SMC Sampler for TARN

March 30, 2015

1 SMC Samplers

Want to use SMC samplers for TARN.

- First we use SIR to sample from the sequence of densities

$$\pi_n(s_{1:n}) \propto \left|\sum_{i=1}^{\tau_n} f(s_i)\right|^{\kappa_n} p(s_{1:n}),\ n = 1, 2, \ldots, m,$$

  where $0 \leq \kappa_1 \leq \kappa_2 \leq \cdots \leq \kappa_m$.

- This is sampling from a sequence of spaces of increasing dimensions (dimension is n, which is increasing from 1 to m).

- Implementing SIR in this case is slightly different from the last time (when we simply used SIR to estimate the price), as in this case we have to keep track of the stopping times of the paths as well.

- Implement SIR to give the following output:

$$S = \begin{pmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & \cdots & s_{\tau_m^{(1)}}^{(1)} & \cdots & \cdots & s_m^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_{\tau_m^{(2)}}^{(2)} & \cdots & \cdots & \cdots & s_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_1^{(M)} & s_2^{(M)} & \cdots & \cdots & \cdots & s_{\tau_m^{(M)}}^{(M)} & \cdots & s_m^{(M)} \end{pmatrix}_{M \times m}$$

  The k-th path looks like $s_1^{(k)} \to s_2^{(k)} \to \ldots \to s_{\tau_m^{(k)}}^{(k)} \to \ldots \to s_m^{(k)}$. The notation looks hideous.

- $\tau_m^{(k)}$ denotes the stoppng time for the k-th path when there are m-time periods in total.

- The whole paths (even after the option has 'died') needs to be stored as the SMC sampler requires this.

1.1 Basic SMC Sampler

- Given the samples from $\pi_n(s_{1:n})$, we are going to use SMC samplers to sample from the sequence of densities

$$\tilde{\pi}_n(s_{1:m}) \propto \left|\sum_{i=1}^{\tau_m} f(s_i)\right|^{\tilde{\kappa}_n} p(s_{1:m}),\ n = 1, 2, \ldots, p,$$

  where $\kappa_m = \tilde{\kappa}_1 \leq \tilde{\kappa}_2 \leq \cdots \leq \tilde{\kappa}_m = 1$.

- This is sampling from a sequence of spaces of the same dimension (dimension is m, which stays fixed).

- We let $x_n^{(l)} = s_{1:m,n}^{(l)}$ denote the l-th particle at time n, $n = 1, 2, \ldots, m$ and $l = 1, 2, \ldots, M$.

- We propagate $x_{n-1}^{(l)} \to x_n^{(l)}$ using a forward Markov kernel $K_n\left(x_{n-1}^{(l)}, x_n\right)$ that is $\tilde{\pi}_n(x_n)$ invariant. Use a Metropolis-Hastings proposal step.

- Given a path $\mathbf{s} = (s_1, s_2, \ldots, s_{\tau_m}, \ldots, s_m)$, propose a new path $\mathbf{S} = (S_1, S_2, \ldots, S_{T_m}, \ldots, S_m)$ as $\mathbf{s} + \mathcal{N}_m(0, \sigma_\star^2 I_m)$ for some $\sigma_\star$ where I_m is the identity matrix.
- T_m is the stoppig time for the proposed path.
- Accept $\mathbf{S}$ with probabilty
$$1 \wedge \frac{|\sum_{i=1}^{T_m} f(S_i)|^{\tilde{\kappa}_n} p(S_{1:m})}{|\sum_{i=1}^{\tau_m} f(s_i)|^{\tilde{\kappa}_n} p(s_{1:m})}$$
- Have to do this for every path, at every time step n ($n = 1, 2, \ldots, p$).

- The backward kernel is chosen as in [1].
- The incremental weights are then

$$\begin{aligned}
\alpha_n^{(l)}\left(x_{n-1}^{(l)}\right) &= \frac{\tilde{\pi}_n\left(x_{n-1}^{(l)}\right)}{\tilde{\pi}_{n-1}\left(x_{n-1}^{(l)}\right)} \\
&= \frac{\left|\sum_{i=1}^{\tau_{m,n-1}^{(l)}} f\left(s_{i,n-1}^{(l)}\right)\right|^{\tilde{\kappa}_n}}{\left|\sum_{i=1}^{\tau_{m,n-1}^{(l)}} f\left(s_{i,n-1}^{(l)}\right)\right|^{\tilde{\kappa}_{n-1}}} \\
&= \left|\sum_{i=1}^{\tau_{m,n-1}^{(l)}} f\left(s_{i,n-1}^{(l)}\right)\right|^{\tilde{\kappa}_n - \tilde{\kappa}_{n-1}}, \; n = 1, 2, \ldots, p, \text{ and } l = 1, 2, \ldots, M
\end{aligned}$$

Implemented a basic version of the SMC sampler.

- Code is slow, need to speed up certain parts. Not immediately sure how to do it though.
- Using a one-step random walk Metropolis-Hastings step for propagation in the SMC part.
- SMC sampler does not seem to do better.

