CSCI-6971 Lecture Notes: Sequential Monte Carlo methods*

Kristopher R. Beevers
Department of Computer Science
Rensselaer Polytechnic Institute
beevek@cs.rpi.edu

March 7, 2006

1 Importance sampling: recap

Recall that if our goal is to compute an expected value of the form:

$$\mathbf{E}_\pi[h(\mathbf{x})] = \int_{\mathcal{A}} h(\mathbf{x})\pi(\mathbf{x})\, d\mathbf{x} \tag{1}$$

an effective strategy is to use an importance sampling Monte Carlo strategy, which approximates the integration by summing m weighted random samples $\mathbf{x}_i, i = 1 \ldots m$, drawn from a *proposal distribution* g (also known as an *importance density*, *trial density*, or *sampling distribution*):

$$\mathbf{E}\left[h(\mathbf{x})\right] \approx \frac{1}{m}\sum_{i=1}^{m} \frac{h(\mathbf{x}_i)\pi(\mathbf{x}_i)}{g(\mathbf{x}_i)} \tag{2}$$

which can be rewritten as:

$$\mathbf{E}\left[h(\mathbf{x})\right] \approx \frac{1}{m}\sum_{i=1}^{m} w_i h(\mathbf{x}_i) \tag{3}$$

where $w_i = \pi(\mathbf{x}_i)/g(\mathbf{x}_i)$ is the importance weight of the ith sample.

2 Sequential importance sampling

For high-dimensional problems it is often difficult to develop a good proposal distribution (i.e., one that closely approximates the target density π). One approach is to "build up" the proposal density sequentially, i.e.:

1. Decompose $\mathbf{x}$ into $\mathbf{x} = (x_1, \ldots, x_d)$, where each x_i may be multidimensional.
2. Construct the proposal distribution $g(\mathbf{x})$ from the product of its marginals, i.e.:

$$g(\mathbf{x}) = g_1(x_1)g_2(x_2|x_1)\ldots g_d(x_d|x_1, \ldots, x_{d-1}) \tag{4}$$

*The primary sources for most of this material are: "Monte Carlo Strategies in Scientific Computing," J.S. Liu, Springer, 2001; "Sequential Monte Carlo Methods in Practice," A. Doucet, N. de Freitas, N. Gordon, Springer, 2001; and the author's own notes.

One important advantage of this approach is that we can hope to obtain some guidance from the target density π as we build g. First, we decompose the target density and the importance weight in a similar manner:

$$\pi(\mathbf{x}) = \pi(x_1)\pi(x_2|x_1)\dots\pi(x_d|x_1,\dots,x_{d-1}) \quad (5)$$

$$w(\mathbf{x}) = \frac{\pi(x_1)\pi(x_2|x_1)\dots\pi(x_d|x_1,\dots,x_{d-1})}{g_1(x_1)g_2(x_2|x_1)\dots g_d(x_d|x_1,\dots,x_{d-1})} \quad (6)$$

If we let $\mathbf{x}_t = (x_1,\dots,x_t)$ then we can rewrite the weight recursively:

$$w_t(\mathbf{x}_t) = w_{t-1}(\mathbf{x}_{t-1})\frac{\pi(x_t|\mathbf{x}_{t-1})}{g_t(x_t|\mathbf{x}_{t-1})} \quad (7)$$

From this, we can see that this approach lets us:

- stop generating further components of the sample $\mathbf{x}$ if the "partial weight" $w_t(\mathbf{x}_t)$ at some time $t < d$ becomes too small; and
- use the marginal distribution $\pi(\mathbf{x}_t)$ to guide our choice of $g_t(x_t|\mathbf{x}_{t-1})$, i.e., pick a better proposal distribution.

Unfortunately decomposing π and w according to Equations 5-6 is hard:

$$\pi(\mathbf{x}_t) = \int \pi(x_1,\dots,x_d)\,dx_{t+1}\dots dx_d \quad (8)$$

(This is the whole reason for employing sampling strategies in the first place.) Thus, we cannot implement sequential importance sampling as we have described it.

