1 Notations

We consider the prediction task of a set of observations $(y^0, \cdots y^T)$ given a set of input $(u^0, \cdots u^T)$. The model can be built on top of a arbitrary encoder model, denoted h, that will be trained by gradient descent. Through this encoder, inputs are mapped to latent vectors:

$$h : (u^0, \cdots u^T) \mapsto (\tilde{u}^0, \cdots \tilde{u}^T)$$

2 Model

Our model can be viewed as a RNN layer followed by a fully connected layer. At time step t,

$$\begin{cases} y_{t+1} = \tanh(W_y x_{t+1}^L + b_y) \\ x_{t+1} = \tanh(W_{xx} x_t + W_{xu} \tilde{u}_{t+1} + b_x) \end{cases}$$

with $x_0 \equiv 0$.

Let's consider the weights of the last RNN and fully connected layers as $\theta \equiv (W_{xx}, \cancel{W_{xu}}, b_x, W_y, b_y)$. We 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:

$$\begin{cases} y_t = f_\theta(x_t) + \epsilon_t & \text{observation model} \\ x_{t+1} = g_\theta(x_t, u_{t+1}) + \eta_{t+1} & \text{state model} \\ \tilde{u}_t = h(u_t) & \text{command model} \end{cases} \tag{1}$$

In the following section, we will focus on maximizing the joint log likelihood:

$$\log\ p_\theta(X_{0:T}, y_{0:T}, u_{0:T}) \tag{2}$$

3 Minimization

We can start by developing the log likelihood:

$$
\begin{aligned}
\log p_\theta(X_{0:T}, y_{0:T}, u_{0:T}) &= \frac{1}{T}\log\left(p_\theta(x_0)p_\theta(y_0|x_0)\prod_{k=1}^{T} p_\theta(x_k|x_{k-1},u_k)p_\theta(y_k|x_k)\right)\\
&= \frac{1}{T}\log p_\theta(x_0)\\
&+ \frac{1}{T}\sum_{k=1}^{T}\log\ p_\theta(x_k|x_{k-1},u_k) + \frac{1}{T}\sum_{k=0}^{T}\log\ p_\theta(y_k|x_k)\\
&= \frac{1}{T}\log p_\theta(x_0) - \frac{1}{2}\log|\Sigma_x| - \frac{1}{2}\log|\Sigma_y| + Cst\\
&- \frac{1}{2T}\sum_{k=1}^{T}(x_k - g_\theta(x_{k-1},u_k))'\Sigma_x^{-1}(x_k - g_\theta(x_{k-1},u_k))\\
&- \frac{1}{2T}\sum_{k=0}^{T}(y_k - f_\theta(x_k))'\Sigma_y^{-1}(y_k - f_\theta(x_k))
\end{aligned}
$$

We aim at maximizing 2 by gradient descent, by leveraging fisher's identity:

$$\nabla \log p_\theta(x_{0:T}, y_{0:T}, u_{0:T}) = \mathbb{E}_\theta\left[\nabla \log p_\theta(x_{0:T}, y_{0:T}, u_{0:T})|Y_{0:T}\right]$$

In order to approximate this expectation, we need to sample from the posterior distribution $p_\theta(x|y)$. In Section 4, we detail a Sequential Monte Carlo approach to this end. In Section 5, we describe the algorithm to train our model through gradient descent.

4 Sequential Monte Carlo Approach

4.1 Filter

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

At time step $k = 0$, $(\xi^l_0)^N_{l=1}$ are sampled independently from the first hidden state, and associated with sampling weights proportional to the observation density q_θ:

$$
\begin{aligned}
\xi^i_0 &\sim \mathcal{N}(x_0, \Sigma_x)\\
\omega^i_0 &\sim q_\theta(\xi^i_0)
\end{aligned}
$$

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:

$$\mathbb{P}(I^i_{k+1} = j) = \omega^j_k \quad \forall 1 \leq j \leq N$$

$$\omega^i_{k+1} \sim q_\theta(\xi^i_{k+1})$$

4.2 Smoother

Using the poor man filter, we get N trajectories:

$$\xi^i_{1:k+1} = (\xi^{I^i_{k+1}}_{1:k}, \xi^i_{k+1})$$

4.3 Approximation

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

$$\mathbb{E}_\theta\left[h(x)|y_{1:T}\right] = \sum_{i=1}^{N} \omega^i_T h(\xi^i_{0:T})$$

5 Gradient descent

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, Σ_x and Σ_y. We initialize the sequence with a initial hidden state sampled from the noise's distribution.

$$x^i_0 \sim \mathcal{N}(0, \Sigma_x)$$

In order to predict each new time step $k+1$, particles from the previous step are attributed weights ω^i_k proportionally to the density probability around the targeted value y_k.

$$\omega^i_k \sim \exp(-\frac{1}{2}(y_k - f_{\theta_p}(x^i_k))'\Sigma^{-1}_{y,p}(y_k - f_{\theta_p}(x^i_k)))$$

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

$$\mathbb{P}(I^i_{k+1} = j) = \omega^j_k \quad \forall 1 \leq j \leq N$$

The current hidden state is computed for the selected particles.

$$x^i_{k+1} = g_\theta(x^{I^i_{k+1}}_k, u_{k+1}) + \eta^i_{k+1}$$

5.2 Loss function

Considering that we have computed a set of N trajectories $(\xi^i_{0:T})$, $1 \leq i \leq N$, associated with weights (ω^i), we can approximate the gradient of the log likelihood by computing the gradient of:

$$\begin{aligned}\mathbb{J}(\theta) &= \log|\Sigma_x| + \log|\Sigma_y| \\ &+ \frac{1}{T}\sum_{k=1}^{T}\sum_{i=1}^{N}\omega^i(y_k - f_\theta(\xi^i_k))'\Sigma_y^{-1}(y_k - f_\theta(\xi^i_k)) \\ &+ \frac{1}{T}\sum_{k=0}^{T}\sum_{i=1}^{N}\omega^i(\xi^i_k - g_\theta(\xi^i_{k-1}, u_k))'\Sigma_x^{-1}(\xi^i_k - g_\theta(\xi^i_{k-1}, u_k))\end{aligned}$$

5.3 Backward pass

During this step, each parameter of the model is updated by gradient descent.

6 Two-step training

In this section, we aim at improving the weights of a model already trained in a traditional fashion.

6.1 Initial training

Given a dataset of samples $(u_{0:T}^{(i)}, y_{0:T}^{(i)})_{i=1}^{m}$, we consider a training of our model as defined in 1 without the added noise. For each sample, we simply compute a prediction $\hat{y}_{0:T}$ by running the input through each layer, and minimize the discrepancy to the target values by gradient descent.

$$\mathbb{J} = \|u_{0:T} - y_{0:T}\|^2$$

This training results in an initial set of weights θ_0.

6.2 Finetuning

Because we trust the initial training has successfully extracted relevant information from the command u, and reached satisfactory parameters for the hidden state model, we will only adapt the parameters of the observation model:

$$\theta_{finetune} \equiv (\cancel{W_{xx}}, \cancel{b_x}, W_y, b_y)$$

Σ_x is set to a small value and will not be updated during the finetuning.