UNROLLING PARTICLES: UNSUPERVISED LEARNING OF SAMPLING DISTRIBUTIONS

Fernando Gama, Nicolas Zilberstein, Richard G. Baraniuk and Santiago Segarra
Department of Electrical and Computer Engineering, Rice University, Houston, TX

Nonlinear Dynamical Systems

- The time evolution of some phenomena under study
  - ⇒ Depends nonlinearly on previous states

Tracking Robotics Communications

- Estimate some unknown quantity dependent on the state of the system
  - ⇒ No access to the state, we have access only to observations

Particle Filtering

- We know the distributions ⇒ System transition, measurements
  - ⇒ Bayesian framework ⇒ Estimate of the state given the observations
- Computing the Bayesian estimate can be computationally intractable
- Particle filtering ⇒ Efficient sampling from a designed distribution
  - ⇒ Average samples to construct an estimator that is good enough

Unrolling Particles

- Designing an efficient sampling distribution that leads to good
  - ⇒ It is difficult ⇒ Balance the model with sampling efficiency
  - ⇒ Avoid weight degeneracy in the resulting sampled particles

[Doucet et al, 2000; Djurić et al, 2003; Godsill, 2019; ur Rehman et al, 2018; Ryu and Boyd, 2015; Elvira and Martino, 2021]

Objective

Learn an efficient sampling distribution based only on observations

- Leverage algorithm unrolling to learn a parametric distribution
- Neural networks to learn the mean and variance of a multivariate normal
- Train the neural networks using an unsupervised learning approach
  - ⇒ Minimize weight degeneracy

Nonlinear Dynamical Systems

- Let $\{\mathbf{x}_t\}_{t\geq 0}$ be a sequence of states $\mathbf{x}_t \in \mathbb{R}^N$ ⇒ Unobservable
- Let $\{\mathbf{y}_t\}_{t\geq 0}$ be a sequence of measurements $\mathbf{y}_t \in \mathbb{R}^M$ ⇒ Observable
- The nonlinear dynamic system is completely characterized by (which are considered known)

Initial state: $\mathbf{x}_0 \sim \mathrm{p}(\mathbf{x}_0)$, Transition: $\mathrm{p}(\mathbf{x}_t|\mathbf{x}_{t-1})$, Measurement: $\mathrm{p}(\mathbf{y}_t|\mathbf{x}_t)$

$$\mathbf{x}_{t-1} \longrightarrow \boxed{\mathrm{p}(\mathbf{x}_t|\mathbf{x}_{t-1})} \xrightarrow{\mathbf{x}_t} \boxed{\mathrm{p}(\mathbf{y}_t|\mathbf{x}_t)} \longrightarrow \mathbf{y}_t$$

References

- A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Stat. Comput.*, vol. 10, no. 3, pp. 197?208, July 2000.
- V. Elvira, L. Martino, M. F. Bugallo, and P. Djurić, "Eluci- dating the auxiliary particle filter via multiple importance sam- pling," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 145?152, 30 Oct. 2019, lecture notes.
- V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Inter- pretable, efficient deep learning for signal and image process- ing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18?44, March 2021.

Particle Filtering

- The objective is to estimate some function f_t of the states
  - ⇒ We only access observations
- Computing the posterior $\mathrm{p}(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ is typically intractable
  - ⇒ Use particle filtering
  - ⇒ Estimate the function by sampling $\{\mathbf{x}_{0:t}^{(k)}\}_{k=1}^K \sim \pi(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$

$$\hat{\mathrm{f}}_t = \sum_{k=1}^{K} w_t^{(k)} \mathrm{f}_t(\mathbf{x}_{0:t}^{(k)})$$

- Design $\pi(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ ⇒ Good estimates, easy to sample
  - ⇒ Assume sequential sampling

$$\pi(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \pi(\mathbf{x}_{0:t-1}|\mathbf{y}_{0:t-1})\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t})$$

