VARIATIONS ON SEQUENTIAL MONTE CARLO METHODS
FOR INFERENCE IN PROBABILISTIC PROGRAMS

AN UNDERGRADUATE HONORS THESIS SUBMITTED TO
THE DEPARTMENT OF COMPUTER SCIENCE
OF STANFORD UNIVERSITY

Ben Mildenhall
Advisor: Pat Hanrahan
June 2015
Acknowledgements

I would like to thank Ron Fedkiw, Matt Pharr, Matt Fisher, Tom Duff, and all other members of the graphics community who have taught me, given me advice, and answered my questions over the course of the past three years.

Extra special thanks go to Daniel Ritchie, who has applied probabilistic programming in so many awesome ways and originally came up with the idea of using SMC for inference, and Pat Hanrahan, who has advised me both over the course of this work and in all things academic.
## Contents

Introduction iv

1 Sampling preliminaries 1
   1.1 Empirical densities and distributions .......................... 1
   1.2 Importance resampling .............................................. 2
   1.3 Sampling posteriors: a special case ............................. 4

2 Sequential Monte Carlo 6
   2.1 A simple model ..................................................... 6
   2.2 Extending the basic model ......................................... 9

3 Probabilistic programs and procedural modeling 13
   3.1 Applying SMC to probabilistic programming ..................... 13
   3.2 Procedural modeling ............................................... 14
   3.3 Designing models and constraints ................................ 15
   3.4 Global constraints ............................................... 18

A Convergence of basic SMC 20
   A.1 Framework ......................................................... 20
   A.2 Applied to SMC ................................................... 21
   A.3 The sampling operator $c^N$ .................................... 23
Introduction

In the context of computer graphics, probabilistic programming can be viewed as a method for automating parameter tweaking. Probabilistic inference allows users to invert high level constraints on the output of complex procedural models. Over the past five years, a variety of increasingly complex techniques for probabilistic inference have been brought to bear on the procedural modeling problem (for example, [4] and [5]).

Sequential Monte Carlo methods were originally developed for online filtering of a time-varying signal. The user provided a probabilistic model, and SMC provided a way to track a belief distribution about where the signal should be, given sequential noisy observations of its position. In a setting where the high level constraints on a model either have semantic meaning for partially completed models or can be broken up into multiple functions indicating partial progress on subcomponents of the model, SMC can be used to guide the growth of a model into regions of high probability.

This thesis is a self-contained introduction to SMC and its applications to probabilistic programming. We first lay the groundwork for the two main concepts underlying SMC: empirical distributions and importance resampling. Then we provide both the signal-processing formulation of SMC as well as its natural extension to a more general setting. Finally, we explain the benefits of using SMC for probabilistic programming and provide some practical advice for users. An accessible non-measure theoretic proof of convergence is also provided for the basic SMC algorithm.
Chapter 1

Sampling preliminaries

A note about notation: we will denote random variables by capital letters, such as $X_t$, and their instantiations as samples by lower case letters, such as $x_t$. We will always use $p$ to denote probability densities. When conditioning is involved, $p(a|b,c)$ is shorthand for $p(a|B=b,C=c)$ with random variables $A,B,C$. Since all these methods are intended to be run on a computer, we always assume densities exist and are available when relevant.

1.1 Empirical densities and distributions

The core idea of SMC involves using an evolving set of $N$ values, or particles, to approximate a sequence of probability distributions over time. Let us clarify what it means to approximate a distribution $F: \Omega \to [0,1]$ with density $p: \Omega \to [0,\infty]$ using a collection of particles $x^{(1)}, \ldots, x^{(N)} \in \Omega$ representing samples from $p$. We define the empirical probability density to be

$$\hat{p}_N(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x),$$

where $\delta$ is the Dirac delta function with $\int_{\Omega} f(x)\delta_i(x)dx = f(t)$. At first glance, this approximation seems quite poor, since it only places mass exactly at each of our particles. If the true density $p$ is bounded and continuous, then for $A = \{x^{(1)}, \ldots, x^{(N)}\}$ we will have $\int_A \hat{p}_N(x)dx = 1$ but $\int_A p(x)dx = 0$. This means that no matter how large $N$ is, the total variation distance between $\hat{p}_N$ and $p$ is equal to 1, the worst possible value! However, assum-
ing that we know no properties of the true density, we have no unbiased way of assigning
mass to any points in \( \Omega \) that we did not sample.

