SMC for Bayes

Joshua J Bon

2019-05-24

or "Sequential Monte Carlo for static Bayesian model"

Prerequisites

Assumed knowledge consists of

1. Some experience with Bayesian modelling and inference.
2. Some understanding of Markov chain Monte Carlo algorithms.
3. Tolerance of vague, possibly misleading subheadings.

Do it for Bayes (some motivation)

Monte Carlo methods (MC) are a suite of probabilistic computational techniques that simulate random variables from probability distributions that have density or mass functions that are difficult to study analytically. For example, such probability laws can have forms that are computationally expensive, or analytically intractable. Analytic intractability primarily manifests as an inability to estimate a multivariate integral, the normalising constants of the target probability distribution. Monte Carlo methods bypass evaluation of this integral by generating draws from the target distribution, from which quantities such as moments or interval probabilities can be approximated. Monte Carlo methods are used in many contexts including in Bayesian statistics to estimate parameter posterior distributions[1], thereby quantifying uncertainty and enabling inference about the real world.

[1] A parameter's posterior distribution is the probability the parameter takes certain values after accounting for prior beliefs and available evidence (or data).

The justification for Monte Carlo integration appeal to a law of large numbers for integrable functions. The integral, or expectation of f, assuming it is integrable,

$$\mathsf{E}_\pi[f(\boldsymbol{\theta})] = \int_\Theta f(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathsf{d}\boldsymbol{\theta} = \lim_{n\to\infty} n^{-1}\sum_{i=1}^{n} f\left(\boldsymbol{\theta}^{(i)}\right) \tag{1}$$

if the random draws $\boldsymbol{\theta}^{(i)}$ have distribution $\pi(\boldsymbol{\theta})$. The approximation from the limit is therefore

$$\mathsf{E}_\pi[f(\boldsymbol{\theta})] \approx n^{-1}\sum_{i=1}^{n} f\left(\boldsymbol{\theta}^{(i)}\right) \quad \text{when} \quad \boldsymbol{\theta}^{(i)} \sim \pi(\boldsymbol{\theta})$$

for large enough n. Modern MC methods focus on simulating from $\pi(\boldsymbol{\theta})$ efficiently, and use the above approximation for inference.

Sequential Monte Carlo (SMC) is an attractive and robust MC method that is commonly used in Bayesian inference. It can be preferred to competing methods, such as Markov chain Monte Carlo (MCMC), as it is adaptive, parallelisable, and better equipped to handle multimodality and non-linear dependence in target distributions. SMC, unlike MCMC, can also be easily applied to modelling data that arrives in real-time, often referred to as sequential Bayesian analysis, as opposed to static analysis.

This research will focus on improving the efficiency, robustness, and applicability of sequential Monte Carlo methods, primarily for static Bayesian inference.

Old school (historical notes)

Contemporary SMC has its origins in importance sampling and particle filtering, which are often motivated by Bayesian statistics and signal processing, respectively. The term 'Sequential Monte Carlo', appears at least as early as Liu and Chen (1998) where importance sampling, resampling, and Markov chain methods were combined to estimate dynamic systems[2] in real-time. This work drew heavily from particle filters such as the Bootstrap (Gordon et al., 1993) and Kalman filters (see Chen, 2003, for example), as well as sequential importance sampling from the time.

[2] State-space models for example.

Whilst the key ingredients of SMC have remained much the same since Liu and Chen (1998), numerous theoretical and algorithmic advances have been made. The work of Del Moral et al. (2006) contains the canonical exposition of modern SMC, describing very general methods with accompanying theory to guide the construction of effective SMC algorithms. The scope of their work was for models with a fixed number of parameters, which is the case for static Bayesian inference.

In general, an SMC algorithm operates by iteratively updating a population of particles (realisations of random variables) to match a sequence of desired distributions. The final distribution in the sequence is the target, which is difficult to sample from directly. The algorithm incrementally progresses particles from one distribution to the next by iteratively reweighting and resampling, followed by refreshing the particles to add diversity. The first two steps are equivalent to importance sampling (Kroese et al., 2011, Ch. 9.7), whilst the third often uses an MCMC kernel to diversify replicated particles in the sample, thereby avoiding degeneracy. We will now describe these steps in more detail.

New wave (modern SMC)

SMC uses a series of densities, $\pi_1, \pi_2, \ldots, \pi_T$, to iteratively generate a realised set of random variables (or particles) whose distribution approaches a desired law, say the density $\pi(\boldsymbol{\theta}) = \pi_T(\boldsymbol{\theta})$ for random variable $\boldsymbol{\theta} \in \Theta$. The sequence, indexed by $t = 1, 2, \ldots, T$, has corresponding particles and weights described by

$$\left\{\boldsymbol{\theta}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{I}$$

for some t and the number of particles $I > 1$. The SMC algorithm is designed such that a particle at index i and iteration t, $\theta_t^{(i)}$, has distribution with density $\pi_t(\boldsymbol{\theta})$. The starting density, $\pi_1(\boldsymbol{\theta})$, is chosen so that it can be simulated from easily, whilst the density we are targeting is that of $\pi(\boldsymbol{\theta}) = \pi_T(\boldsymbol{\theta})$. We proceed through a series of densities (in truth, particles with the current density as their distribution) that are increasingly complex. Each subsequent density is "near" enough to the proceeding one so that evolving the particles from one to the next is feasible. The weights are such that $w_t^{(i)} > 0$, and $\sum_{i=1}^{I} w_t^{(i)} = 1$ for $t = 1, 2, \ldots, T$.

The exact sequence of distributions to move from $\pi_1(\boldsymbol{\theta})$ to $\pi_T(\boldsymbol{\theta})$ can be defined in a number of ways. The most popular schedules are

- *data annealing*, where $\pi_t(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta} \mid y_1, \ldots, y_t)$, where the target is $\pi(\boldsymbol{\theta}) = p(\boldsymbol{\theta} \mid y_1, \ldots, y_T)$ and $T = N$ the number of data observations; and
- *geometric annealing*, where $\pi_t(\boldsymbol{\theta}) \propto \pi(\boldsymbol{\theta})^{\phi_t}\mu(\boldsymbol{\theta})^{1-\phi_t}$ for $0 = \phi_1 < \phi_2 < \cdots < \phi_{T-1} < \phi_T = 1$, and $\mu(\boldsymbol{\theta})$ is the starting distribution of the particles.
- *likelihood annealing* (a special case of geometric for static Bayesian models), where $\pi_t(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{y} \mid \boldsymbol{\theta})^{\phi_t} p(\boldsymbol{\theta})$, the likelihood function is $\pi(\boldsymbol{y} \mid \boldsymbol{\theta})$, and $p(\boldsymbol{\theta})$ is the prior distribution. The annealing parameter ϕ_t is an in geometric annealing.

Data annealing was first used in Chopin (2002) for static Bayesian models, whilst geometric annealing has its origins in papers by Gelman and Meng (1998) as well as Neal (2001). The components that make up an SMC algorithm, can be roughly divided into three iterated steps:

1. *Reweight* particles for the next distribution in the sequence;
2. *Resample* particles to duplicate "good" particles and remove "bad" ones;
3. *Refresh* particles to increase diversity after particles duplicated above.

We will now outline these steps in more detail.

Reweight

The reweighting step is based on importance sampling, which can be seen as a more general form of the Monte Carlo integration identity. Specifically, the expectation in (1) can be rewritten as

$$E_\pi[f(\boldsymbol{\theta})] = \int_\Theta f(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} = \int_\Theta f(\boldsymbol{\theta})\frac{\pi(\boldsymbol{\theta})}{\pi_0(\boldsymbol{\theta})}\pi_0(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} = E_{\pi_0}[w(\boldsymbol{\theta})f(\boldsymbol{\theta})]$$

for some density $\pi_0(\boldsymbol{\theta})$ with the same support as $\pi(\boldsymbol{\theta})$, where $w(\boldsymbol{\theta}) = \frac{\pi(\boldsymbol{\theta})}{\pi_0(\boldsymbol{\theta})}$.

Then by stochastic approximation, we can write

$$E_\pi[f(\boldsymbol{\theta})] \approx \sum_{i=1}^{n} w_i f\left(\boldsymbol{\theta}^{(i)}\right) \quad \text{when} \quad \boldsymbol{\theta}^{(i)} \sim \pi_0(\boldsymbol{\theta}) \tag{2}$$

where $w_i = \frac{\pi\left(\boldsymbol{\theta}^{(i)}\right)}{n\pi_0\left(\boldsymbol{\theta}^{(i)}\right)}$, for large n. The density $\pi_0(\boldsymbol{\theta})$ is called the importance distribution. It is from this approximation that importance sampling, and its derivatives are borne.[3]

[3] Sufficient conditions for workable importance samplers are given in Geweke (1989) and relate to the finiteness of the expected value under the importance distribution and supremum of the ratio of target distribution to importance distribution.

In relation to SMC, importance sampling is performed iteratively, as in sequential or annealed importance sampling (see Neal, 2001, and references therein). At step $t+1$ the previous particles and weights are $\left\{\boldsymbol{\theta}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{I}$. These particles are then evaluated against the new targeted density $\pi_{t+1}(\boldsymbol{\theta})$. That is the previous density, $\pi_t(\boldsymbol{\theta})$ is the importance distribution for the next density $\pi_{t+1}(\boldsymbol{\theta})$. Due to the sequential nature of the algorithm, the random samples from $\pi_t(\boldsymbol{\theta})$ have already been generated. However, each particle will be differently suited to the new density compared to the old one, so the weights must be updated.

In general determining the weights is non-trivial, but an auxiliary parameter approach can be used (Del Moral et al., 2006, Sec. 3.1). There is an important simplification available when the move step is an MCMC kernel with invariant distribution π_{t+1} for a static model. Under this condition, the process of updating the unnormalised weights, W_{t+1} can be performed using the formula (Del Moral et al., 2006, Sec. 3.3.2.3)

$$W_{t+1}^{(i)} = W_t^{(i)} \frac{\gamma_{t+1}\left(\boldsymbol{\theta}_{t+1}^{(i)}\right)}{\gamma_t\left(\boldsymbol{\theta}_t^{(i)}\right)}$$

where $\gamma_t(\boldsymbol{\theta})$ is the unnormalised target density at time t, i.e. $\pi_t(\boldsymbol{\theta}) = \gamma_t(\boldsymbol{\theta})/Z_t$ and Z_t is the normalising constant. The weights can then be normalised by

$$w_t^{(i)} = \frac{W_t^{(i)}}{\sum_{i=1}^{I} W_t^{(i)}}.$$

Resample

The particle weights will degenerate each iteration and eventually most of the weight will be concentrated over a small set of particles. This is the main draw back of standard importance sampling or sequential importance sampling without resampling and refreshment. Degenerate weights cause two problems for SMC and other importance sampling algorithms:

1. It is inefficient to use computational resources to store and manipulate particles with a small weights. In particular, particles with small weights contribute very little to the Monte Carlo integration.
2. Large discrepancies in weight increases the variance of the estimate of the integral. Or in other words, the estimate is less accurate on average.

Resampling addresses the first issue by taking a random sample of the particles with probability proportional to the current weight. This preserves the expectation, removes weights with low values, and duplicates those with high values. There are three main strategies for resampling:

1. *Multinomial resampling*: Sample I particles from the current particles with replacement, with the probability of selecting a given particle proportional to its current weight. After which each particle has unit weight.
2. *Stratified resampling*: Partition the particles into m sets by weight, such that each block (of the partition) has the same total weight, and then sample I/m particles from each block[4] (Kitagawa, 1996). After which each particle has unit weight.
3. *Residual resampling*: Randomly round the current weights to integer values, such that the mean is preserved (see for example, Liu and Chen, 1998). Duplicate particles based on their integer weight, after which each new particle has unit weight.

[4] Assuming I/m is an integer.

Systematic resampling is also popular and can be viewed as a simplified version of stratified sampling. Other variations are also possible but they often lack an intuitive description, unlike the three examples given above. Generally, resampling strategies are functions of empirical inverse tail probabilities (Gerber et al., 2019) which is a standard basis for describing, comparing, and efficiently implementing them.

It is well known that stratified and residual resampling outperform multinomial by reducing the variance of the particles. For general SMC algorithms, stratified sampling is recommended as it is more efficient and has stronger proven theoretical guarantees of convergence than its peers (Gerber et al., 2019). Other specialised resampling techniques also exist for certain situations, for example when resampling particles in a distributed computer setting (Lee and Whiteley, 2016; Murray et al., 2016).

Whilst resampling addresses the first issue, the second issue requires a combination of resampling and refreshment to be effectively dealt with. We detail how particles are refreshed below.

Refresh

After resampling, there are I particles with unit weight. However, many of them may be duplicated. Refreshment[5] increases the diversity of the particles by allowing them to traverse the parameter space. Using a combination of resampling and refreshment addresses the second issue mentioned above: Particles with relatively large weights are duplicated then diversified so that degeneracy in the particles can be avoided and the variance of the integral estimator can be decreased.

[5] Refreshment is also known as mutation, move step, or jittering among other names.

We focus on the situation where an MCMC kernel is used for refreshment because this is the most common practise for static Bayesian models. The resampled particles will be $\left\{\boldsymbol{\theta}_{t+1}^{(i)}\right\}_{i=1}^{I}$ with unit weights. An MCMC kernel, or transition probability density (or mass) function with the Markovian property, $k(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{t+1})$, can be used if the stationary (or invariant) distribution of the Markov chain defined by k is π_{t+1} where $t+1$ is the current time. We update each particle by sampling from this kernel with $\boldsymbol{\theta}_{t+1}$ equal to the current position. This step can be iterated multiple times, and the last value is taken as the current position of each particle. The global current time is not incremented as the particle in the chain still has a π_{t+1} law, due to the MCMC kernel's invariance. Note that the weights need to be updated after mutation, but this can generally be left until the next iteration of the algorithm.

Whilst we have described mutation as the last step, this does not diminish its importance from an algorithmic and historical perspective. In fact the origins of SMC for static models are just as much based on resample-move algorithms (Chopin, 2002, for example), as they are in annealed importance sampling (Neal, 2001).

Improvise. Adapt. Overcome.

Several enhancements are used in modern SMC algorithms to dramatically increase their efficiency, and are outlined in the subsections below. These improvements result from dynamic SMC algorithms which adapt using the current population of particles. Whilst they can improve efficiency, adaptive SMC algorithms are not covered by the theoretical convergence guarantees developed for their non-adaptive counterparts. The theory for some special cases of adaptivity has been developed by Del Moral et al. (2012) and Beskos et al. (2016). When this theory is not available the issue can be circumvented using a pilot run of the entire SMC algorithm. The adapted parameters from the pilot run are fixed for a second, non-adaptive version of SMC, which will be covered by the general convergence results for SMC in static models (see for example, Del Moral et al., 2006).

Adapting the annealing schedule

Basic SMC samplers use a fixed schedule of temperatures, $\phi_1, \phi_2, \ldots, \phi_T$, chosen prior to starting the algorithm. The increments in temperature are required to be close enough so that particles from π_t form a good importance sample for π_{t+1}, but distant enough so avoid excessive run time. If the temperatures are too distant, the particles can be degenerate, that is, only a small proportion of the particles account for most of the weight in the importance sample. If this occurs estimates from the SMC sampler can be unstable (high variance).

A metric for monitoring particle degeneracy is the effective sample size (ESS), which measures the effective number of independent samples[6]. ESS is

[6] Recall the particle weights measure duplication, hence I particles with different weights do not necessarily represent an independent sample of size I.

often approximated using the current weights by

$$\widehat{\mathrm{ESS}}_t = \left(\sum_{i=1}^{I} (w_t^{(i)})^2 \right)^{-1},$$

noting that when $w_t^{(i)} = 1/I$ for all $i \in \{1, 2, \ldots, I\}$ then $\widehat{\mathrm{ESS}}_t = I$, as expected.

Since reweighting is performed to prevent degeneracy, SMC algorithms can monitor the approximate ESS at each iteration and resample only when the ESS falls below a pre-specified threshold. This threshold is often chosen to be $I/2$.

MCMC Kernel tuning

When using an MCMC kernel, the set of particles available prior to refreshment can be used to tune the kernel. Tuning the kernel in a principled manner allows the particles to traverse the parameter space more efficiently, so that particle diversification increases at a low computational cost.

A popular MCMC move for SMC is the multivariate normal random (MVN) walk Metropolis-Hastings (MH) kernel. MVN-MH kernels operates by proposing a random move for each particle, where the move is simulated from a multivariate normal distribution. The proposed new position of each particle is evaluated using MH ratio, and accepted with this probability. If the move is accepted it becomes the new position, otherwise, the previous position prevails. Using the MH ratio to determine the acceptance or rejection of the proposed move is crucial as it preserves the invariant distribution (Kroese et al., 2011, Ch. 6.1).

It is often the case with SMC samplers, that MVN-MH kernel are tuned by calculating the empirical covariance matrix of the current particles, so that the MCMC step (approximately) respects the correlation between different elements of the distribution. That is, the estimated covariance used in the MVN random walk is

$$\hat{Q}_t = (I-1)^{-1} \sum_{i=1}^{I} \left(\boldsymbol{\theta}_t^{(i)} - \bar{\boldsymbol{\theta}}_t \right)^\top \left(\boldsymbol{\theta}_t^{(i)} - \bar{\boldsymbol{\theta}}_t \right)$$

or a scaled version of this.

More generally, tuning generic transition kernels can be performed by monitoring desirable metrics such as the expected square jump distance (ESJD). This has been used in MCMC (Craiu et al., 2009; Sherlock and Roberts, 2009; Pasarica and Gelman, 2010) and SMC (Fearnhead and Taylor, 2013; Salomone et al., 2018) with success. Whilst details of the implementation vary, maximising ESJD promotes better exploration of the parameter space and results in a more efficient sampler. In SMC, typically a pilot run of different tuning parameters is used and the “best”[7] parameters are adopted for the remaining mutation steps for the current time.

