Advanced Simulation Methods

Chapter 9 - Sequential Monte Carlo methods

Sequential Monte Carlo (SMC) methods, aka “particle methods”, constitute a broad and popular class of Monte Carlo algorithms that have been developed over the past twenty years to provide approximate solutions to intractable inference problems. For a detailed treatment of SMC, see [1], [2] and [3]. In these notes, we build upon Sequential Importance Sampling (SIS), which was introduced in the previous notes, in order to describe a generic SMC algorithm. The notation is essentially the same as in Chapter 8.

1 Resampling

We have seen that SIS provides estimates whose variance increases, typically exponentially, with t, because the importance weights become more and more unbalanced when t increases. Resampling is the key ingredient of SMC methods, which partially solves this problem. Resampling refers to the task of transforming a weighted sample $\left(w^i, X^i\right)$, for $i \in \{1, \dots, N\}$, into an unweighted sample $\left(\bar{X}^i\right)$, for $i \in \{1, \dots, N\}$, without “changing the distribution”; in other words, we want $\left(\bar{X}^i\right)$ to represent the same distribution than the weighted sample, but without weights. For instance, if the X^i are drawn from a proposal q, and $w^i \propto \pi(X^i)/q(X^i)$, then the weighted samples are approximately distributed according to π; we want to obtain an unweighted sample also approximately distributed according to π. Of course, simply discarding the weights does not work, since we would then have an unweighted sample distributed according to q.

There are many ways to perform resampling. The most common one is called multinomial resampling, and is described in Algorithm 1. It consists in drawing for each index i, a “ancestry” variable A^i in $\{1, \dots, N\}$ with probabilities

$$\forall k \in \{1, \dots, N\} \quad \mathbb{P}\left(A^i = k\right) = \frac{w^k}{\sum_{j=1}^N w^j}.$$

Then the particle $\bar{X}^i$ is set to be equal to X^{A^i} for all i. This way, particles with higher weights get chosen more often as “ancestors” of the new particles $\left(\bar{X}^i\right)$.

The exact same algorithm can be represented in another way: each $X^i_{1:t}$ gives birth to a certain number of offsprings, proportionally to its weight w^i_t. Both formulations are equivalent. The algorithm based on the offspring representation is described in Algorithm 2.

Algorithm 1 Multinomial resampling based on an ancestry vector. Input $w^{1:N}$ and $X^{1:N}$. Output $\bar{X}^{1:N}$.

- Draw an “ancestry vector” $A^{1:N} = \left(A^1, \dots, A^N\right) \in \{1, \dots, N\}^N$ independently from a categorical distribution
$$A^{1:N} \overset{\text{i.i.d}}{\sim} \mathcal{C}at\left(w^1, \dots, w^N\right),$$
in other words
$$\forall i \in \{1, \dots, N\} \quad \forall k \in \{1, \dots, N\} \quad \mathbb{P}\left[A^i = k\right] = \frac{w^k}{\sum_{j=1}^N w^j}.$$
- Define $\bar{X}^i$ to be X^{A^i} for all $i \in \{1, \dots, N\}$. X^{A^i} is said to be the “parent” or “ancestor” of $\bar{X}^i$.
- Return $\bar{X} = \left(\bar{X}^1, \dots, \bar{X}^N\right)$.

Algorithm 2 Multinomial resampling based on an offspring vector. Input $w^{1:N}$ and $X^{1:N}$. Output $\bar{X}^{1:N}$.

```
• Draw an "offspring vector" $O^{1:N} = \left(O^1, \dots, O^N\right) \in \{0, \dots, N\}^N$ from a multinomial distribution

      $$O_t^{1:N} \sim \mathcal{M}ultinomial\left(N; w^1, \dots, w^N\right),$$

  in other words

      $$\forall i \in \{1, \dots, N\} \quad \mathbb{E}\left[O^i\right] = N \frac{w^i}{\sum_{j=1}^N w^j} \quad \text{and} \quad \sum_{i=1}^N O^i = N.$$

• Each particle $X^i$ is replicated $O^i$ times (possibly zero times) to create the sample $\bar{X}$:
    – $\bar{X} \leftarrow \{\}$
    – For $i = 1, \dots, N$
        ∗ For $k = 0, \dots, O_t^i$
            · $\bar{X} \leftarrow \{\bar{X}, X^i\}$
• Return $\bar{X} = \left(\bar{X}^1, \dots, \bar{X}^N\right)$.
```

The rationale of resampling is that particles with high weights tend to get replicated many times, whereas particles with low weights tend to get removed from the sample. The process is random, so that there is a chance of any particle getting removed, even with high weight; and there is a chance of any particle surviving, even with low weight (except if the weight is exactly zero). Resampling can also be seen as an operator acting on empirical probability measures: it takes a weighted measure $\pi^N(x) = \left(\sum w^j\right)^{-1} \sum w^i \delta_{X^i}(x)$ and returns an unweighted measure $\bar{\pi}^N(x) = N^{-1} \sum \delta_{\bar{X}^i}(x)$.

Of course, not any resampling scheme would lead to consistent estimators; the use of a categorical/multinomial distribution above was not completely arbitrary. In particular, the property $\mathbb{E}\left[O^i\right] = Nw^i / \sum w^j$, or equivalently, $\mathbb{P}\left[A^i = k\right] = w^k / \sum w^j$, is typically required to ensure that the resampled points $\left(\bar{X}^i\right)$ still consistently approximate the same distribution.

Indeed, denoting by $O^{1:N}$ the offspring vector obtained during resampling, we can write for any test function φ:

$$\frac{1}{N} \sum_{k=1}^N \varphi\left(\bar{X}^k\right) = \frac{1}{N} \sum_{k=1}^N O^k \varphi\left(X^k\right).$$

Thus

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{N} \sum_{k=1}^N \varphi\left(\bar{X}^k\right) \mid X^1, \dots, X^N\right] &= \mathbb{E}\left[\frac{1}{N} \sum_{k=1}^N O^k \varphi\left(X^k\right) \mid X^1, \dots, X^N\right] \\
&= \frac{1}{N} \sum_{k=1}^N \mathbb{E}\left[O^k\right] \varphi\left(X^k\right) \\
&= \sum_{k=1}^N \frac{w^k}{\sum_{j=1}^N w^j} \varphi\left(X^k\right)
\end{aligned}$$

so that a Monte Carlo estimator based on $\left(\bar{X}^i\right)$ has the same expectation as the one based on $\left(w^i, X^i\right)$. Using the variance decomposition formula

$$\begin{aligned}
\mathbb{V}\left[\frac{1}{N} \sum_{k=1}^N \varphi\left(\bar{X}^k\right)\right] &= \mathbb{V}\left[\mathbb{E}\left[\frac{1}{N} \sum_{k=1}^N \varphi\left(\bar{X}^k\right) \mid X^1, \dots, X^N\right]\right] + \mathbb{E}\left[\mathbb{V}\left[\frac{1}{N} \sum_{k=1}^N \varphi\left(\bar{X}^k\right) \mid X^1, \dots, X^N\right]\right] \\
&= \mathbb{V}\left[\sum_{k=1}^N \frac{w^k}{\sum_{j=1}^N w^j} \varphi\left(X^k\right)\right] + \mathbb{E}\left[\frac{1}{N}\left[\sum_{k=1}^N \frac{w^k}{\sum_{j=1}^N w^j} \varphi\left(X^k\right)^2 - \left(\sum_{k=1}^N \frac{w^k}{\sum_{j=1}^N w^j} \varphi\left(X^k\right)\right)^2\right]\right],
\end{aligned}$$

Algorithm 3 Sequential Monte Carlo / particle filter

```
At time $t = 1$
• Sample $X_1^i \sim q_1(\cdot)$.
• Compute the weights
      $$w_1^i = \frac{\mu(X_1^i) g(y_1 \mid X_1^i)}{q_1\left(X_1^i\right)}.$$
At time $t \geq 2$
• Resample $\left(w_{t-1}^i, X_{1:t-1}^i\right)$ to obtain $N$ equally-weighted particles $\left(N^{-1}, \overline{X}_{1:t-1}^i\right)$, e.g. using Algorithm 1.
• Sample $X_t^i \sim q_{t|t-1}(\cdot \mid \bar{X}_{1:t-1}^i)$ and define $X_{1:t}^i = \left(\bar{X}_{1:t-1}^i, X_t^i\right)$ (thus redefining $X_1^i, \ldots, X_{t-1}^i$).
• Compute the weights
      $$w_t^i = \frac{f\left(X_t^i \mid X_{t-1}^i\right) g\left(y_t \mid X_t^i\right)}{q_{t|t-1}(X_t^i \mid X_{t-1}^i)}.$$
```

see e.g. [4] for the last line, based on $\mathbb{V}\left[O^i\right] = NW^i\left(1 - W^i\right)$ where W^i refers to the normalized weight. Thus the variance associated with the resampled $\left(\bar{X}^i\right)$ is larger than the one associated with $\left(w^i, X^i\right)$. We can see that, on its own, the resampling step does not do anything useful. On the contrary, it "adds some noise" to the estimator. Various other resampling schemes have been proposed, mainly to reduce the variance of this extra noise; but they all add some noise. So why is resampling useful?

The resampling step is only useful for the subsequent steps. Intuitively, it allows to propagate only the most promising particles (i.e. the ones with high weights). Conversely, it allows to stop propagating particles with low weights, which only contribute little to the overal estimation. In the sequential framework, this is extremely useful since we have only a fixed computational budget "N" to allocate at each time step. Clearly, there is always the possibility that a particle with low weight at time t would have a high weight at time $t + 1$, in which case resampling would be wasteful. There are artificial models for which this is the case. In general, resampling can be seen as a way to provide stability in the future samples, at the cost of an increase in the Monte Carlo variance at the current time.

In terms of computational complexity, remember that Example 2.3 of Chapter 2 described how to sample from a discrete distribution, such as the categorical distribution in Algorithm 1. However, that method would result in a computational cost in $\mathcal{O}\left(N \log N\right)$. In fact, it is possible to sample from a multinomial distribution in only $\mathcal{O}\left(N\right)$ operations, by carefully generating N uniform variables in sorted order. Generally, resampling schemes have a cost in $\mathcal{O}(N)$, similarly to the other steps in the particle filters described below.

2 A Sequential Monte Carlo Algorithm

2.1 Generic algorithm

Let us go back to the problem of sequentially approximating $p\left(x_{1:t} \mid y_{1:t}\right)$ for all $t \geq 1$. Consider first an IS approximation $\pi_t^N\left(x_{1:t} \mid y_{1:t}\right)$ of the target distribution $p\left(x_{1:t} \mid y_{1:t}\right)$. This approximation is based on weighted samples $\left(w_t^i, X_{1:t}^i\right)_{i=1}^N$, where $X_{1:t}^i$ is drawn from some proposal $q_t\left(x_{1:t}\right)$ on $\mathbb{X}^t$, and $w_t^i \propto p(x_{1:t} \mid y_{1:t})/q_t(x_{1:t})$, for all $i \in \{1, \ldots, N\}$.

SMC methods (or particle filters) are a combination of SIS and resampling. Instead of directly propagating each $X_{1:t}^i$ to $X_{1:t+1}^i$ using a proposal distribution $q_{t+1|t}$ as in SIS, SMC first applies a resampling step to select N particles according to their weights. Thus a sample $\bar{X}_{1:t}^{1:N}$ is obtained, all with equal weight N^{-1}. Then these are propagated using the proposal $q_{t+1|t}$ to obtain $X_{1:t+1}^{1:N}$. The algorithm then reads as in Algorithm 3. It is also called particle filter (PF), Sequential Importance Resampling (SIR), or Sequential Importance Sampling with Resampling (SISR). Note that the weights w_t^i are used during the resampling step, and then are reset to N^{-1}, so that the next weights are simply proportional to the "incremental weights" ω_t^i described for the SIS algorithm.

At any time t, this algorithm provides two approximations of $p\left(x_{1:t}\middle| y_{1:t}\right)$. We obtain

$$\pi_t^N\left(x_{1:t}\right) = \sum_{i=1}^{N} \frac{w_t^i}{\sum_{j=1}^{N} w_t^j} \delta_{X_{1:t}^i}\left(x_{1:t}\right) \tag{1}$$

after the sampling step, yielding

$$I^N\left(\varphi_t\right) = \int \varphi_t(x_{1:t})\pi_t^N(x_{1:t})dx_{1:t} = \frac{\sum_{i=1}^{N} w_t^i \varphi_t(X_{1:t}^i)}{\sum_{i=1}^{N} w_t^i}$$

and

$$\bar{\pi}_t^N\left(x_{1:t}\right) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\overline{X}_{1:t}^i}\left(x_{1:t}\right) \tag{2}$$

after the resampling step, yielding

$$\bar{I}^N\left(\varphi_t\right) = \int \varphi_t(x_{1:t})\bar{\pi}_t^N(x_{1:t})dx_{1:t} = \frac{1}{N}\sum_{i=1}^{N} \varphi_t(\bar{X}_{1:t}^i).$$

The approximation (1) is to be preferred to (2), since resampling only adds some noise (as discussed above). Similarly to SIS, we also obtain an approximation of $p\left(y_t\middle| y_{1:t-1}\right)$ through

$$p^N\left(y_t\middle| y_{1:t-1}\right) = \frac{1}{N}\sum_{i=1}^{N} w_t^i.$$

Indeed, remember that

$$\begin{aligned} p\left(y_t\middle| y_{1:t-1}\right) &= \int \frac{f\left(x_t\middle| x_{t-1}\right) g\left(y_t\middle| x_t\right)}{q_{t|t-1}(x_t \mid x_{t-1})} \frac{p(x_{1:t-1} \mid y_{1:t-1})}{q_{t-1}(x_{1:t-1})} q_t\left(x_{1:t}\right) dx_{1:t-1}dx_t \\ &= \int w(x_{t-1}, x_t) q_{t|t-1}(x_t \mid x_{t-1}) \mathrm{p}(x_{1:t-1} \mid y_{1:t-1}) \mathrm{d}x_{1:t-1}\mathrm{d}x_t, \end{aligned}$$

therefore, with $X_{1:t}^i = \left(\bar{X}_{1:t-1}^i, X_t^i\right) \sim \bar{\pi}_{t-1}^N\left(x_{1:t-1}\right) q_{t|t-1}\left(x_t \mid x_{t-1}\right)$ as described in the algorithm, we have

$$p\left(y_t\middle| y_{1:t-1}\right) \approx \frac{1}{N}\sum_{i=1}^{N} w(\bar{X}_{t-1}^i, X_t^i) = \frac{1}{N}\sum_{i=1}^{N} w_t^i.$$

Because of the resampling steps, we cannot write explicitely the law of each trajectory $X_{1:t}$ generated at time t. In particular, it is *not* $q_1(x_1)\prod_{s=2}^{t} q_{s|s-1}(x_s \mid x_{s-1})$, as it was for SIS, nor exactly $p(x_{1:t} \mid y_{1:t})$. The resampling steps indeed introduce dependencies among the N paths, which makes the law of each path intractable. Proving law of large numbers and central limit theorems is therefore harder for particle methods than it was for SIS. Thankfully a rich literature exists and we will state some results at the end of the notes.

2.2 Choices of proposal

Similarly to SIS, we need to choose the proposal distributions q_1 and $q_{t|t-1}$ for all $t \geq 2$ in order to implement the method. As discussed for SIS, a standard choice is to use μ and f, i.e. the model distributions ("prior proposal"). This simplifies the form of the weight functions as follows:

$$\begin{aligned} w(x_1) &= \frac{\mu(x_1)g(y_1 \mid x_1)}{\mu\left(x_1\right)} = g(y_1 \mid x_1) \\ \forall t \geq 2 \quad w(x_{t-1}, x_t) &= \frac{f(x_t \mid x_{t-1}) g\left(y_t\middle| x_t\right)}{f(x_t \mid x_{t-1})} = g(y_t \mid x_t). \end{aligned}$$

Note that, compared to SIS, here the weights are reset to $1/N$ after each resampling step, so that the next weight is just equal to the incremental weight. This proposal performs a "blind" exploration of the state space: the particle X_t^i is drawn from $f(x_t \mid X_{t-1}^i)$, irrespective of the observation y_t.

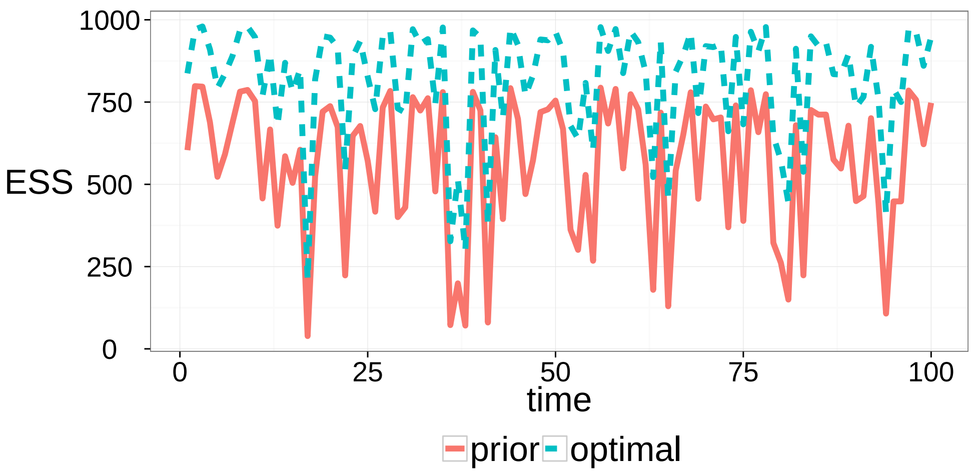


Figure 1: Evolution over time of the Effective Sample Size (ESS) using Sequential Monte Carlo, with the prior proposal and the optimal proposal. Here $N = 1000$, so the ESS is between 1 and 1000.

Another sensible approach consists in selecting a proposal $q_{t|t-1}\left(x_t | x_{t-1}\right)$ that minimizes the variance of the [incremental] weights $(w_t^i)_{i=1}^N$.

Proposition. The proposal $q_{t|t-1}\left(x_t | x_{t-1}\right)$ minimizing $\mathbb{V}_{q_t(x_{1:t})}\left(w_t\left(X_{1:t}\right)\right)$ is given by

$$q_{t|t-1}^{\text{opt}}\left(x_t | x_{t-1}\right) = \frac{f\left(x_t | x_{t-1}\right) g\left(y_t | x_t\right)}{p\left(y_t | x_{t-1}\right)} \tag{3}$$

and the associated incremental weight is given by

$$\omega_t^{\text{opt}}\left(x_{t-1}, x_t\right) = p\left(y_t | x_{t-1}\right).$$

The proof is the same as for SIS. In practice we will rarely be able to implement this locally optimal proposal, but it can provide a guideline to construct smart proposals.

3 A linear Gaussian example revisited

3.1 Filtering and marginal likelihood estimation

We revisit the simple linear Gaussian model described in Chapter 8:

$$\forall t \geq 1 \quad X_t = \phi X_{t-1} + \sigma_V V_t, \tag{4}$$

$$\forall t \geq 1 \quad Y_t = X_t + \sigma_V W_t, \tag{5}$$

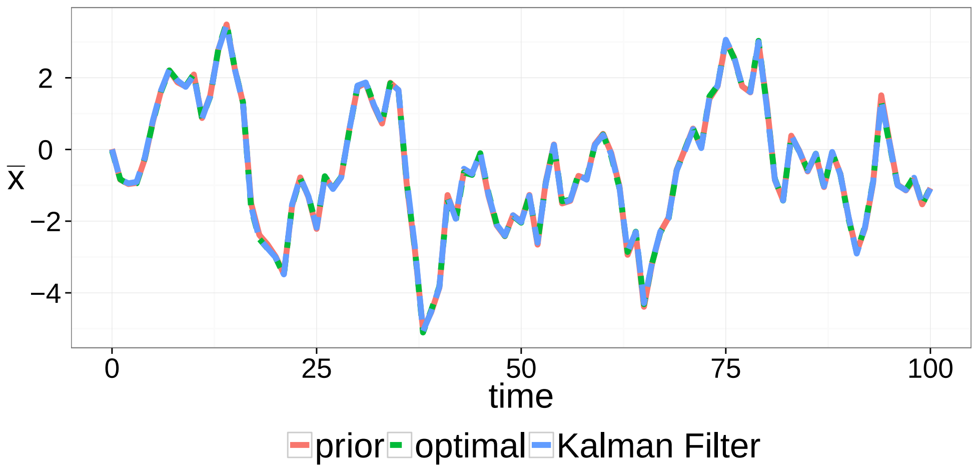
with $X_0 \sim \mathcal{N}\left(0,1\right)$, $V_t, W_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0,1\right)$, $\phi = 0.95$, $\sigma_V = 1$, $\sigma_W = 1$. We simulate $T = 100$ observations from this model.

We use again both the prior and the locally optimal proposals, but now we resample at each time step; i.e. we use SMC instead of SIS. We observe the evolution of the ESS over time when using the prior proposal and the locally optimal proposal within the SMC procedure based on $N = 1000$ particles, on Figure 1. We see that the ESS oscillates between large and small values, but has a stable behaviour over time; whereas the ESS associated with SIS simply dropped to 1 and stayed there.

Figure 2 shows the estimation results: SMC is used to estimate the filtering means $\mathbb{E}\left(x_t \mid y_{1:t}\right)$ and the filtering variances $\mathbb{V}\left(x_t \mid y_{1:t}\right)$, for all $t \geq 1$. The results are compared with the exact means and variances computed using the Kalman filter. We see that for both proposals, the mean is correctly estimated. The estimate of the variance is more noisy but seems to keep track with the real values as time progress.

Finally, we estimate the log likelihood $\log p(y_{1:t})$ for all t. The results are shown in Figure 3. We see that both proposals seem to give satisfactory results.

Thus particle filters/SMC seem to perform much better than SIS for the tasks of estimating expectations of filtering distributions $p(x_{1:t} \mid y_{1:t})$, and estimating the marginal likelihood $p(y_{1:t})$. The resampling steps



