Sequential Monte Carlo methods

Lecture 9 – Maximum likelihood parameter estimation

Thomas Schön, Uppsala University
2019-08-21
Aim: Open up for using the particle filter to learn parameters $\theta$ (and not only states $X_t$) in state-space models.

Outline:

1. The particle filter as likelihood estimator
2. Maximum likelihood estimation of state-space models
   a. Direct optimization
   b. Non-standard $\mu$ on quasi-Newton methods
   c. Expectation maximization
From lecture 2:

\[
X_t = f(X_{t-1}, \theta) + V_t, \\
Y_t = g(X_t, \theta) + E_t,
\]

where \(X_t\) are the states and \(\theta\) the model parameters.

**Only (but important!) difference:** \(X_t\) depends on \(t\), whereas \(\theta\) does not.

The particle filter assumes \(\theta\) is known and computes \(p(X_t \mid y_{1:T}, \theta)\).
The likelihood function

The particle filter assumes $\theta$ is known and computes $p(x_t \mid y_{1:T}, \theta)$. Learning $\theta$ requires

$$p(\theta \mid y_{1:T}) \ (Posterior; \ Bayesian\ inference)$$

or

$$p(y_{1:T} \mid \theta) \ (Likelihood\ function; \ Fisherian\ inference/maximum\ likelihood).$$

$$p(\theta \mid y_{1:T}) = \frac{p(y_{1:T} \mid \theta)p(\theta)}{p(y_{1:T})}$$

This lecture: Focus on maximum likelihood. More on the Bayesian setting in later lectures.
Maximum likelihood parameter inference

Maximum likelihood problem: Select $\theta$ such that the observed data $y_{1:T}$ is as likely as possible to have been observed, i.e.,

$$\hat{\theta} = \arg\max_{\theta} p(y_{1:T} | \theta)$$
Particle filter as likelihood estimator

\[
p(y_{1:T} | \theta) = \prod_{t=1}^{T} p(y_t | y_{1:t-1}, \theta),
\]

\[
p(y_t | y_{1:t-1}, \theta) = \int p(y_t, x_t | y_{1:t-1}, \theta) dx_t
\]

\[
= \int p(y_t | x_t, \theta) p(x_t | y_{1:t-1}, \theta) \, dx_t
\]

\[
\approx \sum_{i=1}^{N} \frac{1}{N} \delta_{x_t}^{x_t^i}(x_t)
\]

\[
= \frac{1}{N} \sum_{i=1}^{N} p(y_t | x_t^i, \theta) = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_t^i
\]

\[
\Rightarrow p(y_{1:T} | \theta) \approx \prod_{t=1}^{T} \left( \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_t^i \right)
\]

\[
(w_t^i \text{ are the unnormalized weights})
\]
Algorithm 1 Bootstrap particle filter (for \( i = 1, \ldots, N \))

1. **Initialization** \((t = 0)\):
   
   (a) Sample \( x_0^i \sim p(x_0 | \theta) \).
   
   (b) Set initial weights: \( w_0^i = 1/N \).

2. **for** \( t = 1 \) **to** \( T \) **do**
   
   (a) Resample: sample ancestor indices \( a_t^i \sim C(\{w_{t-1}^j\}_{j=1}^N) \).
   
   (b) Propagate: sample \( x_t^i \sim p(x_t | x_{t-1}^{a_t^i}, \theta) \).
   
   (c) Weight: compute \( \tilde{w}_t^i = p(y_t | x_t^i, \theta) \) and normalize \( w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j \).

\[
p(y_1:T | \theta) \approx \prod_{t=1}^T \left( \frac{1}{N} \sum_{i=1}^N \tilde{w}_t^i \right)
\]
Log-weights: an important practical aspect

For realistic problems, \( \tilde{w}_t^i \) might be smaller than machine precision \( \rightarrow \tilde{w}_t^i = 0 \) on your computer.

Use **shifted log-weights** \( v_t^i \)!

\[
v_t^i = \log \tilde{w}_t^i - c_t, \quad c_t = \max\{\log \tilde{w}_t^1, \ldots, \log \tilde{w}_t^N\}
\]

Implement using shifted log-weights! Store \( \{v_t^i\}_{i=1}^N \) and \( c_t \).

