连续且非线性

为什么顺序数据就要建立 Sate Space Model?

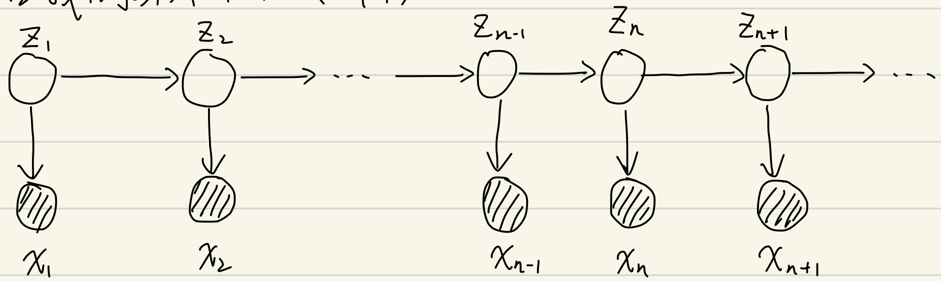
似然函数角度：如果数据集中的数据点是独立同分布的情形，那么我们的似然函数就可以表示为每个数据点处所计算概率分布的**乘积**。但是顺序数据不遵从独立同分布的假设，所以计算似然函数是一个问题。

模型复杂性角度：考虑一个序列 $\{x_1, x_2, \cdots, x_N\}$，如果我们建立一个 x_n 对于 x_1 到 x_{n-1} 的一般的依赖关系，通过这样关系所构建出的模型的复杂度会随着序列的增长而无限制的增长，所以我们考虑引入 Markov Model。但是 Markov Model 对于表达复杂的分布仍有一定的局限性，所以我们需要引入潜在变量，从而得到了 State Space Model。

为什么模型表达复杂的分布仍有一定局限性？

我们用一阶马尔可夫链表达一个分布，若该分布离散拥有 K 个状态，则对单独一个节点来说就有 $K-1$ 个参数，同时若 Markov chain 是一阶的，则为了表示 $p(x_n|x_{n-1})$，就需要有 $K(K-1)$ 个参数。当 Markov chain 扩展到 M 阶时，则有 $K^M(K-1)$ 个参数。这个量会随着 K 和 M 的增长急剧变大，故这就是只使用 Markov chain 的局限性。(PRML中有介绍)

下面我们引入 HMM（一阶）



首先，我们先来分析一下条件独立性

① 对于 z_{n-1} 和 z_{n+1}，由于只存在一条路径，且类型为"head to tail"，若给定 z_n，则 $z_{n-1} \perp z_{n+1} \mid z_n$.

② 对于 x_n，给定之前所有的观测 $x_1, \dots, x_{n-1}$，$p(x_n \mid x_1, \dots, x_{n-1})$ 不会表现出任何的条件独立性.

第二条结论表明了，我们用一阶的 HMM 就可以表示 n-1 阶的 Markov chain.

HMM 的参数 $\lambda = (\pi, A, B)$：

π：初始概率分布，元素为 $\pi_k = p(z_{1k}=1)$，$p(z_1 \mid \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$，$\sum_k \pi_k = 1$

A：转移矩阵，元素为 $A_{ij} = p(z_{nj}=1 \mid z_{n-1,i}=1)$，$p(z_n \mid z_{n-1}, A) = \prod_{i=1}^{K} \prod_{j=1}^{K} A_{ij}^{z_{nj} z_{n-1,i}}$

B：发射概率，我们不限制连续还是离散.

上述中，π 和 A 中，z 都有 K 个状态，对应用"1-of-K"方式来表示.