- Now the weights can be updated sequentially ⇒ All known quantities
  - ⇒ Normalize for computing $\hat{\mathrm{f}}_t$

$$\tilde{w}_t^{(k)} = \tilde{w}_{t-1}^{(k)} \frac{\mathrm{p}(\mathbf{y}_t|\mathbf{x}_t^{(k)})\mathrm{p}(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)})}{\pi(\mathbf{x}_t^{(k)}|\mathbf{x}_{0:t-1}^{(k)}, \mathbf{y}_t)} \quad , \quad w_t^{(k)} = \frac{\tilde{w}_t^{(k)}}{\sum_{k=0}^{K} \tilde{w}_t^{(k)}}$$

- Sequential sampling distributions ⇒ particle degeneracy
  - ⇒ Most weights $w_t^{(k)} \to 0$ except for one
  - ⇒ Minimize it with $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(k)}, \mathbf{y}_{0:t}) = \mathrm{p}(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)}, \mathbf{y}_t)$
- Cannot be avoided ⇒ Resampling
  - ⇒ Estimate effective sample size, keep the largest ones

Learning the Sampling Distribution

- A good sampling distribution may depend only on $\mathbf{x}_{t-1}$ and $\mathbf{y}_t$

$$\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t}) = \pi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$$

- We propose a multivariate normal distribution ⇒ Easy to sample

$$\pi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = \mathcal{N}\Big(\mu_t(\mathbf{x}_{t-1}, \mathbf{y}_t), \Sigma_t(\mathbf{x}_{t-1}, \mathbf{y}_t)\Big)$$

- We use algorithm unrolling for learning the mean and the variance

Learning the Mean

- Algorithm unrolling ⇒ We use a neural network for every time iteration

$$\mathrm{NN}_t^{\mu}(\mathbf{x}_{t-1}, \mathbf{y}_t) = \mathbf{z}_t^{(L_t)} \quad \text{where} \quad \mathbf{z}_t^{(\ell)} = \rho_t\big(\mathbf{A}_t^{(\ell)}\mathbf{z}_t^{(\ell-1)} + \mathbf{b}_t^{(\ell)}\big)$$

  - ⇒ For every t there is a different neural network
  - ⇒ L_t layers and nonlinearity ρ_t
  - ⇒ Each layer is determined by N_t^{ℓ} ⇒ $\mathbf{A}_t^{(\ell)}$ of size $N_t^{(\ell)} \times N_t^{(\ell-1)}$
- The input is given by the concatenation of $\mathbf{x}_{t-1}$ and $\mathbf{y}_t$

$$\mathbf{z}_t^{(0)} = [\mathbf{x}_{t-1}^{\mathsf{T}}, \mathbf{y}_t^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{N+M}$$

- The learned mean is collected at the end of the last layer
  - ⇒ It has size $N_t^{(L_t)} = N$ ⇒ The same size as $\mathbf{x}_t$
- The values of L_t, ρ_t and $N_t^{(\ell)}$ are all design choices ⇒ Hyperparameters

Learning the Covariance

- We learn the covariance matrix with a time-invariant framework

$$\Sigma_t(\mathbf{x}_{t-1}, \mathbf{y}_t) = \Sigma(\mathbf{x}_{t-1}, \mathbf{y}_t) = \mathbf{C}\mathbf{D}(\mathbf{x}_{t-1}, \mathbf{y}_t)\mathbf{C}^{\mathsf{T}}$$

- Here, the $N \times N$ matrix $\mathrm{D}(\mathbf{x}_{t-1}, \mathbf{y}_t)$ represents the gaussian kernel

$$\big[\mathbf{D}(\mathbf{x}_{t-1}, \mathbf{y}_t)\big]_{ij} = \exp\big(-([\mathbf{z}_t]_i - [\mathbf{z}_t]_j)^2\big)$$