Instead, suppose we can find a sequence of "auxiliary distributions" for each joint partial target density, $\pi_1(x_1), \pi_2(\mathbf{x}_2), \dots, \pi_d(\mathbf{x}_d)$, such that $\pi_t(\mathbf{x}_t)$ approximates $\pi(\mathbf{x}_t)$ up to a normalizing constant[1]. This allows us to construct the proper weights without knowing the "real" joint partial target densities. Given these auxiliary distributions, we can implement sequential importance sampling:

SIS step:

1: $X_t \sim g_t(x_t|\mathbf{x}_{t-1})$; $\mathbf{x}_t = (\mathbf{x}_{t-1}, X_t)$
2: $u_t = \dfrac{\pi_t(\mathbf{x}_t)}{\pi_{t-1}(\mathbf{x}_{t-1})g_t(x_t|\mathbf{x}_{t-1})}$
3: $w_t = w_{t-1}u_t$

Here, u_t is an "incremental weight" and as long as $\mathbf{x}_{t-1}$ is properly weighted by w_{t-1} with respect to π_{t-1}, $\mathbf{x}_t$ is also properly weighted by w_t with respect to π_t. We can use the auxiliary distributions in choosing a proposal distribution, e.g. (if possible):

$$g_t(x_t|\mathbf{x}_{t-1}) = \pi_t(x_t|\mathbf{x}_{t-1}) \quad (9)$$

A remaining critical question is how to pick the auxiliary distributions $\pi_t(\mathbf{x}_t)$. In general the choice is problem-dependent, but in nonlinear filtering scenarios $\pi_t(\mathbf{x}_t)$ is usually chosen to be the "current" posterior distribution.

[1] In other words, $\eta\pi_t(\mathbf{x}_t) = \frac{\pi_t(\mathbf{x}_t)}{\int \pi_t(\mathbf{x}_t)\,d\mathbf{x}_t} = \pi(\mathbf{x}_t)$

3 Sequential Monte Carlo

Just as in normal Monte Carlo estimation, we can generate m samples to do estimation. The only difference is that the samples are generated sequentially, using m independent SIS processes in parallel. Thus, at $t = 1$ we have partial samples $\{x_1^{(1)}, \ldots, x_1^{(m)}\}$ with $x_1^(i) \sim g_1(x_1)$, and weights $w_1^{(i)} = 1/m$. At time $t = 2$ we have $\{x_2^{(1)}, \ldots, x_2^{(m)}\}$ with $x_2^{(i)}$ drawn according to $g_2(x_2|x_1^{(i)})$ and having weight:

$$w_2^{(i)} = w_1^{(i)} \frac{\pi_2(x_1^{(i)})}{g_2(x_2|x_1^{(i)})\pi_1(x_1)} \tag{10}$$

As the sampling process proceeds, if the weight $w_t^{(i)}$ becomes too small for some partial sample, we can stop the generation of that sample to save computation. Then, we must replace the sample. One option is to repeat the SIS procedure from the beginning for that sample. From a computational standpoint, this is undesirable. Another more feasible option is to resample from among the currently available partial samples. We will discuss resampling in more detail.

4 Nonlinear filtering

Sequential Monte Carlo can be particularly useful in the context of nonlinear filtering in the state-space model. Here the problem is to estimate the state $\mathbf{x}$ from a sequence of observations or measurements $\vec{z}$. We have (probabilistic) models of the process and measurements:

$$x_t \sim g_t(x_t|x_{t-1}, \theta) \tag{11}$$
$$z_t \sim h_t(z_t|x_t, \phi) \tag{12}$$

where g_t is the process model at time t with parameters θ, and h_t is the measurement model at time t with parameters ϕ. Our goal is the online estimation and prediction ("filtering") of x_t when the z_t arrive sequentially. (For this discussion we will assume the parameters θ and ϕ are known, but they can be incorporated into the filtering process as well.)

Then the posterior $\pi_t(x_t)$ at time t can be computed recursively as:

$$\pi_t(x_t) = P(x_t|\vec{z}_t) \propto \int g_t(x_t|x_{t-1}) h_t(z_t|x_t) \pi_{t-1}(x_{t-1})\, dx_{t-1} \tag{13}$$

This is the well-known *Bayes filter* equation. (Note that the h_t term can be pulled out of the integrand since it is independent of x_{t-1}.)

4.1 Bootstrap/Particle filter

Bayes filtering can be done using sequential Monte Carlo methods with an approach known as the *bootstrap filter* or *particle filter*:

Bootstrap filter:

1: **for** $j = 1 \ldots m$ **do**
2: $\quad x_{t+1}^{(*j)} \sim f_t(x_{t+1}|x_t^{(j)})$
3: $\quad w^{(j)} \propto h_t(z_{t+1}|x_{t+1}^{(*j)})$

4: Resample with replacement from $\{x_{t+1}^{(*1)}, \ldots, x_{t+1}^{(*m)}\}$ with probability proportional to the weights $w^{(j)}$; store the new samples in $\{x_{t+1}^{(1)}, \ldots, x_{t+1}^{(m)}\}$

