Lecture 10: Bayesian Estimation of Parameters in State Space Models

Simo Särkkä

March 20, 2019

Learning Outcomes

1. Summary of the Last Lecture
2. Bayesian estimation of parameters in state space models
3. Computational methods for parameter estimation
4. Practical parameter estimation in state space models
5. Summary

Summary of the Last Lecture

- Extended, statistically linearized and unscented RTS smoothers are the approximate nonlinear smoothers corresponding to EKF, SLF and UKF.
- Gaussian RTS smoothers: cubature RTS smoother, Gauss-Hermite RTS smoothers and various others
- Particle smoothing can be done by storing the whole state histories in SIR algorithm.
- Rao-Blackwellized particle smoother is a combination of particle smoothing and RTS smoothing.

Batch Bayesian estimation of parameters

- State space model with unknown parameters $\theta \in \mathbb{R}^d$:

$$\begin{aligned}
\theta &\sim p(\theta) \\
\mathbf{x}_0 &\sim p(\mathbf{x}_0 \mid \theta) \\
\mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta) \\
\mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k, \theta).
\end{aligned}$$

- The full posterior, in principle, can be computed as

$$p(\mathbf{x}_{0:T}, \theta \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}, \theta)\, p(\mathbf{x}_{0:T} \mid \theta)\, p(\theta)}{p(\mathbf{y}_{1:T})}.$$

- The marginal posterior of parameters is then

$$p(\theta \mid \mathbf{y}_{1:T}) = \int p(\mathbf{x}_{0:T}, \theta \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}_{0:T}.$$

Batch Bayesian estimation of parameters (cont.)

- *Advantages:*
  - A simple static Bayesian model.
  - We can take any numerical method (e.g., MCMC) to attack the model.
- *Disadvantages:*
  - We are not utilizing the Markov structure of the model.
  - Dimensionality is huge, computationally very challenging.
  - Hard to utilize the already developed approximations for filters and smoothers.
  - Requires computation of high-dimensional integral over the state trajectories.
- For computational reasons, we will select another, filtering and smoothing based route.

Filtering-based Bayesian estimation of parameters [1/3]

- Directly approximate the marginal posterior distribution:

$$p(\theta \mid \mathbf{y}_{1:T}) \propto p(\mathbf{y}_{1:T} \mid \theta)\, p(\theta)$$

- The key is the prediction error decomposition:

$$p(\mathbf{y}_{1:T} \mid \theta) = \prod_{k=1}^{T} p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta)$$

- Luckily, the Bayesian filtering equations allow us to compute $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta)$ efficiently.

Filtering-based Bayesian estimation of parameters [2/3]

- Recall that the prediction step of the Bayesian filtering equations computes

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta)$$

- Using the conditional independence of measurements we get:

$$\begin{aligned} p(\mathbf{y}_k, \mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta) &= p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1}, \theta)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta) \\ &= p(\mathbf{y}_k \mid \mathbf{x}_k, \theta)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta). \end{aligned}$$

- Integration over $\mathbf{x}_k$ thus gives

$$p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta) = \int p(\mathbf{y}_k \mid \mathbf{x}_k, \theta)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta)\, \mathrm{d}\mathbf{x}_k$$

Filtering-based Bayesian estimation of parameters [3/3]

Recursion for marginal likelihood of parameters

The marginal likelihood of parameters is given by

$$p(\mathbf{y}_{1:T} \mid \theta) = \prod_{k=1}^{T} p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta)$$

where the terms can be solved via the recursion

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta)\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}, \theta)\, \mathrm{d}\mathbf{x}_{k-1}$$

$$p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta) = \int p(\mathbf{y}_k \mid \mathbf{x}_k, \theta)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta)\, \mathrm{d}\mathbf{x}_k$$

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}, \theta) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k, \theta)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta)}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta)}.$$

Energy function

- Once we have the likelihood $p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})$ we can compute the posterior via

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{\int p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}}$$

- The normalization constant in the denominator is irrelevant and it is often more convenient to work with

$$\tilde{p}(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

- For numerical reasons it is better to work with the logarithm of the above unnormalized distribution.
- The negative logarithm is the energy function:

$$\varphi_T(\boldsymbol{\theta}) = -\log p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}).$$

Energy function (cont.)