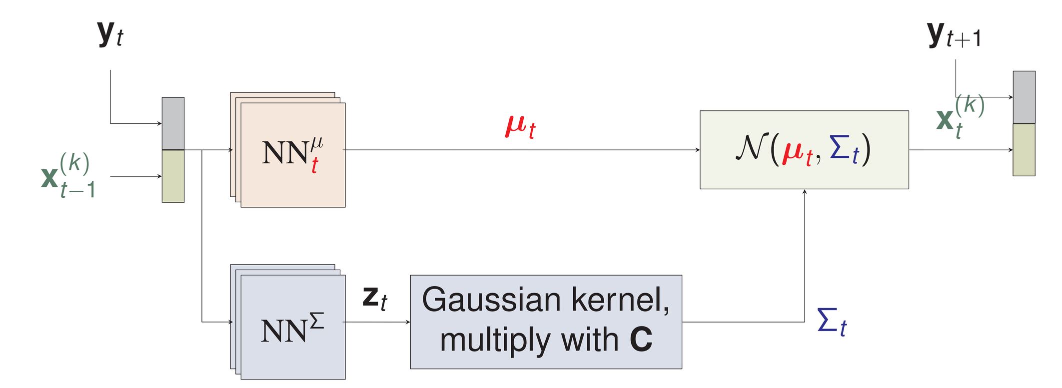
  - ⇒ It is applied on the output of a neural network
  - ⇒ We learn an appropriate representation

$$\mathbf{z}_t = \mathrm{NN}^{\Sigma}(\mathbf{x}_{t-1}, \mathbf{y}_t)$$

- The $N \times N$ matrix $\mathbf{C}$ is also learned
  - ⇒ Learn different covariance directions

Schematic for Learning the Sampling Distribution

- Sampling distribution for each time t and for each sample k
  - ⇒ The neural network for the mean NN_t^{μ} has time-varying parameters
  - ⇒ The NN for the covariance NN^{Σ} has the same parameters for all t
  - ⇒ The output μ_t and Σ_t changes with time because the input $(\mathbf{y}_t, \mathbf{x}_{t-1}^{(k)})$ changes with time



Unsupervised Learning

- Unsupervised learning
  - ⇒ We only have access to the sequence of observations $\{\mathbf{y}_t\}$
  - ⇒ Does not require access to the true trajectories $\{\mathbf{x}_t\}$ of the system
  - ⇒ Allows the sampling distribution to generalize to unseen trajectories
- Learn a distribution such that the weights are similar to each other

$$\mathrm{J}\big(\{w_t^{(k)}\}_{t,k}\big) = \sum_{t=0}^{T-1}\sum_{k=1}^{K} \log\big(w_t^{(k)}\big) \quad \text{with} \quad \tilde{w}_t^{(k)} = \tilde{w}_{t-1}^{(k)} \frac{\mathrm{p}(\mathbf{y}_t|\mathbf{x}_t^{(k)})\mathrm{p}(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)})}{\pi(\mathbf{x}_t^{(k)}|\mathbf{x}_{0:t-1}^{(k)}, \mathbf{y}_t)}$$

  - ⇒ The function J is maximized when all the weights are equal to $1/K$
  - ⇒ When weights $w_t^{(k)}$ get too small, they are penalized by the logarithm
  - ⇒ Reduced benefit for increasing a weight that is already large
- Maximizing J can be done via a stochastic gradient ascent algorithm
  - ⇒ Gradients are propagated via the reparametrization trick

Numerical Experiments

- Given a sequence of measurements $\{\mathbf{y}_t\}$ from some dynamical system
  - ⇒ Estimate $\mathbb{E}[\mathbf{x}_t|\mathbf{y}_{0:t}]$
- Three experimental scenarios to illustrate three different aspects
  - ▷ A linear system with Gaussian noise
    - ⇒ Posterior is known and estimator obtained in closed form
  - ▷ A nonlinear system with Gaussian noise
    - ⇒ Minimum degeneracy distribution can be obtained
  - ▷ A linear system with non-Gaussian noise
    - ⇒ Neither the posterior nor the minimum degeneracy
- The learned distribution is compared with the minimum degeneracy one
  - ⇒ Try with and without resampling (only at test time, not at training time)