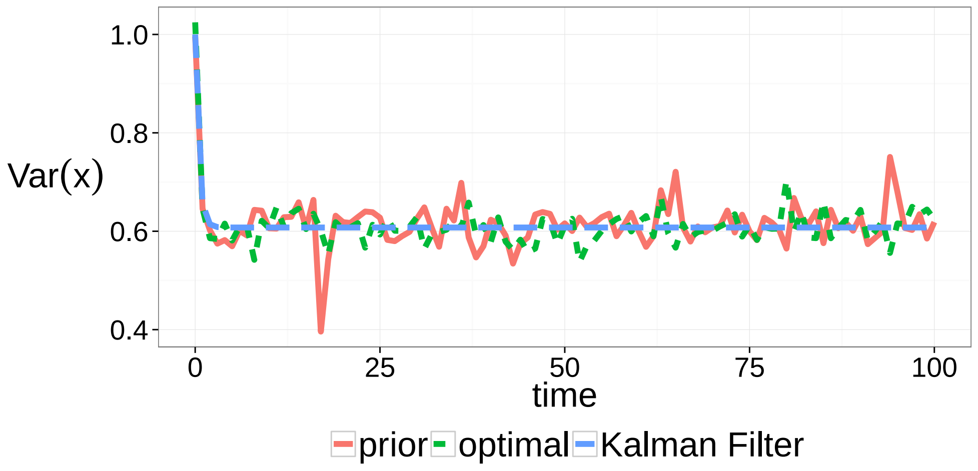


Figure 2: Estimation of the filtering means $\mathbb{E}\left(x_t \mid y_{1:t}\right)$ and the filtering variances $\mathbb{V}\left(x_t \mid y_{1:t}\right)$, using Sequential Monte Carlo, compared to the exact values calculated with the Kalman filter. Both proposals perform reasonably well. For the variances, the optimal proposal seems to result in slightly less noisy estimates.

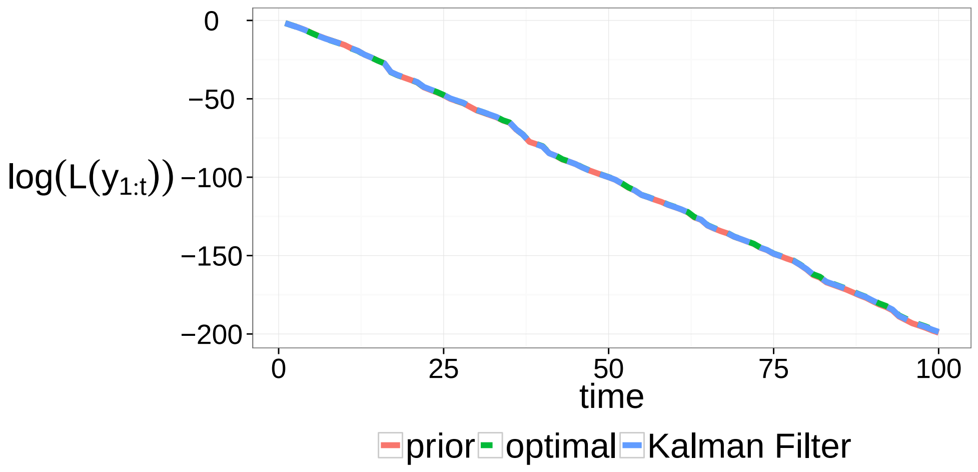


Figure 3: Estimation of the log likelihood $\log p(y_{1:t})$ using Sequential Monte Carlo, compared to the exact values computed with the Kalman filter.

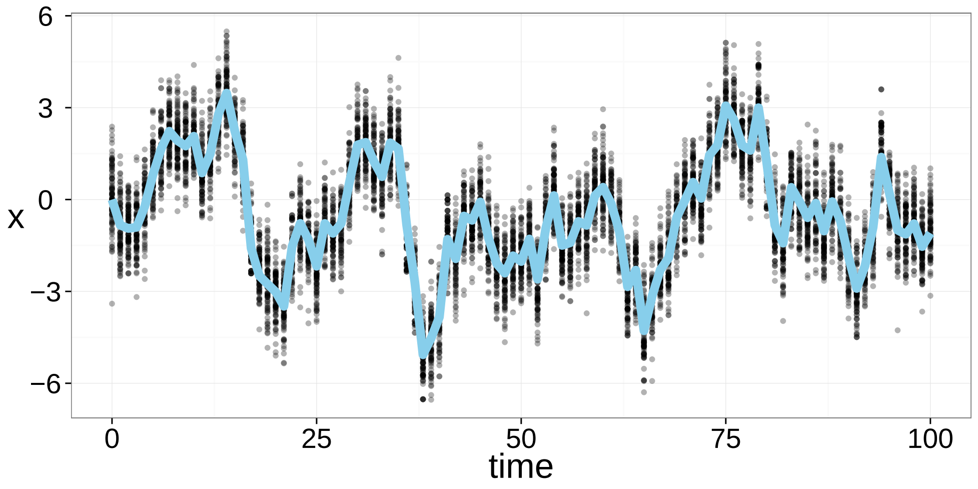


Figure 4: Cloud of points $\bar{X}_t^{1:N}$ after resampling, at each time step t. The filtering means calculated by Kalman filter are represented by the thick blue line.

select the fittest particles at each step, and thus there are always a diverse population of particles with significant weights at each step. On the contrary, SIS quickly resulted in a population where only one particle contributes to the estimates. To show the diversity of the samples, Figure 4 plots a black dot for each $\bar{X}_t^i$ obtained after resampling, at each step. We see that there are always many points (with equal weights $1/N$) in the approximation $\bar{\pi}_t^N$ of $p(x_t \mid y_{1:t})$.

4 Path degeneracy

Despite this good behaviour for filtering, the basic particle filter presented above does not solve all inference problems. Consider for instance the problem of estimating the path distribution $p(x_{1:t} \mid y_{1:t})$, and not only its last component $p(x_t \mid y_{1:t})$. Algorithm 3 provides the particle approximation π_t^N as in Eq. 1 or Eq. 2. As shown on Figure 4, the approximation is satisfactory for $p(x_t \mid y_{1:t})$. Let us plot the full paths $\left(\bar{X}_{1:T}^i\right)$ for $i \in \{1, \dots, N\}$ obtained at the final step T, on Figure 5. We see that most paths "coalesce" into a single trajectory: there are many different values for $\bar{X}_T^i$ at time T but only one value for $\bar{X}_s^i$ at most time steps in $s \in \{1, \dots, T-1\}$. As a result, the trajectories do a poor job at approximating globally the path (or "smoothing") distribution $p(x_{1:T} \mid y_{1:T})$ at time T. For instance, if we are interested in the marginal distribution $p(x_1 \mid y_{1:T})$, then the paths provide only one point $\bar{X}_1$. Clearly this is not good enough: for instance we cannot estimate the variance $\mathbb{V}\left(x_1 \mid y_{1:T}\right)$ at all.

This degeneracy comes from the very resampling steps that makes SMC so efficient for filtering: whenever resampling is performed, some of the paths are depleted. The paths are then extended using $q_{t|t-1}(x_t \mid x_{t-1})$, but nothing is done to regenerate the earlier components x_s for $s < t$. Thus, the number of unique values representing, say, x_1, is monotonically decreasing, from N at time 1, to 1 at some later time t. As can been seen from Figure 5, it does not take many steps for the number of unique values to be 1 or close to 1. This phenomenon is called the "path degeneracy" problem.

If the interest lies in the smoothing distribution, for instance $p(x_1 \mid y_{1:t})$, instead of the filtering distribution $p(x_t \mid y_{1:t})$, the path degeneracy phenomenon is a problem. It has been adressed specifically in the literature, and many algorithms have been proposed, usually called "particle smoother": the most popular are called forward filtering backward smoother, two-filter smoother and fixed-lag smoother (see [3]).

5 Likelihood estimation

So far, the model parameter θ has been assumed constant and SMC methods have been introduced to estimate the hidden process $X_{1:t}$ given the observations $y_{1:t}$. As discussed above, particle filters also provide

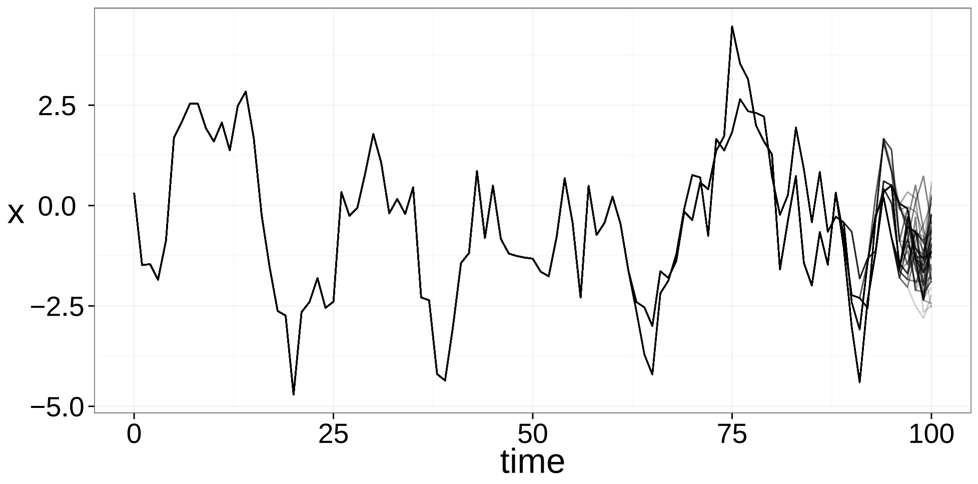


Figure 5: Cloud of paths $\bar{X}^{1:N}_{1:T}$ after resampling at the final step T. Even though there are $N = 100$ paths, most of them share the same components x_s for $s \ll T$.

an estimate of the marginal incremental likelihood:

$$p^N(y_1) = \frac{1}{N}\sum_{i=1}^{N} w_1^i \approx p(y_1)$$

$$\forall t \geq 2 \quad p^N\left(y_t \mid y_{1:t-1}\right) = \frac{1}{N}\sum_{i=1}^{N} w_t^i \approx p(y_t \mid y_{1:t-1}).$$

Multiplying them together, we obtain an estimator of the marginal likelihood:

$$p^N\left(y_{1:t}\right) = \prod_{s=1}^{T}\frac{1}{N}\sum_{i=1}^{N} w_s^i.$$

If we introduce the parameter θ again, then we write $\mu_\theta = \mu(\cdot \mid \theta) = \mu$, $f_\theta = f(\cdot \mid \cdot, \theta) = f$, $g_\theta = g(\cdot \mid \cdot, \theta) = g$, and the proposal distributions q_1 and $q_{t|t-1}$ could depend on θ too. The marginal likelihood is then written $p\left(y_{1:t} \mid \theta\right)$ and can be approximated by $p^N\left(y_{1:t} \mid \theta\right)$ as above. Figure 6 shows the estimated log likelihoods when ϕ varies, in the linear Gaussian example. We see that more particles yield more precision of the estimates.

These estimates can then be used to perform parameter estimation. In particular, "particle Markov chain Monte Carlo methods" have been recently developped, and use particle filter to estimate the parameters associated with hidden Markov models (see [5]).

6 Main theoretical results

Here, we briefly state some selected convergence results for SMC. Denoting by $I^N\left(\varphi_t\right)$ the SMC estimator of $I(\varphi_t) = \int \varphi_t(x_{1:t})p(x_{1:t} \mid y_{1:t})dx_{1:t}$. In general we can prove the following type of result, usually called "L_p" bound:

$$\mathbb{E}\left[\left|I^N\left(\varphi_t\right) - I\left(\varphi_t\right)\right|^p\right]^{1/p} \leq \frac{B(t)c(p)\left|\left|\varphi_t\right|\right|_\infty}{\sqrt{N}},$$

where $B(t)$ is a function of t, which is typically exponential in t as was illustrated by the path degeneracy problem. The term $c(p)$ depends only on p and $\left|\left|\varphi_t\right|\right|_\infty$ is the supremum of φ. The expectation is to be understood with respect to the particle filter, i.e. to all the random variables generated during the algorithm. A central limit theorem holds:

$$\sqrt{N}\left(I^N\left(\varphi_t\right) - I\left(\varphi_t\right)\right) \xrightarrow[N\to\infty]{\mathcal{D}} \mathcal{N}\left(0, \sigma_t^2\right),$$

