Sequential Monte-Carlo Filtering/Smoothing for State and Parameter Estimation in General Nonlinear State Space Systems

Ali Gorji

May 30, 2018

Agenda

- Problem statement: **Nonlinear State Space Systems (NSSS)**
- State and parameter estimation in NSSS
- Expectation Maximization (EM) for joint state and parameter estimation in NSSS
- Handling non-linearity: Sequential Monte Carlo (SMC)
- Particle Filter/Smoother for joint state and parameter estimation

Nonlinear State Space Systems

- We consider a family of discrete-time and affine NSSSs:

$$\begin{aligned} \mathbf{x}_k &= f_{\theta^x}\left(\mathbf{x}_{k-1}, \mathbf{u}_k\right) + \mathbf{v}_k \\ \mathbf{y}_k &= g_{\theta^y}\left(\mathbf{x}_k\right) + \mathbf{w}_k \end{aligned}$$

- Where:
  - $\mathbf{x}_k$: hidden state at the k-th step-time
  - $\mathbf{y}_k$: output observation at the k-th step-time
  - f, g: parametric state transition and output models with parameters θ^x and θ^y, respectively
  - $\mathbf{u}_k$: external input
  - $\mathbf{v}_k, \mathbf{w}_k$: additive noise

NSSS in Time-series Modeling with Missing Data

- Considering $\mathbf{y}_k$ as a vector of output observations at the k-th step:
  - Time-series may suffer from missing-data where some of outputs are not measurable
  - Weather-data: daily max/min temperature is available, but the associated time-of-year may be missing
- **NSSS**: model time-series as a state-space system with missing-data being represented by hidden-states
- Training the model, **infer** missing-data from the received observations

NSSS in Model Reference Control

- The goal in Model Reference Control (MRC):

$$\mathbf{u}_{1:K} = argmin_{\mathbf{u}}\Big\{ \sum_k (y_k - \hat{y}_k)^2 \Big\}$$

  with $\hat{y}_k$ being a desired trajectory
- Consider a parametric structure for the controller: $\mathbf{u}_k = h_{\theta^u}(\mathbf{x}_{k-1})$ and re-construct the NSSS:

$$\mathbf{x}_k = f_{\theta^x}\left(\mathbf{x}_{k-1}, h_{\theta^u}(\mathbf{x}_{k-1})\right) + \mathbf{v}_k$$

- Jointly estimate states and parameters to minimize MRC objective
- Generate control command using the learned h_{θ^u}

NSSS in Observer Trajectory Planning

- Consider a moving object with motion state $\mathbf{x}_k$, and a moving observer with state $\mathbf{x}_k^o$ that gathers measurements y_k
- NSSS in dynamical target tracking:

$$\begin{aligned} \mathbf{x}_k &= A\mathbf{x}_{k-1} + B\mathbf{v}_k \\ y_k &= g_{\mathbf{x}_k^o}(\mathbf{x}_k) + \mathbf{w}_k \end{aligned}$$

  with A, B being known matrices of state-transition model, and g as a nonlinear measurement model such as bearing
- The goal is to estimate observer locations to minimize variance of target state estimation

State and Parameter Estimation in NSSS

- Three main sources in NSSS:
  - Observations: $\{\mathbf{y}_{0:k}, \mathbf{u}_{0:k}\}$
  - Parameters: $\{\theta^x, \theta^y\}$
  - Hidden states: $\mathbf{x}_{0:k}$
- State estimation: estimate posterior density of hidden states given observations:
  - Filtering: $p(\mathbf{x}_{0:k}|\mathbf{y}_{0:k}, \mathbf{u}_{0:k})$
  - Smoothing: $p(\mathbf{x}_{0:k'}|\mathbf{y}_{0:k}, \mathbf{u}_{0:k})$, $k' < k$
- Parameter estimation: obtain an estimate of model parameters by optimizing a proper objective such as likelihood

$$\theta^* = argmax_\theta \left\{ p(\mathbf{y}_{0:k}; \theta) \right\}$$

Maximum Likelihood Estimator

- Parameters can be obtained by maximizing the log-likelihood function

$$\theta^* = argmax_\theta \left(J = \sum_k \log \left\{ p(\mathbf{y}_k; \theta) \right\} \right)$$

- Incomplete likelihood: the likelihood cannot be calculated since hidden states are missing
- Main trick: given the concavity of log-function and using Jensen inequality:

$$\log \{ \underbrace{p(\mathbf{y}_k; \theta)}_{E_q\left(\frac{p(\mathbf{y},\mathbf{x};\theta)}{q(\mathbf{x})}\right)} \} \geq E_{q(\mathbf{x})} \left(\underbrace{p(\mathbf{y}_k | \mathbf{x}_k; \theta)}_{\text{Complete Likelihood}} \right) - \mathcal{H}(\mathbf{x}_k)$$

with $q(.)$ being a distribution over states and $\mathcal{H}(.)$ as the entropy

Expectation Maximization (EM) Algorithm

- Instead of maximizing the likelihood, we aim to maximize the lower-bound
- EM algorithm consists of two main phases:
  - E-phase: assuming parameters of the model are fixed, minimize the bound-gap through minimizing the following kullback leibler divergence:

$$D_{KL}(q||p) = -E_q\left(\log\left(\frac{p(\mathbf{x}_k|\mathbf{y}_k;\theta)}{q(\mathbf{x}_k)}\right)\right)$$

