A Tutorial on Particle Filters

Maarten Speekenbrink*

Experimental Psychology
University College London

Abstract

This tutorial aims to provide an accessible introduction to particle filters, and sequential Monte Carlo (SMC) more generally. These techniques allow for Bayesian inference in complex dynamic state-space models and have become increasingly popular over the last decades. The basic building blocks of SMC – sequential importance sampling and resampling – are discussed in detail with illustrative examples. A final example presents a particle filter for estimating time-varying learning rates in a probabilistic category learning task.

Keywords: Particle filter, Sequential Monte Carlo, State-space model, Sequential Bayesian inference

Particle filters, and sequential Monte Carlo (SMC) techniques more generally, are a class of simulation-based techniques which have become increasingly popular over the last decades to perform Bayesian inference in complex dynamic statistical models (e.g., Doucet, de Freitas, and Gordon, 2001b; Doucet and Johansen, 2011). Particle filters are generally applied

*Department of Experimental Psychology, University College London, Gower Street, London WC1E 6BT, England.

Email address: m.speekenbrink@ucl.ac.uk (Maarten Speekenbrink)

URL: www.ucl.ac.uk/speekenbrink-lab/ (Maarten Speekenbrink)

to so-called filtering problems, where the objective is to estimate the latent states of a stochastic process on-line, such that, after each sequential observation, the state giving rise to that observation is estimated. For instance, in a category learning task, we might want to infer how people use the features of objects to categorize them. Due to learning, we would expect their categorization strategy to change over time. Traditionally, a formal learning model such as ALCOVE (Kruschke, 1992) would be used for this purpose, which describes how feedback on their categorization decisions affects people's momentary strategy. However, these models usually assume a deterministic updating process, which may be too restrictive. Ideally, we would like to estimate someone's strategy – which we can view as the latent state of their decision process – from trial to trial whilst allowing for stochastic transitions between states. Estimating the current categorization strategy is a difficult task, however, as a single categorization decision at each point in time provides relatively little information about people's complete categorization strategy, i.e. their potential categorizations of all possible stimuli. Assuming trial-to-trial changes to a state (strategy) are noisy but relatively small, we may however be able to gain some insight into the current state from all previous categorization decisions someone made. This filtering problem is generally not analytically tractable; analytical results are only available for the restricted class of linear Gaussian state-space models. As particle filters are applicable to the much broader class of non-linear non-Gaussian state-space models, they open up interesting possibilities to study a broad range of dynamic processes in psychology.

A graphical representation of a generic particle filter (see Section 4.3) is given in Figure 1. Particle filters operate on a set of randomly sampled values of a latent state or unknown parameter. The sampled values, generally

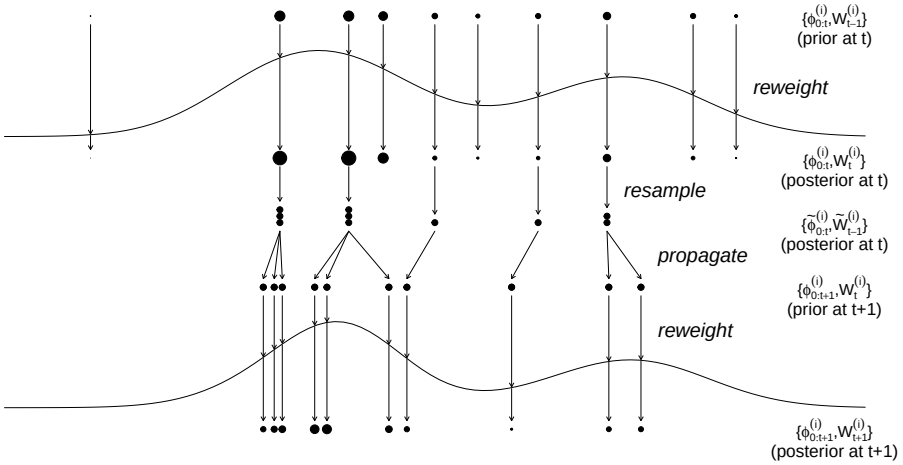

Figure 1: Schematic representation of a generic particle filter (after Doucet et al., 2001a). Standing at time t, we have a set of weighted particles $\{\phi_{0:t}^{(i)}, W_{t-1}^{(i)}\}$ representing the prior distribution at t. Each particle $\phi_{0:t}^{(i)}$ is a multidimensional variable which represents the whole path of the latent state from time 0 up to the current time point t, such that each dimension represents the value of the state at a particular time point. The location of the dots in the graph reflect $\phi_t^{(i)}$, the value of the state at the current time point, i.e. the dimension of each particle reflecting the current state. The size of each dot reflects the weight $W_{t-1}^{(i)}$ ("prior at t"). In the *reweight* step, the weights are updated to $W_t^{(i)}$ partly as a function of $p(y_t|\phi_t^{(i)})$, the likelihood of observation y_t according to each sampled state value $\phi_t^{(i)}$ (solid line). The resulting set $\{\phi_{0:t}^{(i)}, W_t^{(i)}\}$ of weighted particles approximates the posterior distribution ("posterior at t") of the latent state paths. The *resampling* step duplicates values $\phi_{0:t}^{(i)}$ with high weights $W_t^{(i)}$, and eliminates those with low weights, resulting in the set of uniformly weighted particles $\{\tilde{\phi}_{0:t}^{(i)}, \tilde{W}_t^{(i)} = 1/N\}$ which is approximately distributed according to the posterior (second "posterior at t"). In the *propagate* step, values of states $\phi_{t+1}^{(i)}$ at the next time point are sampled and added to each particle to account for state transitions, forming a prior distribution for time $t+1$ ("prior at t+1"). Thus, at each new time point, the particles grow in dimension because the whole path of the latent state now incorporates the new time point as well. The particles are then reweighted in response to the likelihood of the new observation y_{t+1} to approximate the posterior distribution at $t+1$ ("posterior at t+1"), etc.

referred to as "particles", are propagated over time to track the posterior distribution of the state or parameter at each point in time. Each particle is assigned a weight in relation to its posterior probability. To increase their accuracy, SMC techniques resample useful particles from the set according to these weights. This resampling introduces interaction between the particles, and the term "interacting particle filters" was coined by Del Moral (1996), who showed how the method relates to techniques used in physics to analyse the movement of particles.

Particle filters have successfully solved difficult problems in machine learning, such as allowing robots to simultaneously map their environment and localize their position within it (Montemerlo, Thrun, Koller, and Wegbreit, 2002), and the automated tracking of multiple objects in naturalistic videos (Isard and Blake, 1998; Nummiaro, Koller-Meier, and Gool, 2003). More recently, particle filters have also been proposed as models of human cognition, for instance how people learn to categorize objects (Sanborn, Griffiths, and Navarro, 2010), how they detect and predict changes (Brown and Steyvers, 2009) as well as make decisions (Yi, Steyvers, and Lee, 2009) in changing environments.

The aim of this tutorial is to provide readers with an accessible introduction to particle filters and SMC. We will discuss the foundations of SMC, sequential importance sampling and resampling, in detail, using simple examples to highlight important aspects of these techniques. We start with a discussion of importance sampling, which is a Monte Carlo integration technique which can be used to efficiently compute expected values of random variables, including expectations regarding the posterior probabilities of latent states or parameters. We will then move on to sequential importance sampling, an extension of importance sampling which allows for efficient

computation in sequential inference problems. After introducing resampling as a means to overcome some problems in sequential importance sampling, we have all the ingredients to introduce a generic particle filter. After discussing limitations and extensions of SMC, we will conclude with a more complex example involving the estimation of time-varying learning rates in a probabilistic category learning task.

1. Importance sampling

Importance Sampling (IS) is a Monte Carlo integration technique. It can be used to efficiently solve high-dimensional integration problems when analytical solutions are difficult or unobtainable. In statistics, it is often used to approximate expected values of random variables, which is what we will focus on here. If we have a sample of realizations of a random variable Y, we can estimate the expected value by computing a sample average. We do this when we have data from experiments, and it is also the idea behind basic Monte Carlo integration. Importance sampling is based on the same idea, but rather than sampling values from the true distribution of Y, values are sampled from a different distribution, called the importance distribution. Sampling from a different distribution can be useful to focus more directly on the estimation problem at hand, or if it is problematic to sample from the target distribution. To correct for the fact that the samples were drawn from the importance distribution and not the target distribution, weights are assigned to the sampled values which reflect the difference between the importance and target distribution. The final estimate is then a weighted average of the randomly sampled values.

Suppose we wish to compute the expected value of an arbitrary function

f of a random variable Y which is distributed according to a probability distribution p:

$$\mathbb{E}_p[f(Y)] \triangleq \int f(y)p(y)\,\mathrm{d}y.$$

This is just the usual definition of an expected value (we use $\mathbb{E}_p$ to denote an expectation of a random variable with distribution p, and the symbol $\triangleq$ to denote 'is defined as'). The function f depends on what we want to compute. For instance, choosing $f(y) = y$ would result in computing the mean of Y, while choosing $f(y) = (y - \mathbb{E}_p[f(Y)])^2$ would result in computing the variance of Y. It is often not possible to find an analytical solution to the integral above, in which case we have to turn to some form of numerical approximation. A basic Monte Carlo approximation is to draw a number of independent samples from p and then compute a sample average from these random draws:

Algorithm 1. Basic Monte Carlo integration for an expected value $\mathbb{E}_p[f(Y)]$

1. (Sample) For $i = 1, \ldots, N$, sample $y^{(i)} \sim p(y)$.
2. (Estimate) Compute the sample average to obtain the Monte Carlo estimate E^{MC} of the expected value:

$$E^{\mathrm{MC}} \triangleq \frac{1}{N}\sum_{i=1}^{N} f(y^{(i)}). \tag{1}$$

We let $y^{(i)}$ denote the i-th sampled value and for consistency in terminology, we will refer to these sampled values as "particles" from now on. By the law of large numbers, as the number N of particles approaches infinity, this estimate will converge almost surely[1] to the true value (Robert and Casella,

[1] Almost sure convergence means that the probability that the estimate is identical to

2004). A limitation of this procedure is that we need to be able to sample particles according to the distribution p, which is not always possible or efficient. Importance sampling circumvents this limitation, allowing particles to be drawn from an arbitrary "instrumental distribution" q. These particles are then weighted to correct for the fact they were drawn from q and not the target distribution p. Importance sampling relies on the simple algebraic identity $a = \frac{a}{b} \times b$ to derive the following *importance sampling fundamental identity* (Robert and Casella, 2004):

$$\mathbb{E}_p[f(Y)] = \int \frac{p(y)}{q(y)} q(y) f(y)\, \mathrm{d}y = \mathbb{E}_q[w(Y)f(Y)],$$

where we define the importance weight as $w(y) = \frac{p(y)}{q(y)}$. Thus, the expected value of $f(Y)$ under the target distribution p is identical to the expected value of the product $w(Y)f(Y)$ under the instrumental distribution q. The instrumental distribution can be chosen for ease of sampling, or to increase the efficiency of the estimate (as shown in the example below). The only restriction on q is that, in the range where $f(y) \neq 0$, q should have the same support as p (i.e. whenever p assigns non-zero probability to a value y, q should do so also, so $q(y) > 0$ whenever $p(y) > 0$). More compactly, we can state this requirement as: if $f(y)p(y) \neq 0$, then $q(y) > 0$. An IS estimate of the expected value of $f(Y)$ under p is thus obtained by generating a sample from q and computing a weighted average, as in the following algorithm:

Algorithm 2. Importance sampling for an expected value $\mathbb{E}_p[f(Y)]$

1. (Sample) For $i = 1, \ldots, N$, sample $y^{(i)} \sim q(y)$.