- The posterior distribution can be recovered via

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \propto \exp(-\varphi_T(\boldsymbol{\theta})).$$

- $\varphi_T(\boldsymbol{\theta})$ is called energy function, because in physics, the above corresponds to the probability density of a system with energy $\varphi_T(\boldsymbol{\theta})$.
- The energy function can be evaluated recursively as follows:
  - Start from $\varphi_0(\theta) = -\log p(\boldsymbol{\theta})$.
  - At each step $k = 1, 2, \ldots, T$ compute the following:

$$\varphi_k(\boldsymbol{\theta}) = \varphi_{k-1}(\boldsymbol{\theta}) - \log p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$$

- For linear models, we can evaluate the energy function exactly with help of Kalman filter.
- In non-linear models we can use Gaussian filters or particle filters for approximating the energy function.

Maximum a posteriori approximations

- The maximum a posteriori (MAP) estimate:

$$\hat{\boldsymbol{\theta}}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} \left[p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \right].$$

- Can be equivalently computed as

$$\hat{\boldsymbol{\theta}}^{\mathrm{MAP}} = \arg\min_{\boldsymbol{\theta}} \left[\varphi_T(\boldsymbol{\theta}) \right],$$

- The maximum likelihood (ML) estimate of the parameter is a MAP estimate with a formally uniform prior $p(\boldsymbol{\theta}) \propto 1$.
- The minimum (or maximum) can be found by using various gradient-free or gradient-based optimization methods.
- Gradients can be computed by recursive equations called sensitivity equations or sometimes by using Fisher's identity.

Laplace approximations

- The MAP estimate corresponds to a Dirac delta function approximation to the posterior distribution

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \simeq \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{\mathrm{MAP}}),$$

- Ignores the spread of the distribution completely.
- Idea of Laplace approximation is to form a Gaussian approximation to the posterior distribution:

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \simeq \mathrm{N}(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}^{\mathrm{MAP}}, [\mathbf{H}(\hat{\boldsymbol{\theta}}^{\mathrm{MAP}})]^{-1}),$$

where $\mathbf{H}(\hat{\boldsymbol{\theta}}^{\mathrm{MAP}})$ is the Hessian matrix of the energy function evaluated at the MAP estimate.

Markov chain Monte Carlo (MCMC)

- Markov chain Monte Carlo (MCMC) methods are algorithms for drawing samples from $p(\theta \mid \mathbf{y}_{1:T})$.
- Based on simulating a Markov chain which has the distribution $p(\theta \mid \mathbf{y}_{1:T})$ as its stationary distribution.
- The Metropolis–Hastings (MH) algorithm uses a proposal density $q(\theta^{(i)} \mid \theta^{(i-1)})$ for suggesting new samples $\theta^{(i)}$ given the previous ones $\theta^{(i-1)}$.
- Gibbs' sampling samples components of the parameters one at a time from their conditional distributions given the other parameters.
- Adaptive MCMC methods are based on adapting the proposal density $q(\theta^{(i)} \mid \theta^{(i-1)})$ based on past samples.
- Hamiltonian Monte Carlo (HMC) or hybrid Monte Carlo (HMC) method simulates a physical system to construct an efficient proposal distribution.

