UPPSALA
UNIVERSITET

Sequential Monte Carlo methods

Lecture 9 – Likelihood estimator and maximum likelihood

Thomas Schön, Uppsala University

2017-08-25

Outline – Lecture 9

Aim: Show how the particle filter can be used to estimate the likelihood and how we can compute maximum likelihood estimates of unknown model parameters.

Outline:

1. Estimating the likelihood using the particle filter
2. Maximum likelihood estimation of SSMs
   a. Direct optimization
   b. Expectation maximization (if there is time)

Normalizing the weights using shifted log-weights

The normalized weights $\{w_t^i\}_{i=1}^N$ can be computed from the shifted log-weights $\{v_t^i\}_{i=1}^N$, since we have that

$$w_t^i = \frac{\widetilde{w}_t^i}{\sum_{j=1}^N \widetilde{w}_t^j} = \frac{e^{v_t^i + c_t}}{\sum_{j=1}^N e^{v_t^j + c_t}} = \frac{e^{v_t^i}}{\sum_{j=1}^N e^{v_t^j}}$$

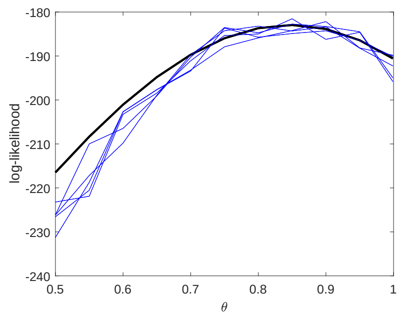
Computing $e^{v_t^i}$ is numerically more well-behaved as compared to computing $\widetilde{w}_t^i$.

ex) Numerical illustration

Simple LG-SSM,

$$\begin{aligned} X_t &= \theta X_{t-1} + V_t, & V_t &\sim \mathcal{N}(0,1), \\ Y_t &= X_t + E_t, & E_t &\sim \mathcal{N}(0,1). \end{aligned}$$

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



Black line – true likelihood computed using the Kalman filter.

Blue thin lines – 5 different likelihood estimates $\widehat{p}^N(y_{1:100} \mid \theta)$ computed using a bootstrap particle filter with $N = 100$ particles.

Maximum likelihood parameter inference

Maximum likelihood problem: Select the θ that according to the observed data $y_{1:T}$ is "as likely as possible" in the sense that

$$\begin{aligned}\widehat{\theta} &= \arg\max_{\theta} p(y_{1:T} \mid \theta) \\ &= \arg\max_{\theta} \sum_{t=1}^{T} \log \int p(y_t \mid x_t, \theta) p(x_t \mid y_{1:t-1}, \theta) \mathrm{d}x_t\end{aligned}$$

Briefly survey two solutions:

1. Direct optimization
2. Expectation maximization (EM)

Direct optimization

Problem: There is one issue **preventing** us from using standard numerical routines for nonlinear optimization:

- The evaluations of the cost function and its derivatives are all noisy.

Solution: Use **probabilistic optimization** methods that can work with noisy function, gradient and Hessian evaluations.

Part of the recent **probabilistic numerics** initiative.

Estimating likelihood gradients using SMC

You know how to approximate the likelihood using the particle filter. We can also approximate the gradient using a particle smoother.

Fisher's identity tells us that

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

where

$$\nabla_\theta \log p(x_{1:T}, y_{1:T} \mid \theta) = \sum_{t=1}^{T} \xi(x_{t-1:t} \mid \theta),$$

with

$$\xi(x_{t-1:t} \mid \theta) = \nabla_\theta \log p(x_t \mid x_{t-1}, \theta) + \nabla_\theta \log p(y_t \mid x_t, \theta).$$

Combining all of the above we obtain

A non-standard (but fruitful) take on quasi-Newton

Our problem is of the form

$$\widehat{\theta} = \arg\max_{\theta} f(\theta)$$

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

$$q(\theta_k, \Delta) = f(\theta_k) + g(\theta_k)^\mathsf{T}\Delta + \frac{1}{2}\Delta^\mathsf{T} H(\theta_k)\Delta$$

Quasi-Newton methods learns the Hessian $H(\theta_k)$ using "observations" of the cost function and the gradient resulting in

$$f_q(\theta_k, \Delta) = f(\theta_k) + g(\theta_k)^\mathsf{T}\Delta + \frac{1}{2}\Delta^\mathsf{T} B_k\Delta$$

A non-standard (but fruitful) take on quasi-Newton

Key question: How do we model and compute an estimate B_k of the Hessian?

It has recently been shown that standard quasi-Newton methods (BFGS, DFP, Broyden's method) can all be interpreted as particular instances of Bayesian linear regression.

Philipp Hennig and Martin Kiefel. **Quasi-Newton methods: A new direction.** *Journal of Machine Learning Research*, 14:843–865, 2013.

Idea: What about modelling the true Hessian using a **Gaussian process** that we then learn using observations of the function and its gradient at different parameter values?

μ on the Gaussian Process (GP)

An abstract idea

In probabilistic linear regression (recall our autoregressive example from lecture 1)

$$y_t = \underbrace{\theta^{\mathsf{T}} \mathbf{x}_t}_{f(\mathbf{x}_t)} + e_t, \quad e_t \sim \mathcal{N}(0, \sigma^2),$$

we place a prior on θ, $\theta \sim \mathcal{N}(0, \sigma^2 I_n)$.

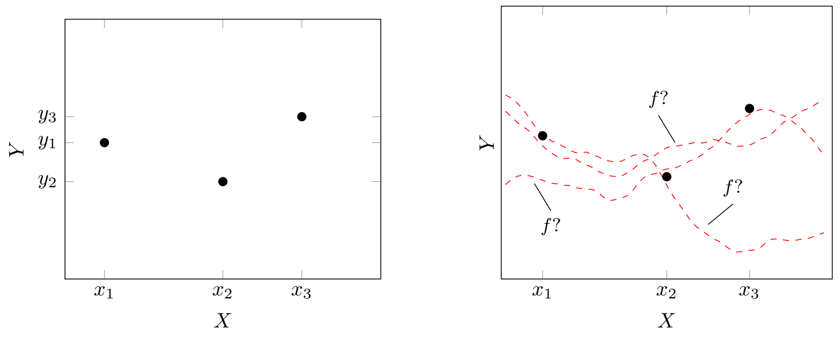
(Abstract) idea: What if we instead place a prior directly on the function $f(\cdot)$

$$f \sim p(f)$$

and look for $p(f \mid \mathbf{y})$ rather than $p(\theta \mid \mathbf{y})$?!

An abstract idea – pictures

What does it actually mean to have a prior over functions?



Can we construct a probabilistic object operating on functions?

One concrete construction

Well, one (arguably simple) idea on how we can reason probabilistically about an unknown function f is by assuming that $f(x)$ and $f(x')$ are jointly Gaussian distributed

$$\begin{pmatrix} f(x) \\ f(x') \end{pmatrix} \sim \mathcal{N}(\mu, K)$$

If we accept the above idea we can without conceptual problems generalize to any *arbitrary* set of input values $\{x_1, x_2, \ldots, x_T\}$.

Definition and its implications

Definition: (Gaussian Process, GP) A GP is a (potentially infinite) collection of random variables such that any finite subset of it is jointly distributed according to a Gaussian.

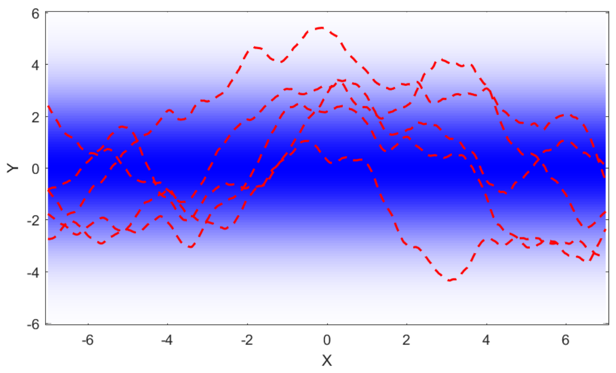
Our definition means that for any *arbitrary* set of input values $\{x_1, x_2, \ldots, x_T\}$ we have

$$\begin{pmatrix} f(x_1) \\ \vdots \\ f(x_T) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(x_1) \\ \vdots \\ m(x_N) \end{pmatrix}, \begin{pmatrix} k(x_1, x_1) & \ldots & k(x_1, x_T) \\ \vdots & \ddots & \vdots \\ k(x_T, x_1) & \ldots & k(x_T, x_T) \end{pmatrix}\right)$$

We now have a prior!

$$f \sim \mathcal{GP}(m, k)$$

The GP is a **generative** model so let us first sample from the prior.



GP regression

Remaining problem: Given training data $\mathcal{T} = \{x_t, y_t\}_{i=1}^{T}$ and our GP prior $f \sim \mathcal{GP}(m, k)$ compute $p(f_\star \mid \mathbf{y})$ for an arbitrary test point $(x_\star, y_\star)$.

$$\begin{pmatrix} \mathbf{y} \\ f_\star \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}) \\ m(x_\star) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}, \mathbf{x}) + \sigma^2 I_T & k(\mathbf{x}, x_\star) \\ k(x_\star, \mathbf{x}) & k(x_\star, x_\star) \end{pmatrix}\right),$$

The conditioning theorem for partitioned Gaussians results in

$$\begin{aligned} f_\star \mid \mathbf{y} &\sim \mathcal{N}(\mu_\star, k_\star), \\ \mu_\star &= m(x_\star) + \mathbf{s}^\mathsf{T}(\mathbf{y} - m(\mathbf{x})), \\ k_\star &= k(x_\star, x_\star) - \mathbf{s}^\mathsf{T} k(\mathbf{x}, x_\star), \end{aligned}$$

where $\mathbf{s}^\mathsf{T} = k(x_\star, \mathbf{x})(k(\mathbf{x}, \mathbf{x}) + \sigma^2 I_T)^{-1}$.

GP regression – illustration

Two different ways of using the GP

1. Model the cost function $f(\theta)$ as a GP

$$f(\theta) \sim \mathcal{GP}(\mu(\theta), k(\theta, \theta'),$$

and possibly the gradient and the Hessian as well.

We can then employ standard GP regression to use the information we have from noisy observations of the cost function and its derivatives.

---

2. Model the Hessian using a GP

$$\widetilde{B}(\tau) \sim \mathcal{GP}(\mu(\tau), k(\tau, t),$$

The Hessian prior model is then updated using noisy observations of the cost function and its derivatives.

Adrian G. Wills and Thomas B. Schön. **On the construction of probabilistic Newton-type algorithms**. Pro-

ex) nonlinear state space model

Consider the following nonlinear state-space model

$$x_t = 0.5x_{t-1} + b\frac{x_{t-1}}{1 + x_{t-1}^2} + 8\cos(1.2t) + qw_t, \qquad w_t \sim \mathcal{N}(0,1)$$

$$y_t = 0.05x_t^2 + e_t, \qquad e_t \sim \mathcal{N}(0, 0.1)$$

Task: Compute the maximum likelihood estimate of $\theta = (b, q)^{\mathsf{T}}$.

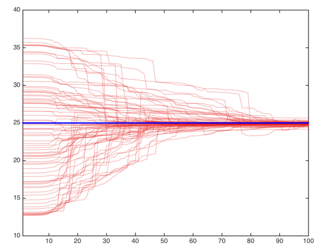
$T = 100$ measurements simulated using the true value $\theta^\star = (25, \sqrt{0.1})$.

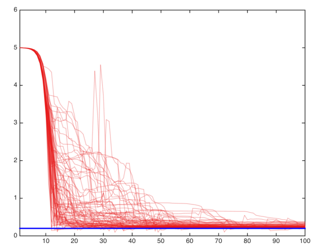
We use $N = 500$ particles in the particle filter to estimate the likelihood.

Fishers' identity together with a particle smoother is used to estimate the likelihood gradient.

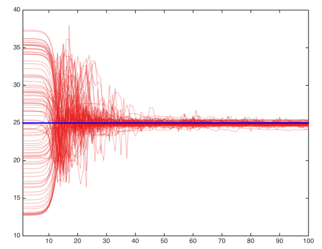
ex) Result placing a GP prior on the Hessian

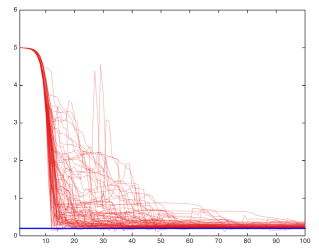
Model the cost function and its gradient as a GP (b left, q right)



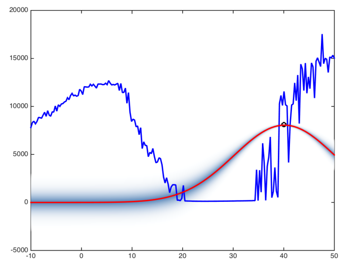


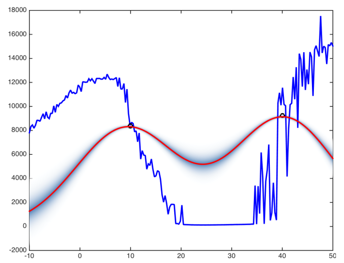
Model the Hessian as a GP (b left, q right)

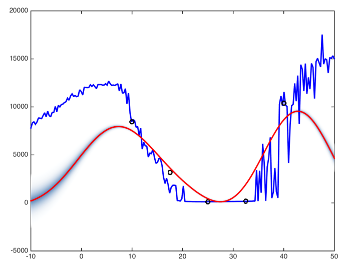


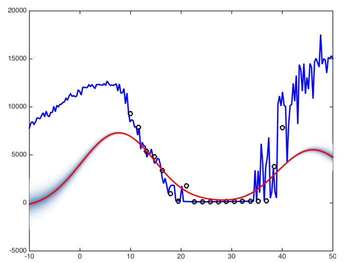


ex) Result placing a GP prior on the function


(a) GP after 1 sample.


(b) GP after 2 samples.





EM in one slide

The expectation maximization (EM) algorithm is an iterative approach to compute maximum likelihood estimates of unknown parameters θ in probabilistic models involving latent variables (e.g. the state trajectory $x_{1:T}$ in the SSM).

EM employs the complete likelihood $p(x_{1:T}, y_{1:T} \mid \theta)$ as a **substitute/surrogate** for the observed likelihood $p(y_{1:T} \mid \theta)$,

$$p(x_{1:T}, y_{1:T} \mid \theta) = p(x_{1:T} \mid y_{1:T}, \theta) p(y_{1:T} \mid \theta).$$

EM works by iteratively computing

$$\mathcal{Q}(\theta, \theta_k) = \int \log p(x_{1:T}, y_{1:T} \mid \theta) p(x_{1:T} \mid y_{1:T}, \theta_k) \mathrm{d}x_{1:T}$$

and then maximizing $\mathcal{Q}(\theta, \theta_k)$ w.r.t θ.

Computing $\mathcal{Q}$

Inserting the following

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

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

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

$$\mathcal{Q}(\theta, \theta_k) = \int \sum_{t=1}^{T} \log p(y_t \mid x_t, \theta) p(x_t \mid y_{1:T}, \theta_k) \mathrm{d}x_t$$
$$+ \int \sum_{t=1}^{T} \log p(x_t \mid x_{t-1}, \theta) p(x_{t-1:t} \mid y_{1:T}, \theta_k) \mathrm{d}x_{t-1:t}$$
$$+ \int \log p(x_0 \mid \theta) p(x_0 \mid y_{1:T}, \theta_k) \mathrm{d}x_0.$$

Final EM algorithm

Inserting particle smoothing approximations now allows for straightforward approximation of $\mathcal{Q}(\theta, \theta_k)$,

$$\widehat{\mathcal{Q}}(\theta, \theta_k) = \sum_{t=1}^{T} \sum_{i=1}^{N} \log p(y_t \mid x_{t\,|\,T}^i, \theta) + \sum_{t=1}^{T} \sum_{i=1}^{N} \log p(x_{t\,|\,T}^i \mid x_{t-1\,|\,T}^i, \theta)$$
$$+ \log \sum_{i=1}^{N} p(x_{0\,|\,T}^i \mid \theta).$$

---

1. Initialize θ_0 and run a particle smoother using an SSM parameterized by θ_0.
3. Use the result from step 2 to compute $\widehat{\mathcal{Q}}(\theta, \theta_0)$.
4. Solve $\theta_1 = \arg\max_{\theta} \widehat{\mathcal{Q}}(\theta, \theta_0)$.
5. Run a particle smoother using an SSM parameterized by θ_1.
6. ....

Further reading

Fairly recent survey/tutorial papers:

Nikolas Kantas, Arnaud Doucet, Sumeetpal S. Singh, Jan Maciejowski and Nicolas Chopin. **On particle methods for parameter estimation in general state-space models.** *Statistical Science*, 30(3):328-351, 2015.

Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wagberg, Christian A. Naesseth, Andreas Svensson and Liang Dai. **Sequential Monte Carlo methods for system identification.** *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, Beijing, China, October 2015.

Maximum likelihood inference using the Gaussian process:

Adrian G. Wills and Thomas B. Schön. **On the construction of probabilistic Newton-type algorithms.** *Proceedings of the 56th IEEE Conference on Decision and Control (CDC)*, Melbourne, Australia, December 2017.

Maximum likelihood inference using EM:

Thomas B. Schön, Adrian Wills and Brett Ninness. **System Identification of Nonlinear State-Space Models.** *Automatica*, 47(1):39-49, January 2011.

Reinterpreting standard quasi-Newton as Bayesian learning:

Philipp Hennig and Martin Kiefel. **Quasi-Newton methods: A new direction.** *Journal of Machine Learning Research*, 14:843–865, 2013.

A few concepts to summarize lecture 9

Likelihood function: A deterministic function obtained by inserting a particular realization of the measurements $Y_{1:T} = y_{1:T}$ into the data distribution.

Maximum likelihood Find the unknown parameters that maximizes the likelihood function.

Expectation maximization (EM) An iterative approach to compute maximum likelihood estimates of unknown parameters in probabilistic models involving latent variables.