5 Sequential Importance Sampling and Particle Filters

We next describe a sequential implementation of Importance Sampling, which applies to a sequence $X = (x_1, \dots, x_n)$ of random elements. We then consider an application and extension of this idea in the context of state estimation in dynamic systems, leading to the important Particle Filtering algorithm.

5.1 Sequential Importance Sampling

Recall the basic IS scheme for estimating $\ell = E_f(H(X)$:

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} H(X_i)W(X_i), \quad X_i \sim g,\ W(X_i) = \frac{f(X_i)}{g(X_i)}\,.$$

Suppose that X includes several components, $X = (x_1, \dots, x_n)$. These can simply represent the elements of a random vector, or, more interestingly, the components of a random process. Recall that $f(x_1, \dots, x_n)$ can always be decomposed as

$$f(x_1, \dots, x_n) = f_1(x_1)f_2(x_2|x_1)\cdots f_n(x_n|x_{1:n-1})\,.$$

Here we use the shorthand notation $x_{1:k} = (x_1, \dots, x_k)$. Note that while this decomposition is always valid, it often reflects the actual definition of X.

Let the IS (importance/proposal/trial) distribution g be defined through a similar decomposition:

$$g(x_1, \dots, x_n) = g_1(x_1)g_2(x_2|x_1)\cdots g_n(x_n|x_{1:n-1})\,.$$

We can now sample the elements of $X \sim g$ sequentially, namely:

1. Sample $x_1 \sim g_1(\cdot)$
2. For $k = 2$ to n, sample $x_k \sim g_k(\cdot|x_{1:k-1})$
3. Output $X = (x_{1:n})$, $W(X) = \dfrac{f(x_{1:n})}{g(x_{1:n})}$

Monte Carlo Methods – Lecture Notes, Nahum Shimkin, Spring 2015

The next step is to compute the weights recursively as well. Note that

$$W(X) = \frac{f_1(x_1)f_2(x_2|x_1)\cdots f_n(x_n|x_{1:n-1})}{g_1(x_1)g_2(x_2|x_1)\cdots g_n(x_n|x_{1:n-1})}\,.$$

This leads to the following *basic SIS scheme* for obtaining a single sample $X \sim g$ and $W(X)$:

1. Sample $x_1 \sim g_1(\cdot)$, set $w_1 = \dfrac{f_1(x_1)}{g_1(x_1)}$
2. For $k = 2$ to n:
   sample $x_k \sim g_k(\cdot|x_{1:k-1})$, set $w_k = \dfrac{f_k(x_k|x_{1:k-1})}{g_k(x_k|x_{1:k-1})} w_{k-1}$
3. Output $X = (x_{1:n})$, $W(X) = w_n$

Example 1: Random walk on the integers. [RK, Example 5.15]

Consider a random walk $x_0, x_1, \ldots$ on the integers, with probabilities p and $q = 1 - p$ for jumping up or down by one unit:

$$p(x_{k+1} = i+1|x_k = i) = p, \;\; p(x_{k+1} = i-1|x_k = i) = q, \quad i \in \mathbb{Z}\,.$$

Suppose $p < q$ (negative drift), and let $x_0 = 0$. Our goal is to estimate the following rare-event probabilities:

a. The probability that $x_n \geq K$, where K and n are given integers so that $1 \ll K < n$.

b. The probability that (x_k) reaches K before it reaches $-k$, where $k \ll K$.

a. Observe first that

$$f(x_k|x_{1:k-1}) = f(x_k|x_{k-1}) = p1_{\{x_k = x_{k-1}+1\}} + q1_{\{x_k = x_{k-1}-1\}}$$

As an IS distribution g we will a random walk with different parameters p_1 and $q_1 = 1 - p_1$, for which the events in interest become less rare (a reasonable first choice is $p_1 = q$). Therefore,

$$\frac{f_k(x_k|x_{1:k-1})}{g_k(x_k|x_{1:k-1})} = \frac{p}{p_1}1_{\{x_k = x_{k-1}+1\}} + \frac{q}{q_1}1_{\{x_k = x_{k-1}-1\}} \triangleq u(x_k|x_{k-1})\,.$$

The above SIS may now be applied.

b. In this case, the length of the sampled sequence is not fixed. Rather, we need to draw $x_1, x_2, \ldots$ up to the first time τ where either $-k$ or K are reaches. Observe that the time τ thus defined is a *stopping time* with respect to the process (x_k).

We can still use the SIS scheme, where now each sample X is the sequence $(x_1, \ldots, x_\tau)$, and the corresponding weight is given by

$$W(X) = \prod_{k=1}^{\tau} u(x_k | x_{k-1}) \,,$$

with an obvious recursive version.

Example 2: Self-avoiding random walk in the plane. [Liu, Section 3.1]

Simulations of molecular chains provided the original motivation for SIS-related algorithms, dating back to the 1950s, and are still of current interest in biology and chemistry. The self-avoiding random walk (SAW) in two or three dimensions is often used as a simple model for chain polymer growth. We describe the most basic model in two dimensions. A chain polymer of length n is described by $x = (x_1, \ldots, x_n)$, where $x_k = (a, b) \in \mathbb{Z}^2$ is the position of the k-th molecule, restricted to the two-dimensional integer grid. The distance between x_k and x_{k+1} has to be exactly one (a unit change in one coordinate), and no two molecules can occupy the same position.