上一页中简单分析了条件独立性，下面我们写出更多更详细的表述。

规定 $X=\{x_1, x_2, \cdots, x_N\}$，我们有：(可以用 d-划分证明)

$$p(X|z_n)=p(x_1,\cdots,x_n|z_n)p(x_{n+1},\cdots,x_N|z_n) \quad (1)$$

$$p(x_1,\cdots,x_{n-1}|x_n,z_n)=p(x_1,\cdots,x_{n-1}|z_n) \quad (2)$$

$$p(x_1,\cdots,x_{n-1}|z_{n-1},z_n)=p(x_1,\cdots,x_{n-1}|z_{n-1}) \quad (3)$$

$$p(x_{n+1},\cdots,x_N|z_n,z_{n+1})=p(x_{n+1},\cdots,x_N|z_{n+1}) \quad (4)$$

$$p(x_{n+2},\cdots,x_N|z_{n+1},x_{n+1})=p(x_{n+2},\cdots,x_N|z_{n+1}) \quad (5)$$

$$p(X|z_{n-1},z_n)=p(x_1,\cdots,x_{n-1}|z_{n-1})p(x_n|z_n)p(x_{n+1},\cdots,x_N|z_n) \quad (6)$$

$$p(x_{N+1}|X,z_{N+1})=p(x_{N+1}|z_{N+1}) \quad (7)$$

$$p(z_{N+1}|z_N,X)=p(z_{N+1},z_N) \quad (8)$$

其中，第(8)条公式我还稍微有点困惑，正常来说应该是 $p(z_{N+1}|z_N,X)=p(z_{N+1}|z_N)$，如果要变成式(8)，就需要乘上 $p(z_N)$。唯一的解释就是 $p(z_N)=1$？

HMM的三个问题：

① Evaluation ： $p(X|\lambda)$ -> Forward-Backward Algorithm

② Learning ： 求 λ -> $\lambda=\underset{\lambda}{\operatorname{argmax}}\, p(X|\lambda)$ -> Baum-Welch / EM Algorithm

③ Decoding ： $p(Z|X,\lambda)$ -> Viterbi Algorithm

引申 ->
- predict ： $p(z_{N+1}|X)$
- filter ： $p(z_n|x_1,\cdots,x_n)$
- smoothing ： $p(z_n|x_1,\cdots,x_N)$

前向-后向算法
(Forward-Backward Algorithm)

$\boxed{\text{HMM的似然函数}}$. => 链长度为 N，单个隐变量的状态个数为 K

由上述可知：Evaluation 和 Learning 问题，都是与HMM的似然函数有关，所以我们先在这里讨论HMM的似然函数 $p(X|\lambda)$，简便起见，我们省略 $\lambda \to p(X)$

$$p(X) = \sum_{Z} p(X, Z) = \sum_{Z} p(z_1) \prod_{n=2}^{N} p(z_n|z_{n-1}) \prod_{n=1}^{N} p(x_n|z_n)$$

我们注意到求和项 $\sum_{Z}$，由于我们不能将 $p(X, Z)$ 在 n 上进行分解，故我们有 N 个变量需要求和，每个变量有 K 个状态，所以我们一共有 K^N 个求和项（即将这 N 个节点，根据状态进行排列组合），所以求和式中项的数量随着 N 的增大指数增长，所以我们想要直接对 $p(X)$ 进行计算是不可行的，需要借助其他方法。

(其实前向-后向算法和 sum-product 算法是等价的，具体见 PRML)

考虑到一阶的HMM是一棵树，因此我们知道潜变量的后验概率分布可以使用两阶段的信息传递算法高效求出。

前向-后向算法其实可以拆分为前向算法与后向算法。

引入两个记号：

$$\alpha(z_n) = p(x_1, \cdots, x_n, z_n) \quad \text{前向过程}$$

$$\beta(z_n) = p(x_{n+1}, \cdots, x_N | z_n) \quad \text{后向过程}$$

下面分别就这两个记号的递归表达式进行推导.

$$
\begin{aligned}
\alpha(z_n) &= p(x_1,\cdots,x_n,z_n) \\
&= \sum_{z_{n-1}} p(x_1,\cdots,x_n,z_n,z_{n-1}) \\
&= \sum_{z_{n-1}} p(z_n,x_n \mid x_1,\cdots,x_{n-1},z_{n-1})\, p(x_1,\cdots,x_{n-1},z_{n-1}) \\
&= \sum_{z_{n-1}} p(z_n,x_n \mid z_{n-1})\,\alpha(z_{n-1}) \\
&= \sum_{z_{n-1}} p(x_n \mid z_n,z_{n-1})\, p(z_n \mid z_{n-1})\,\alpha(z_{n-1}) \\
&= \sum_{z_{n-1}} p(x_n \mid z_n)\, p(z_n \mid z_{n-1})\,\alpha(z_{n-1}) \\
&= p(x_n \mid z_n) \sum_{z_{n-1}} p(z_n \mid z_{n-1})\,\alpha(z_{n-1})
\end{aligned}
$$