Metropolis–Hastings

- Draw the starting point, $\boldsymbol{\theta}^{(0)}$ from an arbitrary initial distribution.
- For $i = 1, 2, \ldots, N$ do
  1. Sample a candidate point $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)})$.
  2. Evaluate the acceptance probability

$$\alpha_i = \min \left\{ 1, \exp(\varphi_T(\boldsymbol{\theta}^{(i-1)}) - \varphi_T(\boldsymbol{\theta}^*)) \, \frac{q(\boldsymbol{\theta}^{(i-1)} \mid \boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)})} \right\}.$$

  3. Generate a uniform random variable $u \sim \mathrm{U}(0, 1)$ and set

$$\boldsymbol{\theta}^{(i)} = \begin{cases} \boldsymbol{\theta}^*, & \text{if } u \leq \alpha_i \\ \boldsymbol{\theta}^{(i-1)}, & \text{otherwise.} \end{cases}$$

Expectation–maximization (EM) algorithm [1/5]

- Expectation–maximization (EM) is an algorithm for computing ML and MAP estimates of parameters when direct optimization is not feasible.
- Let $q(\mathbf{x}_{0:T})$ be an arbitrary probability density over the states, then we have the inequality

$$\log p(\mathbf{y}_{1:T} \mid \theta) \geq F[q(\mathbf{x}_{0:T}), \theta].$$

where the functional F is defined as

$$F[q(\mathbf{x}_{0:T}), \theta] = \int q(\mathbf{x}_{0:T}) \log \frac{p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \theta)}{q(\mathbf{x}_{0:T})} \, \mathrm{d}\mathbf{x}_{0:T}.$$

- Idea of EM: We can maximize the likelihood by iteratively maximizing the lower bound $F[q(\mathbf{x}_{0:T}), \theta]$.

Expectation–maximization (EM) algorithm [2/5]

Abstract EM

The maximization of the lower bound can be done by coordinate ascend as follows:

1. Start from initial guesses $q^{(0)}$, $\theta^{(0)}$.
2. For $n = 0, 1, 2, \ldots$ do the following steps:
   1. *E-step:* Find $q^{(n+1)} = \arg\max_q F[q, \theta^{(n)}]$.
   2. *M-step:* Find $\theta^{(n+1)} = \arg\max_\theta F[q^{(n+1)}, \theta]$.

- To implement the EM algorithm we need to be able to do the maximizations in practice.
- Fortunately, it can be shown that

$$q^{(n+1)}(\mathbf{x}_{0:T}) = p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \theta^{(n)}).$$

Expectation–maximization (EM) algorithm [3/5]

- We now get

$$
\begin{aligned}
& F[q^{(n+1)}(\mathbf{x}_{0:T}), \theta] \\
&= \int p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \theta^{(n)}) \, \log p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \theta) \, \mathrm{d}\mathbf{x}_{0:T} \\
&- \int p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \theta^{(n)}) \, \log p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \theta^{(n)}) \, \mathrm{d}\mathbf{x}_{0:T}.
\end{aligned}
$$

- Because the latter term does not depend on θ, maximizing $F[q^{(n+1)}, \theta]$ is equivalent to maximizing

$$
\mathcal{Q}(\theta, \theta^{(n)}) = \int p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \theta^{(n)}) \, \log p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \theta) \, \mathrm{d}\mathbf{x}_{0:T}.
$$

Expectation–maximization (EM) algorithm [4/5]

EM algorithm

The EM algorithm consists of the following steps:

1. Start from an initial guess $\theta^{(0)}$.
2. For $n = 0, 1, 2, \ldots$ do the following steps:
   1. *E-step:* compute $\mathcal{Q}(\theta, \theta^{(n)})$.
   2. *M-step:* compute $\theta^{(n+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(n)})$.

- In state space models we have

$$\log p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \theta)$$

$$= \log p(\mathbf{x}_0 \mid \theta) + \sum_{k=1}^{T} \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta) + \sum_{k=1}^{T} \log p(\mathbf{y}_k \mid \mathbf{x}_k, \theta).$$

Expectation–maximization (EM) algorithm [5/5]

- Thus on E-step we compute

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) = &\int p(\mathbf{x}_0 \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \log p(\mathbf{x}_0 \mid \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_0 \\ &+ \sum_{k=1}^{T} \int p(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \\ &\qquad \times \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_k \, \mathrm{d}\mathbf{x}_{k-1} \\ &+ \sum_{k=1}^{T} \int p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \log p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_k.\end{aligned}$$

- In linear models, these terms can be computed from the RTS smoother results.
- In non-Gaussian models we can approximate these using Gaussian RTS smoothers or particle smoothers.
- On M-step we maximize $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$ with respect to $\boldsymbol{\theta}$.