---

the true value approaches 1 as N approaches infinity, i.e., $p(\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(y^{(i)}) = \mathbb{E}_p(f(Y))) = 1$.

2. (Weight) For $i = 1, \ldots, N$, compute the importance weight $w^{(i)} = \frac{p(y^{(i)})}{q(y^{(i)})}$.

3. (Estimate) Compute a weighted average to obtain the IS estimate:

$$E^{\mathrm{IS}} \triangleq \frac{1}{N} \sum_{i=1}^{N} w^{(i)} f(y^{(i)}). \tag{2}$$

As for basic Monte Carlo estimation, the law of large numbers assures that E^{IS} convergences to $\mathbb{E}_p[f(Y)]$ as the number of particles approaches infinity (Robert and Casella, 2004).

It should be stressed that IS is a Monte Carlo integration method, which can be used to approximate expected values of random variables. It is not a method to directly generate samples according to the target distribution. However, we can generate samples which are approximately distributed according to the target distribution by resampling particles with replacement from the set of particles, where we sample a particle $y^{(i)}$ with a probability proportional to the importance weight $w(y^{(i)})$. This importance sampling resampling algorithm, which will be discussed in more detail later, can provide an "empirical" approximation to the distribution p (in the sense that we use a finite random sample drawn from p to approximate p, just like a histogram of observations from an experiment approximates the distribution of possible observations that could be made in that experiment). We can also use IS to compute any probability within the distribution p. For instance, if we want to compute the probability that the value of Y is between a and b, we can use IS with the indicator function $f(y) = \mathbb{I}(a \geq y \geq b)$, where the indicator function $\mathbb{I}$ equals 1 when its argument is true and 0 otherwise. We can do this as it is easy to show that the required probability equals the expected value of this indicator function: $p(a \geq Y \geq b) = \mathbb{E}_p[\mathbb{I}(a \geq y \geq b)]$. In practice, the estimated probability is then simply the sum of the impor-

tance weights of the particles that lie between a and b. This is illustrated in Figure 2. A few remarks are in order. Firstly, while the estimated probabilities are unbiased, in practice, we can only estimate the probability if at least one particle falls within the interval. Secondly, given a set of particles, we can vary the bounds of the interval in between the particles and we will obtain the same estimates, because if two regions capture the same subset of particles, the sum of the weights of those particles will also be identical. Finally, to obtain a precise estimate of a probability, it is wise to tailor the importance distribution to sample solely in the required region, as will be shown in the following example.

1.1. Example: computing the tail probability of the Ex-Gaussian distribution

The ex-Gaussian distribution is a popular distribution to model response times (Van Zandt, 2000). The ex-Gaussian distribution is defined as the sum of an exponential and normal (Gaussian) distributed variable, and has three parameters: μ, σ, and τ, which are respectively the mean and standard deviation of the Gaussian variable, and the rate of the exponential variable. See Figure 3 for an example of an ex-Gaussian distribution.

Suppose that for a certain person the distribution of completion times for a task approximately follows an ex-Gaussian distribution with parameters $\mu = 0.4$, $\sigma = 0.1$ and $\tau = 0.5$, and that we want to know on how many trials that person would fail to complete the task within a time limit of 3 seconds. Looking at Figure 3, we can already see that the probability of non-completion is rather small. As the ex-Gaussian distribution is relatively easy to draw samples from, we can use basic Monte Carlo integration (Algorithm 1) to approximate this probability. With a sample size of $N = 2000$, this gave an estimate $p(Y \geq 3) \approx 0.0040$. Compared to the true value,

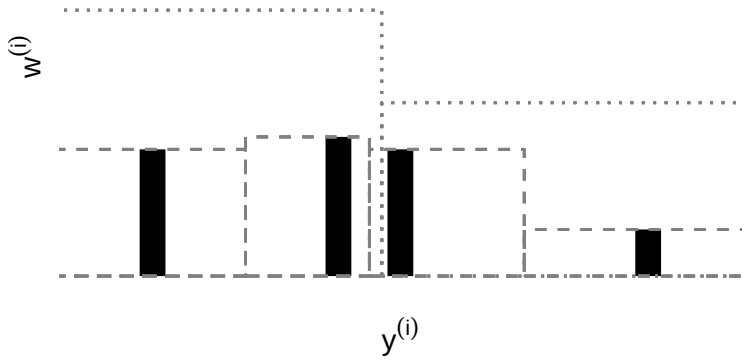

Figure 2: Probability estimation with importance sampling. The locations of the black bars represent particle values ($y^{(i)}$) and the height of the black bars represents the corresponding weights ($w^{(i)}$). The broken and dotted lines represent two different estimates of probabilities from these particles. The broken lines involve smaller regions which each include a single particle, while the dotted lines involve larger regions which include multiple particles. Small changes to the bounds of the regions would leave the estimates unchanged as long as the same particles fall within each region.

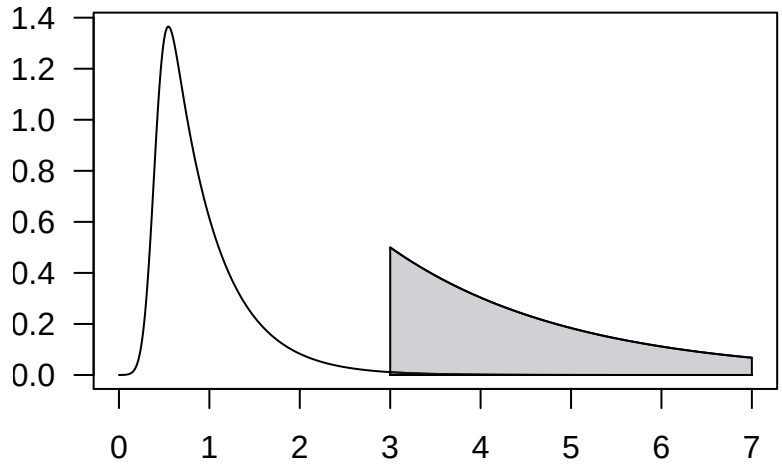

Figure 3: Tail probability estimation for an ex-Gaussian distribution through importance sampling with a shifted Exponential distribution. The solid line with an unshaded region below it reflects the ex-Gaussian distribution. Overlaid and shaded grey is the shifted Exponential distribution which is used as importance distribution.

$p(Y \geq 3) = 0.0056$, this estimate is too low by 28.93%. Basic Monte Carlo integration fails here because the exceedance probability $p(Y \geq 3)$ is relatively small and we therefore need many samples to obtain an estimate with adequate precision. Because the importance distribution can be tailored to the estimation problem at hand, IS can be much more efficient. To apply IS, we first formulate the desired result as the expected value

$$p(Y \geq 3) = \mathbb{E}_p[\mathbb{I}(y \geq 3)].$$

We then need to choose an importance distribution. Recall that the only requirement is that the instrumental distribution q has the same support as the target distribution p in the range where $f(y) \neq 0$. For this example, q need thus only be defined over the range $[3; \infty)$. In fact, choosing an importance distribution which does not extend beyond this range is a good idea, because samples outside this range are wasteful as they will not affect the estimate. A reasonable choice is a shifted exponential distribution, shifted to the right to start at 3 rather than 0. With a sample of $N = 2000$ and matching $\tau = 0.5$ to the same value as in the ex-Gaussian distribution, this gives the estimate $p(Y \geq 3) \approx 0.0055$, which deviates from the true value by only 2.63%. This is a representative example and shows the IS estimator is much better than the basic Monte Carlo estimator. While using an importance distribution defined over the range $[3, \infty)$, such as the shifted exponential, is a good way to increase the precision of the estimate, we must be careful when choosing the importance distribution. For example, a Normal distribution truncated below at 3 with parameters $\mu = 3$ and $\sigma = 0.1$, resulted in the estimate $P(Y \geq 3) \approx 0.0036$, which is too low by 35.55% and worse than the basic Monte Carlo estimate. The problem with this truncated Normal is that the parameter $\sigma = 0.1$ is set too low, resulting in a

distribution with a right tail which is too light compared to the ex-Gaussian distribution. Recall that the importance weights are given by the ratio $\frac{p(y)}{q(y)}$. If the instrumental distribution has lighter tails than the target distribution, there will be relatively few particles that fall in the tails, but for these rare particles, $p(y)$ may be very large compared to $q(y)$, leading to very large weights. In the extreme case, when one or a few importance weights are very large compared to the other weights, the estimate is effectively determined by only one or a few particles, which is obviously bad. For example, in the most extreme estimate resulting from the truncated Normal, there was one particle with a weight of 88.4, while the next highest weight was 0.6. For comparison, in the most extreme estimate from the shifted Exponential distribution, the largest and second-largest importance weights were both 0.02. Large variation in importance weights results in an estimator with a high variance. This can be clearly seen in Figure 4, which shows the variation in the estimates of the three estimators when applying them repeatedly for 1000 times. While the distribution of the estimates obtained for IS with a shifted exponential distribution is tightly clustered around the true value, both basic Monte Carlo integration and IS with the truncated Normal provide much more variable estimates. Note that these estimates are still unbiased, in the sense that on average, they are equal to the true value. However, the large variance of the estimates means that in practice, we are often quite far off the true value. The positive skew for the truncated Normal shows that IS with this importance distribution underestimates the probability most of the time, but in the rare cases that a particle falls in the right tail, the high weight assigned to that particle results in a large overestimation of the probability. Note that there is no problem with using a truncated Normal per se: increasing the parameter to $\sigma = 1$ results in a

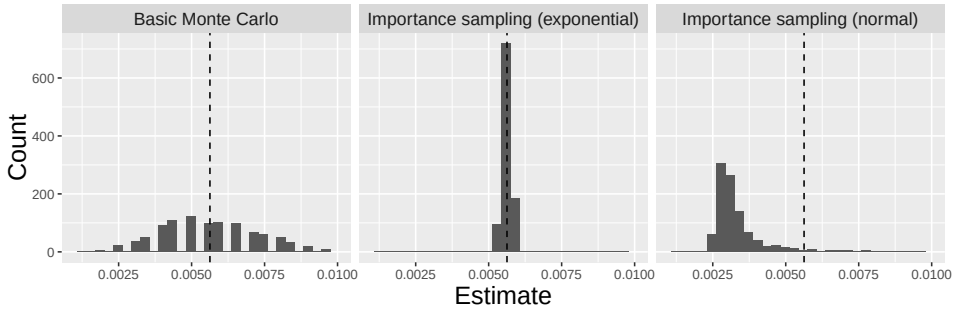

Figure 4: Distribution of estimates of the tail probability for an Ex-Gaussian distribution using basic Monte Carlo integration, importance sampling with a shifted exponential distribution, and importance sampling with a truncated Normal distribution. The dotted lines show the true value of the tail probability. Each estimator used $N = 2000$ particles and distributions were obtained by applying each estimator repeatedly for 1000 times.

heavier-tailed importance distribution which gives much better results.