蓝线部分是发射概率和转移概率.

$$
\begin{aligned}
\beta(z_n) &= p(x_{n+1},\cdots,x_N \mid z_n) \\
&= \sum_{z_{n+1}} p(x_{n+1},\cdots,x_N,z_{n+1} \mid z_n) \\
&= \sum_{z_{n+1}} p(x_{n+1},\cdots,x_N \mid z_{n+1},z_n)\, p(z_{n+1} \mid z_n) \\
&= \sum_{z_{n+1}} p(x_{n+2},\cdots,x_N \mid x_{n+1},z_{n+1})\, p(x_{n+1} \mid z_{n+1})\, p(z_{n+1} \mid z_n) \\
&= \sum_{z_{n+1}} p(x_{n+2},\cdots,x_N \mid z_{n+1})\, p(x_{n+1} \mid z_{n+1})\, p(z_{n+1} \mid z_n) \\
&= \sum_{z_{n+1}} \beta(z_{n+1})\, p(x_{n+1} \mid z_{n+1})\, p(z_{n+1} \mid z_n)
\end{aligned}
$$

蓝线部分为发射概率和转移概率

两者的递归已经确定之后，需要确定初始条件.

对于 $\alpha(z_n)$ 来说，由于其依赖上一时刻，所以需要确定 $\alpha(z_1)$

$$\boxed{\alpha(z_1)=p(x_1,z_1)=p(x_1|z_1)\,p(z_1)}$$

第一个概率是发射矩阵
第二个就是 π

对于 $\beta(z_n)$ 就需要确定 $\beta(z_N)$，对于 $\beta(z_N)$ 的确定需要转变一点思路

我们先看下面的式子

$$p(z_n|X)=\frac{p(X|z_n)p(z_n)}{p(X)}=\frac{p(x_1,\cdots,x_n|z_n)\,p(x_{n+1},\cdots x_N|z_n)\,p(z_n)}{p(X)}$$

$$=\frac{p(x_1,\cdots,x_n,z_n)\,p(x_{n+1},\cdots,x_N|z_n)}{p(X)}$$

$$=\frac{\alpha(z_n)\beta(z_n)}{p(X)}$$

我们注意到 $p(z_n|X)$ 可以由 $\alpha(z_n)$ 和 $\beta(z_n)$ 表示. (实际上，α 和 β 在 PRML 上就是为了求 $p(z_n|X)$ 和 $p(z_n,z_{n-1}|X)$ 而设计的)

当 $n=N$ 时，

$$p(z_N|X)=\frac{\alpha(z_N)\beta(z_N)}{p(X)}=\frac{p(X,z_n)\beta(z_N)}{p(X)}$$

我们注意到，当 $\beta(z_N)=1$ 时，等式得以成立. 故 $\boxed{\beta(z_N)=1}$.

从而，我们能够递归地计算所有的 α 和 β. 根据 α 和 β，我们就能计算 $p(X)$，甚至，其实 α 和 β 单独拿出一个就能计算似然函数，如下所示：

$$p(X)=\sum_{z_n}\alpha(z_n)\beta(z_n)$$

计算代价：$O(NK^2)$
N为链长度，K为隐变量的状态数

$$p(X)=\sum_{z_N}p(x_1,\cdots,x_N,z_N)=\sum_{z_N}\alpha(z_N)$$

$$p(X)=\sum p(x_1,\cdots,x_N,z_1)=\sum p(z_1)p(x_1,\cdots,x_N|z_1)=\sum_{z_1}p(z_1)p(x_1|z_1)\beta(z_1)$$

(尽管 $p(X)$ 可以这样计算，但是对于 Baum-Welch Algorithm 来说，其思想同 EM 相近，即我们要求的是 $p(X,Z)$，所以我们需要借助上面提到的 $p(z_n|X)$ 和 $p(z_n,z_{n-1}|X)$ 来求解)