State augmentation

- Consider a model of the form

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \theta) + \mathbf{q}_{k-1}$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \theta) + \mathbf{r}_k$$

- We can now rewrite the model as

$$\theta_k = \theta_{k-1}$$
$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \theta_{k-1}) + \mathbf{q}_{k-1}$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \theta_k) + \mathbf{r}_k$$

- Redefining the state as $\tilde{\mathbf{x}}_k = (\mathbf{x}_k, \theta_k)$, leads to the augmented model with without unknown parameters:

$$\tilde{\mathbf{x}}_k = \tilde{\mathbf{f}}(\tilde{\mathbf{x}}_{k-1}) + \tilde{\mathbf{q}}_{k-1}$$
$$\mathbf{y}_k = \mathbf{h}(\tilde{\mathbf{x}}_k) + \mathbf{r}_k$$

- This is called state augmentation approach.
- The disadvantage is the severe non-linearity and singularity of the augmented model.

Energy function for linear Gaussian models [1/3]

- Consider the following linear Gaussian model with unknown parameters θ:

$$\mathbf{x}_k = \mathbf{A}(\theta)\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}$$
$$\mathbf{y}_k = \mathbf{H}(\theta)\,\mathbf{x}_k + \mathbf{r}_k$$

- Recall that the Kalman filter gives us the Gaussian predictive distribution

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-(\theta), \mathbf{P}_k^-(\theta))$$

- Thus we get

$$\begin{aligned} & p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta) \\ &= \int \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}(\theta)\,\mathbf{x}_k, \mathbf{R}(\theta))\ \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-(\theta), \mathbf{P}_k^-(\theta))\,\mathrm{d}\mathbf{x}_k \\ &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}(\theta)\,\mathbf{m}_k^-(\theta), \mathbf{H}(\theta)\,\mathbf{P}_k^-(\theta)\,\mathbf{H}^\mathsf{T}(\theta) + \mathbf{R}(\theta)). \end{aligned}$$

Energy function for linear Gaussian models [2/3]

Energy function for linear Gaussian model

The recursion for the energy function is given as

$$\varphi_k(\boldsymbol{\theta}) = \varphi_{k-1}(\boldsymbol{\theta}) + \frac{1}{2}\log|2\pi\,\mathbf{S}_k(\boldsymbol{\theta})| + \frac{1}{2}\mathbf{v}_k^{\mathsf{T}}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\mathbf{v}_k(\boldsymbol{\theta}),$$

where the terms $\mathbf{v}_k(\boldsymbol{\theta})$ and $\mathbf{S}_k(\boldsymbol{\theta})$ are given by the Kalman filter with the parameters fixed to $\boldsymbol{\theta}$:

- *Prediction:*

$$\begin{aligned}\mathbf{m}_k^-(\boldsymbol{\theta}) &= \mathbf{A}(\boldsymbol{\theta})\,\mathbf{m}_{k-1}(\boldsymbol{\theta})\\ \mathbf{P}_k^-(\boldsymbol{\theta}) &= \mathbf{A}(\boldsymbol{\theta})\,\mathbf{P}_{k-1}(\boldsymbol{\theta})\,\mathbf{A}^{\mathsf{T}}(\boldsymbol{\theta}) + \mathbf{Q}(\boldsymbol{\theta}).\end{aligned}$$

(continues ...)

Energy function for linear Gaussian models [3/3]

Energy function for linear Gaussian model (cont.)