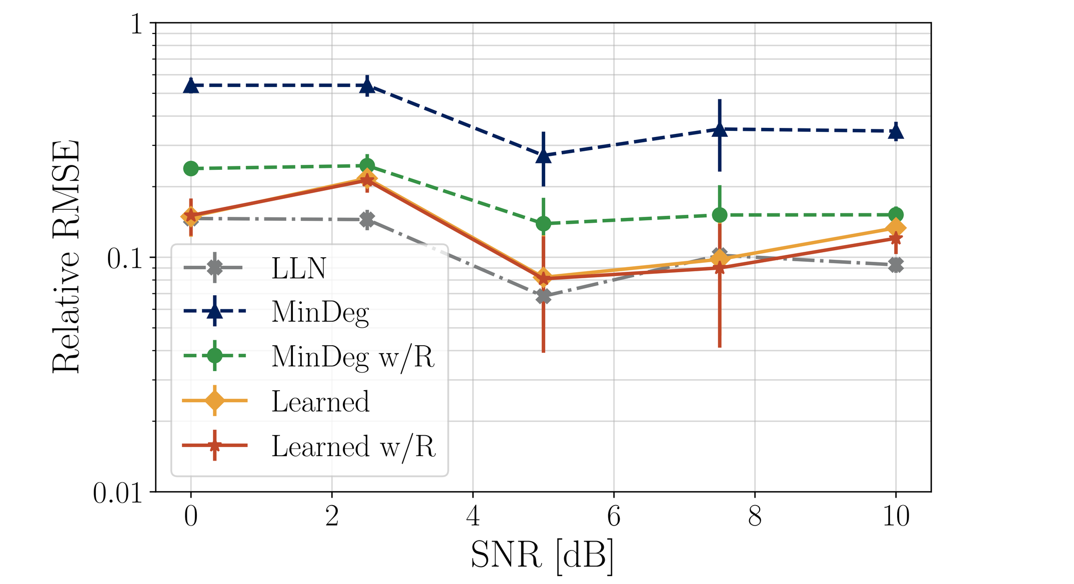
Numerical Experiments: Setting

- Set $L_t = 2$ layers and nonlinearity $\rho_t = \tanh$
- Set $N_t^{(1)} = 256$ and $N_t^{(2)} = 512$ for all t
- Train by running the particle filtering, computing the loss
  - ⇒ Using ADAM with learning rate 0.001
- The number of particles simulated on each run is $K = 25$
- The particle filter is run 200 times, updating the parameters each time
- Test by running the particle filters 100 times, drawing $K = 25$ particles
- Compute the relative RMSE between the estimate and the target value
- Set $K_{\text{thres}} = K/3$ as the threshold for resampling during test time
- Run the entire training and testing for 10 times,
  - ⇒ Report median and standard deviation

Linear System with Gaussian Noise

- $N = 10$ size of the state, $M = 8$ number of measurements
- **A** adjacency matrix, SNR $= \|\mu^0\|_2^2/\|\Sigma_v\|_2^2$

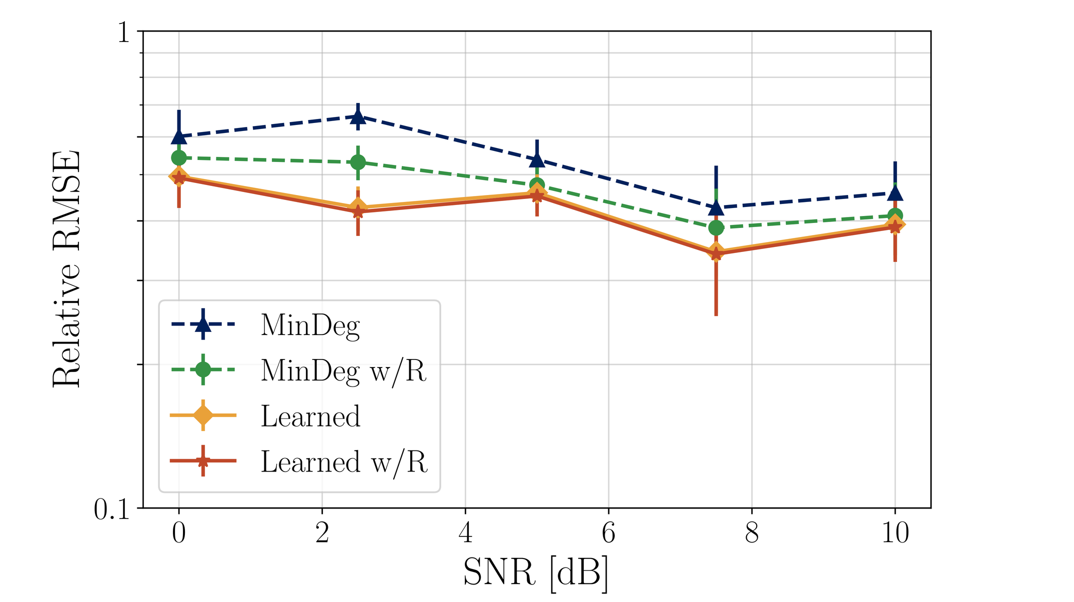
$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{v}_t \quad , \quad \mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{w}_t,$$
$$\mathbf{x}_0 \sim \mathcal{N}(\mu^0, \Sigma^0)\ ,\ \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_v)\ ,\ \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_w),$$