1.2. Efficiency

As the previous example illustrates, some importance distributions are better than others. In the extreme case, a bad choice of importance distribution can result in an estimator with an infinite variance. The optimal importance distribution q^*, in terms of minimizing the variance of the estimator, is

$$q^*(y) = \frac{|f(y)|p(y)}{\int |f(y)|p(y)\,\mathrm{d}y} \tag{3}$$

(Kahn and Marshall, 1953). For example, the optimal importance distribution in the previous example is a truncated ex-Gaussian. The optimal importance distribution is mainly of theoretical interest; to be able to use it, we would have to know the value of the integral $\int |f(z)|p(z)\,\mathrm{d}z$, which is pretty much the quantity we want to estimate in the first place. Nevertheless, we should aim to use an importance distribution which is as close to

this distribution as possible.

A more practical way to reduce the variance of the estimator is to normalize the importance weights so they sum to 1 (Casella and Robert, 1998). Using normalized weights

$$W^{(i)} \triangleq \frac{w^{(i)}}{\sum_{j=1}^{N} w^{(j)}} \tag{4}$$

results in the "self-normalised" IS estimator

$$E^{\mathrm{ISn}} \triangleq \sum_{i=1}^{N} W^{(i)} f(y^{(i)}). \tag{5}$$

It should be noted that the estimator E^{ISn} is biased, but the bias is generally small, diminishes as the number of particles increases, and is often offset by the gain in efficiency (the reduction in the variance of the estimator).[2] Self-normalized weights are particularly useful in situations where the distribution p is only known up to a normalizing constant. For instance, we may be interested in the conditional distribution $p(y|x) = \frac{p(x,y)}{\int p(x,y)\,\mathrm{d}y}$, but although we can compute $p(x, y)$, the marginal distribution $p(x) = \int p(x, y)\,\mathrm{d}y$ is intractable. In that case, we can still use importance sampling, as the normalizing constant cancels out in the computation of the self-normalized

[2] One reason why the variance of the self-normalized IS estimate can be smaller is that the expected value of each weight is

$$\mathbb{E}_q[w(Y)] = \int \frac{p(y)}{q(y)} q(y)\,\mathrm{d}y = \int p(y)\,\mathrm{d}y = 1,$$

and the expected value of the sum of the weights is thus N. In practice, the summed importance weights will deviate from this value. Using self-normalizing weights ensures that the sum of the weights is always equal to 1 (not N, but we have accounted for this by removing the $1/N$ term in Equation 5), which removes one source of variance.

weights[3]. So, when using self-normalized weights, the target distribution only has to be known up to the normalizing constant.

2. Sequential importance sampling and online Bayesian inference

We often need to infer unknown parameters of a statistical model sequentially after each new observation comes in. Such online inference is crucial in a wide range of situations, including adaptive design of experiments (e.g., Amzal, Bois, Parent, and Robert, 2006; Myung, Cavagnaro, and Pitt, 2013) and real-time fault diagnosis in nuclear power plants. From a Bayesian viewpoint, this means we want to compute a sequence of posterior distributions $p(\theta|y_1), p(\theta|y_{1:2}), \ldots, p(\theta|y_{1:t})$, where $y_{1:t} = (y_1, y_2, \ldots, y_t)$ denotes a sequence of observations, and θ a vector of parameters. To approximate such a posterior distribution $p(\theta|y_{1:t})$ with importance sampling, we need an importance distribution $q_t(\theta)$ to generate an importance sample of particles, and compute the importance weights

$$w_t^{(i)} = \frac{p(\theta^{(i)}|y_{1:t})}{q_t(\theta^{(i)})}.$$

While we could generate a fresh importance sample at each point in time, this will usually increase the computational burden at each consecutive time point, as at each time we would have to browse the whole history of observations to compute the importance weights. Moreover, when tracking an evolving latent state over time, we would also have to generate larger and

[3]This is shown as

$$\sum_{i=1}^{N} \frac{\frac{p(y^{(i)}|x)}{q(y^{(i)})} f(y^{(i)})}{\sum_{j=1}^{N} \frac{p(y^{(i)}|x)}{q(y^{(i)})}} = \sum_{i=1}^{N} \frac{\frac{1}{p(x)} \frac{p(x,y^{(i)})}{q(y^{(i)})} f(y^{(i)})}{\frac{1}{p(x)} \sum_{j=1}^{N} \frac{p(x,y^{(i)})}{q(y^{(i)})}} = \sum_{j=1}^{N} \frac{\frac{p(x,y^{(i)})}{q(y^{(i)})} f(y^{(i)})}{\sum_{j=1}^{N} \frac{p(x,y^{(i)})}{q(y^{(i)})}}$$

larger importance samples as, at each time point, we would have to sample the whole trajectory of the latent state thus far. For real-time applications, it is important to devise an algorithm with an approximately fixed computational cost at each time point. Sequential importance sampling (SIS) serves this purpose. In addition, by using information from previous observations and samples, SIS can provide more efficient importance distributions than a straightforward application of IS.

A key idea in SIS is to compute the importance weights incrementally, by multiplying the importance weight at the previous time $t-1$ by an incremental weight update $a_t^{(i)}$. It is always possible to formulate the importance weights in such a form by trivially rewriting the importance weights above as

$$\begin{aligned} w_t^{(i)} &= \frac{p(\theta^{(i)}|y_{1:t})q_{t-1}(\theta^{(i)})}{p(\theta^{(i)}|y_{1:t-1})q_t(\theta^{(i)})} \frac{p(\theta^{(i)}|y_{1:t-1})}{q_{t-1}(\theta^{(i)})} \\ &= a_t^{(i)} w_{t-1}^{(i)}, \end{aligned} \tag{6}$$

where we define the incremental weight update as

$$a_t^{(i)} \triangleq \frac{p(\theta^{(i)}|y_{1:t})}{p(\theta^{(i)}|y_{1:t-1})} \times \frac{q_{t-1}(\theta^{(i)})}{q_t(\theta^{(i)})}. \tag{7}$$

This is of course not immediately helpful, as we still need to compute $p(\theta^{(i)}|y_{1:t})$ and $q_t(\theta^{(i)})$ in full at each time point. However, there are some important cases where we can simplify the incremental weight update further. In this section, we will focus on one such case, involving on-line inference of time-invariant parameters. This will help to illustrate the basics of SIS and its main shortcoming. We will see that while sequential importance sampling is computationally efficient, allowing one to approximate the distributions of interest sequentially without having to revisit all previous observations or completely redraw the whole importance sample, the performance of SIS

degrades over time because after a large number of iterations, all but one particle will have negligible weight. After introducing resampling as a way to overcome this problem of "weight degeneracy", we then return to a second important application of SIS, namely the inference of latent states in state-space models. Combining SIS with resampling provides us with a recipe for a particle filter which is highly effective for these applications.

2.1. Sequential importance sampling for time-invariant parameters

We will now focus on the problem of computing a sequence of posterior distributions $p(\theta|y_{1:t})$, $t = 1, \ldots, T$ for a vector of unknown parameters θ. Assuming the observations are conditionally independent given the parameters, we can express each of these posteriors through Bayes' theorem as

$$p(\theta|y_{1:t}) = \frac{p(\theta)\prod_{i=1}^{t} p(y_i|\theta)}{p(y_{1:t})}.$$

In this case, the left-hand ratio in (7) simplifies to

$$\frac{p(\theta^{(i)}|y_{1:t})}{p(\theta^{(i)}|y_{1:t-1})} = p(y_t|\theta^{(i)})p(y_t|y_{1:t-1})$$

If we also use a single importance distribution $q_t(\theta) = q_{t-1}(\theta) = q(\theta)$ to approximate each posterior, the right-hand ratio in (7) evaluates to $\frac{q_{t-1}(\theta^{(i)})}{q_t(\theta^{(i)})} = 1$, so that the incremental weight update is simply $a_t^{(i)} = p(y_t|\theta^{(i)})p(y_t|y_{1:t-1})$. Using self-normalized importance weights, we can ignore the $p(y_t|y_{1:t-1})$ term, resulting in a simple importance sampling scheme where we sequentially update the weights of an initial importance sample in light of each new observation:

Algorithm 3. SIS for time-invariant parameters

1. (Initialize) For $i = 1, \ldots, N$, sample $\theta^{(i)} \sim q(\theta)$, and compute the normalized weights $W_0^{(i)} \propto \frac{p(\theta)}{q(\theta)}$ with $\sum_{j=1}^{N} W_0^{(i)} = 1$.

2. For $t = 1, \ldots, t$:

   (a) (Reweight) For $i = 1, \ldots, N$, compute $W_t^{(i)} \propto p(y_t|\theta^{(i)})W_{t-1}^{(i)}$, with $\sum_{i=1}^{N} W_t^{(i)} = 1$.

   (b) (Estimate) Compute the (self-normalized) SIS estimate

$$E_t^{\text{SISn}} = \sum_{i=1}^{N} W_t^{(i)} f(\theta^{(i)})$$

2.1.1. Example: inferring the mean and variance of a Gaussian variable

To illustrate how this algorithm works, we will apply it to sequentially infer the (posterior) mean and variance of a random variable. The observations are assumed to be independent samples from a Gaussian distribution with unknown mean μ and variance σ^2, so the parameters are $\theta = (\mu, \sigma)$. As prior distributions, we will use a Gaussian distribution for μ (with a mean of 0 and a standard deviation of 10) and a uniform distribution for σ (in the range between 0 and 50). As these priors are easy to draw from, we use them also as importance distributions. We apply the algorithm to a total of 100 observations from a Gaussian distribution with mean $\mu = 5$ and standard deviation $\sigma = 5$, using an importance sample of size $N = 200$ (note that the sample size is kept low to enhance the clarity of the results; in real applications a larger sample size would be recommended). Figure 5 shows the resulting estimates (posterior means) as well as the normalized importance weights for each particle. We can see that the estimated posterior mean of σ comes reasonably close to the true value as t increases. However, the estimated posterior mean of μ converges to a value which is further off the true value. The problem is that, as t increases, the weight of almost all the particles becomes negligible. In the end, the posterior mean is effectively estimated by a single particle $\theta^{(*)} = (\mu^{(*)}, \sigma^{(*)})$. While this single particle

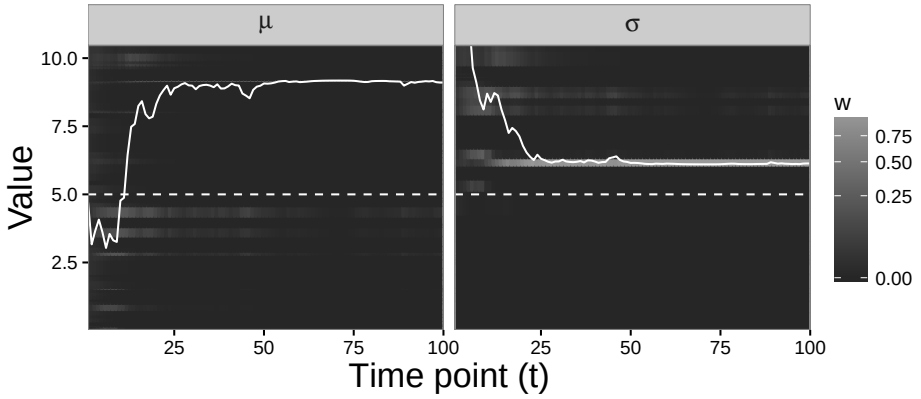

Figure 5: Online Bayesian inference of μ and σ of a Normal distribution. Solid white lines represent the posterior mean after each sequential observation and broken white lines the true values. Tiles in the background are centered around the particle values, such that each edge lies halfway between a particle and the immediately adjacent particle. The shade of each tile reflects the normalized weight of the corresponding particle. Initially, weights are almost uniformly distributed over the particles, but in the end (at time $t = 100$) only a single particle has non-negligible weight.

is, in some sense, the best one in the set, it does not have to be close to the true values. In this run of the algorithm, $\sigma^{(*)}$ was quite close to the true value, but $\mu^{(*)}$ was relatively far off from the true value. Indeed, there were other particles with values $\mu^{(i)}$ which were closer to the true value. However, for these particles, $\sigma^{(i)}$ was further off the true value, such that taken together as a pair of parameter values, particle $\theta^{(*)}$ was better than any of the other ones. The problem that the weights of almost all particles approach 0 is referred to as weight degeneracy and a driving force behind it is that the importance distribution becomes less and less efficient over time. While the posterior distribution at first is close to the prior distribution that was used as importance distribution, as more observations come in, the posterior distribution becomes more and more peaked around the true parameter values. The prior distribution is then too dispersed compared to the posterior distribution and far from optimal as importance distribution.

As illustrated in Figure 2, we can think of the set of particles as defining a random and irregularly-spaced grid and the SIS algorithm as approximating posterior distributions by computing the posterior probability of each grid point. The algorithm becomes less and less efficient because after sampling the particles, the grid is fixed. To make the algorithm more efficient, we should adapt the grid points to each posterior distribution we want to approximate. This is precisely what SMC algorithms do by resampling from the particles (grid points), replicating particles with high and eliminating those with low posterior probability at t. Insofar as the posterior probability of particles at time $t+1$ is not wildly different from the posterior probability at time t, this will thus provide useful grid points for time $t+1$. However, resampling provides exact replicates of useful particles, and using the same grid point multiple times does not increase the precision of the estimate.

After resampling, the particles are therefore "jittered" to rejuvenate the set, increasing the number of unique grid points and hence the precision of the approximation. Such jittering is natural for time-varying parameters. For instance, if instead of a fixed mean μ, we assume the mean μ_t changes from time to time according to a transition distribution $p(\mu_t|\mu_{t-1})$, we can use this transition distribution to generate samples of the current mean based on the samples of the previous mean. When the parameters are time-invariant, there is no immediately obvious way to "jitter" the particles after resampling. Some solutions have been proposed and we return to the problem of estimating static parameters with SMC in section 5.2. We will now first describe how resampling can be combined with (sequential) importance sampling, before turning to particle filters, which iterate sequential importance sampling and resampling steps to allow for flexible and efficient approximations to posterior distributions of latent states in general state-space models.

3. Resampling

Due to the problem of weight degeneracy, after running an SIS algorithm for a large number of iterations (time points), all but one particle will have negligible weight. Clearly, this is not a good situation, as we effectively approximate a distribution with a single particle. Moreover, this single particle does not even have to be in a region of high probability. A particle can have such a large relative weight that it will take very long for it to reduce, even though the target distribution has "moved on". A useful measure to detect weight degeneracy is the *effective sample size* (Liu, 2001, p. 35-36), defined as

$$N_{\text{eff}} \triangleq \frac{1}{\sum_{i=1}^{N}(W^{(i)})^2} \tag{8}$$

This measure varies between 1 (all but one particle have weight 0) and N (all particles have equal weight). Thus, the lower the effective sample size, the stronger the weight degeneracy.

To counter the problem of weight degeneracy, SMC algorithms include a resampling step, in which particles are sampled with replacement from the set of all particles, with a probability that depends on the importance weights. The main idea is to replicate particles with large weights and eliminate those with small weights, as the latter have little effect on the estimates anyway. The simplest sampling scheme is multinomial sampling, which draws N samples from a multinomial distribution over the particle indices $i = 1, \ldots, N$, with probabilities $p(i) = W^{(i)}$. After resampling, the weights are set to $W^{(i)} = 1/N$, because, roughly put, we have already used the information in the weights to resample the set of particles. It is straightforward to show that resampling does not change the expected value of the estimator (5).

In addition to countering weight degeneracy, a further benefit of resampling is that while the SIS samples themselves are not distributed according to the target distribution p (they are distributed according to the instrumental distribution q and to approximate p we need to use the importance weights), the resampled values are (approximately) distributed according to p. A drawback of resampling is that it increases the variance of the estimator. To reduce this effect of resampling, alternatives to multinomial resampling have been proposed with smaller variance.

The idea behind *residual resampling* (Liu and Chen, 1998) is to use a deterministic approach as much as possible, and then use random resampling for the remainder. To preserve the expected value of the estimator, the expected number of replications of each particle i should be $NW^{(i)}$.

This is generally not an integer, and hence we can't use these expectations directly to generate the desired number of replications. Residual resampling takes the integer part of each $NW^{(i)}$ term and replicates each particle deterministically according to that number. The remaining particles are then generated through multinomial resampling from a distribution determined by the non-integer parts of each $NW^{(i)}$ term.

Stratified resampling (Carpenter, Clifford, and Fearnhead, 1999) is another scheme which results in partly deterministic replication of particles. As the name suggests, it is based on the principles of stratified sampling used in survey research. Practically, the method consists of using the weights to form an "empirical" cumulative distribution over the particles. This distribution is then split into N equally sized strata, and a single draw is taken from each stratum.

The most popular resampling scheme, *systematic resampling* (Kitagawa, 1996), is based on the same intuition as stratified resampling, but reduces the Monte Carlo variance further by using a single random number, rather than a different random number, to draw from each stratum. Letting $\{\theta_t, W_t^{(i)}\}$ represent the set of particles before resampling, and $\{\tilde{\theta}_t^{(i)}, \tilde{W}_t^{(i)}\}$ the set of particles after resampling, systematic resampling can be summarized as follows:

Algorithm 4. Systematic resampling

1. Draw $u \sim \text{Unif}(0, 1/N)$.
2. Define $U^i = (i-1)/N + u$, $i = 1, \ldots, N$.
3. For $i = 1, \ldots, N$, find r such that $\sum_{k=1}^{r-1} W^{(k)} \leq U^i < \sum_{k=1}^{r} W_t^{(k)}$ and set $j(i) = r$.
4. For $i = 1, \ldots, N$, set $\tilde{\theta}_t^{(i)} = \theta_t^{(j(i))}$ and $\tilde{W}_t^{(i)} = 1/N$.

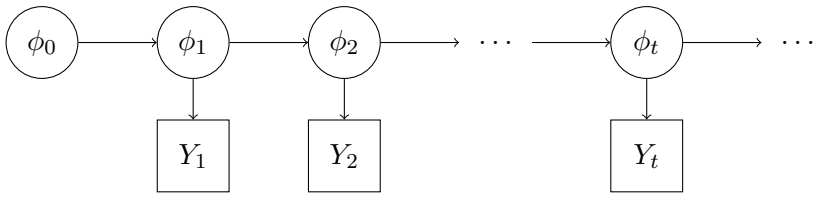

Figure 6: Schematic representation of a state-space model. Each observation Y_t at time t depends only on the current state ϕ_t. Each consecutive state ϕ_t depends only on the previous state ϕ_{t-1}.

Systematic resampling is simple to implement and generally performs very well in practice (Doucet and Johansen, 2011). However, in contrast to residual and stratified resampling, it is not guaranteed to outperform multinomial resampling (see Douc, Cappé, and Moulines, 2005, for this point and a thorough comparison of the theoretical properties of different resampling schemes).

4. Particle filters: SMC for state-space models

In filtering problems, we are interested in tracking the latent states ϕ_t of a stochastic process as each observation y_t comes in. In a Bayesian framework, we do this by computing a sequence of posterior distributions $p(\phi_1|y_1)$, ..., $p(\phi_t|y_{1:t})$. These posterior distributions of the current state ϕ_t, given all the observations thus far, are also called *filtering distributions*.

4.1. State-space models

State-space models are an important class of models to describe time-series of observations. State-space models describe an observable time series $y_{1:t}$ through a time-series of latent or hidden states $\phi_{0:t}$. In state-space models, we make two important assumptions about the relation between

states and observations. A graphical representation of a state-space model in the form of a Bayesian network is given in Figure 6. Firstly, we assume that each observation y_t depends solely on the current state ϕ_t, such that the observations are conditionally independent given the states ϕ_t:

$$p(y_{1:T}|\phi_{0:T}) = \prod_{t=1}^{T} p(y_t|\phi_t),$$

Secondly, we assume that the hidden states change over time according to a first-order Markov process, such that the current state depends only on the state at the immediately preceding time point:

$$p(\phi_{0:T}) = p(\phi_0) \prod_{t=1}^{T} p(\phi_t|\phi_{t-1}).$$

Given these two assumptions, we can write the posterior distribution over the hidden states as

$$p(\phi_{0:T}|y_{1:T}) = \frac{p(\phi_0) \prod_{t=1}^{T} p(y_t|\phi_t)p(\phi_t|\phi_{t-1})}{p(y_{1:T})}.$$

Moreover, we can compute the posteriors recursively as

$$p(\phi_{0:t}|y_{1:t}) = \frac{p(y_t|\phi_t)p(\phi_t|\phi_{t-1})}{p(y_t|y_{1:t-1})} p(\phi_{0:t-1}|y_{1:t-1}) \tag{9}$$

where

$$p(y_t|y_{1:t-1}) = \iint p(y_t|\phi_t)p(\phi_t|\phi_{t-1})p(\phi_{t-1}|y_{1:t-1}) \, \mathrm{d}\phi_{t-1}\mathrm{d}\phi_t$$

Models with this structure are also known as hidden Markov models (e.g., Visser, 2011).