(...continues)

- *Update:*

$$
\begin{aligned}
\mathbf{v}_k(\theta) &= \mathbf{y}_k - \mathbf{H}(\theta)\,\mathbf{m}_k^-(\theta) \\
\mathbf{S}_k(\theta) &= \mathbf{H}(\theta)\,\mathbf{P}_k^-(\theta)\,\mathbf{H}^\mathsf{T}(\theta) + \mathbf{R}(\theta) \\
\mathbf{K}_k(\theta) &= \mathbf{P}_k^-(\theta)\,\mathbf{H}^\mathsf{T}(\theta)\,\mathbf{S}_k^{-1}(\theta) \\
\mathbf{m}_k(\theta) &= \mathbf{m}_k^-(\theta) + \mathbf{K}_k(\theta)\,\mathbf{v}_k(\theta) \\
\mathbf{P}_k(\theta) &= \mathbf{P}_k^-(\theta) - \mathbf{K}_k(\theta)\,\mathbf{S}_k(\theta)\,\mathbf{K}_k^\mathsf{T}(\theta).
\end{aligned}
$$

EM algorithm for linear Gaussian models

The expression for $\mathcal{Q}$ for the linear Gaussian models can be written as

$$
\begin{aligned}
&\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) \\
&= -\frac{1}{2} \log |2\pi \mathbf{P}_0(\boldsymbol{\theta})| - \frac{T}{2} \log |2\pi \mathbf{Q}(\boldsymbol{\theta})| - \frac{T}{2} \log |2\pi \mathbf{R}(\boldsymbol{\theta})| \\
&- \frac{1}{2} \operatorname{tr} \left\{ \mathbf{P}_0^{-1}(\boldsymbol{\theta}) \left[\mathbf{P}_0^s + (\mathbf{m}_0^s - \mathbf{m}_0(\boldsymbol{\theta})) (\mathbf{m}_0^s - \mathbf{m}_0(\boldsymbol{\theta}))^\mathsf{T} \right] \right\} \\
&- \frac{T}{2} \operatorname{tr} \left\{ \mathbf{Q}^{-1}(\boldsymbol{\theta}) \left[\boldsymbol{\Sigma} - \mathbf{C}\, \mathbf{A}^\mathsf{T}(\boldsymbol{\theta}) - \mathbf{A}(\boldsymbol{\theta})\, \mathbf{C}^\mathsf{T} + \mathbf{A}(\boldsymbol{\theta})\, \boldsymbol{\Phi}\, \mathbf{A}^\mathsf{T}(\boldsymbol{\theta}) \right] \right\} \\
&- \frac{T}{2} \operatorname{tr} \left\{ \mathbf{R}^{-1}(\boldsymbol{\theta}) \left[\mathbf{D} - \mathbf{B}\, \mathbf{H}^\mathsf{T}(\boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\theta})\, \mathbf{B}^\mathsf{T} + \mathbf{H}(\boldsymbol{\theta})\, \boldsymbol{\Sigma}\, \mathbf{H}^\mathsf{T}(\boldsymbol{\theta}) \right] \right\},
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{\Sigma} &= \frac{1}{T} \sum_{k=1}^{T} \mathbf{P}_k^s + \mathbf{m}_k^s \, [\mathbf{m}_k^s]^\mathsf{T} \\
\boldsymbol{\Phi} &= \frac{1}{T} \sum_{k=1}^{T} \mathbf{P}_{k-1}^s + \mathbf{m}_{k-1}^s \, [\mathbf{m}_{k-1}^s]^\mathsf{T} \\
\mathbf{B} &= \frac{1}{T} \sum_{k=1}^{T} \mathbf{y}_k \, [\mathbf{m}_k^s]^\mathsf{T} \\
\mathbf{C} &= \frac{1}{T} \sum_{k=1}^{T} \mathbf{P}_k^s \, \mathbf{G}_{k-1}^\mathsf{T} + \mathbf{m}_k^s \, [\mathbf{m}_{k-1}^s]^\mathsf{T} \\
\mathbf{D} &= \frac{1}{T} \sum_{k=1}^{T} \mathbf{y}_k \, \mathbf{y}_k^\mathsf{T}.
\end{aligned}
$$

EM algorithm for linear Gaussian models (cont.)