1.2 Some Immediate Issues

- *Speed*: can this be implemented more efficiently? I suspect it can be.
- *Kernel*: how to choose the kernel K_n? This is an important issue.
- κ: as in SIR, how to choose the sequence of κ's and $\tilde{\kappa}$'s?

1.3 What else can we do?

- As in SIR, can choose pretty much any sequence of target densities $\{\pi_n(s_{1:n})\}_{1 \le n \le m}$ and $\{\tilde{\pi}_n(s_{1:m})\}_{1 \le n \le p}$ (just make sure that they are integrable).
- Just make sure that $\pi_m(s_{1:m})$ and $\tilde{\pi}_1(s_{1:m})$ match and that
$$\tilde{\pi}_p(s_{1:m}) = \left|\sum_{i=1}^{\tau_m} f(s_i)\right| p(s_{1:m}).$$
- But how to do this really? (Of course, the issue of choosing the kernel K_n will still remain.)
- Well, what we are trying to do is that construct artificially a sequence of target probability densities so that the normalizing constant is estimated with the least variance. Is there any literature on this?
- Is it worthwhile to look at the asymptotic variance of Z_n and try to minimize it?

2 A Particular Problem

- $S_0 = 100$, $\sigma = 8\%$ (not sure if this means $\sigma = 8$ or $\sigma = 0.08$ in our implementation).
- $m = 24$, $k = 30$.
- $$f(s) = \begin{cases} 2(s-110)+20 & \text{if } s > 110 \\ 2(80-s)+20 & \text{if } s < 90 \\ -20 & \text{if } 90 \leq s \leq 110 \end{cases}$$
- $\Gamma_G = 200$, $\Gamma_L = 100$.

Some diagnostics:

- We plot the ESS of SIR and SMC.
  - ESS for SIR is always very high (more than threshold of $M/2$). This means that the SIR never resamples and thus does exactly the same as naive MC.
  - SMC only resamples once.
- We look at the average acceptance rate of the MH step.
  - High average acceptance rate, more than 85%.
- Can use the adaptive algorithm of [2].
  - But then this tries to make the ESS high. We anyway have the ESS high.

Discovered a bug in the MH step of the SMC sampler. Was not calculating the stopping time correctly. Fixed this bug. No noticable change.

2.1 Tempering

We want to construct a sequence of continuous functions

$$\{g_0\}_{1 \leq n \leq p} : \mathbb{R}^m \to \mathbb{R}$$

such that

$$g_p \equiv g := \sum_{i=1}^{\tau_m} f(s_i)$$

and then set the sequence of target distributions to be

$$\tilde{\pi}_n(s_{1:m}) = |g_n(s_{1:m})| p(s_{1:m})$$

The idea is that if the functions g_n approach g gradually, then moving through

$$g_0 \to g_1 \to \cdots g_{p-1} \to g_p = f$$

would be a sensible thing to try when trying to sample from $\tilde{\pi}_p$.

The g_n's are functions in $\mathbb{R}^m$, and so it is hard to select them. An easier thing to do is the following:

- Since we have fixed m, let us drop the m from τ_m.
- Choose a sequence of functions
  $$\{f_n\}_{0 \leq n \leq p} : \mathbb{R} \to \mathbb{R}$$
  with $f_0 \equiv 1$ and $f_p \equiv f$.

- Then set the the sequence of target distributions to be

$$\tilde{\pi}_n(s_{1:m}) = \left| \sum_{i=1}^{\tau_{f_n}} f_n(s_i) \right| p(s_{1:m})$$

  In this case, the stopping time is with respect to the function f_n.

- We note that this is going to be computationally quite expensive.

How to choose the f_n's?

2.1.1 First attempt

- Choose a sequence of positive numbers $\delta = \delta_1 > \delta_2 > \cdots > \delta_{p-1} > 0$ and set

$$f_n(s) = \begin{cases} 2(80-s)+20 & \text{if } s < 90 \\ -\frac{20}{\delta_n}(s-90) & \text{if } 90 \le s < 90+\delta_n \\ -20 & \text{if } 90+\delta_n \le s < 110-2\delta_n \\ \frac{20}{\delta_n}(s-110)+20 & \text{if } 110-2\delta_n \le s < 110 \\ 2(s-110)+20 & \text{if } s > 110 \end{cases}$$

- Managed to implement this.

- Very very slow, calculating the stoping time for every path with respect to every f_n at every iteration takes a lot of time. Not sure if possible to implement faster.

- Still using random walk MH step as the kernel K_n.

2.1.2 Second Attempt

- Choose $f_n(s) = \delta_n |s-110|^{1.01} - 20$, $1 \le n \le p-1$.

This seems to point towards using a sequence of parametric functions $f_n = f_{\delta_n}$, where δ_n is the parameter.

References

[1] Jasra, A. and Del Moral, P., *Sequential Monte Carlo for Option Pricing*, Stochastic Analysis and Applications, (2011)

[2] Del Moral, P., Doucet, A. and Jasra, A., *An Adaptive Sequential Monte Carlo Method for Approximate Bayesian Computation*, Stat Computing, (2011)