4.2. SIS for state-space models

We can view the problem of estimating the hidden states $\phi_{0:t}$ as estimating a vector of parameters θ which increases in dimension at each time

point t, such that, at time t, we estimate $\theta = \phi_{0:t}$, and at time $t+1$ we add another dimension to the parameter vector to estimate $\theta = \phi_{0:t+1}$. Using IS, we could draw a new importance sample $\{\phi_{0:t}^{(i)}\}$ at each time t, but this would result in an increase in computational burden over time, which we would like to avoid in real-time applications. Using sequential importance sampling, we can incrementally build up the importance sample, starting at time $t = 0$ with a sample $\{\phi_0^{(i)}\}$, then sampling values $\phi_1^{(i)}$ at time 1 conditional on the sample at time 0, then adding sampled values $\phi_2^{(i)}$ at time 2 conditional on the sample at time 1, etc. Formally, this means we define the importance distribution at time t as

$$q_t(\phi_{0:t}) = q_t(\phi_t|\phi_{0:t-1})q_{t-1}(\phi_{0:t-1}). \tag{10}$$

Using this conditional importance distribution, and noting that $q_{t-1}(\phi_{0:t}) = q_{t-1}(\phi_{0:t-1})$, the right-hand ratio in (7) simplifies to $\frac{q_{t-1}(\phi_{0:t})}{q_t(\phi_{0:t})} = \frac{1}{q_t(\phi_t|\phi_{0:t-1})}$. Combining this with Equation 9, we can write the incremental weight update as

$$a_t^{(i)} = \frac{p(y_t|\phi_t^{(i)})p(\phi_t^{(i)}|\phi_{t-1}^{(i)})}{p(y_t|y_{1:t-1})q_t(\phi_t^{(i)}|\phi_{0:t-1}^{(i)})}.$$

Using normalized importance weights, we can ignore the $p(y_t|y_{1:t-1})$ term, which is often difficult to compute. As when using SIS to sequentially estimate time-invariant parameters, we now have an algorithm of (approximately) constant computational cost. At each time t, we add a new dimension to our particles by drawing values $\phi_t^{(i)}$ from a conditional importance distribution, and we update the weights without the need to revisit all the previous observations and hidden states.

As usual, the choice of importance distribution $q_t(\phi_t|\phi_{0:t-1})$ should be carefully considered, as a poor choice can result in an estimator with high

variance. An equivalent formulation of the incremental weight update is

$$a_t^{(i)} = \frac{p(\phi_t^{(i)}|y_t, \phi_{t-1}^{(i)})p(y_t|\phi_{t-1}^{(i)})}{p(y_t|y_{1:t-1})q_t(\phi_t^{(i)}|\phi_{0:t-1}^{(i)})},$$

which is generally more involved to compute, but indicates that the optimal importance distribution (in terms of minimizing the variance of the importance weights) is

$$q_t^*(\phi_t|\phi_{0:t-1}) = p(\phi_t|y_t, \phi_{t-1}).$$

The optimal importance distribution is again mostly of theoretical interest, but when possible, an importance distribution should be used which matches it as closely as possible.

Unfortunately, even when the optimal importance distribution can be used, SIS for state-space models will suffer from the same weight degeneracy problem we observed when estimating time-invariant parameters. Suppose that at time t, we have a "perfect" sample $\phi_{0:t}^{(i)} \sim p(\phi_{0:t}|y_{1:t})$, i.e., the particles are distributed according to the target distribution and the normalized importance weights are all $W^{(i)} = 1/N$. Moving to the next time point, we can sample the new state $\phi_{t+1}^{(i)} \sim p(\phi_{t+1}|y_{t+1}, \phi_t^{(i)})$, but without redrawing $\phi_{0:t}^{(i)}$, the resulting particles $\phi_{0:t+1}^{(i)}$ will not be distributed according to the current target distribution $p(\phi_{0:t+1}|y_{1:t+1})$. Thus, in a sequential algorithm where we keep the values $\phi_{0:t}^{(i)}$, it will not be possible to generate a perfect sample at time $t+1$. To do this, the samples $\phi_{0:t}^{(i)}$ would already have to be distributed according to $p(\phi_{0:t}|y_{1:t+1})$, and not according to $p(\phi_{0:t}|y_{1:t})$, the target distribution at time t. These two distributions are generally different, because the new observation y_{t+1} provides information about the likely values of $\phi_{0:t}$ that was not available at the time of drawing $\phi_{0:t}^{(i)}$. Hence,

repeated application of SIS necessarily introduces variability in the importance weights, which builds up over time, increasing the variance of the estimator and ultimately resulting in weight degeneracy.

4.3. A generic particle filter

To counter weight degeneracy, SMC methods combine SIS with resampling, replicating particles with high weights and eliminating those with low weights. After resampling, the particles have uniform weights and are (approximately) distributed according to the target distribution $p(\phi_{0:t}|y_{1:t})$. At the next time point, a new dimension ϕ_{t+1} is added to the particles by drawing from a conditional importance distribution $q_{t+1}(\phi_{t+1}^{(i)}|\phi_{0:t}^{(i)})$. Resampling will have focussed the set $\{\phi_{0:t}^{(i)}\}$ on useful "grid points" and while this set contains exact copies of particles, the values of the new dimension $\phi_{t+1}^{(i)}$ will be jittered and hence provide an adapted and useful set of grid points to estimate ϕ_{t+1}. As estimation of a current state ϕ_t is generally of more interest than estimating the whole path $\phi_{0:t}$, the fact that we now have a multidimensional grid where only the last dimension is adapted (as the dimensions reflecting earlier states contain exact replicates) is of little concern. If we are interested in estimating the whole path $\phi_{0:t}$, we would need to jitter the grid points on all dimensions. We return to this issue in Section 5.1.

A generic particle filter (see Figure 1) to approximate a sequence of posterior distributions $p(\phi_{0:1}|y_1)$, $p(\phi_{0:2}|y_{1:2})$, $\ldots$, $p(\phi_{0:t}|y_{1:t})$, proceeds as follows:

Algorithm 5. A generic particle filter

1. (Initialize) For $i = 1, \ldots, N$, sample $\tilde{\phi}_0^{(i)} \sim q(\phi_0)$ and compute the normalized importance weights $\tilde{W}_0^{(i)} \propto \frac{p(\tilde{\phi}_0^{(i)})}{q(\tilde{\phi}_0^{(i)})}$ with $\sum_{i=1}^N \tilde{W}_t^{(i)} = 1$.

2. For $t = 1, \ldots, T$:

   (a) (Propagate) For $i = 1, \ldots, N$, sample $\phi_t^{(i)} \sim q_t(\phi_t | \tilde{\phi}_{0:t-1}^{(i)})$, and add this new dimension to the particles, setting $\phi_{0:t}^{(i)} = (\tilde{\phi}_{0:t-1}^{(i)}, \phi_t^{(i)})$.

   (b) (Reweight) For $i = 1, \ldots, N$, compute normalized weights

$$W_t^{(i)} \propto \frac{p(y_t|\phi_t^{(i)})p(\phi_t^{(i)}|\phi_{t-1}^{(i)})}{q_t(\phi_t^{(i)}|\phi_{0:t-1}^{(i)})}\tilde{W}_{t-1}^{(i)}$$

   with $\sum_{i=1}^N W_t^{(i)} = 1$.

   (c) (Estimate) Compute the required estimate

$$E_t^{\mathrm{PFn}} = \sum_{i=1}^N f(\phi_{0:t}^{(i)})W_t^{(i)}.$$

   (d) (Resample) If $N_{\mathrm{eff}} \leq cN$, resample $\{\tilde{\phi}_{0:t}^{(i)}\}$ with replacement from $\{\phi_{0:t}^{(i)}\}$ using the normalized weights $W_t^{(i)}$ and set $\tilde{W}_t^{(i)} = 1/N$ to obtain a set of equally weighted particles $\{\tilde{\phi}_t^{(i}, \tilde{W}^{(i)} = 1/N\}$; else set $\{\tilde{\phi}_{0:t}^{(i)}, \tilde{W}_t^{(i)}\} = \{\phi_{0:t}^{(i)}, W_t^{(i)}\}$.

In this generic particle filter, we allow for optional resampling, whenever the effective sample size is smaller than or equal to a proportion c of the number of particles used N. While particle filters often set $c = 1$, so that resampling is done on every step, choosing a different value can be beneficial, as resampling introduces additional variance in the estimates. If the importance weights show little degeneracy, then this additional variance is unnecessary. Therefore, setting $c = .5$, which is another common value, we only resample when there is sufficient evidence for weight degeneracy.

At each iteration, we effectively have two particle approximations, the set $\{\phi_{0:t}^{(i)}, W_t^{(i)}\}$ before resampling, and the set $\{\tilde{\phi}_{0:t}^{(i)}, \tilde{W}^{(i)}\}$ after resampling. While both provide unbiased estimates, the estimator before resampling

generally has lower variance. It should also be noted that this particle filter provides a weighted sample of state *sequences* $\phi_{0:t}^{(i)} = (\phi_0^{(i)}, \phi_1^{(i)}, \ldots, \phi_t^{(i)})$, approximating a posterior distribution over state sequences $p(\phi_{0:t}|y_{1:t})$. The estimator E_t^{PFn} is also defined over these sequences and not a single state ϕ_t. In filtering problems, we are generally only interested in estimating the current state, not the whole path $\phi_{0:t}$. As the posterior distribution over a single state is a marginal distribution of the joint posterior distribution over all states, we can write the required expected value as

$$\begin{aligned}\mathbb{E}_p[f(\phi_t)] &= \int f(\phi_t) p(\phi_t|y_{1:t}) \,\mathrm{d}\phi_t \\ &= \iint f(\phi_t) p(\phi_t, \phi_{0:t-1}|y_{1:t}) \,\mathrm{d}\phi_t \mathrm{d}\phi_{0:t-1}.\end{aligned}$$

This means that we can effectively ignore the previous states, and use the estimator

$$E^{\mathrm{PFn}} = \sum_{i=1}^{n} W_t^{(i)} f(\phi_t^{(i)}).$$

Several variants of this generic particle filter can be found in the literature. In the bootstrap filter (Gordon, Salmond, and Smith, 1993), the state transition distribution is used as the conditional importance distribution, i.e. $q_t(\phi_t|\phi_{0:t-1}^{(i)}) = p(\phi_t|\phi_{t-1})$. This usually makes the propagate step simple to implement and also simplifies the reweighting step, as the weights can now be computed as $W_t^{(i)} \propto p(y_t|\phi_t^{(i)})\tilde{W}_{t-1}^{(i)}$. The auxiliary particle filter, introduced in Pitt and Shephard (1999) and later improved by Carpenter et al. (1999), effectively switches the resampling and propagate steps, resulting in a larger number of distinct particles to approximate the target. The auxiliary particle filter can be implemented as a variant of the generic particle filter by adapting the importance weights to incorporate information from the observation at the next time point; the subsequent resampling step is

then based on information from the next time point, and thus particles will be resampled which are likely to be useful for predicting this time point. For more information, see, e.g., Doucet and Johansen (2011) and Whiteley and Johansen (2011).