- If $\theta \in \{\mathbf{A}, \mathbf{H}, \mathbf{Q}, \mathbf{R}, \mathbf{P}_0, \mathbf{m}_0\}$, we can maximize $\mathcal{Q}$ analytically by setting the derivatives to zero.
- Leads to an iterative algorithm: run RTS smoother, recompute the estimates, run RTS smoother again, recompute estimates, and so on.
- The parameters to be estimated should be identifiable for the ML/MAP to make sense: for example, we cannot hope to blindly estimate all the model matrices.
- EM is only an algorithm for computing ML (or MAP) estimates.
- Direct energy function optimization often converges faster than EM and should be preferred in that sense.
- If a RTS smoother implementation is available, EM is sometimes easier to implement.

Gaussian filtering based energy function approximation

- Let's consider parameter estimation in non-linear models of the form

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \theta) + \mathbf{q}_{k-1}$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \theta) + \mathbf{r}_k$$

- We can now approximate the energy function by replacing Kalman filter with a Gaussian filter.
- The approximate energy function recursion becomes

$$\varphi_k(\theta) \simeq \varphi_{k-1}(\theta) + \frac{1}{2}\log|2\pi\,\mathbf{S}_k(\theta)| + \frac{1}{2}\mathbf{v}_k^{\mathsf{T}}(\theta)\,\mathbf{S}_k^{-1}(\theta)\,\mathbf{v}_k(\theta),$$

where the terms $\mathbf{v}_k(\theta)$ and $\mathbf{S}_k(\theta)$ are given by a Gaussian filter with the parameters fixed to θ.

Gaussian smoothing based EM algorithm

The approximation to $\mathcal{Q}$ function can now be written as

$$
\begin{aligned}
&\mathcal{Q}(\theta, \theta^{(n)}) \\
&\simeq -\frac{1}{2}\log|2\pi\,\mathbf{P}_0(\theta)| - \frac{T}{2}\log|2\pi\,\mathbf{Q}(\theta)| - \frac{T}{2}\log|2\pi\,\mathbf{R}(\theta)| \\
&- \frac{1}{2}\operatorname{tr}\left\{\mathbf{P}_0^{-1}(\theta)\left[\mathbf{P}_0^s + (\mathbf{m}_0^s - \mathbf{m}_0(\theta))\,(\mathbf{m}_0^s - \mathbf{m}_0(\theta))^{\mathsf{T}}\right]\right\} \\
&- \frac{1}{2}\sum_{k=1}^{T}\operatorname{tr}\left\{\mathbf{Q}^{-1}(\theta)\;\mathrm{E}\left[(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \theta))\,(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \theta))^{\mathsf{T}} \mid \mathbf{y}_{1:T}\right]\right\} \\
&- \frac{1}{2}\sum_{k=1}^{T}\operatorname{tr}\left\{\mathbf{R}^{-1}(\theta)\;\mathrm{E}\left[(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \theta))\,(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \theta))^{\mathsf{T}} \mid \mathbf{y}_{1:T}\right]\right\},
\end{aligned}
$$

where the expectations can be computed using the Gaussian RTS smoother results.

Particle filtering approximation of energy function [1/3]

- In the particle filtering approach we can consider generic models of the form

$$\begin{aligned}\theta &\sim p(\theta)\\ \mathbf{x}_0 &\sim p(\mathbf{x}_0 \mid \theta)\\ \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta)\\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k, \theta),\end{aligned}$$

- Using particle filter results, we can form an importance sampling approximation as follows:

$$p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta) \approx \sum_i w_{k-1}^{(i)}\, v_k^{(i)},$$

where

$$v_k^{(i)} = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}, \theta)\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \theta)}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k})}$$

and $w_{k-1}^{(i)}$ are the previous particle filter weights.

Particle filtering approximation of energy function [2/3]

SIR based energy function approximation

1. Draw samples $\mathbf{x}_k^{(i)}$ from the importance distributions

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}), \qquad i = 1, \ldots, N.$$

2. Compute the following weights

$$v_k^{(i)} = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}, \theta)\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \theta)}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k})}$$

and compute the estimate of $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta)$ as

$$\hat{p}(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta) = \sum_i w_{k-1}^{(i)}\, v_k^{(i)}$$

Particle filtering approximation of energy function [3/3]

SIR based energy function approximation (cont.)