Luckily, our approximation is not so poor by some other metrics. If \( \Omega = \mathbb{R} \), then the
empirical distribution function is

\[ \hat{F}_N(x) = \int_{-\infty}^{x} \hat{p}_N(t) dt = \frac{1}{N} \sum_{i=1}^{N} \int_{-\infty}^{x} \delta_{x^{(i)}}(t) dt = \frac{1}{N} \sum_{i=1}^{N} 1\{x^{(i)} \leq x\} = \frac{\# \text{ samples} \leq x}{\text{total \# samples}}. \]

Here \( 1\{x \leq x\} \) is equal to 1 if \( x \leq x \), and 0 otherwise. We can also view our particles as
\( N \) independent random variables \( X^{(1)}, \ldots, X^{(N)} \) distributed according to \( p \) rather than fixed
sample values, in which case \( \hat{F}_N(x) = \frac{1}{n} \sum 1\{X^{(i)} \leq x\} \) also becomes a random variable.
Since

\[ E[1\{X^{(i)} \leq x\}] = \int_{-\infty}^{\infty} 1\{t \leq x\} p(t) dt = \int_{-\infty}^{x} p(t) dt = P(X^{(i)} \leq x) = F(x), \]

the strong law of numbers tells us that \( \lim_{N \to \infty} \hat{F}_N(x) = F(x) \) almost surely, or with prob-
ability 1, a more satisfying result. Pointwise convergence of distributions is also known as
weak convergence. (In the appendix, we show that the approximate distributions provided
by SMC converge weakly to the true distributions almost surely.)

The Glivenko-Cantelli theorem, an even stronger result about the convergence of empir-
cial distributions, tells us that

\[ \lim_{N \to \infty} \left( \sup_{x \in \mathbb{R}} |\hat{F}_N(x) - F(x)| \right) = 0 \]

with probability 1, so the convergence is uniform. Thus we can rest assured that in some
sense, the empirical approximation does converge to the true distribution as \( N \) increases.

### 1.2 Importance resampling

Importance resampling allows us to transform an approximation for one density into an
approximation for another. Suppose we take \( N \) samples \( x^{(i)}, \ldots, x^{(N)} \) from a density \( p \) to
form the empirical density \( \hat{p}_N = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}. \) However, we would now like an estimate for a
different distribution $G$ with density $q$. Instead of sampling directly from $q$, we can reweight our original samples with weights $w^{(i)} = q(x^{(i)})/p(x^{(i)})$, to get

$$
\hat{q}_N(x) = \frac{\sum_{i=1}^N w^{(i)} \delta_{x^{(i)}}(x)}{\sum_{i=1}^N w^{(i)}}.
$$

For densities defined on $\mathbb{R}$, we then have a corresponding empirical distribution

$$
\hat{G}_N(x) = \frac{\sum_{i=1}^N w^{(i)} 1\{x^{(i)} \leq x\}}{\sum_{i=1}^N w^{(i)}}.
$$

Once again, we can consider $\hat{G}_N(x)$ as a random variable, with $X^{(i)}$ distributed i.i.d. according to $p$. Applying the strong law of large numbers to the numerator and denominator shows

$$
\frac{\sum_{i=1}^N w^{(i)} 1\{X^{(i)} \leq x\}}{\sum_{i=1}^N w^{(i)}} \xrightarrow{N \to \infty} \frac{E[w^{(i)} 1\{X^{(i)} \leq x\}]}{E[w^{(i)}]},
$$

where for all $i$,

$$
E[w^{(i)}] = E\left[\frac{q(X^{(i)})}{p(X^{(i)})}\right] = \int \frac{q(t)}{p(t)} p(t) dt = \int q(t) dt = 1
$$

$$
E[w^{(i)} 1\{X^{(i)} \leq x\}] = \int \frac{q(t)}{p(t)} 1\{t \leq x\} p(t) dt = \int 1\{t \leq x\} q(t) dt = G(x).
$$

Just as in the case of the basic empirical distribution, we see that $\hat{G}_N(x) \to G(x)$ with probability 1. This convergence proof is from Smith and Gelfand [3].

A key feature of importance resampling is that we can also write our density as

$$
\hat{q}_N(x) = \sum_{i=1}^N \tilde{w}^{(i)} \delta_{x^{(i)}}(x), \quad w^{(i)} = \frac{w^{(i)}}{\sum_{i=1}^N w^{(i)}}.
$$

In this form, it becomes clear that the importance weights $\tilde{w}^{(i)}$ do not change if we scale $q$ by some positive number. This means that $q$ can be unnormalized, since we will get the same weights using either $q$ or

$$
q'(x) = \frac{q(x)}{\int q(x) dx}.
$$
1.3 Sampling posteriors: a special case

We can use importance resampling to turn an empirical prior distribution into an empirical posterior for Bayesian inference. Suppose our current beliefs about the distribution of a random variable $X$ are given by the prior $p(x)$. If we subsequently observe that $Y = y$ and have access to the likelihood function $L(x; y) = p(y|x)$, we can update our beliefs (exactly) using Bayes’ theorem:

$$p(x|y) = \frac{p(x,y)}{p(y)} = \frac{p(x,y)}{\int p(x,y)dx} = \frac{p(y|x)}{\int p(y|x)p(x)dx} \cdot p(x).$$

If we are tracking the distribution $p(x)$ with particles $x^{(1)}, \ldots, x^{(N)}$, then the importance weights for updating to $p(y|x)$ are given by

$$w^{(i)} = \frac{p(y|x^{(i)})}{\int p(y|x)p(x)dx} \propto p(y|x^{(i)}).$$

Note that the denominator $\int p(y|x)p(x)dx = p(y)$ is a constant since $y$ is fixed, so it ends up canceling out when we calculate the normalized weights:

$$\tilde{w}^{(i)} = \frac{p(y|x^{(i)})}{\sum_{i=1}^{N} p(y|x^{(i)})}.$$

Being able to condition on an observation $p(y|x)$ without calculating the normalizing constant $p(y)$ makes formulating constraints much easier, since one does not have to worry about whether an analytic formula for $p(y)$ exists.

**Example 1.3.1.** Suppose the random vector $X = (X_1, X_2, \ldots, X_d) \in \mathbb{R}^d$ is distributed according to some density $p$. We can form an empirical distribution by sampling $N$ particles $x^{(i)}$ from $p$. Given this setup, a common way to enforce a desired constraint $f(X) = y$ is to condition on an observation $y$ of a random variable $Y \sim \mathcal{N}(f(X), \sigma^2)$. This is often called “soft equality,” and the value of $\sigma$ determines how strictly the constraint is enforced. In this case, our normalized importance resampling weights would be

$$\tilde{w}^{(i)} = \frac{g(y - f(x^{(i)}))}{\sum_{i=1}^{N} g(y - f(x^{(i)}))}, \quad g(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{t^2}{2\sigma^2} \right).$$
Since we do not need to know the normalizing constant, we are saved from having to calculate the marginal density

\[ p(y) = \int_{\mathbb{R}^d} g(y - f(x))p(x)dx. \]

which is impossible to evaluate analytically for many reasonable choices of \( f \).
Chapter 2