Nonlinear System with Gaussian Noise

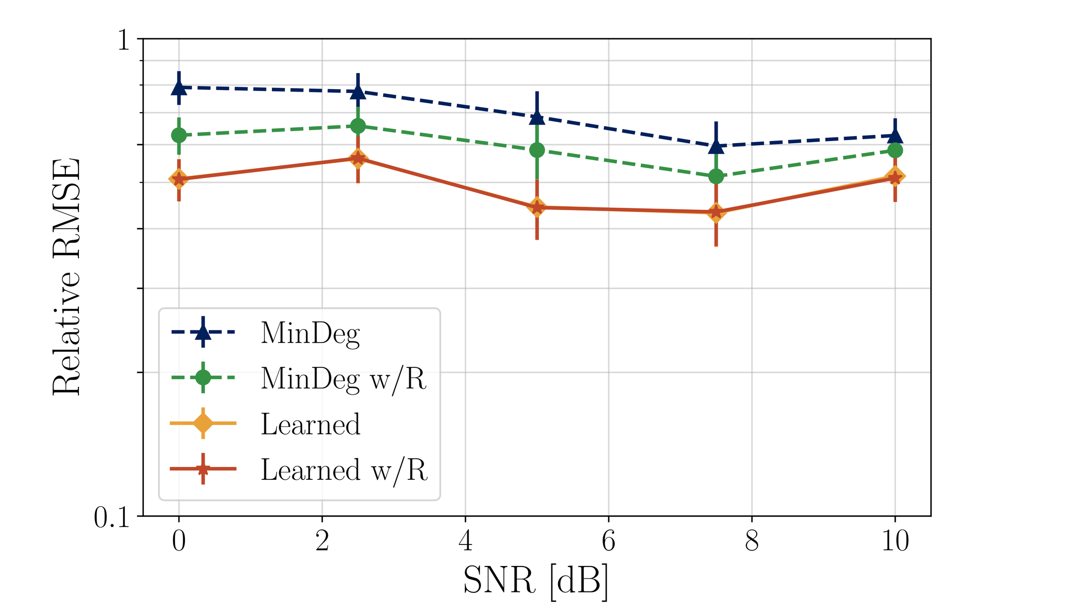
- $N = 10$ size of the state, $M = 8$ number of measurements
- ϕ absolute value, SNR $= \|\mu^0\|_2^2/\|\Sigma_v\|_2^2$

$$\mathbf{x}_t = \phi(\mathbf{A}\mathbf{x}_{t-1}) + \mathbf{v}_t, \quad , \quad \mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{w}_t,$$
$$\mathbf{x}_0 \sim \mathcal{N}(\mu^0, \Sigma^0)\ ,\ \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_v)\ ,\ \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_w),$$



Linear System with Uniform Noise

- Linear system with linear measurements
  - ⇒ The initial state and the noise are uniform
  - ⇒ The noise samples are i.i.d. with covariance matrix given by $\Sigma = \sigma^2\mathbf{I}$



Conclusions

- Learning sampling distributions for particle filters
  - ⇒ Algorithm unrolling for learning a multivariate normal
- Train in unsupervised learning framework
  - ⇒ Requires access only to the sequence of measurements
  - ⇒ Does not require access to the true trajectories of the system
  - ⇒ Allows the sampling distribution to generalize to unseen trajectories
- Train to minimize degeneracy by maximizing a logarithm of the weights
- Numerical experiments showcase improved performance