Assuming equal energy for all legal configurations, the corresponding Boltzmann distribution is the uniform one. One would like to estimate relevant physical quantities of the chain, such as the mean square extension $E||x_n - x_1||^2$. A naive way to sample uniformly would be to start with $x_1 = (0, 0)$, and grow the chain sequentially by choosing $x_k - x_{k-1} \in \{(0, \pm 1), (\pm 1, 0)\}$ with equal probabilities. If self-intersection occurs the sequence is rejected. The problem with this method is that most samples are rejected: already for $n = 50$, less than 1% of the trials are retained.

An apparent remedy is to avoid occupied positions, and continue to sample randomly over the free adjacent ones. However, such a sampling strategy results in non-uniform distribution, which favors compact configurations (why?).

Using the SIS framework, we can identify the IS distribution g and associated weights. For a given subsequence $(x_1, \ldots, x_{k-1})$, $k \geq 3$, let $m_k \leq 3$ denote the number of free spots for x_k. Then x_k is chosen as one of these positions with equal probabilities (of $1/m_k$). It further follows that for any chosen x_k,

$$\frac{f_k(x_k | x_{1:k-1})}{g_k(x_k | x_{1:k-1})} = (?)$$

This implies

$$W(X) = (?)$$

We note that the problem of "attrition" (sample rejection) is not completely solved, as in some configurations there will not be any free neighbors ($m_k = 0$): the process runs into a dead end. This becomes severe again for very long chains ($n > 200$). A number of methods have been devised to further address the problem. We only mention one – that of looking several steps ahead, and avoiding moves that inevitably lead to an impasse (with a suitable compensation in the weights). In this context, the method described above can be considered as a one-step lookahead, and its extensions as multiple-step lookahead.

5.2 Degeneracy and Resampling

A major problem in sequential SIS is that of *sample degeneracy.* If n is large, the weights of most samples $(X_i, W(X_i))$ become small, while only a few remain significant. This does not allow accurate estimates. Many schemes have been proposed to address this issue. We briefly describe here the basic approach of *resampling* at intermediate steps. This will also lead us to the topic of particle filters for nonlinear state estimation, where this method has apparently originated.

Recall that in SIS we generate each sample $(X_i, W(X_i))$ recursively, by generating the intermediate sequence

$$(x_1^i, w_1^i),\ (x_{1:2}^i, w_2^i), \dots, (x_{1:n}^i, w_n^i)\,.$$

We conveniently refer to $(x_{1:k}^i, w_k^i)$ as the i-th *particle* at stage k. Here w_k^i is the weight of the particle, and $x_{1:k}^i$ its value. The degeneracy problem can be seen as the gradual decay (as k increases) of the weights w_k^i for most particles.

To apply resampling, we evolve all particle in parallel rather than sequentially. Consider the particle population $(x_{1:k}^i, w_k^i)_{i=1}^N$ after stage k. Some particles will have relatively large weights, other have smaller ones. The idea is to *split* particles with large weights into several particles with a unit weight, and eliminate other particles with small weights, before continuing to stage $k+1$. To avoid introducing bias, this can be done using random sampling as follows.

Random resampling: Sample N' particles (with replacement) from the particle population $(x_{1:k}^i, w_k^i)_{i=1}^N$, with probabilities p_i proportional to w_k^i, that is $p_i = w_k^i / \sum_i w_k^i$.

Let $(\tilde{x}^i_{1:k})_{i=1}^{N'}$ denote the values of the sampled particles. We next re-assign the weights evenly, namely set

$$\tilde{w}^i_k \equiv \tilde{w}_k \stackrel{\triangle}{=} \frac{1}{N'} \sum_{i=1}^{N} w^i_k \,,$$

and proceed to stage $k+1$ with the new particle set $(\tilde{x}^i_{1:k}, \tilde{w}^i_k)$.

Remarks:

1. The choice of $\tilde{w}_k$ above preserves the total weight of all particles. In the basic case where all particles are evolved an equal number of steps (n), the total weight need not be preserved as we may simply set $\tilde{w}^i_k = 1$ or $\frac{1}{N'}$. In the more general case where each particle may be evolved up to some stopping time, the total weight should be preserved.

2. Resampling does not introduce bias into the estimates. In particular, for a given particle set $(x^i_{1:k}, w^i_k)_{i=1}^N$, the sampled set satisfies

$$\frac{\sum_{i=1}^N w^i_k h(x^i_{1:k})}{\sum_{i=1}^N w^i_k} = E(h(\tilde{x}^i_{1:k})) = \frac{1}{N} E(\sum_{i=1}^{N} h(\tilde{x}^i_{1:k}))$$

   for any function h.

3. Resampling need not be carried out at each stage, but rather once every number of stages.

3. The number or resampled particles N' at each stage does not have to be the same as the original one N, although it commonly is.

4. There exist some *deterministic* resampling methods, which allegedly avoid the added variance introduced by random resampling. Care must be taken of course to avoid introducing bias.

5.3 Particle Filters

(separate slides)