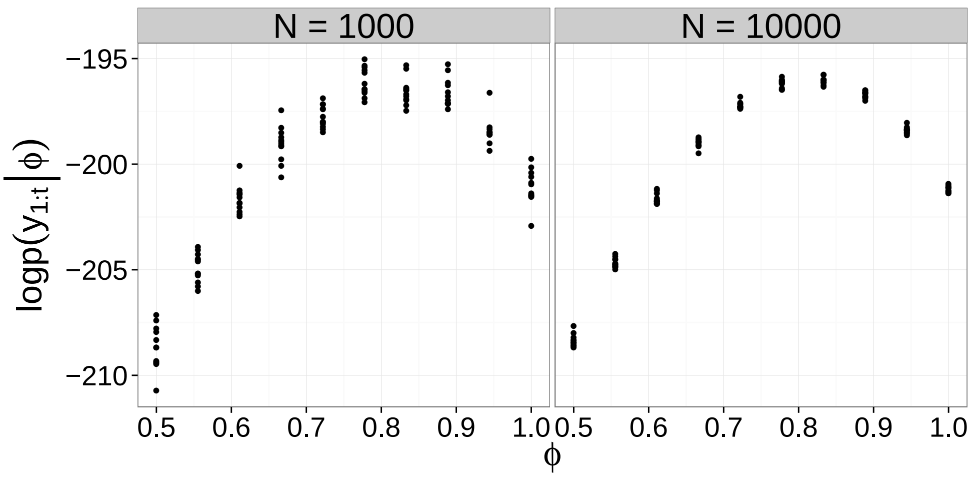


Figure 6: Log-likelihood $\log p(y_{1:t} \mid \phi)$ estimated by $\log p^N(y_{1:t} \mid \phi)$ as a function of ϕ for the linear Gaussian model. For each ϕ, 12 independent estimates are drawn. The left panel shows the results using $N = 1000$ particles, while the right panel shows the results for $N = 10000$ particles.

where again σ_t^2 grows exponentially fast with t. Hence, these are very weak results: if it was for them, particle filters would be useless.

When we look at the filtering distribution only, we obtain much stronger results. Define $\varphi_t(x_{1:t}) = \varphi_t(x_t)$, i.e. the test function depends only on the latest component x_t. Then we obtain

$$\mathbb{E}\left[\left|I^N(\varphi_t) - I(\varphi_t)\right|^p\right]^{1/p} \leq \frac{B_1 c(p)\,||\varphi_t||_\infty}{\sqrt{N}}$$
$$\sqrt{N}\left(I^N(\varphi_t) - I(\varphi_t)\right) \xrightarrow[N\to\infty]{\mathcal{D}} \mathcal{N}\left(0, \sigma_t^2\right),$$

where now, B_1 does not depend on t, and σ_t^2 can be uniformly bounded (over t) by some constant B_2. In other words, the asymptotic variance of the SMC filtering estimators is stable in time! This remarkable property is often called "time uniform stability" of particle filters. It means that a fixed number of particles N will be enough to keep track of the filtering distribution, as time t goes to infinity.

For the marginal likelihood $p(y_{1:t})$, the variance is typically increasing with t but linearly, and not exponentially. Some recent articles prove

$$\mathbb{E}\left(\left(\frac{p^N(y_{1:t})}{p(y_{1:t}} - 1\right)^2\right) \leq \frac{B_3 t}{N}$$

where B_3 is another constant; the result holds for all N and t, so it is non-asymptotic.

Another interesting property of the marginal likelihood estimator $p^N(y_{1:t})$ is that it is unbiased:

$$\mathbb{E}\left(p^N(y_{1:t})\right) = p(y_{1:t}),$$

as proven e.g. in [1], Chapter 7. This proves very useful for parameter estimation, as emphasized in [5]. Indeed, unbiased estimators of the likelihood can indeed be used to design MCMC algorithms that target exactly the posterior distribution, as if we had access to the likelihood.

7 Back to the general sampling

The efficiency of particle methods in hidden Markov models has led to consider similar methods for general models, outside time series analysis. Consider again the problem of sampling from a target distribution π, defined on a space $\mathbb{X}$, such that the probability density function $\pi(x)$ can be evaluated point-wise up to a normalizing constant. We can introduce an artificial sequence of target distributions Introduce intermediate distributions

$$\pi_0, \pi_1, \ldots, \pi_T$$

such that:

- π_0 is easy to sample from,
- π_t and π_{t+1} are not too different,
- $\pi_T = \pi$.

For example, in the case of a posterior distributioon $\pi(\theta) \propto p(\theta)p(y_{1:T} \mid \theta)$, we can introduce the partial posterior $\pi_t(\theta) \propto p(\theta)p(y_{1:t} \mid \theta)$. The first distribution is $\pi_0(\theta) = p(\theta)$, the prior distribution, and the last distribution $\pi_T(\theta) = \pi(\theta) = p(\theta \mid y_{1:T})$ is the full posterior distribution. We can expect $\pi_t(\theta)$ to be similar to $\pi_{t+1}(\theta)$, at least when t is large and when the data is i.i.d. Note that for i.i.d data, the ordering of $y_{1:T}$ should not matter, so any arbitrary permutation of the dataset $y_1, \ldots, y_T$ can used.

Another generic choice consists in introducing a simple parametric distribution q, and the sequence of distributions

$$\pi_t(x) \propto \pi(x)^{\gamma_t} q(x)^{1-\gamma_t}$$

where $0 = \gamma_0 < \gamma_1 < \ldots < \gamma_T = 1$. Then $\pi_0 = q$, $\pi_T = \pi$, and if γ_t is close to γ_{t+1}, then π_t is close to π_{t+1}.