Sequential Monte Carlo

2.1 A simple model

We begin by introducing a simple time-series model for SMC in order to build intuition for how the algorithm updates, weights, and resamples its set of particles at each time step. The notation here is modified from Crisan and Doucet [2]. Suppose we have two sequences of random variables, $X = \{X_1, X_2, \ldots\}$ and $Y = \{Y_1, Y_2, \ldots\}$, with $X_t \in S_1$ and $Y_t \in S_2$ for all $t \in \mathbb{N}$. From a time-series filtering perspective, $X$ represents the true signal and $Y$ represents the measurements that we are capable of making of the signal. For now, assume that $X$ satisfies the Markov property, i.e. the distribution of the signal at time $t$ depends only on the signal’s value at time $t - 1$:

$$p(x_t|x_{t-1}) = p(x_t|x_{1:t-1}, y_{1:t-1}).$$

In addition, assume the observation at time $t$ is randomly generated based only on the signal at time $t - 1$:

$$p(y_t|x_t) = p(y_t|x_{1:t}, y_{1:t-1}).$$

We use $K(x_t|x_{t-1})$ to denote the Markov transition kernel providing the density of $X_t$ given that $X_{t-1} = x_{t-1}$. The conditional density of $Y_t$ given $X_t$ is provided by $g(y_t|x_t)$. For now, we’ll keep $K$ and $g$ the same at all time steps.
We will use the notation $\pi_{k:m}(x_{k:t}) = p(x_{k:t}|Y_{1:m} = y_{1:m})$. Additionally, let $\pi_1(x_1) = p(x_1)$ be the original density from which $X_1$ is sampled. Given this setup, our goal is to use a set of particles $\{x_t^{(i)}\}_{i=1}^N$ with empirical density $\pi_t^N(x_t)$ to approximately track the true posterior density $\pi_t(x_t) = p(x_t|y_{1:t})$, given the measurements $y_{1:t}$. SMC does this by repeatedly transforming $\pi_{t-1|t-1}^N$ into $\pi_{t}^N$ using the following recursive relationships:

1. $\pi_{t|t-1}(x_t) = p(x_t|y_{1:t-1}) = \int p(x_t, x_{t-1}|y_{1:t-1})dx_{t-1} = \int K(x_t|x_{t-1})\pi_{t-1|t-1}(x_{t-1})dx_{t-1}$.
   
   (“Given what I know about $y_{1:t-1}$, where do I believe $x_t$ will fall?”)

2. $\pi_t(x_t) = p(x_t|y_{1:t}) = \frac{p(x_t, y_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{g(y_t|x_t)\pi_{t|t-1}(x_t)}{\int g(y_t|x_t)\pi_{t|t-1}(x_t)dx_t} \propto g(y_t|x_t)\pi_{t|t-1}(x_t)$.
   
   (“How do I modify my beliefs about the distribution of $x_t$ given this observation $y_t$?”)

Ideally, we would perform each of these steps analytically, but without restrictions on the form of $K$ and $g$, we must resort to approximate methods such as SMC. Given particles $\{x_{t-1}^{(i)}\}_{i=1}^N$ tracking the empirical density $\pi_{t-1|t-1}^N = \frac{1}{N} \sum_{i=1}^N \delta_{x_t^{(i)}}$, SMC uses the following two update steps to approximate the exact analytic case:

1. First, exploit the fact that the empirical density is a sum of delta functions to note that

$$\int K(x_t|x_{t-1})\pi_{t-1|t-1}^N(x_{t-1})dx_{t-1} = \frac{1}{N} \sum_{i=1}^N K(x_t|x_{t-1}^{(i)}).$$

We can thus update our empirical density from $\pi_{t-1|t-1}^N$ to $\tilde{\pi}_{t|t-1}^N$ (approximating $\pi_{t|t-1}$) by sampling $\tilde{x}_t^{(i)} \sim \frac{1}{N} \sum_{j=1}^N K(x_t|x_{t-1}^{(j)})$ for $i = 1, \ldots, N$. This step pushes forward our particles to represent guesses about the value of $X_t$, given our estimates for $X_{t-1}$.