From this, the likelihood estimate is obtained

\[
\prod_{t=1}^{T} \left( \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_t^i \right) = \prod_{t=1}^{T} \exp \left( c_t + \log \sum_{i=1}^{N} e^{v_t^i} - \log N \right)
\]

Also the normalized weights \( \{w_t^i\}_{i=1}^N \) are available from \( \{v_t^i\}_{i=1}^N \),

\[
w_t^i = \frac{\tilde{w}_t^i}{\sum_{j=1}^{N} \tilde{w}_t^j} = \frac{e^{v_t^i} + c_t}{\sum_{j=1}^{N} e^{v_t^j} + c_t} = \frac{e^{v_t^i}}{\sum_{j=1}^{N} e^{v_t^j}}
\]
**ex) Numerical illustration**

Simple LG-SSM,

\[ X_t = \theta X_{t-1} + V_t, \quad V_t \sim \mathcal{N}(0, 1), \]
\[ Y_t = X_t + E_t, \quad E_t \sim \mathcal{N}(0, 1). \]

**Task:** estimate \( p(y_{1:100} | \theta) \) for a simulated data set. True \( \theta^* = 0.9 \).

Black line – true likelihood computed using the Kalman filter.

Blue thin lines – 5 different likelihood estimates \( \hat{p}^N(y_{1:100} | \theta) \) computed using a bootstrap particle filter with \( N = 100 \) particles.
The particle filter as likelihood estimator

- **Good news:** Each run of the particle filter returns an estimate of $p(y_{1:T} | \theta)$ — in addition to the state estimates!
- **Challenge:** The particle filter contains randomness → the estimate of $p(y_{1:T} | \theta)$ contains randomness or ‘noise’.
- More on its stochastic properties in the next lecture.
\[ \hat{\theta} = \arg \max_{\theta} \ p(y_1:T \mid \theta) \]

Can we use standard optimization routines?

Say, `scipy.optimize.minimize(fun=-my_BPF_function,x0 = .2)`

No. The evaluation of the cost function is ‘noisy’.

**Solution:** Use (or design) **stochastic optimization** methods that can work with noisy cost functions.
We can also get noisy approximations for the gradient of the likelihood.

**Fisher’s identity** states that

\[
\nabla_\theta \log p(y_{1:T} | \theta) = \mathbb{E}_\theta [\nabla_\theta \log p(x_{1:T}, y_{1:T} | \theta) | y_{1:T}],
\]

where

\[
\nabla_\theta \log p(x_{1:T}, y_{1:T} | \theta) = \sum_{t=1}^{T} \nabla_\theta \log p(x_t | x_{t-1}, \theta) + \nabla_\theta \log p(y_t | x_t, \theta),
\]

\[
\Rightarrow \nabla_\theta \log p(y_{1:T} | \theta) = \sum_{t=1}^{T} \int [\nabla_\theta \log p(x_t | x_{t-1}, \theta) + \nabla_\theta \log p(y_t | x_t, \theta)] p(x_{t-1:t} | y_{1:T}, \theta) dx_{t-1:t}.
\]

Here, \( p(x_{t-1:t} | y_{1:T}, \theta) \) requires a particle *smoother*. Several SMC-based alternative exists.
Stochastic optimization (very brief)
Our problem is of the form

$$\min_{\theta} f(\theta)$$

Idea underlying (quasi-)Newton methods: Learn a local quadratic model \(q(\theta_k, \delta)\) of the cost function \(f(\theta)\) around the current iterate \(\theta_k\)

\[
q(\theta_k, \delta) = f(\theta_k) + g(\theta_k)^T \delta + \frac{1}{2} \delta^T H(\theta_k) \delta
\]

\[
g(\theta_k) = \nabla f(\theta)\big|_{\theta = \theta_k}, \quad H(\theta_k) = \nabla^2 f(\theta)\big|_{\theta = \theta_k}, \quad \delta = \theta - \theta_k.
\]

We have measurements of

- the cost function \(f_k = f(\theta_k)\),
- and its gradient \(g_k = g(\theta_k)\).
Useful basic facts

Line segment connecting two adjacent iterates $\theta_k$ and $\theta_{k+1}$:

$$r_k(\tau) = \theta_k + \tau(\theta_{k+1} - \theta_k), \quad \tau \in [0, 1].$$

1. The **fundamental theorem of calculus** states that

$$\int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) d\tau = \nabla f(r_k(1)) - \nabla f(r_k(0)) = \nabla f(\theta_{k+1}) - \nabla f(\theta_k).$$