Given a sequence of targets, one can apply the Sequential Importance Sampling idea. Here, each target is defined on the same space $\mathbb{X}$, so there is no need to "extend the space" as in hidden Markov models (remember, the sequence of targets was $p(x_{1:t} \mid y_{1:t})$, defined on $\mathbb{X}^t$). The procedure starts with $\theta^1, \ldots, \theta^N$ drawn from π_0, and then at any time $t \geq 1$

$$\begin{aligned} w_t^i &= \frac{\pi_t(\theta^i)}{\pi_0(\theta^i)} \\ &= \frac{\pi_{t-1}(\theta^i)}{\pi_0(\theta^i)} \frac{\pi_t(\theta^i)}{\pi_{t-1}(\theta^i)} = w_{t-1}^i \times \frac{\pi_t(\theta^i)}{\pi_{t-1}(\theta^i)}. \end{aligned}$$

If we simply keep updating the weights as above, then one of the particles generated from the initial distribution π_0 will end up having all the weight. As in particle filters, we can introduce resampling steps. If we only do that, then the resampled particles $\left(N^{-1}, \bar{\theta}^i\right)_{i=1}^N$ will have fewer unique values than the particles before resampling $(w_t^i, \theta^i)_{i=1}^N$. Thus resampling depletes the number of unique values in the population, and after a few steps the population contains only one unique value. This clearly implies that the approximation of the posterior distribution is poor.

Note that this is different from the particle filter setting: there, we had less unique values in $(N^{-1}, \bar{X}_t^i)$ than in $\left(w_t^i, X_t^i\right)$, but at the next step, we drew $X_{t+1}^i \sim q_{t+1|t}\left(\cdot \mid \bar{X}_t^i\right)$. Thus the particles get diversified again. The particle depletion effect was only damaging the path approximation $p^N\left(x_{1:t} \mid y_{1:t}\right)$, not the filtering approximation $p^N\left(x_t \mid y_{1:t}\right)$.

For static problems, we can introduce a similar rejuvation mechanism. Introduce a sequence of Markov kernels K_t, such that K_t leaves π_t invariant. For instance, K_t might correspond to a Metropolis–Hastings kernel targeting π_t, using some random walk proposal, or independent proposal. Then, after resampling, we might draw

$$\forall i \in \{1, \ldots, N\} \quad \theta_t^i \sim K_t(\bar{\theta}_t, d\theta).$$

The algorithm thus goes as follows: start with $\theta_0^1, \ldots, \theta_0^N$ drawn from π_0, and $w_0^i = N^{-1}$ for all i, and then at any time $t \geq 1$:

- Resample $\left(w_{t-1}^i, \theta_{t-1}^i\right)$ to obtain N equally-weighted particles $\left(N^{-1}, \bar{\theta}_{t-1}^i\right)$,
- Move the particle using a π_{t-1}-invariant Markov kernel K_{t-1}: $\forall i \in \{1, \ldots, N\} \quad \theta_t^i \sim K_{t-1}(\bar{\theta}_{t-1}, d\theta)$.
- [At this point $\left(N^{-1}, \theta_t^i\right)$ still approximates π_{t-1}.]
- Compute $w_t^i = \pi_t\left(\theta_t^i\right) / \pi_{t-1}\left(\theta_t^i\right)$.
- [At this point $\left(w_t^i, \theta_t^i\right)$ approximates π_t.]

Algorithm 4 Sequential Monte Carlo Sampler for static problems. $\tilde{N}$ refers to the effective sample size threshold.

```
At time $t = 0$
• Sample $\theta_0^i \sim \pi_0(\cdot)$ for $i \in \{1, \dots, N\}$.
• Compute the weights
      $\forall i \in \{1, \dots, N\} \quad w_0^i = \frac{1}{N}.$
At time $t \geq 1$
  • If ESS $< \tilde{N}$, then:
      – Resample $\left(w_{t-1}^i, \theta_{t-1}^i\right)$ to obtain $N$ equally-weighted particles $\left(N^{-1}, \bar{\theta}_{t-1}^i\right)$,
      – Reset the weights $w_{t-1}^i$ to $1/N$. At this point $\left(N^{-1}, \bar{\theta}_{t-1}^i\right)$ approximates $\pi_{t-1}$.
      – Draw for all $i$, $\theta_t^i \sim K_{t-1}(\bar{\theta}_{t-1}^i, d\theta)$, where $K_{t-1}$ is a Markov kernel leaving $\pi_{t-1}$ invariant. $(N^{-1}, \theta_t^i)$ still approximates $\pi_{t-1}$.
  • Compute the weights
      $\forall i \in \{1, \dots, N\} \quad w_t^i = w_{t-1}^i \times \frac{\pi_t(\theta_t^i)}{\pi_{t-1}(\theta_t^i)}.$
    The particles $\left(w_t^i, \theta_t^i\right)$ now approximate $\pi_t$.
```

This algorithm would perform well but would be expensive: each draw from K_t typically has a cost in $\mathcal{O}(t)$. For instance, if $\pi_t(\theta) \propto p(\theta)p(y_{1:t} \mid \theta)$, then a Metropolis–Hastings step targeting π_t involves computing the likelihood of t observations $y_{1:t}$. To save some computational time, one typically uses an "adaptive resampling scheme": that is, the resampling and move steps are not performed at every step t of the algorithm, but only when necessary. One criterion to decide whether it is necessary to perform this step relies on the effective sample size (ESS). If it gets too low (for instance, below $N/2$ or $N/10$), then a resample move step is performed. Denoting the ESS threshold by $\tilde{N}$, the full algorithm is described in Algorithm 4.

References

[1] Del Moral, P. (2004) *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications.* Series: Probability and Applications, Springer-Verlag, New York.

[2] Doucet, A., de Freitas, N. and Gordon, N.J. (eds.) (2001) *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, New York.

[3] Doucet, A. and Johansen, A.M. (2011), A tutorial on particle filtering and smoothing: fifteen years later. In *Handbook of Nonlinear Filtering*, Cambridge University Press.

[4] Douc, R., Cappé, O. and Moulines, E. (2005), Comparison of Resampling Schemes for Particle Filtering. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis.

[5] Andrieu, C., Doucet, A. and Holenstein, R. Particle markov chain monte carlo methods. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 72.3 (2010): 269-342.