---

**Figure 2.1:** The dependencies of the basic setup as a Bayesian network.
2. Since \( \pi_t | t(x_t) \) is a posterior distribution that is proportional to the product of a prior \( \pi_{t-1}(x_t) \) and a likelihood \( g(y_t|x_t) \) and we have a set of samples \( \tilde{x}_t^{(i)} \sim \tilde{\pi}_{t-1}^N \), we have the importance sampling approximation

\[
\tilde{\pi}_t^N = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{\tilde{x}_t^{(i)}}, \quad \tilde{w}_t^{(i)} = \frac{g(y_t|\tilde{x}_t^{(i)})}{\sum_{j=1}^{N} g(y_t|\tilde{x}_t^{(j)})},
\]

We resample \( N \) times from \( \tilde{\pi}_t^N \) to get the final updated particles \( x_t^{(i)} \) for our empirical density

\[
\pi_t^N = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_t^{(i)}}
\]

where the particles are equally weighted. This step hones our guesses about \( X_t \) by conditioning on the given information that \( Y_t = y_t \).

**Figure 2.2: SMC’s inductive step.**

**Example 2.1.1.** Suppose that \( X_t, Y_t \in \mathbb{R}^2 \), with

\[
K(x_t|x_{t-1}) = \mathcal{N}(x_{t-1} + (1, 0), \sigma_m^2 I)
\]

\[
g(y_t|x_t) = \mathcal{N}(x_t, \sigma_0^2 I).
\]

A model like this could be used to track the movement of a robot that was attempting to move 1 meter to the right at each time step. In this case, \( \sigma_m \) accounts for error introduced by the imperfection of the robot’s mechanical parts and \( \sigma_0 \) accounts for error in the robot’s measurements of its own position.

Figure 2.3 demonstrates the realization of a single SMC step in the case where \( \sigma_m = \sigma_0 = 0.1 \). We start in the top left with a set of particles \( \{x_{t-1}^{(i)}\} \). In the top right, we push the particles forward to \( \{\tilde{x}_t^{(i)}\} \) by sampling from \( K \). In the bottom left, we show the location of \( y_t \) (indicated by the square marker) and the relative weights (visualized by area) assigned
to each particle by the likelihood function $g(y_t|\cdot)$. In the bottom right we show the final resampled set of particles $\{x_t^{(i)}\}$. The more opaque markers indicate particles that have been resampled multiple times.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{smc_step.png}
\caption{One SMC step, visualized.}
\end{figure}

2.2 Extending the basic model

The above SMC framework is most suited to filtering time series data. In order to make SMC useable as a framework for inference in probabilistic programs, we will introduce a few modifications and generalizations. Most of these extensions do not require much mathematical justification.

One obvious and simple extension is making the kernels $K_t(x_t|x_{t-1})$ and observation
densities \( g_t(y_t|x_t) \) time dependent. In other words, we do not require the signal to evolve
the same way at each time step, and we allow each observation to have its own conditional
density. This modification changes nothing mathematically; the only difference is that each
occurrence of \( K_t \) or \( g_t \) now needs a subscript.

Much more significantly, we can lift our restrictions on the structure of the random
variables’ dependency graph. We can allow \( X_t \) to depend on any subset of the previously
sampled random variables, rather than just \( X_{t-1} \). Similarly, \( Y_t \) can depend on more than
just \( X_t \).

\[ g_t(y_t|x_t) \]

![Diagram of dependencies](image)

\[ K_t(x_t|x_U), g_s(y_s|x_V) \]

As pictured above, this extension allows us to apply SMC to more complex probabilistic
models, such as the one pictured on the right. We can adjust our notation to account for
the additional freedom which this provides. We will use capital letters such as \( U \) and \( V \) to
denote sets of indices, and write \( x_U \) as shorthand for \( \{x_u : u \in U\} \). Now our kernel and
observation density functions take the form \( K_t(x_t|x_U), g_s(y_s|x_V) \).

In this more general case, the variables \( X_t \) no longer have a total ordering, so there is
now some flexibility in the order in which we sample each \( X_t \). We are only constrained
by the partial order induced by the edges of the dependency graph. These edges are given
by the kernels \( K_t(x_t|x_U) \), which provide edges \( X_u \to X_t \) for each \( u \in U \), and the densities
\( g_s(y_s|x_V) \), which provide edges \( X_v \to Y_s \) for each \( v \in V \). Note that the subscripts for \( X_t \) and
\( Y_s \) are now distinct since there no longer exists a one to one correspondence between signal
Given that each \( X_t \) and \( Y_s \) can depend on an arbitrary subset of previously sampled variables, each particle cannot only represent a single variable \( X_t \) but must also track the full history of all variables previously sampled by that particle. We will use \( T \) to keep track of which variables we have sampled and \( S \) to keep track of which observations we have conditioned on. This also means that we will now be tracking the density \( p(x_T|y_S) \) rather than simply \( p(x_t|y_{1:t}) \). The fully generalized SMC algorithm can be seen in Figure 2.5.