[7] For example, Salomone et al. (2018) use the kernel parameter corresponding to the highest median estimated ESJD.

Number of MCMC steps

The refreshment step can fail to effectively diversify the current set of particles if too large a proportion of the MCMC proposals result in a rejection. Whilst running multiple MCMC steps can resolve this, determining the number of iterations required is not trivial. A general solution to this issue is to run a single MCMC step, then use a criteria in order to decide how many more iterations are required.

The most established criteria aims to ensure that the overall expected acceptance probability for k iterations is above a certain threshold (Drovandi and Pettitt, 2011; South et al., 2018). The probability that at least one MCMC step is accepted in a sequence of k iterations of an MCMC kernel is

$$\alpha_k = 1 - (1 - \alpha_1)^k$$

for $k \in \{1, 2, \ldots\}$, where α_1 is the probability of accepting a single MCMC step. A pilot MCMC step can be used to estimate the average acceptance rate across the particles in a single step, $\widehat{\alpha_1}$. We can then find k such that $\alpha_{k+1} \geq p$ some threshold $0 < p < 1$. The term $k + 1$ is used as we have already run the pilot MCMC step once. The formula for k is then

$$k \geq \frac{\log(1-p)}{\log(1-\widehat{\alpha_1})} - 1$$

where $\widehat{\alpha_1}$ is the estimated acceptance rate from one run. The number of mutation steps, k, can be selected as the smallest positive integer that satisfies the above equation.

A possibly improved criteria that has recently been proposed involves ensuring the expected jump distance (EJD) is greater than a pre-specified level (Salomone et al., 2018). The MCMC kernel is run for multiple steps until a proportion of particles, ρ, has exceeded a threshold for the total EJD, J. Both ρ and J are prespecified. The benefit in comparison to the expected acceptance rate method is its robustness to poor quality MCMC moves. For example, a kernel that proposes very short jumps will have a high acceptance rate, at the cost of efficient exploration of the target distribution. The expected acceptance method will not be able to mitigate this issue, whilst the EJD would be expected to do much better.

References

Alexandros Beskos, Ajay Jasra, Nikolas Kantas, Alexandre Thiery, et al. On the convergence of adaptive sequential monte carlo methods. *The Annals of Applied Probability*, 26(2):1111–1146, 2016.

Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics: A Journal of Theoretical and Applied Statistics*, 182(1):1–69, 2003.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

Radu V Craiu, Jeffrey Rosenthal, and Chao Yang. Learn from thy neighbor: Parallel-chain and regional adaptive mcmc. *Journal of the American Statistical Association*, 104(488):1454–1466, 2009.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

Pierre Del Moral, Arnaud Doucet, Ajay Jasra, et al. On adaptive resampling strategies for sequential monte carlo methods. *Bernoulli*, 18(1):252–278, 2012.

Christopher C Drovandi and Anthony N Pettitt. Likelihood-free bayesian estimation of multivariate quantile distributions. *Computational Statistics & Data Analysis*, 55(9):2541–2556, 2011.

Paul Fearnhead and Benjamin M Taylor. An adaptive sequential monte carlo sampler. *Bayesian analysis*, 8(2):411–438, 2013.

Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, 13(2):163–185, 1998.

Mathieu Gerber, Nicolas Chopin, and Nick Whiteley. Negative association, ordering and convergence of resampling methods. *Annals of Statistics*, (forthcoming), 2019.

John Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–1339, 1989.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1): 1–25, 1996.

Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of Monte Carlo methods*. John Wiley & Sons, Hoboken, New Jersey, 2011.

Anthony Lee and Nick Whiteley. Forest resampling for distributed sequential Monte Carlo. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 9(4):230–248, 2016.

Jun S Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.

Lawrence M Murray, Anthony Lee, and Pierre E Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3): 789–805, 2016.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11 (2):125–139, 2001.

Cristian Pasarica and Andrew Gelman. Adaptively scaling the metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, 20(1): 343–364, 2010.

Robert Salomone, Leah F South, Christopher C Drovandi, and Dirk P Kroese. Unbiased and consistent nested sampling via sequential monte carlo. *arXiv preprint arXiv:1805.03924*, 2018.

Chris Sherlock and Gareth Roberts. Optimal scaling of the random walk metropolis on elliptically symmetric unimodal targets. *Bernoulli*, 15(3): 774–798, 2009.

L. F. South, A. N. Pettitt, and C. C. Drovandi. Sequential monte carlo samplers with independent markov chain monte carlo proposals. *Bayesian Analysis*, 2018. DOI: 10.1214/18-BA1129. URL https://doi.org/10.1214/18-BA1129. Advance publication.