评论 Tan 等人 2019 年的《针对不完整多模态时间序列的深度马尔可夫模型的因式推理》

2020 年 4 月 4 日

潘旻琦

1. [Tan et al., 2019] 是对两年前的 [Wu and Goodman, 2018]（在模态之间条件独立的大前提下，将多模态的后验分布因式分解为每个模态的后验分布的高斯专家乘积）和三年前的 [Krishnan et al., 2017]（构造推理网络用 RNN 估计时间序列的一串潜在变量的后验分布，可以说是在时间轴上复制了一串变分自动编码机）的工作成果的拓展，从而为深度马尔可夫模型添加了两个新功能——多模态、丢失数据完形填空

2. 我认为 [Tan et al., 2019] 最核心的设计是在时间和模态上对潜变量的后验概率进行因式分解；在数学变换的过程中，作者有选择性地扔掉了很多因子，又有选择性地换顺序、加括号，最后弄出来 “昨天”、“今天”、“明天” 三大因子；这种大胆变形是值得学习的，可以想象作者可能读了 [Krishnan et al., 2017] 在时间轴上对后验分布进行因式分解之后觉得还不够过瘾，继续进一步因式分解，于是有了 [Tan et al., 2019]，但也不禁让我思考为什么作者可以这么大刀阔斧地变形：

   (a) 客观原因：首先是马尔可夫结构，马尔可夫性质提供了时间维度上乘积因式分解的可能性，而马氏概率图的变量之间的依赖如此之少又使得有很多条件独立性可供利用来做后验概率的形式化简；另一个原因是贝叶斯定理，这个定理提供了很大的代数变形的空间；最后，因为我们只是来取样，可以用概率 “成正比” 代替概率 “相等”，所以可以扔掉一些因子

   (b) 主观原因：作者应该是在追求变形到一种可以按照模态展开的乘积形式，这样如果有某个模态上的数据缺失，那就不乘这一项就好了，这样可以实现对丢失数据的支持

3. 论文里的 $p(z_t|z_{t-1}, x_{t:T}^{1:M}) = p(x_{t+1:T}^{1:M}|z_t)p(x_t^{1:M}|z_t)\frac{p(z_t|z_{t-1})}{p(x_{t:T}^{1:M}|z_{t-1})}$ 这一步代数变换很难

理解，我猜中间步骤可能是这样的：

$$
\begin{aligned}
p(z_t|z_{t-1}, x_{t:T}^{1:M}) &= \frac{p(z_{t-1}, z_t, x_{t:T}^{1:M})}{p(z_{t-1}, x_{t:T}^{1:M})} \\
&= \frac{p(x_{t:T}^{1:M}|z_{t-1}, z_t)p(z_{t-1}, z_t)}{p(z_{t-1}, x_{t:T}^{1:M})} \\
&= \frac{p(z_{t-1})p(z_t|z_{t-1})p(x_{t:T}^{1:M}|z_t)}{p(x_{t:T}^{1:M}|z_{t-1})p(z_{t-1})} \\
&= \frac{p(z_{t-1})p(z_t|z_{t-1})p(x_t^{1:M}|z_t)p(x_{t+1:T}^{1:M}|z_t)}{p(x_{t:T}^{1:M}|z_{t-1})p(z_{t-1})} \\
&= \frac{p(z_t|z_{t-1})p(x_t^{1:M}|z_t)p(x_{t+1:T}^{1:M}|z_t)}{p(x_{t:T}^{1:M}|z_{t-1})} \\
&= p(x_{t+1:T}^{1:M}|z_t)p(x_t^{1:M}|z_t)\frac{p(z_t|z_{t-1})}{p(x_{t:T}^{1:M}|z_{t-1})}
\end{aligned}
$$

上面第三个等号我的理由是因为概率图中从 z_{t-1} 到 $x_{t:T}^{1:M}$ 的每条路径都包含 $\cdots \to z_t \to \ldots$，所以满足 d-separation，于是 $x_{t:T}^{1:M} \perp\!\!\!\perp z_{t-1}|z_t$；上面第四个等号我的理由是概率图中 x_t 的父节点是 z_t，而 $x_{t+1:T}$ 都不是 x_t 的后继，于是局部马尔可夫性蕴含了 $x_t \perp\!\!\!\perp x_{t+1:T}|z_t$；然后作者通过扔掉下面这个因子把 $=$ 号换成了 $\propto$ 号