**Figure 2.5: The generalized SMC algorithm.**

\[
T = \{0\}, \quad x_T^{(i)} \sim p(x_0)
\]
\[
S = \emptyset
\]
\[
\textbf{while } K_t, g_s \text{ remaining with } t \notin T \text{ or } s \notin S \text{ do}
\]
\[
\text{if } \exists g_s(y_s|x_V) \text{ with } V \subseteq T \text{ then}
\]
\[
w^{(i)} = g(y_s|x_V^{(i)}), \quad w = \sum_{i=1}^{N} w^{(i)}
\]
\[
\tilde{w}^{(i)} = w^{(i)}/w
\]
\[
\text{for } i = 1 \text{ to } N \text{ do}
\]
\[
j \sim \text{Multinomial}(\tilde{w}^{(1)}, \ldots, \tilde{w}^{(N)})
\]
\[
\tilde{x}_T^{(i)} = x_T^{(j)}
\]
\[
\text{end for}
\]
\[
S = S \cup \{s\}, \quad x_T^{(i)} = \tilde{x}_T^{(i)}
\]
\[
\text{else if } \exists K_t(x_t|x_U) \text{ with } U \subseteq T \text{ then}
\]
\[
\tilde{x}_t^{(i)} \sim K_t(x_t|x_U^{(i)})
\]
\[
T = T \cup \{t\}, \quad x_T^{(i)} = (x_T^{(i)}, \tilde{x}_t^{(i)})
\]
\[
\text{end if}
\]
\[
\textbf{end while}
\]

Our modified density update equations corresponding to the two halves of the while loop in Figure 2.5 are

\[
p(x_T|y_{S'}) \propto g_s(y_s|x_V)p(x_T|y_S) \quad \text{for } V \subseteq T, \quad S' = S \cup \{s\}
\]

\[
p(x_T|y_S) = K_t(x_t|x_U)p(x_T|y_S) \quad \text{for } U \subseteq T, \quad T' = T \cup \{t\}
\]

The pseudocode purposefully leaves the ordering choices ambiguous. However, it does indicate that each observation \( g_s \) should be conditioned on as soon as possible, that is, before any more kernels are used to sample additional variables. We will illustrate the benefits of
incorporating observations as soon as possible with an example.

**Example 2.2.1.** Suppose we have

\[
p(x_0) = \mathcal{N}(0,1), \quad p(y_0|x_0) \propto 1\{x_0 \geq 0\}
p(x_1|x_0) = \mathcal{N}(x_0, 1), \quad p(y_1|x_1) \propto 1\{x_1 \geq 0\}
\]

(Note that generally, it is inadvisable to have observation densities with finite support, since this can lead to a case where all particle weights are zero.) We will be comparing the performance of SMC when incorporating the \(y_0\) observation either after or before sampling \(X_1\). One metric by which we can judge SMC’s quality is its level of particle depletion, which we will define as the chance that any individual particle will have nonzero or nonnegligible importance weight at a given conditioning step.

First, consider sampling \(X_0\) and \(X_1\) subsequently without enforcing the \(x_0 \geq 0\) constraint. In this case, the probability that both constraints \(x_0 \geq 0\) and \(x_1 \geq 0\) are satisfied for an individual particle can be calculated to be \(3/8 = 0.375\) exactly.

Now, consider the case where first \(X_0\) is sampled, then the \(x_0 \geq 0\) constraint is enforced by resampling, and finally \(X_1\) is sampled. With numerical experiments, we can determine that the probability \(x_1 \geq 0\) is satisfied for an individual particle is approximately 0.750. The first constraint is guaranteed to hold for all particles because of the resampling step.