缩放因子

我们继续观察 $\alpha(z_n)$ 和 $\beta(z_n)$ 的递推式.

$$\alpha(z_n) = p(x_n|z_n)\sum_{z_{n-1}} p(z_n|z_{n-1})\,\alpha(z_{n-1})$$

$$\beta(z_n) = \sum_{z_{n+1}} p(x_{n+1}|z_{n+1})\,p(z_{n+1}|z_n)\,\beta(z_{n+1})$$

这里面的每一项都是小于1的，因此，随着链的推进，$\alpha(z_n)$ 和 $\beta(z_n)$ 会指数地趋近于零. 在计算时出现下溢现象。

在独立同分布的情况下，我们可以使用log，将乘法变为加法，避免数值下溢，但是HMM中不能这样做，因为log中存在对乘积求和（其实就是对所有可能路径求和）

所以考虑使用缩放因子，将 $\alpha(z_n)$ 和 $\beta(z_n)$ 的值保持在一个量级上

以 $\alpha(z_n)$ 为例，我们考虑将缩放后的 $\alpha(z_n)$ 记为 $\hat{\alpha}(z_n)$.

定义 $\hat{\alpha}(z_n) = p(z_n|x_1,\cdots,x_n) = \dfrac{\alpha(z_n)}{p(x_1,\cdots,x_n)}$

（将 $\hat{\alpha}(z_n)$ 定义为该值的原因是，对于任意的 n，都是K个变量上的一个概率分布）

从而，引入缩放因子，$c_n = p(x_n|x_1,\cdots,x_{n-1}) \Rightarrow p(x_1,\cdots,x_n) = \prod_{i=1}^{n} c_i$

因此，$\alpha(z_n) = \left(\prod_{m=1}^{n} c_m\right)\hat{\alpha}(z_n)$

代入递推式 $c_n\hat{\alpha}(z_n) = p(x_n|z_n)\sum_{z_{n-1}} p(z_n|z_{n-1})\,\hat{\alpha}(z_{n-1})$

在每一次计算时，都需要存储 c_n，以便最后还原成 $\alpha(z_n)$，

（c_n 其实就是归一化因子，因为 $\hat{\alpha}(z_n) = p(z_n|x_1,\cdots,x_n)$，所以 c_n 就是等式右侧对 z_n 求和的结果）

同理，得到 $\beta(z_n)$ 的缩放递推式. $c_{n+1}\hat{\beta}(z_n) = \sum_{z_{n+1}} p(x_{n+1}|z_{n+1})\,p(z_{n+1}|z_n)\,\hat{\beta}(z_{n+1})$

其中 $\hat{\beta}(z_n) = \dfrac{p(x_{n+1},\cdots,x_N|z_n)}{p(x_{n+1},\cdots,x_N|x_1,\cdots,x_n)}$

Baum-Welch 算法

虽然，上一节中已经有一种能够以 $O(NK^2)$ 的复杂度求解似然函数的方法，但是，我们仍然不能直接使用最大似然法来求解。这会导致复杂的表达式且没有解析解。(具体原因可见混合模型那一节的说明)

所以我们使用 Baum-Welch 算法(也就是EM算法)来对 Learning 问题进行求解。

下面直接写出EM的Q函数。

$$
\begin{aligned}
Q(\lambda, \lambda^{(t)}) &= E_{p(Z|X,\lambda^{(t)})}\left[\log P(X,Z|\lambda)\right] \\
&= \sum_Z p(Z|X,\lambda^{(t)}) \log P(X,Z|\lambda) \\
&= \sum_Z p(Z|X,\lambda^{(t)}) \log\left[p(z_1)\prod_{i=2}^{N} p(z_i|z_{i-1}) \prod_{i=1}^{N} p(x_i|z_i)\right] \\
&= \sum_Z p(Z|X,\lambda^{(t)})\left[\log p(z_1) + \sum_{i=2}^{N}\log p(z_i|z_{i-1}) + \sum_{i=1}^{N} p(x_i|z_i)\right]
\end{aligned}
$$

我们注意到，后面括号中的部分是连加，则最前方的求和符号可以单独作用于后面的每一个子式中。那就可以进行化简，考虑作用于 $\log p(z_1)$ 的情景：$\sum_Z p(Z|X,\lambda^{(t)}) \log p(z_1) = \underset{\sim\sim\sim\sim\sim\sim\sim\sim}{\sum_{z_1} p(z_1|X,\lambda^{(t)})} \log p(z_1)$

最终可以看到只剩 $p(z_1|X,\lambda^{(t)})$

同理，另外的子式也可以化简为，$\sum_{i=2}^{N}\sum_{z_{i-1}}\sum_{z_i} p(z_{i-1},z_i|X,\lambda^{(t)}) \log p(z_i|z_{i-1})$

$\sum_{i=1}^{N}\sum_{z_i} p(z_i|X,\lambda^{(t)}) \log p(x_i|z_i)$

为了方便，我们引入两个记号：

$\gamma(z_i) = p(z_i | X, \lambda^{(t)})$ ， $\Rightarrow \gamma(z_{ik}) = p(z_{ik}=1 | X, \lambda^{(t)})$

$\xi(z_{i-1}, z_i) = p(z_{i-1}, z_i | X, \lambda^{(t)})$ $\Rightarrow \xi(z_{i-1,k}, z_{ij}) = p(z_{i-1,k}=1, z_{ij}=1 | X, \lambda^{(t)})$

由于二值随机变量的期望就是取值为1的概率，则

$\gamma(z_{ik}) = E[z_{ik}] = \sum_{z_i} \gamma(z_i) z_{ik}$

$\xi(z_{i-1,k}, z_{ij}) = E[z_{i-1,k}, z_{ij}] = \sum_{z_{i-1}} \sum_{z_i} \gamma(z_{i-1}, z_i) z_{i-1,k} \cdot z_{ij}$

这一句我认为不能将其看成是0-1分布，而是n=1的多项式分布，两者结果相同

两个画红线部分还需要理解。

则 $Q(\lambda, \lambda^{(t)}) = \sum_{z_1} \gamma(z_1) \log p(z_1) + \sum_{i=2}^{N} \sum_{z_{i-1}} \sum_{z_i} p(z_{i-1}, z_i | X, \lambda^{(t)}) \log p(z_i | z_{i-1})$

$+ \sum_{i=1}^{N} \sum_{z_i} p(z_i | X, \lambda^{(t)}) \log p(x_i | z_i)$

$= \sum_{z_1} \gamma(z_1) \sum_{k=1}^{K} z_{1k} \log \pi_k + \sum_{i=2}^{N} \sum_{z_{i-1}} \sum_{z_i} \xi(z_{i-1}, z_i) \sum_{k=1}^{K} \sum_{j=1}^{K} z_{i-1,k} \cdot z_{ij} \log A_{kj}$

$+ \sum_{i=1}^{N} \sum_{z_1} \gamma(z_i) \sum_{k=1}^{K} z_{ik} \log p(x_i | \phi_k)$ $p(x_i | z_i) = \prod_{k=1}^{K} p(x_i | \phi_k)^{z_{ik}}$

混合模型

$= \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{i=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} \xi(z_{i-1,k}, z_{ij}) \log A_{kj}$

$+ \sum_{i=1}^{N} \sum_{k=1}^{K} \log p(x_i | \phi_k)$

应用拉格朗日乘数法，关于π和A最大化 $Q(\lambda, \lambda^{(t)})$，可得：

$\pi_k = \dfrac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}$ ， $A_{kj} = \dfrac{\sum_{n=2}^{N} \xi(z_{n-1,k}, z_{nj})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,k}, z_{nl})}$

（PRML上还有对 ϕ_k 的求解，可以看一下）

Viterbi算法.

该算法解决的是Decoding问题，即给定观测序列X和参数λ，求隐变量序列. 即我们需要求解 $p(Z|X,\lambda)$ 其实就是寻找最大可能性的隐变量路径

回顾一下，之前在概率图模型那一章节有学习 max-product 算法，就是求概率最大的变量集合，包括后面改进的 max-sum 算法.

