SIR for TARN

March 30, 2015

1 SIR

1.1 Basic

Want to use Sequential Importance Resampling for TARN.

- Choose $\gamma_n(s_{1:n}) = |\sum_{i=1}^{\tau_n} f(s_i)| p(s_{1:n})$ and $Z_n = \int \gamma_n(s_{1:n}) p(s_{1:n}) \mathrm{d}s_{1:n} = \int |\sum_{i=1}^{\tau_n} f(s_i)| p(s_{1:n}) \mathrm{d}s_{1:n}$.
- τ_n because stopping time depends on the number of months.
- Choose proposal density $q_n(s_{1:n}) = p_n(s_{1:n})$ (process density).
- Want to estimate Z_m.
- Incremental weights:
$$\alpha_n(s_{1:n}) = \frac{\gamma_n(s_{1:n})}{\gamma_{n-1}(s_{1:n-1}) q_n(s_n | s_{1:n-1})} = \frac{|\sum_{i=1}^{\tau_n} f(s_i)|}{\left|\sum_{i=1}^{\tau_{n-1}} f(s_i)\right|}$$
- At time n:
  - For k-th path, propagate it forward if still alive after time $n-1$. If it is dead, do nothing.
  - Incremental weight will be
$$\alpha_n\left(s_{1:n}^{(k)}\right) = \begin{cases} \frac{\left|\sum_{i=1}^{n} f\left(s_i^{(k)}\right)\right|}{\left|\sum_{i=1}^{n-1} f\left(s_i^{(k)}\right)\right|} & \text{if } k\text{-th path is alive after time } n-1 \\ 1 & \text{if } k\text{-th path is dead after time } n-1 \end{cases}$$
  - Resample.
  - After resampling, find out whether the k-th resampled path is alive after time n:
    - If it was replaced by a path that was already dead, then it remains dead.
    - If it was replaced by a path that was previously alive, then it might or might not die now.

This should ultimately lead to an unbiased estimator of $\mathbb{E}\,|\sum_{i=1}^{\tau_m} f(S_i)|$.

If $f > 0$, then the resampling step would lead to paths that are still alive being given higher weights. Of course, in that case we would not need to take an absolute value.

- For example, choose $f(x) = e^{-x}$. Then $f > 0$.
- Longer paths given higher weights.
- Do not notice SIR doing any better, but need to do more experiments.
- SIR and MC taking similar running time.

2 Slightly Modified

Let us next try SIR as in SMC samplers, with $0 \leq \kappa_1 \leq \kappa_2 \leq \cdots \leq \kappa_m = 1$:

- Choose $\gamma_n(s_{1:n}) = |\sum_{i=1}^{\tau_n} f(s_i)|^{\kappa_n} p(s_{1:n})$.
- Incremental weights:
$$\alpha_n(s_{1:n}) = \frac{|\sum_{i=1}^{\tau_n} f(s_i)|^{\kappa_n}}{\left|\sum_{i=1}^{\tau_{n-1}} f(s_i)\right|^{\kappa_{n-1}}}.$$
- At time n:
  - For k-th path, propagate it forward if still alive after time $n-1$. If it is dead, do nothing.
  - Incremental weight will be
$$\alpha_n\left(s_{1:n}^{(k)}\right) = \begin{cases} \frac{\left|\sum_{i=1}^{n} f\left(s_i^{(k)}\right)\right|^{\kappa_n}}{\left|\sum_{i=1}^{n-1} f\left(s_i^{(k)}\right)\right|^{\kappa_{n-1}}} & \text{if } k\text{-th path is alive after time } n-1 \\ 1 & \text{if } k\text{-th path is dead after time } n-1 \end{cases}$$
- Easy to implement.
- But even this is not doing any better.
- Tried to select the κ's randomly, yet to no avail.

THOUGHTS:

- We have control over the sequence of κ's.
- This is been referred to as 'temperature' in literature somewhere.
- For the barrier option case, we were estimating a bunch of conditional survival probabilites efficiently using SMC instead of using MC to estimate the overall survival probability. This is like importance splitting.

What else can we do?