This particular case features two observations that somewhat “reinforce” one another, since if \(x_0 \geq 0\) holds it makes \(x_1 \geq 0\) more likely to be true. However, we can demonstrate the same benefit in a case where the opposite holds: change the first constraint to \(p(y_0|x_0) \propto 1\{x_0 \leq 0\}\) and leave the second one the same.

In this case, the chance an individual particle can survive the resampling step or steps after \(X_1\) is sampled is equal to exactly 1/8 = 0.125 with the first method and approximately 0.250 with the second method. The absolute performance is worse than in the reinforcing constraint case since here the two constraints contradict one another, but incorporating the constraint earlier still works better. The intuition behind why this is true is that every observation step depletes the quality of the particles, and every sampling step improves the quality, so applying multiple constraints in a row is bad for performance.
Chapter 3

Probabilistic programs and procedural modeling

3.1 Applying SMC to probabilistic programming

In a probabilistic programming language, regular programming language expressions and constructs combined with statements sampling from probability distributions together define a prior distribution over possible outputs. Condition statements allow a user to modify this distribution via Bayesian inference. It is the language designer’s job to create an engine to perform this inference. This is typically implemented using a variant of the Metropolis-Hastings algorithm or other more complex Markov chain Monte Carlo (MCMC) techniques (see [4], [5]). Whereas most of these methods run the whole program forward before evaluating the conditioning densities, SMC is able to incorporate a form of partial feedback by conditioning on observations at the point where they are placed in the program’s code.

In the extended framework described in the previous section, each random variable sampled in a probabilistic program’s execution trace serves as one of the variables $X_t$, and each condition statement provides both a likelihood (density function) $g_s$ and an observed value $Y_s = y_s$. The program’s structure and the order in which it defines variables implicitly provide the partial order that SMC needs to follow to honor dependencies. Since a program cannot use a variable it has not declared, it is impossible to reach a sampling or observation statement before defining every variable on which that statement depends.
Probabilistic programming languages provide more expressive power than Bayesian networks or factor graphs since they can succinctly encode probabilistic models that would be cumbersome to describe in other ways. For example, even the simple program in Figure 3.1 samples an unbounded number of random variables and thus cannot be represented as finite Bayesian network. Though this problem could be addressed by replacing the `while` loop by a single sample from a geometric distribution, with probabilistic programming it is not necessary for either the programmer and compiler to know that fact.

```
p = uniform(0,1)
x = 0
while (bernoulli(p)) {
    x = x + 1
}
condition(x,poisson(5))
```

*Figure 3.1: A simple probabilistic program.*

In a variety of situations, one of the benefits of probabilistic programming is that it can often supplant users’ technical knowledge with computational power. Unlike in SMC’s original signal processing context, in this domain the line between “signal” and “observation” is not so clear. A tradeoff exists between the complexity that users account for in their probabilistic model and the complexity that users enforce by conditioning on desired constraints. More complex models may be trickier to design and have a longer runtime, whereas more complex or stringent constraint functions can result in severe particle degeneracy and a dearth of usable results. For the remainder of this section, we will explore some of the finer points of applying SMC to modeling problems in particular.

### 3.2 Procedural modeling

Procedural modeling is a computer graphics technique used to generate large amounts of two and three dimensional content, usually in order to populate the virtual worlds underlying video games or films. For example, an artist might want to fill a field with grass or generate
coastlines for an entire planet. Rather than crafting each blade or border by hand, it is often easier for an artist to write a short set of rules (a probabilistic program) that can generate the desired content. Randomness is critical to these models, since the human brain will quickly dismiss any excessively uniform pattern as artificial.

Figure 3.2 shows one example how conditioning the output of such models can achieve an interesting effect. Here the procedural model simply walks and branches randomly in three dimensional space. The shadow cast by a typical sample from this model will has no significance whatsoever. But by conditioning on a function measuring the pixel-wise similarity of the model’s shadow to a target image like the one on the left, we can get outputs like the one on the right.