  - M-phase: given estimates of hidden-states, maximize the expectation over the complete likelihood:

$$\theta^* = argmax_\theta E_q\left(\log\left(p(\mathbf{y}_k|\mathbf{x}_k;\theta)\right)\right)$$

Iterative State/Parameter Estimation Using EM

- Initialize model parameters, training iteration $n = 0$ and model parameters $\theta[0]$
- Do until convergence:
  - E-phase: by estimating $q(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{y}_k; \theta[n]), k \in \{0, \ldots, K\}$
  - M-phase: update parameters by maximizing $\sum_{k=0}^{K} E_q \left(\log \left(p(\mathbf{y}_k|\mathbf{x}_k; \theta)\right)\right)$ with q being used from the E-phase
  - Calculate log-likelihood using estimated states and new parameter values
  - If no significant change in log-likelihood **break**, otherwise set $n \leftarrow n + 1$
- Theorem: Under **exact state estimation**, EM guarantees log-likelihood increases at each step: $\mathcal{L}(\theta[n+1]) \geq \mathcal{L}(\theta[n])$ with $\mathcal{L}(\theta)$ being the complete likelihood

Nonlinearity and State/parameter Estimation

- Under additive Gaussian noises and linear state-transition and output models:
  - Optimal filtering such as **Kalman filter** and Gaussian distributed posterior $p(\mathbf{x}_k|\mathbf{y}_k;\theta) = \mathcal{N}\left(\mathbf{x}_{k|k}, P_{k|k}\right)$ with $\mathbf{x}_{k|k}$ and $P_{k|k}$ being mean and covariance of estimates, respectively
  - Obtain a **closed-form** for $\sum_{k=1}^{K} E_q\left(\log\left(p(\mathbf{y}_k|\mathbf{x}_k;\theta)\right)\right)$ and find the optimal parameter value
- Under general nonlinear state-transition and/or ourput models:
  - Local linearization and Extended Kalman Filter for approximate E-phase and Gradient updates for M-phase
  - Approximate the posterior distribution of states using samples and find an unbiased estimate of the log-likelihood that could be maximized against parameters of the model

Approximate State Estimation Using Sequential Monte Carlo (SMC)

- Main idea is to represent the posterior distribution as a weighted summation of samples:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \sim \sum_{m=1}^{M} \tilde{w}_k^m \delta(\mathbf{x}_k - \mathbf{x}_k^m)$$

with $\tilde{w}_k^m$ being normalized importance weights, $\delta(.)$ as the delta-Dirac function and M as the total number of particles

- Approximate M-phase with SMC sampler:

$$\theta^* = argmax_\theta \sum_{k=1}^{K} \sum_{m=1}^{M} \tilde{w}_k^m \log\left(\mathbf{y}_k|\mathbf{x}_k^m;\theta\right)$$

Sequential Importance Sampling

- Draw samples from a proposal distribution $s(\mathbf{x}_{0:k}|\mathbf{y}_{0:k})$ and update importance weights recursively:

$$w_k^m = w_{k-1}^m \frac{p(\mathbf{x}_k^m|\mathbf{x}_{k-1}^m)p(\mathbf{y}_k|\mathbf{x}_k^m)}{s(\mathbf{x}_k^m|\mathbf{x}_{k-1}^m, \mathbf{y}_k)}$$

  and normalize the weights $\tilde{w}_k^m = \frac{w_k^m}{\sum_{m=1}^M w_k^m}$
- Choice of proposal density function:
  - State-transition: $s(\mathbf{x}_k^m|\mathbf{x}_{k-1}^m, \mathbf{y}_k) = p(\mathbf{x}_k^m|\mathbf{x}_{k-1}^m)$
  - Using the outputs and sample from $p(\mathbf{x}_k^m|\mathbf{x}_{k-1}^m)p(\mathbf{y}_k|\mathbf{x}_k^m)$
  - Use sub-optimal Extended Kalman Filter (EKF) to obtain an estimate of the posterior $p(\mathbf{x}_k|\mathbf{y}_k)$ and assign the distribution to the proposal