Chopin (2004) shows how a general class of SMC algorithms, including the generic particle filter, satisfy a Central Limit Theorem, such that, as the number of particles approaches infinity, SMC estimates follow a Gaussian distribution centered around the true value. For other convergence results and proofs, see, e.g., Del Moral (2013), Douc and Moulines (2008), and Whiteley (2013).

4.4. Example: A particle filter for a simple Gaussian process

We will illustrate SIS with an example of a latent Gaussian process with noisy observations. Suppose there is a latent variable ϕ which moves in discrete time according to a random walk

$$\phi_{t+1} = \phi_t + \xi_t \qquad \xi_t \sim N(0, \sigma_\xi^2), \tag{11}$$

where the initial distribution at $t = 0$ is given as

$$\phi_0 \sim N(\mu_0, \sigma_0^2). \tag{12}$$

The value of the latent variable can only be inferred from noisy observations Y_t that depend on the latent process through

$$Y_t = \phi_t + \epsilon_t \qquad \epsilon_t \sim N(0, \sigma_\epsilon^2). \tag{13}$$

This model is a relatively simple state-space model with

$$p(y_t|\phi_t) = N(\phi_t, \sigma_\epsilon^2)$$

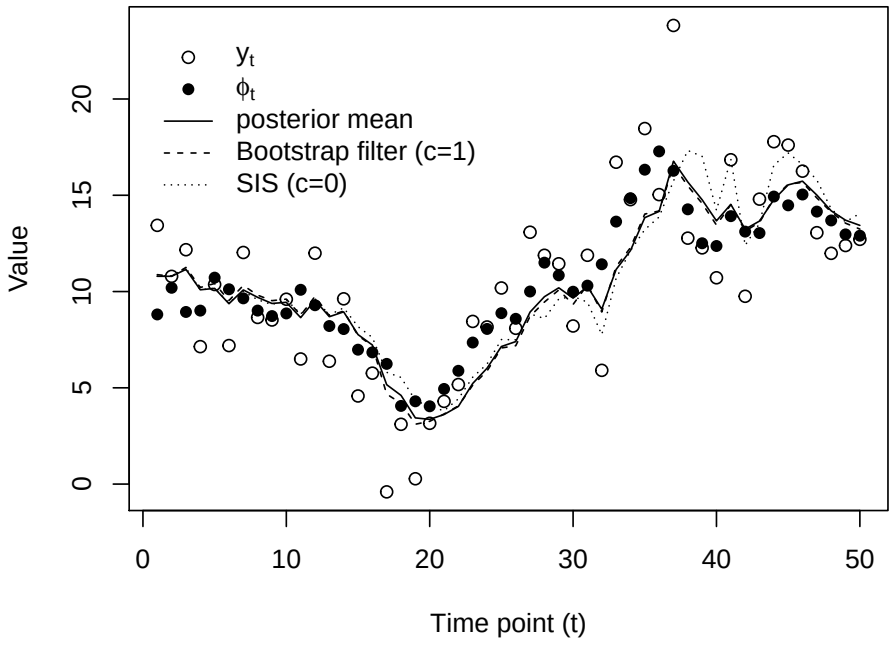

Figure 7: Example of the one-dimensional Gaussian latent process with $\mu_0 = 10$, $\sigma_0^2 = 2$, $\sigma_\xi^2 = 1$, and $\sigma_\epsilon^2 = 10$. Dots show the observations y_t and latent states ϕ_t. Lines show the true posterior means and particle filter estimates of the posterior means.

and

$$p(\phi_t|\phi_{t-1}) = N(\phi_{t-1}, \sigma_\xi^2).$$

Figure 7 contains example data from this process.

Suppose that the observations y_t are made sequentially, and after each new observation, we wish to infer the value of the underlying latent variable ϕ_t. The distributions of interest are thus $p(\phi_1|y_1)$, $p(\phi_2|y_{1:2}), \ldots, p(\phi_t|y_{1:t})$. As the process is linear and Gaussian, the distributions can be computed

analytically by the Kalman filter (Kalman, 1960; Kalman and Bucy, 1961). However, approximating the posterior means by a particle filter will illustrate some key features of the algorithm and the availability of analytical estimates offers a useful standard to evaluate its quality.

We will use a bootstrap filter, using as conditional importance distributions the transition distribution of the latent process, i.e. $q_t(\phi_t|\phi_{0:t-1}) = p(\phi_t|\phi_{t-1})$. The self-normalized weights are then easily computed as $W_t \propto p(y_t|\phi_t^{(i)})\tilde{W}_{t-1}^{(i)}$. We also set $c = 1$, so that we resample at each iteration, using systematic resampling. Figure 7 contains the resulting estimates E_t^{PFn} of the posterior means as well as the analytical (true) values computed with a Kalman filter. As can be seen there, the estimates are very close to the analytical posterior means. For comparison, if we run the filter with $c = 0$ (so that we never resample, turning it into a straightforward SIS algorithm), we see that while the estimates are quite good initially, at later time points, the deviation between the estimated and actual posterior means increases. Again, this is due to weight degeneracy, which is countered by the resampling step in the particle filter. The effect of resampling can be clearly seen in Figure 8, which depicts the variation in the estimates when the algorithms are applied repeatedly to the same data. While there clearly is an increase in the variance of SIS over time, the estimates of the bootstrap filter remain close to the analytical values. For comparison, we also plot the results of an SIS and particle filter algorithm with the optimal importance distribution $q_t(\phi_t|\phi_{0:t-1}) = p(\phi_t|y_t, \phi_{t-1})$. The increase in efficiency due to the optimal importance distribution is clearly seen in the case of SIS. While still present, the difference between the two particle filters appears less marked.

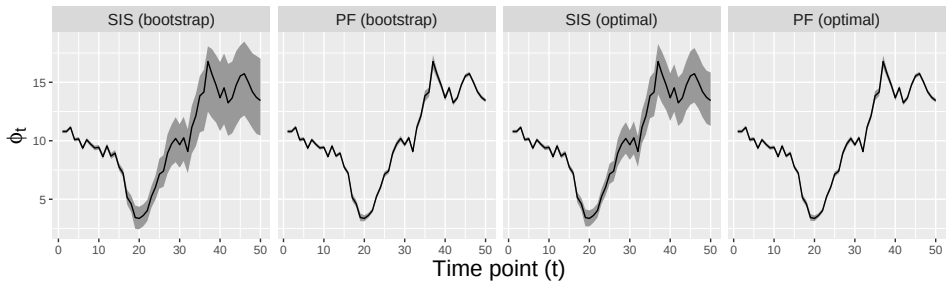

Figure 8: A comparison of SIS (no resampling) and particle filters (with resampling) shows that resampling clearly improves the accuracy of estimation. Each panel shows the true posterior mean (solid line) and 95% interpercentile region of estimated posterior means (shaded region) when applying the algorithms to the data of Figure 7. SIS (bootstrap) is the bootstrap filter with $c = 0$ (no resampling), PF (bootstrap) is the bootstrap filter with $c = 1$ (always resampling), and SIS (optimal) and PF (optimal) are similar to these but use the optimal importance distribution. Each algorithm uses $N = 500$ particles and the interpercentile regions were computed by running each algorithm on the same data for 2000 replications.

5. Further issues and extensions

While particle filters generally work well to approximate the filtering distributions of latent states in general (non-linear and/or non-Gaussian) state-space models, the application of SMC beyond this domain requires further consideration. Here we briefly highlight some issues and solutions for using SMC to approximate the posterior distributions over whole state trajectories and time-invariant or static parameters. We end with a brief discussion of how to combine sampling with analytical integration in order to increase the efficiency of SMC.

5.1. Sample impoverishment and particle smoothing

Although resampling counters the problem of weight degeneracy, it introduces a new problem of "sample impoverishment". By replicating and removing particles, resampling reduces the total number of unique values present in the set of particles. This is no major issue in filtering, where we are interested in estimating the current state ϕ_t. The particle values $\phi_t^{(i)}$ used for this are "jittered" in the propagate step, and estimation proceeds before resampling affects the number of unique values of this state in the set of particles. However, resampling does reduce the unique values of $\phi_{0:t-1}^{(i)}$ reflecting the states at earlier time points, and over time this problem becomes more and more severe for the initial states. When the algorithm has run for a large number of time points, it can be the case that all the particles have the same value for ϕ_1. This sample impoverishment can severely affect the approximation of the so called *smoothing* distributions $p(\phi_t|y_{1:T})$, $1 \leq t \leq T$.

To provide better approximations of smoothing distributions, several alternatives to basic SMC have been proposed. A simple procedure is to use

a *fixed lag* approximation (Kitagawa and Sato, 2001). This approach relies on the exponential forgetting properties of many state-space models, such that

$$p(\phi_{0:t}|y_{1:T}) \approx p(\phi_{0:t}|y_{1:(t+\Delta)}) \tag{14}$$

for a certain integer $0 < \Delta < T - t$; that is, observations after time $t + \Delta$ provide no additional information about $\phi_{0:t}$. If this is the case, we do not need to update the estimates of $\phi_{0:t}$ after time $t+\Delta$. For the SMC algorithm, that means we do not need to update (e.g., resample) the particle values $\phi_{0:t}^{(i)}$ then. Generally, we do not know Δ, and hence have to choose a value D which may be smaller or larger than Δ. If $D > \Delta$, we have not reduced the degeneracy problem as much as we could. If $D < \Delta$, then $p(\phi_{0:t}|y_{1:(t+D)})$ is a poor approximation of $p(\phi_{0:t}|y_{1:T})$.

A better, but computationally more expensive option is to store the particle approximations of the filtering distributions (i.e., the particle values $\phi_t^{(i)}$ and weights $W_t^{(i)}$ approximating $p(\phi_t|y_{1:t})$) and then reweight these using information from observations $y_{(t+1):T}$ to obtain an approximation of $p(\phi_t|y_{1:T})$. Particle variants of the Forward Filtering Backwards Smoothing and Forward Filtering Backwards Sampling algorithms have been proposed for this purpose (see e.g., Douc, Garivier, Moulines, and Olsson, 2011). One issue is that as these methods reweight or resample the particle values used to approximate the filtering distributions, but do not generate new particle values, they can be expected to perform poorly when the smoothing distributions differ substantially from the filtering distributions. An alternative approach, which can be expected to perform better in these situations, is the two-filter formulation of Briers, Doucet, and Maskell (2010).