- We could try SMC samplers. This is much more computationally intensive, and there is another pdf file for this.
- We could just try to do MCMC. This will have its own issues.
- We could try to do some of the more advanced SMC methods, for example from [1]. But none of these seem to focus on the normalizing constant.
- We can in general change the sequence of target densities pretty much as we like:
$$\gamma_n(s_{:n}) = g_n(s_{1:n})$$
for any sequence of positive functions $\{g_n\}_{1 \leq n \leq m}$ with
$$g_m(s_{1:m}) = \left|\sum_{i=1}^{\tau_m} f(s_i)\right| p(s_{1:n}).$$
Getting a sequence of g_n's that estimates the normalizing constant well is the challenge.

Let us focus on the last idea at the moment. In-fact, let us focus on tempering.
Could do this within the SMC samplers as well.

3 A Particular Problem

Trivial case. No longer working on a log-scale.

1. $S_0 = 100$, $\sigma = 8\%$, $\Gamma_G = 200$, $\Gamma_L = 100$, $m = 24$, $k = 30$.

$$f(s) = \begin{cases} 2(s-110)+20 & \text{if } s > 110 \\ 2(80-s)+20 & \text{if } s < 90 \\ -20 & \text{if } 90 \leq s \leq 110 \end{cases}$$

2. Constant sequence of κ's:
   - Doing same as MC.
3. Linearly increasing sequence of κ's:
   - Interestingly, MC does marginally better.
   - The ESS is always very high. This means that the SIR algorithm is never resampling and that is why it is doing same as naive MC.
4. Exponentially increasing sequence of κ's:
   - Not doing any better, in-fact doing slightly worse.
   - Perhaps because the κ's increase too slowly initially.
   - Never resamples.
5. Under these dynamics, the estimated price is 476 and the estimated standard error is about 1.5% for MC. This is probably not very good.
6. This means that a 95% (say) confidence interval for the price is about (460,490), which is not a desirable thing.

THOUGHTS:

- There seems to be a trade-off between SMC and naive MC. For SMC, we do not want to resample much as we fear that this depletes the particle system. On the other hand, if we do not really resample much then it is similar to doing naive MC. So where exactly is the benefit of doing SMC?

3.1 Tempering

We observe that $f(s)$ is discontinuous at $s = 90$ and at $s = 110$. In fact, there is quite a big jump at these points. Our idea is to have a sequence of continuous functions approximating f and use these in the tempering strategy:

- Choose $\{f_n\}_{n=1}^m$ continuous such that $f_m \equiv f$ and f_n approaches f as $n \to m$.
- Set target density to be

$$\gamma_n(s_{1:n}) = \left| \sum_{i=1}^{\tau_n} f_n(s_i) \right| p(s_{1:n})$$

- In this context, we can choose τ_n to be the stopping time either depending on f_n or simply on f. If it is simply on f, then it is much easier to implement and that is what we try first.

3.1.1 Stopping time depending on f only

- Incremental weights are

$$\alpha_n\left(s_{1:n}^{(l)}\right) = \frac{\left|\sum_{i=1}^{\tau_{n,f}^{(l)}} f_n\left(s_i^{(l)}\right)\right|}{\left|\sum_{i=1}^{\tau_{n-1,f}^{(l)}} f_{n-1}\left(s_i^{(l)}\right)\right|}, \quad n = 1, 2, \ldots, m$$

- This is more complicated than SIR with a temperature scheme to implement. Can it be done more efficiently?

Choosing approximating functions:

1. We choose a sequence of positive numbers $\delta = \delta_1 > \delta_2 > \cdots > \delta_{m-1} > 0$ and set

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

   This does about the same as MC.

2. Choose some other sequences of approximating functions. In both cases we are doing worse than MC.

3.1.2 Stopping times depending on each f_n

- Incremental weights are

$$\alpha_n\left(s_{1:n}^{(l)}\right) = \frac{\left|\sum_{i=1}^{\tau_{n,f_n}^{(l)}} f_n\left(s_i^{(l)}\right)\right|}{\left|\sum_{i=1}^{\tau_{n-1,f_{n-1}}^{(l)}} f_{n-1}\left(s_i^{(l)}\right)\right|}, \quad n = 1, 2, \ldots, m$$

Even this does the same as naive MC.

References

[1] Doucet, A. and Johansen, A.M., *A Tutorial on Particle Filtering and Smothing: Fifteen Years Later*, Oxford Handbook of Nonlinear Filtering (2011)