The approach is to first "predict" the evolution of the process, using the previous posterior as the proposal distribution, and then weight and resample the partial samples according to the likelihood of the most recent observation for each partial sample. The "current" target distribution which we are trying to approximate can be seen to be:

$$\pi_t(\mathbf{x}_t) \propto h_t(z_t|x_t) f_t(x_t|x_{t-1}) \pi_{t-1}(\mathbf{x}_{t-1}) \tag{14}$$

but because the state-space model is Markovian, we only need to estimate $\pi_t(x_t)$ instead of the full joint distribution $\pi(\mathbf{x}_t)$ to attain this approximation. This is essential for the bootstrap filter to be applicable.

There are two major problems with the bootstrap filter, which we will address later:

- It does not use all currently available information in the SIS step since it does not consider z_{t+1}, the current measurement, when sampling from the process model.
- It resamples every time step, which can decrease efficiency significantly if the current set of samples is already sufficient.

5 General SMC framework

We can think of sequential Monte Carlo as a *probabilisitic dynamic system* (PDS).

Definition 5.1. A *probabilistic dynamic system* is a sequence of PDFs $\pi_t(\mathbf{x}_t)$ indexed by discrete times $t = 0, 1, 2, \ldots$ where $\mathbf{x}_t$ can either increase in dimensionality with time ($\mathbf{x}_t \in \mathcal{S}_1, x_{t+1} \in \mathcal{S}_2, \mathbf{x}_{t+1} \in \mathcal{S}_1 \times \mathcal{S}_2$); or stay of the same dimensionality ($\mathbf{x}_t, \mathbf{x}_{t+1} \in \mathcal{S}_1$).

A PDS is really just a sequence of posterior distributions conditional on information "up to time t." The final PDF is constructed sequentially as new information is incorporated. Thus, in a PDS, the "configuration space" of the system changes with time.

In the state-space model where $\pi_t(\mathbf{x}_t) = P(\mathbf{x}_t|\vec{z}_t, x_0)$ (the posterior at time t), we can think of the estimation of x_t as an evaluation of the Monte Carlo average of a random variable under distribution π_t.

5.1 Sampling distributions

Recall one of the problems with the bootstrap filter: it does not use the current measurement in choosing a sampling distribution. The general problem of choosing a sampling distribution can be solved in several ways. Frequently, we choose

$$g_t(x_t|\mathbf{x}_{t-1}) = \pi_t(x_t|\mathbf{x}_{t-1}) \tag{15}$$

i.e., the sampling distribution is picked to be the posterior from the previous step. Another alternative is to employ a "coarsening" strategy, where we group a sequence of consecutive state variables $(x_{bt+1}, \ldots, x_{b(t+1)})$ into a "mega"-state x'_t, and then apply SIS to draw a block of b partial samples at a time.

Yet another approach is to employ some look-ahead if "future" information is available. The idea is to construct a sampling distribution that is as close to the future target density as possible. For example, a two-step forward look-ahead yields:

$$g_t(x_t|\mathbf{x}_{t-1}) \propto \int \pi_{t+1}(x_{t+1}|\mathbf{x}_t)\, dx_{t+1} \tag{16}$$

More generally, an s-step look-ahead gives:

$$g_t(x_t|\mathbf{x}_{t-1}) \propto \pi_{t+s}(x_t|\mathbf{x}_{t+1}) \quad (17)$$

$$= \int \pi_{t+s}(x_{t+s},\ldots,x_t|\mathbf{x}_{t-1})\,dx_{t+1}\ldots dx_{t+s} \quad (18)$$

The look-ahead approach tries to make use of as much future information as possible. If $t+s=n$, i.e. all future information is available, then g_t becomes the *ideal* sampling distribution ($\prod_t g_t = \pi_n$). Of course, the problem is that computing a forward-looking distribution is hard as s increases, but for small horizons it can be an effective strategy.

6 Resampling strategies

A problem with sequential Monte Carlo methods is that the variance in the importance weights $w_t^{(j)}$ increases with time. Thus after a while, there are a few partial samples with large weights and many with small weights. We would like to have "good samples" and not waste effort on "bad" ones. One approach is to discard bad samples and regenerate replacements from the start of the process, but this is computationally infeasible. Another more reasonable approach is to "resample" from among the current good samples. Several strategies can be employed for doing resampling.