$$
\frac{1}{p(x_{t:T}^{1:M}|z_{t-1})}
$$

然后，作者没明说，但显然是做了与 [Wu and Goodman, 2018] 一样的假设——假设模态之间条件独立，即 $p(x_t^{1:M}|z_t) = \prod_{m=1}^{M} p(x_t^m|z_t)$，所以

$$
\begin{aligned}
p(z_t|z_{t-1}, x_{t:T}^{1:M}) &\propto p(x_{t+1:T}^{1:M}|z_t)p(x_t^{1:M}|z_t)p(z_t|z_{t-1}) \\
&= p(x_{t+1:T}^{1:M}|z_t)\left[\prod_{m=1}^{M} p(x_t^m|z_t)\right]p(z_t|z_{t-1}) \\
&= \frac{p(z_t|x_{t+1:T}^{1:M})p(x_{t+1:T}^{1:M})}{p(z_t)}\left[\prod_{m=1}^{M}\frac{p(z_t|x_t^m)p(x_t^m)}{p(z_t)}\right]p(z_t|z_{t-1})
\end{aligned}
$$

最后作者又扔掉了下面这个因子

$$
p(x_{t+1:T}^{1:M})\prod_{m=1}^{M} p(x_t^m) = p(x_{t:T}^{1:M})
$$

从而得到最终作者想要的一个未来 $\times$ 现在 $\times$ 过去的形式

$$
p(z_t|z_{t-1}, x_{t:T}) \propto p(z_t|x_{t+1:T})\left[\prod_m \frac{p(z_t|x_t^m)}{p(z_t)}\right]\frac{p(z_t|z_{t-1})}{p(z_t)}
$$

4. 论文里的 $p(z_t|x_{t+1:T}) = \mathbb{E}_{p(z_{t+1}|x_{t+1:T})}[p(z_t|z_{t+1})]$ 这一步代数变换很难理解，我猜中间步骤可能是这样的：

$$\begin{aligned} p(z_t|x_{t+1:T}) &= \int_{z_{t+1}} p(z_t, z_{t+1}|x_{t+1:T}) dz_{t+1} \\ &= \int_{z_{t+1}} p(z_t|z_{t+1}, x_{t+1:T}) p(z_{t+1}|x_{t+1:T}) dz_{t+1} \\ &= \int_{z_{t+1}} p(z_t|z_{t+1}) p(z_{t+1}|x_{t+1:T}) dz_{t+1} \\ &= \mathop{\mathbb{E}}_{p(z_{t+1}|x_{t+1:T})} [p(z_t|z_{t+1})] \end{aligned}$$

上面第三个等号我的理由是因为概率图中从 z_t 到 $x_{t+1:T}$ 的每条路径都包含 $\cdots \to z_{t+1} \to \ldots$，所以满足 d-separation，于是 $z_t \perp\!\!\!\perp x_{t+1:T}|z_{t+1}$

5. [Tan et al., 2019] 为了支持缺失数据真是一波三折，非常精彩；首先作者把后验概率分解成昨天 $\times$ 今天 $\times$ 明天的形式，分解开了昨天的观测数据、今天的观测数据、未来的观测数据对潜变量后验分布的影响，然后分而治之：

   (a) 昨天 $= \frac{1}{p(z_t)} p(z_t|z_{t-1})$，这个因子不涉及观测数据，缺数据无关紧要

   (b) 今天 $= \prod_m \frac{1}{p(z_t)} p(z_t|x_t^m)$，这个因子已经分解到了最佳的形式，每个模态上的数据的贡献一目了然，如果模态 m_i 上缺数据，那么在今天这个时间点上就不乘 $\frac{1}{p(z_t)} p(z_t|x_t^{m_i})$ 就可以了，后验分布不变，自然而然使得今天缺的某模态的数据对后验分布不产生任何影响

   (c) 明天 $= p(z_t|x_{t+1:T})$，这个因子很难继续因式分解了，但是不把缺失的数据的贡献因子分解出来就没法支持 $x_{t+1:T}$ 里那些缺失的数据，怎么办呢？这里非常有技巧性，作者先通过条件独立性做了下面这个代数变换，从 $p(z_t|x_{t+1:T})$ 里拆了一个 $p(z_t|z_{t+1})$ 出来：

   $$p(z_t|x_{t+1:T}) = \mathop{\mathbb{E}}_{p(z_{t+1}|x_{t+1:T})} [p(z_t|z_{t+1})]$$

   拆完之后再假定 $p(z_t|x_{t+1:T})$、$p(z_t|z_{t+1})$ 均满足高斯分布；目测作者的主观动机可能是要把丢失的数据从“明天”拉回到“今天”，因为 $t+1$ 时刻的缺失数据可以在求 $p(z_{t+1}|x_{t+1:T})$ 的时候处理，原本的求 $p(z_t|x_{t+1:T})$ 就变成了求 $p(z_t|z_{t+1})$ 的期望，就不涉及观测数据了；而高斯分布的假设使得作者可以套用 [Huber et al., 2011] 的矩估计方法，通过在 $p(z_{t+1}|x_{t+1:T})$ 下做粒子取样来估计 $p(z_t|x_{t+1:T})$，再用 $p(z_t|x_{t+1:T})$ 估计 $p(z_t|x_{t:T})$，此时递归结构就出现了，变成动态规划了：

   $$p(z_T|x_{T:T}) \to p(z_{T-1}|x_{T:T}) \to p(z_{T-1}|x_{T-1:T}) \to \cdots \to p(z_1|x_{1:T})$$

6. [Tan et al., 2019] 的损失函数设计得很精致，模型关于每批训练样本的损失 L 整体上是这样定义的：

$$L \equiv \frac{1}{2}L_{\text{filter}} + \frac{1}{2}L_{\text{smooth}} + \frac{1}{100}L_{\text{match}}$$

(a) L_{filter} 的设计目标是实现反向预测时的最大似然学习，由 $M+1$ 个 ELBO 损失组成，这么多 ELBO 是因为要让模型学习到如何联合 M 个模态以及在 $1 \dots M$ 的某个模态下孤立地执行推理：

$$\begin{aligned}
L_{\text{filter}} &\equiv L_{\text{filter}}^{1:M} + \sum_{m=1}^{M} L_{\text{filter}}^{m} \\
L_{\text{filter}}^{1:M} &\equiv \sum_{t=1}^{T}[\mathop{\mathbb{E}}_{q(z_t|x_{t:T}^{1:M})} \sum_{m=1}^{M} \lambda_m \log p(x_t^m|z_t) - \\
&\qquad \mathop{\mathbb{E}}_{q(z_{t+1}|x_{t+1:T}^{1:M})} \beta \text{KL}\left(q(z_t|z_{t+1}, x_t^{1:M}) \| p(z_t|z_{t+1})\right)] \\
L_{\text{filter}}^{m} &\equiv \sum_{t=1}^{T}[\mathop{\mathbb{E}}_{q(z_t|x_{t:T}^{m})} \lambda_m \log p(x_t^m|z_t) - \\
&\qquad \mathop{\mathbb{E}}_{q(z_{t+1}|x_{t+1:T}^{m})} \beta \text{KL}\left(q(z_t|z_{t+1}, x_t^{m}) \| p(z_t|z_{t+1})\right)]
\end{aligned}$$

其中 $\lambda_1, \dots, \lambda_M, \beta$ 这些加权处理是为了让 ELBO 损失在 “促进各个模态上的最大似然重建” 和 “促进潜变量对于先验分布的忠诚” 这两个效果之间求得平衡，其中 $\lambda_1, \dots, \lambda_M$ 被设定为常数超参数，然后让 β 在训练早期的数个纪元内从 0 线性退火到 1，所以一开始着重优化各个模态上的最大似然重建，后期开始着重促进潜变量对于先验分布的忠诚

(b) L_{smooth} 与 L_{filter} 的设计完全类似，只是把反向预测改成正向平滑：

$$\begin{aligned}
L_{\text{smooth}} &\equiv L_{\text{smooth}}^{1:M} + \sum_{m=1}^{M} L_{\text{smooth}}^{m} \\
L_{\text{smooth}}^{1:M} &\equiv \sum_{t=1}^{T}[\mathop{\mathbb{E}}_{q(z_t|x_{1:T}^{1:M})} \sum_{m=1}^{M} \lambda_m \log p(x_t^m|z_t) - \\
&\qquad \mathop{\mathbb{E}}_{q(z_{t-1}|x_{1:T}^{1:M})} \beta \text{KL}(q(z_t|z_{t-1}, x_{t:T}^{1:M}) \| p(z_t|z_{t-1}))] \\
L_{\text{smooth}}^{m} &\equiv \sum_{t=1}^{T}[\mathop{\mathbb{E}}_{q(z_t|x_{1:T}^{m})} \lambda_m \log p(x_t^m|z_t) - \\
&\qquad \mathop{\mathbb{E}}_{q(z_{t-1}|x_{1:T}^{m})} \beta \text{KL}\left(q(z_t|z_{t-1}, x_{t:T}^{m}) \| p(z_t|z_{t-1})\right)]
\end{aligned}$$