2. The **chain rule** tells us that

$$\frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) = \nabla^2 f(r_k(\tau)) \frac{\partial r_k(\tau)}{\partial \tau} = \nabla^2 f(r_k(\tau))(\theta_{k+1} - \theta_k).$$

$$g_{k+1} - g_k = \int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) d\tau = \int_0^1 \nabla^2 f(r_k(\tau)) d\tau (\theta_{k+1} - \theta_k).$$
Result – the quasi-Newton integral

With the definitions $y_k \triangleq g_{k+1} - g_k$ and $s_k \triangleq \theta_{k+1} - \theta_k$ we have

$$y_k = \int_0^1 \nabla^2 f(r_k(\tau))d\tau s_k.$$ 

**Interpretation:** The difference between two consecutive gradients ($y_k$) constitute a line integral observation of the Hessian.

**Problem:** Since the Hessian is unknown there is no functional form available for it.
Two different solutions

1. Recover existing quasi-Newton algorithms by assuming the Hessian to be constant

\[ \nabla^2 f(r_k(\tau)) \approx H_{k+1}, \quad \tau \in [0, 1], \]

implying the following approximation of the integral (secant condition)

\[ y_k = H_{k+1}s_k. \]

2. Recall that the problem is stochastic and nonlinear. Use a flexible nonlinear model.

Idea: Represent the Hessian using a Gaussian process learnt from data.
Expectation Maximization
As an alternative to direct optimization of $p(y_{1:T} | \theta)$, we can use **Expectation Maximization** (EM).


\[
Q(\theta, \theta_{k-1}) = \int \log p(y_{1:T}, x_{0:T} | \theta)p(x_{0:T} | y_{1:T}, \theta_{k-1}) \, dx_{0:T}
\]

(M) Solve $\theta_k \leftarrow \arg\max_{\theta} Q_k(\theta, \theta_{k-1})$

Iterate until convergence.

*Note: Does not make use of the particle filter as a likelihood estimator, but uses a particle smoother.*
Computing $Q$

Inserting

$$\log p(x_0:T, y_1:T \mid \theta) = \log \left( \prod_{t=1}^{T} p(y_t \mid x_t, \theta) \prod_{t=1}^{T} p(x_t \mid x_{t-1}, \theta) p(x_0 \mid \theta) \right)$$

$$= \sum_{t=1}^{T} \log p(y_t \mid x_t, \theta) + \sum_{t=1}^{T} \log p(x_t \mid x_{t-1}, \theta) + \log p(x_0 \mid \theta)$$

into the expression for $Q(\theta, \theta_k)$ results in

$$Q(\theta, \theta_k) = \int \sum_{t=1}^{T} \log p(y_t \mid x_t, \theta) p(x_t \mid y_1:T, \theta_k) d x_t$$

$$+ \int \sum_{t=1}^{T} \log p(x_t \mid x_{t-1}, \theta) p(x_{t-1:t} \mid y_1:T, \theta_k) d x_{t-1:t}$$

$$+ \int \log p(x_0 \mid \theta) p(x_0 \mid y_1:T, \theta_k) d x_0.$$
Inserting particle smoothing approximations now allows for straightforward approximation of $Q(\theta, \theta_k)$,

$$\hat{Q}(\theta, \theta_k) = \sum_{t=1}^{T} \sum_{i=1}^{N} \log p(y_t | x^i_t | T, \theta) + \sum_{t=1}^{T} \sum_{i=1}^{N} \log p(x^i_t | T | x^i_{t-1} | T, \theta) + \log \sum_{i=1}^{N} p(x^i_0 | T | \theta).$$

1. Initialize $\theta_0$ and run a particle smoother conditional on $\theta_0$.
2. Use the result from previous step to compute $\hat{Q}(\theta, \theta_0)$.
3. Solve $\theta_1 = \arg \max_{\theta} \hat{Q}(\theta, \theta_0)$.
4. Run a particle smoother conditional on $\theta_1$.
5. ....

Requires $N \to \infty$ and infinitely many iterations. There are more intricate solutions.
Further reading

Fairly recent survey/tutorial papers:

Anna Wigren, Johan Wågberg, Fredrik Lindsten, Adrian Wills and Thomas B. Schön. **Nonlinear system identification – Learning while respecting physical models using Sequential Monte Carlo.** *IEEE Control Systems Magazine*, 2022. (accepted for publication).


Maximum likelihood inference using stochastic optimization:


Maximum likelihood inference using EM:
