Chapter 1

Model definition

1.1 Notations

We consider the prediction task of a set of observations $(y^1, \cdots y^T)$ given a set of input $(u^1, \cdots u^T)$.

1.2 Model

We define a L layer RNN followed by a fully connected layer. At time step t,

$$\begin{cases} y_{t+1} = \tanh(W_y x_{t+1}^L + b_y) \\ x_{t+1}^l = \tanh(W_{xx}^l x_t^l + W_{xu}^l x_{t+1}^{l-1} + b_x^l) \quad \forall 1 \leq l \leq L \end{cases}$$

with $x_t^0 \equiv u_t \; \forall t$ and $x_0^l \equiv 0 \; \forall 1 \leq l \leq L$.

Let's consider the weights of the last RNN and fully connected layers as $\theta \equiv (W_{xx}^L, W_{xu}^L, b_x^L, W_y, b_y)$. We can define a new matrix x_t at each time step corresponding to the concatenation of all RNN layers: $x_t \equiv (x_t^1 \cdots x_t^L)$. We also introduce two sequences of random noises as i.i.d real valued random variables ϵ and η, with covariance matrices Σ_y and Σ_x. We can now write our model in terms of two functions f and g as:

$$\begin{cases} y_{t+1} = f_\theta(x_{t+1}) + \epsilon_{t+1} & \text{observation model} \\ x_{t+1} = g_\theta(x_t, u_{t+1}) + \eta_{t+1} & \text{state model} \end{cases} \tag{1.1}$$

In the following section, we will focus on minimizing the log likelihood

$$\log \; p_\mu(X_{1:T}, y_{1:T}, u_{1:T}) \tag{1.2}$$

1.3 Minimization

In order to minimize 1.2, we apply an EM strategy. Let $\mu_p = (\theta_p, \Sigma_{x,p}, \Sigma_{y,p})$, we will compute at each EM step:

$$Q(\hat{\mu}_p, \mu) = \mathbb{E}_{\hat{\mu}_p}\left[\log\ p_\mu(X_{1:T}, y_{1:T}, u_{1:T}) | y_{1:T}\right] \tag{1.3}$$

We can start by developing the log likelihood:

$$\begin{aligned}
\log p_\mu(X_{1:T}, y_{1:T}, u_{1:T}) &= \frac{1}{T}\log\left(\prod_{k=1}^{T} p_\mu(x_k|x_{k-1}, u_k) p_\mu(y_k|x_k)\right) \\
&= \frac{1}{T}\sum_{k=1}^{T}\log\ p_\mu(x_k|x_{k-1}, u_k) + \log\ p_\mu(y_k|x_k) \\
&= \frac{1}{T}\sum_{k=1}^{T}\log\left(\det(2\pi\Sigma_x)^{-1/2}\exp(-\frac{1}{2}(x_k - g_\theta(x_{k-1}, u_k))^T\Sigma_x^{-1}(x_k - g_\theta(x_{k-1}, u_k))\right) \\
&+ \log\left(det(2\pi\Sigma_y)^{-1/2}\exp(-\frac{1}{2}(y_k - f_\theta(x_k))^T\Sigma_y^{-1}(y_k - f_\theta(x_k))\right) \\
&= -\frac{1}{2}\log|\Sigma_x| - \frac{1}{2}\log|\Sigma_y| \\
&- \frac{1}{2T}\sum_{k=1}^{T}(x_k - g_\theta(x_{k-1}, u_k))^T\Sigma_x^{-1}(x_k - g_\theta(x_{k-1}, u_k)) \\
&- \frac{1}{2T}\sum_{k=1}^{T}(y_k - f_\theta(x_k))^T\Sigma_y^{-1}(y_k - f_\theta(x_k))
\end{aligned}$$

We can identify two approaches to jointly minimizing Σ_x, Σ_y and θ iteratively:

1. At each step of the EM algorithm, we can compute the maximum expectation for both covariant matrices, given the previous value of θ, then approximate the new θ by minimizing an argmin, through gradient descent for example. To express the maximum expectation of Σ_y, we start by searching for the zeros of the derivate of the convex function $\Sigma_y \mapsto p_\mu(X_{1:T}, y_{1:T}, u_{1:T})$. Results are similar for Σ_x.

$$\frac{\partial p_\mu(X_{1:T}, y_{1:T}, u_{1:T})}{\partial \Sigma_y^{-1}} = \frac{1}{2}\Sigma_y - \frac{1}{2T}\sum_{k=1}^{T}(x_k - f_\theta(x_k)) \cdot (x_k - f_\theta(x_k))'$$

$$\frac{\partial p_\mu(X_{1:T}, y_{1:T}, u_{1:T})}{\partial \Sigma_y^{-1}} = 0 \implies \Sigma_y = \frac{1}{T}\sum_{k=1}^{T}(y_k - f_\theta(x_k))(y_k - f_\theta(x_k))'$$

$$\frac{\partial p_\mu(X_{1:T}, y_{1:T}, u_{1:T})}{\partial \Sigma_x^{-1}} = 0 \implies \Sigma_x = \frac{1}{T}\sum_{k=1}^{T}(x_k - g_\theta(x_{k-1}, u_k))(x_k - g_\theta(x_{k-1}, u_k))'$$

