Sequential Monte Carlo methods

Lecture 9 – Maximum likelihood parameter estimation

Andreas Lindholm, Uppsala University
2019-08-27
Aim: Open up for using the particle filter for inference about 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. 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 \( \theta \) doesn’t.

The particle filter assumes \( \theta \) is known and computes \( p(x_t | y_{1:t}, \theta) \).
The likelihood function

The particle filter assumes $\theta$ is known and computes $p(x_t | y_{1:t}, \theta)$.

Inference about $\theta$ requires

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

or

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

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

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

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

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

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

\[
= \int p(y_t | x_t, \theta) \, \delta_{x_t}^{bPF}(x_t) \, dx_t \approx
\]

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

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

(\(w^i_t\) are the unnormalized weights)
Algorithm 1 Bootstrap particle filter (for $i = 1, \ldots, N$)

1. **Initialization** ($t = 0$):
   (a) Sample $x^i_0 \sim p(x_0 | \theta)$.
   (b) Set initial weights: $w^i_0 = 1/N$.

2. **for** $t = 1$ **to** $T$ **do**
   (a) Resample: sample ancestor indices $a^i_t \sim C(\{w^j_{t-1}\}_{j=1}^N)$.
   (b) Propagate: sample $x^i_t \sim p(x_t | x^i_{t-1}, \theta)$.
   (c) Weight: compute $\tilde{w}^i_t = p(y_t | x^i_t, \theta)$ and normalize $w^i_t = \tilde{w}^i_t / \sum_{j=1}^N \tilde{w}^j_t$.

\[
p(y_{1:T} | \theta) \approx \prod_{t=1}^{T} \left( \frac{1}{N} \sum_{i=1}^{N} \tilde{w}^i_t \right)
\]
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 your particle filter 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 $\{w_t^i\}_{i=1}^N$ can be computed from $\{v_t^i\}_{i=1}^N$,

$$w_t^i = \frac{\tilde{w}_t^i}{\sum_{j=1}^N \tilde{w}_t^j} = \frac{e^{v_t^i+c_t}}{\sum_{j=1}^N e^{v_t^j+c_t}} = \frac{e^{v_t^i}}{\sum_{j=1}^N e^{v_t^j}}$$
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 $p(y_{1:T} \mid \theta)$ — in addition to the state estimates!
- **Challenge:** The particle filter contains randomness → the estimate of $p(y_{1:T} \mid \theta)$ contains randomness or ‘noise’.
- More on its stochastic properties in the next lecture.
Direct optimization

\[ \hat{\theta} = \arg \max_\theta p(y_{1:T} \mid \theta) \]

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) **probabilistic optimization** methods that can work with noisy cost functions.

For example using Gaussian processes
Estimating likelihood gradients

We can also get noisy approximations for the gradient of the likelihood.

**Fisher’s identity**

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

where

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

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

Here, \( p(x_{t-1:t} | y_{1:T}, \theta) \) requires a particle *smoother*. Several SMC-based alternative exists, but are not in this course.
Expectation Maximization

As an alternative to direct optimization of \( p(y_{1:T} \mid \theta) \), we can use the **Expectation Maximization** (EM) method.


Idea:

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

\[
(M) \quad \text{Solve } \theta_k \leftarrow \text{argmax}_\theta Q_k(\theta, \theta_{k-1})
\]

Iterate until convergence.

*Note: Does not make use of the particle filter as a likelihood estimator, but uses a particle smoother (again: not in this course).*
Computing $Q$

Inserting

$$\log p(x_{0:T}, y_{1:T} | \theta) = \log \left( \prod_{t=1}^{T} p(y_t | x_t, \theta) \prod_{t=1}^{T} p(x_t | x_{t-1}, \theta) p(x_0 | \theta) \right)$$

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

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

$$Q(\theta, \theta_k) = \int \sum_{t=1}^{T} \log p(y_t | x_t, \theta) p(x_t | y_{1:T}, \theta_k) dx_t$$

$$+ \int \sum_{t=1}^{T} \log p(x_t | x_{t-1}, \theta) p(x_{t-1:t} | y_{1:T}, \theta_k) dx_{t-1:t}$$

$$+ \int \log p(x_0 | \theta) p(x_0 | y_{1:T}, \theta_k) dx_0.$$
Final EM algorithm

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:


Maximum likelihood inference using the Gaussian process:


Maximum likelihood inference using EM: