Exercises set IV
PhD course on Sequential Monte Carlo methods 2017

Department of Information Technology, Uppsala University

August 28, 2017

This document contains exercises to make you familiar with the content of the course. *The exercises in this document are not mandatory, and you do not need to hand in your solutions.* The mandatory assignment is found in a separate document named "Hand-in". We strongly recommend that you carefully work through these exercises before starting with the mandatory assignments.

IV.1 Particle Metropolis-Hastings

Consider the following state-space model

$$x_t = \cos(\theta x_{t-1}) + v_t, \qquad v_t \sim \mathcal{N}(0,1) \tag{1a}$$
$$y_t = x + e_t, \qquad e_t \sim \mathcal{N}(0,1) \tag{1b}$$
$$x_0 \sim \mathcal{N}(0,1). \tag{1c}$$

Generate $T = 50$ data points $y_{1:T}$ from this model with $\theta = 1$, and then pretend that you forgot the true value of θ, but assume that you drew it from $\mathcal{N}(0,1)$. Use particle Metropolis-Hastings to infer the posterior distribution of θ, $p(\theta \mid y_{1:T})$, as follows:

(a) Implement a particle filter of your choice to deliver an estimate $\widehat{z}_\theta$ of the likelihood $p(y_{1:T} \mid \theta)$.

(b) Design a proposal $q(\theta' \mid \theta[k-1])$ for θ' as, e.g., a Gaussian random walk on θ-space.

(c) Implement a Metropolis-Hastings sampler, where the acceptance ratio is computed as

$$\min\left(1, \frac{\widehat{z}_{\theta'}}{\widehat{z}_{\theta[k-1]}} \frac{p(\theta')}{p(\theta[k-1])} \frac{q(\theta' \mid \theta[k-1])}{q(\theta[k-1] \mid \theta')}\right) \tag{2}$$

($p(\theta)$ is the prior).

Note: the standard Metropolis-Hastings algorithm requires the target, in this case $p(\theta \mid y_{1:T}) \propto p(y_{1:T} \mid \theta)p(\theta)$, to be evaluated exactly (up to proportionality). Here, however, we only have access to estimates $\widehat{z}_\theta$ of $p(y_{1:T} \mid \theta)$, and the fact that this still works is depending on the so-called pseudo-marginal Metropolis-Hastings. The key here is the unbiasedness of $\widehat{z}_\theta$.

IV.2 Conditional particle filter

Return to any of the particle filter implementations you have done earlier, and turn it into a *conditional* particle filter as follows:

- Introduce the concept of a reference state trajectory $x_{1:T}[k-1]$
- In each propagation step, use multinomial resampling but replace the propagation of the N:th particle x_{t-1}^N (after resampling) to x_t^N by just taking $x_t[k-1]$ from the reference state trajectory (instead of sampling from $p(x_t \,|\, x_{t-1}^N)$).
- After the final time step, $t = T$, sample a new reference trajectory $x_{1:T}[k]$ by sampling j from the categorical distribution defined by $\{w_T^i\}_{j=1}^N$, and take x_T^j and its ancestors $x_{T-1}^{a_t^j}$ etc.

This is the particle Gibbs Markov kernel, which maps one reference state trajectory $x_{1:T}[k-1]$ onto another $x_{1:T}[k]$, which can be combined with sampling the unknown parameters in a Gibbs fashion (i.e., sample θ conditional on $x_{1:T}[k-1]$ as $\theta[k] \sim p(\theta \,|\, x_{1:T}[k-1])$).

IV.3 Conditional importance sampling

(a) Implement an MCMC procedure to sample from $\pi(x) = \mathcal{N}(x \,|\, 1, 1)$ by using a conditional importance sampling kernel with proposal $q(x) = \mathcal{N}(x \,|\, 0, 1)$. Verify that you get samples from the target in the long run, even using $N = 2$ particles in the importance sampler.

(b) Let $A \subset \mathcal{X}$. Note that we can write $\int_A \kappa_N(x, x^\star)\mathrm{d}x^\star$, i.e. the probability that a draw from the conditional importance sampling kernel falls in A, as:

$$\int_A \kappa_N(x, x^\star)\mathrm{d}x^\star = \mathbb{E}\left[\frac{\sum_{i=1}^N \mathbb{1}(X^i \in A)\omega(X^i)}{\sum_{j=1}^N \omega(X^j)}\right]$$

Assume that $\omega(x) \leq c$ and verify that

$$\mathbb{E}\left[\frac{\sum_{i=1}^N \mathbb{1}(X^i \in A)\omega(X^i)}{\sum_{j=1}^N \omega(X^j)}\right] \geq \left(1 - \frac{1}{1 + d(N-1)}\right) \int_A \pi(x)\mathrm{d}x$$

for some constant d.