L_{filter} 和 L_{smooth} 之所以要各取一半同时优化，部分原因是正向平滑的时候要从反向预测得到的分布上做粒子取样，即 L_{smooth} 依赖于 L_{filter}

(c) L_{match} 损失定义如下：

$$L_{\text{match}} \equiv \text{KL}\left(p(z_t) \| \mathop{\mathbb{E}}_{z_{t-1}} p(z_t|z_{t-1})\right) + \text{KL}\left(p(z_t) \| \mathop{\mathbb{E}}_{z_{t+1}} p(z_t|z_{t+1})\right)$$

这个损失看似什么都没做，因为数学上有

$$\mathop{\mathbb{E}}_{z_{t-1}} p(z_t|z_{t-1}) = \int_{z_{t-1}} p(z_t|z_{t-1})p(z_{t-1})dz_{t-1} = p(z_t)$$
$$\mathop{\mathbb{E}}_{z_{t+1}} p(z_t|z_{t+1}) = \int_{z_{t+1}} p(z_t|z_{t+1})p(z_{t+1})dz_{t+1} = p(z_t)$$

但是因为 $p(z_t)$ 是粒子取样的，上述等式在工程上不成立；因此我认为这个损失的设计目标是缩小工程与理论的距离，促进潜变量的分布在没有新的观测数据出现的情况下要保持不变，作者很可能担心在反向预测、正向平滑的过程中这个稳定性遭到破坏

7. 这里记录一下我从源代码反推得到的 [Tan et al., 2019] 的噪声双向螺旋数据集的一些细节：

(a) 这个数据集要求生成 500 条顺时针螺旋、500 个逆时针螺旋，每条螺旋都定义成如下的一个离散的时间序列 $\mathbf{x}(t) : \{0, 1, 2 \dots 99\} \to \mathbb{R}^2$

$$\mathbf{x}(t) \equiv \begin{bmatrix} \sqrt{R} \cdot r(t) \cos\theta(t) + 0.1 \cdot \mathcal{N} \\ \frac{1}{\sqrt{R}} \cdot r(t) \sin\theta(t) + 0.1 \cdot \mathcal{N} \end{bmatrix}$$

其中长宽比 $R \in 2^{\mathcal{U}_{[-1,1)}}$，$r(0) \dots r(99)$ 和 $\theta(0) \dots \theta(99)$ 是等差数列：

$$r(0) \equiv 0.25 + \mathcal{U}_{[0,0.5)} \dots r(99) \equiv 2.25 + \mathcal{U}_{[0,0.5)}$$
$$\theta(0) \equiv \mathcal{U}_{[0,\pi)} \dots \theta(99) \equiv \mathcal{U}_{[4\pi,5\pi)}$$
或
$$\theta(0) \equiv \mathcal{U}_{[0,-\pi)} \dots \theta(99) \equiv \mathcal{U}_{[-4\pi,-5\pi)}$$

其中 $\mathcal{U}$ 代表从均匀分布上取样，最后添加标准正态分布噪声 $\mathcal{N}$ 可能是为了使数据集更加逼真，概因此作者把它叫做“嘈杂螺旋”

(b) 每条螺旋包含 100 个 $\mathbb{R}^2$ 中的点，该实验以这个点的横、纵坐标为两个模态，因此每个模态的观测数据 $x_t^m \in \mathbb{R}$

(c) 该实验规定潜变量 $z_t \in \mathbb{R}^5$，潜变量后验分布估计 $\tilde{q}(z_t|x_t^m) = \frac{p(z_t|x_t^m)}{p(z_t)}$ 被定义成一些五元正态分布 $\mathcal{N}(\boldsymbol{\mu}(x_t^m), \boldsymbol{\Sigma}(x_t^m))$，其中

$$\begin{aligned}\boldsymbol{\mu} &\equiv (\mathbb{R}^5 \leftarrow \mathbb{R}^{20}) \circ \mathrm{ReLU} \circ (\mathbb{R}^{20} \leftarrow \mathbb{R}) \\ \boldsymbol{\Sigma} &\equiv 0.001 I_5 + \mathrm{SoftPlus} \circ (\mathbb{R}^5 \leftarrow \mathbb{R}^{20}) \circ \mathrm{ReLU} \circ (\mathbb{R}^{20} \leftarrow \mathbb{R})\end{aligned}$$

(d) 发射分布 $p(x_t^m|z_t)$ 被定义成一些正态分布 $\mathcal{N}(\mu(z_t), \sigma(z_t))$，其构造与潜变量后验分布估计类似：

$$\begin{aligned}\mu &\equiv \mathrm{FC}_\mu \circ \mathrm{ReLU}_h \circ \mathrm{FC}_h \\ \sigma &\equiv 0.001 + \mathrm{SoftPlus}_\sigma \circ \mathrm{FC}_\sigma \circ \mathrm{ReLU}_h \circ \mathrm{FC}_h \\ \mathrm{FC}_h &: \mathbb{R}^5 \to \mathbb{R}^{20} \\ \mathrm{FC}_\mu &: \mathbb{R}^{20} \to \mathbb{R} \\ \mathrm{FC}_\sigma &: \mathbb{R}^{20} \to \mathbb{R}\end{aligned}$$

8. [Tan et al., 2019] 的损失的 ELBO 项的 KL 散度项之所以可以设计成

$$\sum_{t=1}^{T} [\mathbb{E}_{q(z_{t+1}|x_{t+1:T})} \mathrm{KL}\left(q(z_t|z_{t+1}, x_t) \| p(z_t|z_{t+1})\right)]$$

和

$$\sum_{t=1}^{T} [\mathbb{E}_{q(z_{t-1}|x_{1:T})} \mathrm{KL}(q(z_t|z_{t-1}, x_{t:T}) \| p(z_t|z_{t-1}))]$$

的形式，而不是直接用

$$\mathrm{KL}\left(q(z_{1:T}|x_{1:T}) \| p(z_{1:T})\right),$$

是出于三年前的 [Krishnan et al., 2017] 的贡献，它当时贡献的一种使用潜变量的先验分布得到的 ELBO 损失的分解方法，利用高斯分布的假设使得 KL 项具有解析梯度；否则 KL 项没有解析形式，必须诉诸于蒙特卡洛采样来估计 KL 项的梯度，但这会导致估计和梯度的方差较高，不是很稳定

9. [Tan et al., 2019] 的嘈杂螺实验除了用了神经网络去参数化潜变量后验分布估计

$$\tilde{q}(z_t|x_t^m) = \frac{1}{p(z_t)} p(z_t|x_t^m)$$

和发射分布 $p(x_t^m|z_t)$ 之外，还用神经网络参数化了潜变量状态转移分布估计

$$\tilde{q}(z_t|z_{t-1}) = \frac{1}{p(z_t)} p(z_t|z_{t-1}), \tilde{q}(z_t|z_{t+1}) = \frac{1}{p(z_t)} p(z_t|z_{t+1}),$$

假定了它们俩都是五元正态分布，而且都使用了类似于 [Krishnan et al., 2017] 的门控转移函数的一种参数化结构；这个 “门控” 是为了灵活地为某些维度选择线性状态转移、而为其他维度选择非线性状态转移；例如 $\tilde{q}(z_t|z_{t-1}) \equiv \mathcal{N}(\boldsymbol{\mu}(z_{t-1}), \boldsymbol{\Sigma}(z_{t-1}))$，其中

$$
\begin{aligned}
\text{门控} \equiv & \text{Sigmoid} \circ (\mathbb{R}^5 \leftarrow \mathbb{R}^{20}) \circ \text{ReLU} \circ (\mathbb{R}^{20} \leftarrow \mathbb{R}^5) \\
\boldsymbol{\mu} \equiv & [(1 - \text{门控}) \odot] \circ (\mathbb{R}^5 \leftarrow \mathbb{R}^5) + \\
& [\text{门控} \odot] \circ (\mathbb{R}^5 \leftarrow \mathbb{R}^{20}) \circ \text{ReLU} \circ (\mathbb{R}^{20} \leftarrow \mathbb{R}^5) \\
\boldsymbol{\Sigma} \equiv & 0.001 I_5 + \text{Softplus} \circ (\mathbb{R}^5 \leftarrow \mathbb{R}^5) \circ (\mathbb{R}^5 \leftarrow \mathbb{R}^{20}) \circ \text{ReLU} \circ (\mathbb{R}^{20} \leftarrow \mathbb{R}^5)
\end{aligned}
$$

潜变量的全局先验分布 $p(z_t)$ 的参数化十分简单，一共才 10 个参数而已：

$$
p(z) \sim \mathcal{N}\left(\boldsymbol{\mu} = \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix}, \boldsymbol{\Sigma} = 0.001 I_5 + \begin{bmatrix} e^{\sigma_0} & 0 & 0 & 0 & 0 \\ 0 & e^{\sigma_1} & 0 & 0 & 0 \\ 0 & 0 & e^{\sigma_2} & 0 & 0 \\ 0 & 0 & 0 & e^{\sigma_3} & 0 \\ 0 & 0 & 0 & 0 & e^{\sigma_4} \end{bmatrix} \right)
$$

10. [Tan et al., 2019] 之所以用神经网络去参数化 $\tilde{q}(z_t|x_t^m), \tilde{q}(z_t|z_{t-1}), \tilde{q}(z_t|z_{t+1})$ 的时候藏了一个 $\frac{1}{p(z_t)}$ 因子在里面，是为了绕开高斯分布做除法需要满足的数学约束这个技术细节，两年前 [Wu and Goodman, 2018] 的 §2.1 也采用了类似的做法；但是，这里绕开了后面就要补上，所以在 $\tilde{q}$ 上做粒子取样之后，得到的粒子还需要与 $p(z_t)$ 取样出来的粒子做高斯专家乘积；因为 $\tilde{q}$ 之前做过的那些高斯假设，每个粒子在这里都可以被看作一位高斯专家，同理 $p(z_t)$ 取样出来的每个粒子也都是一位高斯专家，他们相乘还是高斯分布，而且有解析解 [Cao and Fleet, 2014]；乘完之后粒子的数量减半，最后再统计得到的粒子的均值和方差，才得到想要的不带波浪号的 $q(z_t|x_t^m), q(z_t|z_{t-1}), q(z_t|z_{t+1})$ 的参数，得以去掉了 $\frac{1}{p(z_t)}$ 这个因子

11. [Tan et al., 2019] 的 Weizmann 人类动作实验规定了潜变量 $z_t \in \mathbb{R}^{256}$，参数一共有四百万多个，存储在外存储器上有 17MB 之大，我从源代码反推的各模态、各分布的参数化方法如下：

    (a) 原视频模态的潜变量后验分布估计 $\tilde{q}(z_t|x_t) \equiv \mathcal{N}(\boldsymbol{\mu}(x_t), \boldsymbol{\Sigma}(x_t))$ 是用如下的卷

积神经网络去参数化的：

$$
\begin{aligned}
\mathrm{CNN} \equiv (\mathbb{R}^{64\times8\times8} &\xleftarrow{\text{3×3 卷积核，步伐 2，填充 1}} \mathbb{R}^{32\times16\times16})\circ \\
&\mathrm{ReLU}\circ\mathrm{BN}\circ(\mathbb{R}^{32\times16\times16} \xleftarrow{\text{3×3 卷积核，步伐 2，填充 1}} \mathbb{R}^{16\times32\times32})\circ \\
&\mathrm{ReLU}\circ\mathrm{BN}\circ(\mathbb{R}^{16\times32\times32} \xleftarrow{\text{3×3 卷积核，步伐 2，填充 1}} \mathbb{R}^{3\times64\times64}) \\
\boldsymbol{\mu} &\equiv (\mathbb{R}^{256} \leftarrow \mathbb{R}^{64\times8\times8})\circ\mathrm{CNN} \\
\boldsymbol{\Sigma} &\equiv \mathrm{Softplus}\circ(\mathbb{R}^{256} \leftarrow \mathbb{R}^{64\times8\times8})\circ\mathrm{CNN}
\end{aligned}
$$

(b) 轮廓模态的潜变量后验分布估计与原视频模态的类似，只是一开始的输入由 $\mathbb{R}^{3\times16\times16}$ 改为 $\mathbb{R}^{1\times16\times16}$

(c) 人名标签、动作标签模态的潜变量后验分布估计 $\tilde{q}(z_t|x_t) \equiv \mathcal{N}(\boldsymbol{\mu}(x_t), \boldsymbol{\Sigma}(x_t))$ 的参数化方法如下：

$$
\begin{aligned}
\text{嵌入} &\equiv \mathrm{ReLU}\circ(\mathbb{R}^{256} \leftarrow \{0,1,\ldots,9\}) \\
\boldsymbol{\mu} &\equiv (\mathbb{R}^{256} \leftarrow \mathbb{R}^{256})\circ\mathrm{ReLU}\circ(\mathbb{R}^{256} \leftarrow \mathbb{R}^{256})\circ\text{嵌入} \\
\boldsymbol{\Sigma} &\equiv 0.001I_5 + \mathrm{SoftPlus}\circ(\mathbb{R}^{256} \leftarrow \mathbb{R}^{256})\circ\mathrm{ReLU}\circ(\mathbb{R}^{256} \leftarrow \mathbb{R}^{256})\circ\text{嵌入}
\end{aligned}
$$

(d) 原视频模态的发射分布 $p(x_t|z_t)$ 的参数化方法如下：

$$
\begin{aligned}
p(x_t|z_t) \equiv[&\mathrm{Sigmoid}\circ(\mathbb{R}^{3\times64\times64} \xleftarrow{\text{4×4 卷积核，步伐 2，填充 1}} \mathbb{R}^{16\times32\times32})\circ \\
&(\mathbb{R}^{16\times32\times32} \xleftarrow{\text{4×4 卷积核，步伐 2，填充 1}} \mathbb{R}^{32\times16\times16})\circ \\
&(\mathbb{R}^{32\times16\times16} \xleftarrow{\text{4×4 卷积核，步伐 2，填充 1}} \mathbb{R}^{64\times8\times8})\circ \\
&\mathrm{ReLU}\circ(\mathbb{R}^{64\times8\times8} \leftarrow \mathbb{R}^{256})(z_t)](x_t)
\end{aligned}
$$

(e) 轮廓模态的发射分布与原视频模态的类似，只是一开始的输入由 $\mathbb{R}^{3\times16\times16}$ 改为 $\mathbb{R}^{1\times16\times16}$

(f) 人名标签、动作标签模态的发射分布 $p(x_t|z_t)$ 的参数化方法如下：

$$
p(x_t|z_t) \equiv [\mathrm{Softmax}\circ(\mathbb{R}^{10} \leftarrow \mathbb{R}^{256})\circ\mathrm{ReLU}\circ(\mathbb{R}^{256} \leftarrow \mathbb{R}^{256})(z_t)](x_t)
$$

12. [Tan et al., 2019] 训练完的模型如何进行完形填空呢？我根据源代码反推的步骤如下：

(a) 先收集多模态观测时序数据 $x_{1:T}^{1:M}$，这些数据可能来源于嘈杂或异步的传感器，作为对训练完的模型的输入

(b) 之后通过 $x_{1:T}^{1:M}$ 求各个模态 $m \in \{1,\ldots,M\}$ 的潜变量后验分布估计 $\tilde{q}(z_t|x_t^m)$

(c) 之后，以训练好的全局先验 $p(z_T)$ 作为 $t=T$ 时刻的先验分布 $p(z_t|z_{t+1})$，从最后一个时刻倒着出发，对潜变量执行反向预测；期间的每次后验推断

$$q(z_t|x_t, z_{t+1}) \leftarrow p(z_t|z_{t+1}) \prod_m \tilde{q}(z_t|x_t^m)$$

并不对该结果进行粒子取样，而直接采用得到的 $q(z_t|x_t, z_{t+1})$ 的高斯均值作为下次迭代 $t-1$ 时刻的 z_t，当作输入去产生 $p(z_{t-1}|z_t)$；这样迭代下去，直到求出所有的 $\{q(z_t|z_{t+1}) : t \in [1, T]\}$ 放在一边待用

(d) 之后，以训练好的全局先验 $p(z_1)$ 作为 $t=1$ 时刻的先验分布 $p(z_t|z_{t-1})$，从第一个时刻出发，对潜变量执行正向平滑；期间的每次后验推断

$$q(z_t|z_{t-1}, x_{t:T}) \leftarrow q(z_t|z_{t+1}) \prod_{\mathcal{M}} [\tilde{q}(z_t|x_t^m)] \frac{p(z_t|z_{t-1})}{p(z_t)}$$