5.2. Inferring time-invariant (static) parameters

While particle filtering works generally well to estimate latent states ϕ_t, which can be viewed as time-varying parameters, estimation of time-invariant or static parameters θ is more problematic. For instance, consider the simple Gaussian process defined in (11) - (13), and suppose both σ_ξ and σ_ϵ are unknown. The inference problem is then to approximate

$$p(\phi_{0:t}, \theta | y_{1:t}) \propto p(\phi_{0:t} | \theta, y_{1:t}) p(\theta | y_{1:t})$$

where $\theta = (\sigma_\xi, \sigma_\epsilon)$. The main problem for a particle filter approximation is again sample impoverishment: resampling will reduce the number of unique particle values that represent θ. For time-invariant parameters, there is no natural way to "jitter" the particles after resampling. One solution is to use an artificial dynamic process for the static parameters, e.g., drawing new particles from a (multivariate) Normal distribution centered around the old particle values

$$\theta_{t+1}^{(i)} \sim N(\theta_t^{(i)}, \Sigma_\theta),$$

where Σ_θ is a covariance matrix, and subscript t indicates that the parameter values reflect the time t posterior and not that the parameters are actually time-varying. While this reduces the problem of sample impoverishment, the artificial process will inflate the variance of the posterior distributions. Liu and West (2001) propose to view the artificial dynamics as a form of kernel smoothing: drawing new particle values from a Normal distribution centered around the old particle values is akin to approximating the distribution of θ by a (multivariate) Gaussian kernel density:

$$p(\theta | y_{1:t}) \approx \sum_{i=1}^{N} W^{(i)} N(\theta | \theta_t^{(i)}, \Sigma_\theta).$$

To reduce variance inflation, Liu and West (2001) suggest to use a form of shrinkage, shifting the kernel locations $\theta_t^{(i)}$ closer to their overall mean by a factor which ensures that the variance of the particles equals the actual posterior variance.

An alternative is to incorporate Markov chain Monte Carlo (MCMC) moves in the particle filter (Andrieu, Doucet, and Holenstein, 2010; Chopin, 2002; Chopin, Jacob, and Papaspiliopoulos, 2013; Gilks and Berzuini, 2001). The idea is to rejuvenate the particle set by applying a Markov transition kernel with the correct invariant distribution as the target. As they leave the target distribution intact, inclusion of MCMC moves in a particle filter algorithm is generally allowed. For a recent comparison of various approaches to parameter estimation with SMC techniques, see Kantas, Doucet, Singh, Maciejowski, and Chopin (2015).

5.3. Rao-Blackwellized particle filters

There are models in which, conditional upon some parameters, the distributions of the remaining parameters can be solved analytically. For instance, in the example of the Gaussian process, we could assume that the process can switch between periods of high and low volatility. This can be represented by assuming a second latent process $\omega_{1:T}$, where ω_t is a discrete latent state indicating low or high volatility, and letting the innovation variance $\sigma_\xi(\omega_t)$ be a function of this discrete latent state. This is an example of a switching linear state-space model, which is analytically intractable. Writing the joint posterior as

$$p(\phi_{0:t}, \omega_{1:t}|y_{1:t}) = p(\phi_{0:t}|\omega_{1:t}, y_{1:t})p(\omega_{1:t}|y_{1:t})$$

and realizing that, conditional upon $\omega_{1:t}$, $\phi_{0:t}$ is a linear Gaussian state-space model, the Kalman filter can be used to analytically compute the

conditional distributions $p(\phi_{0:t}|\omega_{1:t}, y_{1:t})$. Hence, we only need to approximate $p(\omega_{1:t}|y_{1:t})$ through sampling (cf. Chen and Liu, 2000). Solving part of the problem analytically reduces the variance in the importance weights, and hence increases the reliability of the estimates (Chopin, 2004). The main message here is that sampling should be avoided whenever possible. The combination of sampling and analytical inference is also called Rao-Blackwellisation (Casella and Robert, 1996).

6. A particle filter to track changes in learning rate during probabilistic category learning

As a final example of SMC estimation, we use a particle filter to estimate changes in learning rate in probabilistic category learning. In probabilistic category learning tasks, people learn to assign objects to mutually exclusive categories according to their features, which are noisy indicators of category membership. Tracking the learning rate throughout a task is theoretically interesting, as it reflects how people adapt their learning to the volatility in the task. If the relation between features and category membership is time-invariant, people should ideally show "error discounting" (Craig, Lewandowsky, and Little, 2011; Speekenbrink, Channon, and Shanks, 2008), where they accept an unavoidable level of error and stabilize their classification strategy by slowly stopping to learn. On the other hand, if the relation between features and category membership changes over time, then people should continue to adapt their categorization strategy to these changes (Behrens, Woolrich, Walton, and Rushworth, 2007; Speekenbrink and Shanks, 2010). How quickly they adapt (i.e. their learning rate) should ideally depend on the rate at which the feature-category relation changes

(i.e. the volatility in the task). When the volatility is unknown, this itself has to be inferred from experience, such that people effectively have to "learn how (much) to learn".

Previous investigations of dynamic changes in learning rate have either estimated learning rates separately in consecutive blocks of trials (Behrens et al., 2007), or assumed the changes in learning rate followed a predetermined schedule (Craig et al., 2011; Speekenbrink et al., 2008). Using a particle filter, we can estimate the learning rate on a trial-by-trial basis without making too restrictive assumptions. In this example, we use unpublished data collected in 2005 by David A. Lagnado at University College London. Nineteen participants (12 female, average age 25.84) performed the Weather Prediction Task (WPT, Knowlton, Squire, and Gluck, 1994). In the WPT, the objective is to predict the state of the weather ("fine" or "rainy") on the basis of four cues (tarot cards with different geometric patterns, which are either present or absent). Two cues are predictive of fine weather and two cues of rainy weather. The version of the WPT used here included a sudden change, whereby from trial 101 until the final (200th) trial, cues that were first predictive of fine weather become predictive of rainy weather, and vice versa.

As in Speekenbrink et al. (2008), we will assume people learn an associative weight, v_j, for each cue. Over trials, these weights are updated by the delta-rule

$$v_{j,t+1} = v_{j,t} + \eta_t \left(y_t - p_t\right) x_{j,t},$$

where $x_{j,t}$ is a binary variable reflecting whether cue j was present on trial t, y_t is a binary variable reflecting the state of the weather, and p_t is the

predicted probability of the state of the weather:

$$p_t = \frac{1}{1 + \exp\left(-\sum_{j=1}^{4} v_{j,t} x_{j,t}\right)}$$

People's categorization responses are also assumed to follow these predicted probabilities.

Our interest is in the learning rate $\eta_t > 0$, which we will allow to vary from trial to trial according to the following transition distribution:

$$p(\eta_{t+1}|\eta_t) = TN(\eta_t, \sigma_\eta^2) \tag{15}$$

where TN is a normal distribution truncated below at 0 (as the learning rate is positive). We also use a truncated normal distribution for the initial learning rates:

$$\eta_0 \sim TN(\mu_0, \sigma_0^2)$$

Estimating the learning rates η_t is a difficult problem. Each response is a random variable drawn from a Bernoulli distribution with parameter p_t, which depends on the cues on trial t and the associative weights $v_{j,t}$. These associative weights in turn depend on the starting weights $v_{j,1}$ (which we fix to 0), the previous cues and states of the weather, and the sequence of unknown learning rates $\eta_{1:t-1}$. While the delta rule is deterministic, uncertainty about the learning rates induces uncertainty about the associative weights. Unfortunately, as we only have a single binary response on each trial, we obtain relatively little information to reduce the uncertainty about the associative weights and with that about the learning rates which gave rise to these weights. All in all, we can thus expect the estimated learning rates to be somewhat noisy.

For each participant, we estimated the time-varying learning rates with a bootstrap filter with $N = 2000$ particles and selective resampling when

$N_{\text{eff}} < .5N$, i.e. when the effective sample size was half the number of particles. The hyper-parameters were $\mu_0 = 0.05$, $\sigma_0 = 0.795$ and $\sigma_\eta = 0.101$, which were determined maximizing the likelihood of the responses for all participants.

We expected learning rates to start relatively high at the beginning of the task and then to gradually decrease due to error discounting. In response to the abrupt change in task structure at trial 101, learning rates were expected to increase again, possibly decreasing again thereafter. Figure 9 shows the estimated means and 5% and 95% quantiles of the posterior (filtering) distributions $p(\eta_t|\mathbf{x}_{1:t+1}, y_{1:t}, r_{1:t+1})$. The results show that many participants show an increase in learning rate after trial 101, reflecting the change in task structure at that point. Thus, many participants appeared to indeed adapt their learning to the volatility in the environment. While some participants, such as S5, show the expected pattern with initially relatively high learning rate which decreases until the change at trial 101, then increasing and decreasing again thereafter, other participants, such as S14 do not show slowed learning towards the end of the task. This might be due to expecting more abrupt changes. There is quite some individual variability in the dynamics of learning rate over time. The best performing participants have relatively high learning rates and marked changes throughout the task. The less well performing participants had relatively low learning rates, indicative of a slow adaptation of their strategy to the task structure. While the posterior distributions are generally wide, because the responses provide limited information about the learning rates, the results were consistent over multiple runs of the algorithm and deviations in the hyper-parameters.

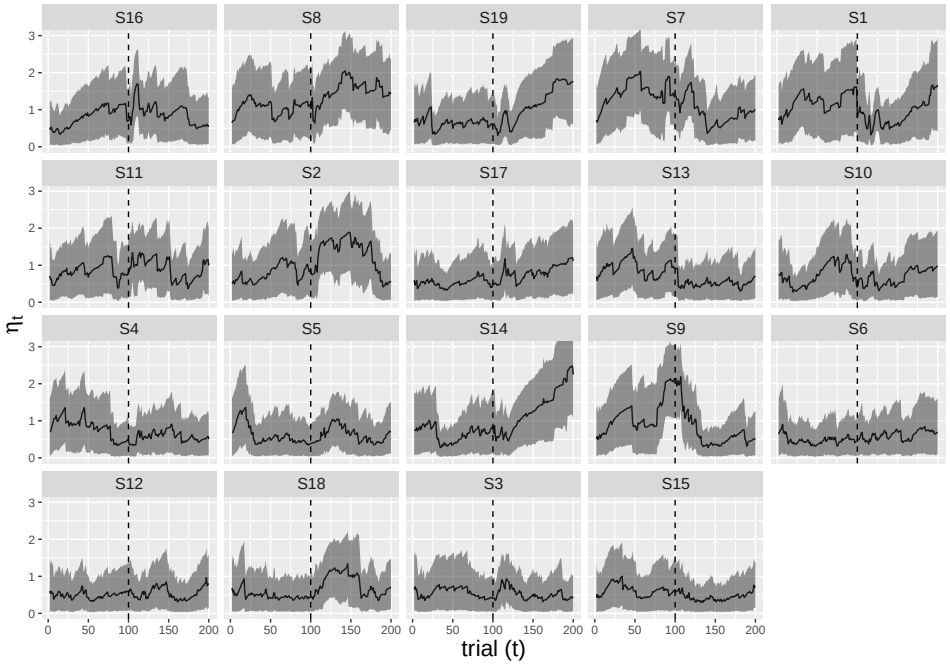

Figure 9: Particle filter estimates of time-varying learning rates. Solid lines represent the estimated mean (solid line) and shaded areas the 5%-95% interpercentile range of the posterior (filtering) distributions of η_t for each participant. Participants are ordered row-wise according to the expected performance of their predictions (the probability that their prediction was correct, rather than whether it was actually correct on a particular trial), with the best performing participant appearing in the top-left panel.