This is referred to as a minorization of the conditional importance sampline kernel, and underlies the decomposition of the kernel as discussed in the lectures.

IV.4 An SMC sampler for localization

We want to localize an object positioned at x_0 in the world $[-12, 12]^2$, as illustrated in Figure 1. We have access to a bunch of measurements $y_{1:M}$ of the position, corrupted by heavy-tail noise. As the measurements are corrupted by heavy-tailed noise from the exponential distribution, we are not really interested in just a point estimate of x_0, but instead we want the entire posterior distribution $p(x_0|y_{1:M})$ of its position, reflecting the uncertainty inherent in the problem.

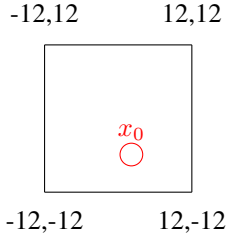


Figure 1: Illustration of our world $[-12, 12]^2$, with the true position x_0 indicated using a red circle.

(a) Prepare code for simulating M independent measurements from the following model

$$y_t^1 = x_0^1 + n_m^1 b_m^1, \tag{3a}$$
$$y_t^2 = x_0^2 + n_m^2 b_m^2, \tag{3b}$$

for $m = 1, \ldots, M$, where x_0^1 and x_0^2 are the components of x_0, and n_m^1 and n_m^2 are exponentially distributed with scale parameter 2, and $\mathbb{P}\left(b_m^1 = 1\right) = \mathbb{P}\left(b_m^1 = -1\right) = \frac{1}{2}$ and similarly for b_m^2.

(b) Prepare code for evaluating the density $p(y_m \,|\, x_0)$ based on the definitions above. This defines the likelihood in your problem, $p(y_{1:m} \,|\, x_0) = \prod_{m=1}^{M} p(y_m \,|\, x_0)$.

(c) Based on our background knowledge on the problem, you know that a reasonable prior for the position x_0 is the following,

$$p(x_0) = \mathcal{N}\left(x_0 \,|\, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 7 & 0 \\ 0 & 7 \end{bmatrix}\right). \tag{4}$$

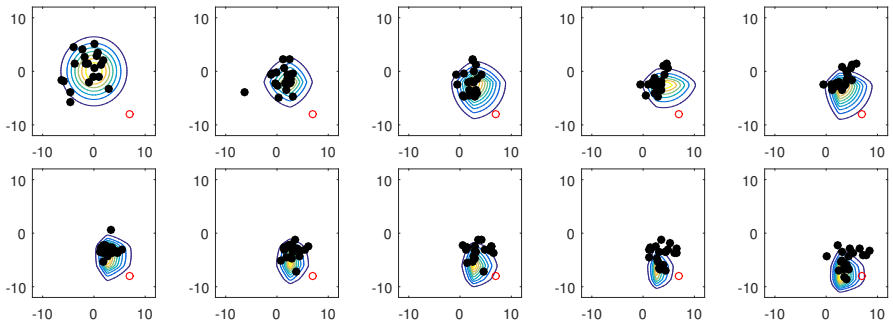
Design a likelihood tempered transition $\pi_0, \ldots, \pi_P$ from the prior $\pi_0(x_0) = p(x_0)$ to the posterior $\pi_P(x_0) = p(x_0|Y) \propto p(x_0) \prod_{t=1}^{T} p(y_t|x_0)$.

(d) Implement a π_n-invariant Metropolis-Hastings (MH) kernel based on a random walk proposal.

(e) Put everything together in an SMC sampler .

*There are alternative ways to update the particle weights in an SMC sampler. The easiest is probably to set the weight of particle x_{n-1}^i as $\pi_n(x_{n-1}^i)/\pi_{n-1}(x_{n-1}^i)$ [1, eq. (31)].

(f) Test your algorithm by making sure that it converges to something close to x_0 when, say, $N = 100$ particles and $M = 50$ measurements are used. Make plots to follow the evolution of the particles in the SMC sampler, such as



The black dots are the particles, the red circle is the true position of x_0 and the contours are proportional to π_n, for $n = 1, \ldots, 10$.

(g) Make a comparison between the SMC sampler and simply using the Metropolis-Hastings sampler to sample from π.

(h) OPTIONAL. To make the problem we are solving more interesting, assume another measurement model. Instead of measuring the objects absolute position, we are now measuring its relative distance to some sensors s_j,

$$y_j = \|x_0 - s_j\| + n, \tag{5}$$

where n still is exponentially distributed, and s_j denotes the coordinates for sensor j. That is, the distance between the sensor and the object is known (except for the noise), but not the angle.

Update your code using this measurement model instead. What does the posterior look like for the case of only one sensor in the origin? What about multiple sensors with different locations? How well is the SMC sampler doing in each scenario, respectively?

References

[1] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. “Sequential Monte Carlo samplers”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (2006), pp. 411–436.