3. Compute the normalized weights as

$$w_k^{(i)} \propto w_{k-1}^{(i)} \, v_k^{(i)}$$

4. If the effective number of particles is too low, perform resampling.

The approximation of the marginal likelihood of the parameters is:

$$p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) \approx \prod_k \hat{p}(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}),$$

and the corresponding energy function approximation is

$$\varphi_T(\boldsymbol{\theta}) \approx -\log p(\boldsymbol{\theta}) - \sum_{k=1}^{T} \log \hat{p}(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}).$$

Particle Markov chain Monte Carlo (PMCMC)

- The particle filter based energy function approximation can now be used in Metropolis–Hastings based MCMC algorithm.
- With finite N, the likelihood is only an approximation and thus we would expect the algorithm to be an approximation only.
- Surprisingly, it turns out that this algorithm is an exact MCMC algorithm also with finite N.
- The resulting algorithm is called particle Markov chain Monte Carlo (PMCMC) method.
- Computing ML and MAP estimates via the particle filter approximation is problematic, because resampling causes discontinuities to the likelihood approximation.

Particle smoothing based EM algorithm

- Recall that on E-step of EM algorithm we need to compute

$$Q(\theta, \theta^{(n)}) = l_1(\theta, \theta^{(n)}) + l_2(\theta, \theta^{(n)}) + l_3(\theta, \theta^{(n)}),$$

where

$$l_1(\theta, \theta^{(n)}) = \int p(\mathbf{x}_0 \mid \mathbf{y}_{1:T}, \theta^{(n)}) \log p(\mathbf{x}_0 \mid \theta) \, \mathrm{d}\mathbf{x}_0$$

$$l_2(\theta, \theta^{(n)}) = \sum_{k=1}^{T} \int p(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:T}, \theta^{(n)}) \times \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta) \, \mathrm{d}\mathbf{x}_k \, \mathrm{d}\mathbf{x}_{k-1}$$

$$l_3(\theta, \theta^{(n)}) = \sum_{k=1}^{T} \int p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \theta^{(n)}) \log p(\mathbf{y}_k \mid \mathbf{x}_k, \theta) \, \mathrm{d}\mathbf{x}_k.$$

- It is also possible to use particle smoothers to approximate the required expectations.

Particle smoothing based EM algorithm (cont.)

- For example, by using backward simulation smoother, we can approximate the expectations as

$$l_1(\theta, \theta^{(n)}) \approx \frac{1}{S} \sum_{i=1}^{S} \log p(\tilde{\mathbf{x}}_0^{(i)} \mid \theta)$$

$$l_2(\theta, \theta^{(n)}) \approx \sum_{k=0}^{T-1} \frac{1}{S} \sum_{i=1}^{S} \log p(\tilde{\mathbf{x}}_{k+1}^{(i)} \mid \tilde{\mathbf{x}}_k^{(i)}, \theta)$$

$$l_3(\theta, \theta^{(n)}) \approx \sum_{k=1}^{T} \frac{1}{S} \sum_{i=1}^{S} \log p(\mathbf{y}_k \mid \tilde{\mathbf{x}}_k^{(i)}, \theta).$$

Summary

- The marginal posterior distribution of parameters can be computed from the results of Bayesian filter.
- Given the marginal posterior, we can e.g. use optimization methods to compute MAP estimates or sample from the posterior using MCMC methods.
- Expectation–maximization (EM) algorithm can also be used for iterative computation of ML or MAP estimates using Bayesian smoother results.
- The parameter posterior for linear Gaussian models can be evaluated with Kalman filter.
- The expectations required for implementing EM algorithm for linear Gaussian models can be evaluated with RTS smoother.
- For non-linear/non-Gaussian models the parameter posterior and EM-algorithm can be approximated with Gaussian filters/smoothers and particle filters/smoothers.