6.1 PERM

A simple approach, known as the *prune-enriched Rosenbluth method* (PERM), is as follows:

PERM:

1: Define time-dependent upper/lower weight cuttoffs C_t/c_t
2: **if** $w_t^{(j)} > C_t$ **then**
3: Split $\mathbf{x}_t^{(j)}$ into r copies each with weight $w_t^{(j)}/r$
4: **if** $w_t^{(j)} \leq c_t$ **then**
5: Flip a fair coin
6: **if** coin is heads **then**
7: Keep $\mathbf{x}_t^{(j)}$ and set $w_t^{(j)} = 2w_t^{(j)}$
8: **else**
9: Discard $\mathbf{x}_t^{(j)}$

It can be shown that PERM is unbiased as long as C_t and c_t are specified in advance (i.e., not determined from the data). While this offers flexibility, it also significantly reduces the usefulness of the approach.

6.2 Simple random resampling

An even simpler approach is to just resample with replacement according to the importance weights:

Simple random resampling:

1: Draw with replacement m times from $\{\mathbf{x}_t^{(1)},\ldots,\mathbf{x}_t^{(m)}\}$ according to the $w_t^{(j)}$s
2: For $j=1\ldots m$ set $w_t^{(j)} = \frac{1}{m}\sum_{i=1}^m w_t^{(i)}$

This is the technique used in the vanilla bootstrap filter. It works reasonably well in practice and because of its simplicity is the most widely employed resampling approach, but with some more work we can do better.

6.3 Residual resampling

Rather than resampling randomly we can enforce that the number of copies retained of a partial sample is (approximately) proportional to the weight of the sample. This technique is known as *residual resampling*:

Residual resampling:

1: Let $w_t^{(*j)} = w_t^{(j)} / \sum_{i=1^m} w_t^{(i)}$
2: Retain $k_j = \lfloor m w_t^{(*j)} \rfloor$ copies of $\mathbf{x}_t^{(j)}$
3: Let $m_r = m - k_1 - \ldots - k_m$
4: Obtain m_r i.i.d. draws (with replacement) from $\{\mathbf{x}_t^{(1)}, \ldots, \mathbf{x}_t^{(m)}\}$ with probabilities proportional to $m w_t^{(*j)} - k_j$
5: For $j = 1 \ldots m$ set $w_t^{(j)} = \frac{1}{m} \sum_{i=1}^{m} w_t^{(i)}$

It can be shown that residual resampling dominates simple random resampling in that it has both smaller variance and computation time. Furthermore, it does not seem to have disadvantages in any other aspects.

6.4 A generalized strategy

A generalization of simple random resampling that does not directly use the importance weights but instead uses an "alternative" probability vector $(a^{(1)}, \ldots, a^{(m)})$ is as follows:

Generalized random resampling:

1: **for** $j' = 1 \ldots \tilde{m}$ **do**
2: Draw $\tilde{\mathbf{x}}_t^{(j')}$ i.i.d. from the $\mathbf{x}_t^{(j)}$s according to $(a^{(1)}, \ldots, a^{(m)})$
3: Suppose $\tilde{\mathbf{x}}_t^{(j')} = \mathbf{x}_t^{(j)}$; then set $\tilde{w}_t^{(j')} = w_t^{(j)} / a^{(j)}$
4: Return the $\tilde{\mathbf{x}}_t^{(j')}$s and $\tilde{w}_t^{(j')}$s

Note that it is important that $a^{(j)}$ be monotone in $w_t^{(j)}$, since our goal is to prune "bad" samples and copy "good" ones.

Using different probabilities than the importance weights can offer some useful flexibility. For example, the $a^{(j)}$s can be chosen to balance the need for sample "focus" (giving more presence to samples with high weights) with the need for diversity in the samples. One way to do this is to assign $a^{(j)} = (w_t^{(j)})^{\alpha}$ where $0 < \alpha \leq 1$ can vary according to, e.g., the variance in the importance weights. Choosing, e.g., $\alpha = 1/2$ amplifies the weight of seemingly poor samples slightly, giving them a "second chance." Note though that the resampling process does not reset the weights uniformly; thus, if a sample must still improve when given a second chance in order to survive in the next round of resampling.

6.5 Resampling schedules

A remaining question is: when should we resample? While the bootstrap filter resamples at every step, this can lead quickly to a lack of diversity in the samples. One approach is to pick a deterministic schedule, i.e. resampling at times $t_0, 2t_0, \ldots$ where t_0 is given in advance. Another approach is to examine the *effective sample size* N_{eff} and resample if it falls below some threshold, e.g. $m/2$. N_{eff} can be approximated as:

$$\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{m} (w_t^{(i)})^2} \tag{19}$$