因此，可以用 max-sum 算法求解Decoding问题，该算法在HMM中就称之为Viterbi算法.

我们这里用联合概率推出Viterbi算法.

$$Z = \underset{Z}{\arg\max}\ p(x_1,\cdots,x_N,z_1,\cdots,z_N)$$

$$= \underset{Z}{\arg\max}\ p(z_1)\prod_{n=2}^{N} p(z_n|z_{n-1})\prod_{n=1}^{N} p(x_n|z_n)$$

因此，我们要求 $\underset{Z}{\max}\ p(z_1)\prod_{n=2}^{N} p(z_n|z_{n-1})\prod_{n=1}^{N} p(x_n|z_n)$

对其取对数. $\underset{Z}{\max}\left[\log p(z_1)+\sum_{n=2}^{N}\log p(z_n|z_{n-1})+\sum_{n=1}^{N}\log p(x_n|z_n)\right]$ 避免下溢.

交换max和求和的位置：

$$\max_{z_N}\left[\cdots\max_{z_2}\left[\log p(x_2|z_2)+\max_{z_1}\left[\log p(z_2|z_1)+\left[\log p(z_1)+\log p(x_1|z_1)\right]\right]\right]\cdots\right]$$

我们令
$$\begin{cases} w(z_1) = \log p(z_1)+\log p(x_1|z_1) \\ w(z_n) = \log p(x_n|z_n)+\underset{z_{n-1}}{\max}\left\{\log p(z_n|z_{n-1})+w(z_{n-1})\right\} \end{cases}$$

简单说就是记录最大值的来源

就得到了递推式. 即 $z_n = \underset{z_n}{\arg\max}\ w(z_n)$

只需要应用反向跟踪算法，就可以得到可能性最大的序列 (具体见PRML)

(其实思想类似于动态规划寻找价值最大路径，可以看白板视频，有简单说了一下)

小结

（白板视频）

HMM → time + mixture model.

时间序列　隐变量到观测变量是混合模型.

- Learning → $\lambda = \arg\max P(X|\lambda)$ (Baum-Welch / EM)
- Inference
  - decoding : $P(z_1, \cdots, z_N | x_1, \cdots, x_N)$ (Viterbi)
  - prob of evidence : $P(X|\lambda) = P(x_1, \cdots, x_N | \lambda)$ (Forward-Backward)
  - filtering : $P(z_n | x_1, \cdots, x_n)$ → online
  - smoothing : $P(z_n | x_1, \cdots, x_N)$ → offline.
  - prediction : $\begin{cases} P(z_{N+1} | x_1, \cdots, x_N) \\ P(x_{N+1} | x_1, \cdots, x_N) \end{cases}$

对于 filtering:

$$P(z_n | x_1, \cdots, x_n) = \frac{\alpha(z_n)}{P(x_1, \cdots, x_n)} = \hat{\alpha}(z_n)$$

对于 smoothing:

$$P(z_n | x_1, \cdots, x_N) = \frac{P(x_1, \cdots, x_N, z_n)}{P(x_1, \cdots, x_N)} = \frac{P(x_1, \cdots, x_n, z_n) P(x_{n+1}, \cdots, x_N | z_n)}{P(x_1, \cdots, x_N)} = \frac{\alpha(z_n)\beta(z_n)}{P(x_1, \cdots, x_N)}$$

对于 prediction:

① $$P(z_{N+1} | x_1, \cdots, x_N) = \sum_{z_N} P(z_{N+1}, z_N | x_1, \cdots, x_N) = \sum_{z_N} P(z_{N+1} | z_N) \underbrace{P(z_N | x_1, \cdots, x_N)}_{\text{filtering / smoothing}}$$

② $$P(x_{N+1} | x_1, \cdots, x_N) = \sum_{z_{N+1}} P(x_{N+1}, z_{N+1} | x_1, \cdots, x_N) = \sum_{z_{N+1}} P(x_{N+1} | z_{N+1}) \underbrace{P(z_{N+1} | x_1, \cdots, x_N)}_{\text{prediction ①}}$$