Sequential Importance Re-sampling

- Importance sampling provides estimates whose variance increases exponentially over time
  - Only a few particles survive with a high weight with other particles will be assigned to a zero-weight
- We define re-sampling as an operation that maps the set $\{\tilde{w}_k^m, \mathbf{x}_k^m\}$ to a set of equally-weighted states $\{\frac{1}{M}, \bar{\mathbf{x}}_k^m\}$
- **Idea**: samples with higher weights will be replicated, but lower-weighted samples also survive
- Different re-sampling strategies:
  - Residual re-sampling, systematic re-sampling and multinomial re-sampling
  - Adaptive re-sampling: re-sample only if the effective sample-size of weights is below a threshold:

$$\left(\sum_{m=1}^{M} (\tilde{w}_k^m)^2\right)^{-1} \leq \tau$$

with $\tau = \frac{M}{2}$

General SMC Sampler for Filtering

- Sample initial particles $\mathbf{x}_0^m$ from an arbitrary distribution
- Compute initial normalized weights $\tilde{w}_0^m$
- Check the re-sampling criterion and if needed, run re-sampling and generate $\{\frac{1}{N}, \bar{\mathbf{x}}_0^m\}$
- For $k > 0$:
  - Draw new samples from the proposal distribution $s(\mathbf{x}_k^m | \bar{\mathbf{x}}_{k-1}^m, \mathbf{y}_k)$
  - Compute the normalized weights $\tilde{w}_k^m$
  - Check for the re-sampling criterion:
    - If effective sample-size is lower than the threshold, run re-sampling and $\{\tilde{w}_k^m, \mathbf{x}_k^m\} \rightarrow \{\frac{1}{M}, \bar{\mathbf{x}}_k^m\}$
    - If effective sample-size is above the threshold, use $\{\tilde{w}_k^m, \mathbf{x}_k^m\}$ as the set of particles and associated weights

Summary: EM with SMC

- **Initialization**: randomly initialize parameters $\theta[0]$ and M particles with associated weights $\{\frac{1}{M}, \mathbf{x}_0^m\}$
- Do until convergence:
  - E-phase: run one step of SMC sampler and obtain sampled states and associated weights $\{\tilde{w}_k^m, \bar{\mathbf{x}}_k^m\}, m \in \{1, \ldots, M\}, k \in \{0, \ldots, K\}$
  - M-phase: approximate the complete log-likelihood and obtain new parameters through $\theta[n+1] = argmax_\theta \sum_{k=1}^{K} \sum_{m=1}^{M} \tilde{w}_k^m \log(\mathbf{y}_k | \bar{\mathbf{x}}_k^m; \theta)$
  - Stop if the likelihood does not change significantly $|\mathcal{L}(\theta[n+1]) - \mathcal{L}(\theta[n])| \leq \epsilon$ or $n \geq n_{max}$, otherwise $n \leftarrow n+1$

State Estimation: Smoothing

- Given a batch of observations ($\mathbf{y}_{0:k}$), estimate $p(\mathbf{x}_k|\mathbf{y}_{0:K})$ instead of filtering distribution:
  - Makes state estimation more robust to the process or observation noise
  - Allows the training procedure to have a better estimate of model parameters
- Different perspectives:
  - Forward-backward: Use estimated states at the filtering stage (forward path) to approximate smoothing distribution
  - Two-filter smoothing: Break down the smoothing distribution into two separate filtering stages and estimate each stage through a single filter
  - Maximum a Posteriori smoothing: Run the forward path and estimate states. Then, obtain smoothed estimates by maximizing the posterior distribution of hidden states over measurements (through Dynamic Programming)

Forward-backward Particle Smoothing

- Modeling the smoothing distribution as $p(\mathbf{x}_k|\mathbf{y}_{0:K}) \sim \sum_{m=1}^{M} \tilde{w}_{k|K}^m \delta(\mathbf{x} - \mathbf{x}_k^m)$
- Updating weights using the following forward-backward rule:

$$\tilde{w}_{k|K}^m = \tilde{w}_k^m \left\{ \sum_{m=1}^{M} \tilde{w}_{k+1|K}^m \frac{p(\mathbf{x}_{k+1}^m|\mathbf{x}_k^m)}{\sum_{m'=1}^{M} \tilde{w}_k^{m'} p(\mathbf{x}_{k+1}^{m'}|\mathbf{x}_k^{m'})} \right\}$$

- Some considerations: the computational complexity is of order M^3
- There are approximations to reduce to $M^2 \log(M)$ by partitioning particles
- EM Particle Smoothing: calculate the complete likelihood using samples generated by smoothing distribution

Summary

- Family of discrete-time nonlinear state space systems was discussed
- An NSSS is fully characterized through hidden states and its model parameters
- EM algorithm is used to estimate hidden states and learn parameters of the model
- SMC sampler (particle filtering) aims to provide an approximate low-variance estimate of the posterior distribution of states
- EM with particle filtering/smoothing forms a universal framework for jointly estimating hidden states and learning parameters of a general nonlinear state space system

References

- A. Doucet, and A. M. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later", in Handbook of Nonlinear Filtering (eds. D. Crisan et B. Rozovsky), Oxford University Press, 2011.
- M. Klaas, et al. "Fast particle smoothing: If I had a million particles"", Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006.
- S. Roweis, and Z. Ghahramani, "An EM algorithm for identification of nonlinear dynamical systms", Proceedings of the 1998 conference on Advances in neural information processing systems.

Questions