$$\Sigma_{y,p+1} = \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}_{\hat{\mu}_p}\left[(y_k - f_{\theta_p}(x_k))(y_k - f_{\theta_p}(x_k))'|y_{1:T}\right]$$
$$\Sigma_{x,p+1} = \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}_{\hat{\mu}_p}\left[(x_k - g_{\theta_p}(x_{k-1},u_k))(x_k - g_{\theta_p}(x_{k-1},u_k))'|y_{1:T}\right]$$
$$\theta_{p+1} = \underset{\theta}{\operatorname{argmin}}\,\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}_{\hat{\mu}_p}[(y_k - f_{\theta_p}(x_k))'\Sigma_{y,p+1}^{-1}(y_k - f_{\theta_p}(x_k))$$
$$+ (x_k - g_{\theta_p}(x_{k-1},u_k)'\Sigma_{x,p+1}^{-1}(x_k - g_{\theta_p}(x_{k-1},u_k))|y_{1:T}]$$

2. We can also ignore the explicit expression for the covariant matrices, and approximate both θ and Σ by gradient descent at each time step. Although we're putting aside a valuable result about Σ, this method could prove more efficient from an implementation perspective.

In the following sections, we will detail the second option. In Section 1.4, we detail the approximation of the posterior law through Sequential Monte Carlo approaches. In Section 1.5, we describe the algorithm to train our model through gradient descent.

1.4 Sequential Monte Carlo Approach

1.4.1 Filter

In order to compute the conditional expectations in the previous expressions, we will iteratively sample trajectories $\xi_{1:T}^i$ associated with weights ω^i with respect to the density $p_\theta(x|y)$, using a sequential Monte Carlo particle filter.

At time step $k = 1$, $(\xi_1^l)_{l=1}^N$ are sampled independently from the first hidden state, and associated with sampling weights proportional to the observation density q_θ:

$$\xi_1^i \sim \mathcal{N}(x_1, \Sigma_x)$$
$$\omega_1^i \sim q_\theta(\xi_1^i)$$

At time step $k + 1$, we sample indices I of the particles to propagate, based on their previous weights. After propagation, particles weights are computed following the observation density function:

$$\mathbb{P}(I_{k+1}^i = j) = \omega_k^j \quad \forall 1 \le j \le N$$

$$\omega_{k+1}^i \sim q_\theta(\xi_{k+1}^i)$$

1.4.2 Smoother

Using the poor man filter, we get N trajectories:

$$\xi_{1:k+1}^i = (\xi_{1:k}^{I_{k+1}^i}, \xi_{k+1}^i)$$

1.4.3 Approximation

We can now approximate this conditional expectation for any measurable bounded function h:

$$\begin{aligned}\Phi_k^M[h] &= \mathbb{E}_{\hat{\mu}_p}\left[h(x)|y_{1:T}\right] \\ &= \sum_{i=1}^{N} \omega_T^i h(\xi_{1:T}^i)\end{aligned}$$

1.5 Gradient descent

1.5.1 Forward pass

During the forward pass, we generate a set of N particles under the law $p(x|y)$ for fixed values of θ_p, $\Sigma_{x,p}$ and $\Sigma_{y,p}$. In order to predict each new time step $k+1$, particles from the previous step are attributed weights ω_k^i proportionally to the density probability around the targeted value y_k. should we add ϵ_k ?

$$\omega_k^i \sim \exp\left(-\frac{1}{2}(y_k - f_{\theta_p}(x_k^i))'\Sigma_{y,p}^{-1}(y_k - f_{\theta_p}(x_k^i))\right)$$

We then select a new population from these particles indexed by I_{k+1}^i, based on their weights.

$$\mathbb{P}(I_{k+1}^i = j) = \omega_k^j \quad \forall 1 \leq j \leq N$$

The current hidden state is computed for the selected particles.

$$x_{k+1}^i = g_{\theta_p}(x_k^{I_{k+1}^i}, u_{k+1}) + \eta_{k+1}^i$$

We initialize the sequence with a random initial hidden state.

1.5.2 Loss function

Considering that we have computed a set of N trajectories $(\xi_{1:T}^i)$, $1 \leq i \leq N$, associated with weights (ω^i), we define our loss function as an approximation of the log likelihood:

$$\begin{aligned}\mathbb{J}(\mu) = {} & \log|\Sigma_x| + \log|\Sigma_y| \\ & + \frac{1}{TN}\sum_{k=1}^{T}\sum_{i=1}^{N}\omega^i(y_k - f_\theta(\xi_k^i))'\Sigma_{y,p}^{-1}(y_k - f_\theta(\xi_k^i)) \\ & + \frac{1}{TN}\sum_{k=1}^{T}\sum_{i=1}^{N}\omega^i(\xi_k^i - g_\theta(\xi_{k-1}^m, u_k))'\Sigma_{x,p+1}^{-1}(\xi_k^m - g_\theta(\xi_{k-1}^m, u_k))\end{aligned}$$

1.5.3 Backward pass