7. Conclusion

This tutorial introduced sequential Monte Carlo (SMC) estimation and, in particular, particle filters. These techniques provide sampling-based approximations of a sequence of posterior distributions over parameter vectors which increase in dimension, and allow inference in complex dynamic statistical models. They rely on a combination of sequential importance sampling and resampling steps. Sequential importance sampling provides sets of weighted random samples (particles), while resampling reduces the problem of weight degeneracy that plagues sequential importance sampling. SMC has proven especially useful in filtering problems for general state-space models, where the objective is to estimate the current value of a latent state given all previous observations, but its use extends to other problems including maximum likelihood estimation (e.g., Johansen, Doucet, and Davy, 2008) and optimizing experimental designs (Amzal et al., 2006). SMC is an active research field in statistics and machine learning and recent developments have focussed on combining SMC with Markov chain Monte Carlo (MCMC) techniques in order to provide efficient inference for statistical models with both dynamic states and static parameters (e.g., Andrieu et al., 2010; Chopin et al., 2013). Recent software to implement SMC techniques include Biips (`http://alea.bordeaux.inria.fr/biips/`) and LibBi (`http://libbi.org/`). All analyses in this tutorial were programmed in the R language and all code and data are available in the supplementary material and on the Open Science Framework (`http://osf.io/b6gsk/`).

8. Acknowledgements

This work was supported by the U.K. Economic and Social Research Council (ESRC), grant RES-062-23-1511.

9. References

Amzal, B., Bois, F. Y., Parent, E., Robert, C. P., 2006. Bayesian-optimal design via interacting particle systems. Journal of the American Statistical Association 101, 773–785.

Andrieu, C., Doucet, A., Holenstein, R., 2010. Particle Markov chain Monte Carlo methods. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 72 (3), 269–342.

Behrens, T. E. J., Woolrich, M. W., Walton, M. E., Rushworth, M. F. S., Sep. 2007. Learning the value of information in an uncertain world. Nature neuroscience 10 (9), 1214–21.

Briers, M., Doucet, A., Maskell, S., 2010. Smoothing algorithms for state?space models. Annals of the Institute of Statistical Mathematics 62, 61–89.

Brown, S. D., Steyvers, M., 2009. Detecting and predicting changes. Cognitive Psychology 58, 49–67.

Carpenter, J., Clifford, P., Fearnhead, P., 1999. Improved particle filter for nonlinear problems. IEE Proceedings-Radar, Sonar and Navigation 146 (1), 2–7.

Casella, G., Robert, C. P., 1996. Rao-blackwellisation of sampling schemes. Biometrika 83 (1), 81–94.

928 Casella, G., Robert, C. P., 1998. Post-processing accept-reject samples: Re-
929 cycling and rescaling. Journal of Computational and Graphical Statistics
930 7 (2), pp. 139–157.

931 Chen, R., Liu, J. S., 2000. Mixture kalman filters. Journal of the Royal
932 Statistical Society: Series B (Statistical Methodology) 62 (3), 493–508.

933 Chopin, N., 2002. A sequential particle filter method for static models.
934 Biometrika 89 (3), 539–552.

935 Chopin, N., 2004. Central limit theorem for sequential monte carlo methods
936 and its application to bayesian inference. The Annals of Statistics 32 (6),
937 pp. 2385–2411.

938 Chopin, N., Jacob, P. E., Papaspiliopoulos, O., 2013. Smc2: an efficient
939 algorithm for sequential analysis of state space models. Journal of the
940 Royal Statistical Society: Series B (Statistical Methodology) 75 (3), 397
941 – 426.

942 Craig, S., Lewandowsky, S., Little, D. R., May 2011. Error discounting
943 in probabilistic category learning. Journal of experimental psychology.
944 Learning, memory, and cognition 37 (3), 673–87.

945 Del Moral, P., 1996. Non-linear filtering: interacting particle resolution.
946 Markov processes and related fields 2 (4), 555–581.

947 Del Moral, P., 2013. Mean field simulation for Monte Carlo integration.
948 Chapman & Hall/CRC Press, London.

949 Douc, R., Cappé, O., Moulines, E., 2005. Comparison of resampling schemes
950 for particle filtering. In: Proceedings of the 4th International Symposium

on Image and Signal Processing and Analysis. ISPA 2005. IEEE, pp. 64–69.

Douc, R., Garivier, A., Moulines, E., Olsson, J., 2011. Sequential Monte Carlo smoothing for general state space hidden Markov models. The Annals of Applied Probability 21, 2109–2145.

Douc, R., Moulines, E., 10 2008. Limit theorems for weighted samples with applications to sequential Monte Carlo methods. Ann. Statist. 36 (5), 2344–2376.

Doucet, A., de Freitas, N., Gordon, N., 2001a. An introduction to sequential monte carlo methods. In: Doucet, A., de Freitas, N., Gordon, N. (Eds.), Sequential Monte Carlo methods in practice. Springer, New York, pp. 3–14.

Doucet, A., de Freitas, N., Gordon, N. (Eds.), 2001b. Sequential Monte Carlo methods in practice. Springer, New York.

Doucet, A., Johansen, A. M., 2011. A tutorial on particle filtering and smoothing: Fifteen years later. In: Crisan, D., Rozovskii, B. (Eds.), The Oxford handbook of nonlinear filtering. Oxford University Press, New York, NY, pp. 656–704.

Gilks, W. R., Berzuini, C., 2001. Following a moving target: Monte Carlo inference for dynamic Bayesian models. Journal of the Royal Statistical Society. Series B (Statistical Methodology) 63 (1), pp. 127–146.

Gordon, N. J., Salmond, D. P., Smith, A. F. M., 1993. A novel approach to non-linear/non-Gaussian Bayesian state estimation. IEE Proceedings-F 140, 107–113.

Isard, M., Blake, A., 1998. Condensation—conditional density propagation for visual tracking. International Journal of Computer Vision 29 (1), 5–28. URL http://dx.doi.org/10.1023/A:1008078328650

Johansen, A. M., Doucet, A., Davy, M., 2008. Particle methods for maximum likelihood estimation in latent variable models. Statistics and Computing 18 (1), 47–57.

Kahn, H., Marshall, A. W., 1953. Methods of reducing sample size in monte carlo computations. Journal of the Operations Research Society of America 1 (5), 263–278.

Kalman, R. E., 1960. A new approach to linear filtering and prediction problems. Transactions of the American Society of Mechanical Engineers, Series D, Journal of Basic Engineering 82, 35–45.

Kalman, R. E., Bucy, R. S., 1961. New results in linear filtering and prediction theory. Transactions of the American Society of Mechanical Engineers, Series D, Journal of Basic Engineering 83, 95–108.

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J. M., Chopin, N., 2015. On particle methods for parameter estimation in general state-space models. Statistical Science 30 (3), 328–351.

Kitagawa, G., 1996. Monte carlo filter and smoother for non-gaussian nonlinear state space models. Journal of Computational and Graphical Statistics 5, 1–25.

Kitagawa, G., Sato, S., 2001. Monte carlo smoothing and self-organizing state?space model. In: Doucet, A., de Freitas, J. F. G., Gordon, N. J.

(Eds.), Sequential Monte Carlo methods in practice. Springer, New York, pp. 177–195.

Knowlton, B. J., Squire, L. R., Gluck, M. A., 1994. Probabilistic classification learning in amnesia. Learning & Memory 1, 106–120.

Kruschke, J. K., 1992. ALCOVE: an exemplar-based connectionist model of category learning. Psychological Review 99, 22–44.

Liu, J., West, M., 2001. Combined parameter and state estimation in simulation-based filtering. In: Doucet, A., de Freitas, N., Gordon, N. (Eds.), Sequential Monte Carlo Methods in Practice. Springer, New York, NY.

Liu, J. S., 2001. Monte Carlo Strategies in Scientific Computing. Springer, New York.

Liu, J. S., Chen, R., 1998. Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association 93 (443), 1032–1044.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., 2002. Fastslam: A factored solution to the simultaneous localization and mapping problem. In: Dechter, R., Kearns, M., Sutton, R. (Eds.), Eighteenth National Conference on Artificial Intelligence. American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 593–598.

Myung, J. I., Cavagnaro, D. R., Pitt, M. A., 2013. A tutorial on adaptive design optimization. Journal of Mathematical Psychology 57 (3–4), 53–67.

Nummiaro, K., Koller-Meier, E., Gool, L. V., 2003. An adaptive color-based

particle filter. Image and Vision Computing 21 (1), 99 – 110.
URL http://www.sciencedirect.com/science/article/pii/S0262885602001294

Pitt, M. K., Shephard, N., 1999. Filtering via simulation: Auxiliary particle filters. Journal of the American Statistical Association 94 (446), 590.

Robert, C. P., Casella, G., 2004. Monte Carlo statistical methods, 2nd Edition. Springer, New York.

Sanborn, A. N., Griffiths, T. L., Navarro, D. J., 2010. Rational approximations to rational models: Alternative algorithms for category learning. Psychological Review 117 (4), 1144–1167.

Speekenbrink, M., Channon, S., Shanks, D. R., 2008. Learning strategies in amnesia. Neuroscience and Biobehavioral Reviews 32, 292–310.

Speekenbrink, M., Shanks, D. R., May 2010. Learning in a changing environment. Journal of Experimental Psychology. General 139 (2), 266–98.

Van Zandt, T., Sep. 2000. How to fit a response time distribution. Psychonomic Bulletin & Review 7 (3), 424–65.

Visser, I., 2011. Seven things to remember about hidden Markov models: A tutorial on Markovian models for time series. Journal of Mathematical Psychology 55 (6), 403–415.

Whiteley, N., 2013. Stability properties of some particle filters. The Annals of Applied Probability 23 (6), 2500–2537.

Whiteley, N., Johansen, A. M., 2011. Bayesian time series models. Cambridge University Press, Cambridge, U.K., Ch. Auxiliary particle filtering: Recent developments, pp. 52–81.

Yi, M. S., Steyvers, M., Lee, M., 2009. Modeling human performance in restless bandits with particle filters. The Journal of Problem Solving 2 (2), 5.