并不对推断结果 $q(z_t|z_{t-1}, x_{t:T})$ 进行粒子取样，而直接采用得到的 $q(z_t|z_{t-1}, x_{t:T})$ 的高斯均值作为下次迭代 $t+1$ 时刻的 $\hat{z}_t$，当作输入去产生 $p(\hat{z}_{t+1}|\hat{z}_t)$；这样迭代下去，实则递归地取出了所有条件平滑后验的高斯均值：

$$\hat{z}_t \equiv \arg\max_{z_t} q(z_t|\hat{z}_{t-1}, x_{t:T})$$

(e) 与三年前的 [Krishnan et al., 2017] 一样绕开 Viterbi 算法，直接采用上述 $\hat{z}_{1:T}$ 作为对潜变量的最大后验概率 $z_{1:T}^*$ 的估算：

$$z_{1:T}^* \equiv \arg\max_{z_{1:T}} p(z_{1:T}|x_{1:T}^{1:M})$$

(f) 然后开始重建，用各个模态训练好的发射分布 $p(x_t^m|z_t)$ 求观测值的最大似然估计 $\hat{x}_{1:T}^{1:M}$：

$$\hat{x}_{1:T}^{1:M} \equiv \arg\max_{x_{1:T}^{1:M}} p(x_{1:T}^{1:M}|\hat{z}_{1:T})$$

(g) 最后，采用 $\hat{x}_{1:T}^{1:M}$ 作为完形填空的结果

13. 尝试了复现 [Tan et al., 2019] 的嘈杂螺旋实验，我的实验结果如下：

(a) 训练前，我生成了 1000 条嘈杂螺旋，其中 500 条是顺时针螺旋、500 条是逆时针螺旋；我保留了其中的 400 条作为测试集，剩下的 600 条作为训练集

(b) 训练时，我以 100 条螺旋为一批，以 6 批为一个纪元，对反向预测我使用了 25 个粒子，对先验分布 $p(z_t)$ 我使用了 50 个粒子，共训练了五百个纪元；在训练过程中，损失 L 出现先上升后下降的现象：首个纪元之后，损失为 3.8；二十六至七十一个纪元之间损失一度增长到 4.0；之后经过四百多个纪元，渐渐降到 3.2；为什么先升后降呢？我猜测这与上述 ELBO 损失的 β 退火有关

(c) 对于嘈杂螺旋实验，评估发现完全没有重建能力，回顾日志发现其训练时的重建均方误差最终稳定在一个很高的水平，说明训练是失败的，这可能是与某些超参数的选择有关

14. 复现了 [Tan et al., 2019] 的 Weizmann 人类动作实验，评估了嘈杂螺旋实验的推理性能，我的实验结果如下：

    (a) 我观察了一下人类动作数据集的情况，那些动作小短片的帧率是 25，分辨率是 180×144，视频长度有 1 秒至 5 秒不等，因此包含 25 至 125 个帧不等；[Tan et al., 2019] 为了加快视频数据集的训练速度，将每个输入序列分割成了 25 个时间步长的片段，定死了每个小短片 25 个帧，亦即 $t \in \{0, \ldots, 24\}$，这个分割也导致了样本数量的增加

    (b) 我训练人类动作的时候只使用了原视频、人名标签、动作标签这三个模态，没有使用轮廓模态；我训练了五百个纪元，每个纪元 10 批次，每批 25 个样本；第一批训练结束后的损失是 17300，几百个纪元之后基本稳定在 16131 左右

    (c) 训练后，加载最佳参数，分别在训练集和测试集上针对每个模态、每个维度随机删除 50% 的观测数据后，用训练好的模型进行完形填空，评估效果如下：不管是训练集还是测试集，重建效果整体是很模糊的，就连观测到的那几帧重建之后也比原先观察到的模糊了很多，但对于那些被删除的帧重建后的视频效果大体上是流畅的；对于训练集来说，标签大部分都是正确的，能正确补全重建的帧的人和动作标签；对于测试集来说，人的标签肯定是不正确的(因为这个人从来没有见过)，但是动作标签的完形填空基本上依然是正确的

参考文献

[Cao and Fleet, 2014] Cao, Y. and Fleet, D. J. (2014). Generalized product of experts for automatic and principled fusion of gaussian process predictions. *CoRR*, abs/1410.7827.

[Huber et al., 2011] Huber, M. F., Beutler, F., and Hanebeck, U. D. (2011). Semi-analytic gaussian assumed density filter. In *Proceedings of the 2011 American Control Conference*, pages 3006–3011.

[Krishnan et al., 2017] Krishnan, R. G., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Thirty-first aaai conference on artificial intelligence*.

[Tan et al., 2019] Tan, Z.-X., Soh, H., and Ong, D. C. (2019). Factorized inference in deep markov models for incomplete multimodal time series. *ArXiv*, abs/1905.13570.

[Wu and Goodman, 2018] Wu, M. and Goodman, N. (2018). Multimodal generative models for scalable weakly-supervised learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5575–5585. Curran Associates, Inc.