![Figure 3.2: A target image for a model’s shadow, left. One high-scoring sample from a procedural model with the target image constraint applied, right.](image)

This example also demonstrates the reusability of probabilistic programs. Simply by changing the target image or the constraint function, we can achieve another desired look using the same underlying program. However, crafting constraint functions to be effective while avoiding particle depletion takes care.

### 3.3 Designing models and constraints

This section will provide some advice in formulating models and constraints to yield interesting and nondegenerate results. In procedural models, the types of constraints usually fall into two categories:
1. **Hard constraints**, which must be satisfied for a model to be usable. For example, a model might randomly generate the positions and sizes of tables in a restaurant. Requiring the tables not to intersect one another would be a hard constraint.

2. **Soft constraints**, which have some degree of desirability. For example, in the above model, the user might want approximately $2/3$ of the floor’s area covered by tables, without being too insistent on the fraction’s exact value.

Hard constraints can sometimes be used when MCMC methods are being used for inference, since MCMC has many chances to adjust variables’ values over multiple runs of the program. However, in SMC the value of each variable is fixed once and for all the first time it is sampled because the program only runs forward once per particle. Any particle that does not satisfy the hard constraint is guaranteed to die off in the resampling step since its weight will be zero, so hard constraints accelerate particle depletion.

As such, it is better to incorporate rejection sampling into the model itself if possible, rather than conditioning on hard constraints. This means that variables subject to a hard constraint should simply be repeatedly resampled until the constraint in question is satisfied. The downside of rejection sampling is that it increases the model’s complexity and runtime to a degree dependent on the prior probability that the hard constraint is satisfied.

Soft constraints typically are encoded by a distance-to-target metric $d_f(X_T, X_{\text{target}}) = f(X_T)$ wrapped in a probability density $p$. A particularly common choice is $\mathcal{N}(f(X_T), \sigma_f^2)$ because the normal distribution is well understood and has only a single parameter to tune.

We can analyze the tradeoff between number of particles $N$ and the variance value for a particular Gaussian constraint $f$. Suppose that before the resampling step for $f$, the best particle $i$ has score $f(\tilde{x}^{(i)}_T) = \alpha$ and the next best particle has score $k\alpha$ (since lower scores are better, we must have $k \geq 1$). If our constraint density is $\mathcal{N}(f(X_T), \sigma_f^2)$, the particle $i$ will have a normalized importance weight $\tilde{w}^{(i)}$ such that

$$\tilde{w}^{(i)} \geq \frac{e^{-\alpha^2/(2\sigma^2)}}{e^{-\alpha^2/(2\sigma^2)} + (N-1)e^{-(k\alpha)^2/(2\sigma^2)}} \geq \frac{\beta}{\beta + N\beta k^2}$$
for $\beta = \exp(-\alpha^2/(2\sigma^2))$. We will thus have total particle degeneracy (every resampled particle will be a copy of particle $i$) with probability at least $1 - \varepsilon$ if

$$\left[\tilde{w}^{(i)}\right]^N \geq \left[\frac{\beta}{\beta + N\beta^{k^2}}\right]^N \geq 1 - \varepsilon$$

$$\frac{1}{1 + N\beta^{k^2 - 1}} \leq (1 - \varepsilon)^{1/N} \approx e^{-\varepsilon/N}$$

$$N\beta^{k^2 - 1} \leq e^{\varepsilon/N} - 1 \approx \frac{\varepsilon}{N}$$

$$(k^2 - 1) \log \beta \leq \log \left(\frac{\varepsilon}{N^2}\right)$$

$$k^2 - 1 \geq -\frac{2\sigma^2}{\alpha^2} \log \left(\frac{\varepsilon}{N^2}\right)$$

$$k \geq \sqrt{1 + 2(\sigma/\alpha)^2 \log(N^2/\varepsilon)}.$$ 

We apply the first order approximation $e^x \approx 1 + x$ twice in the above derivation, justified by the small magnitude of $\varepsilon$. Note that $\alpha/\sigma$ is the number of standard deviations of particle $i$'s score from the goal. The intuitive interpretation of this inequality is that if we expect our best particle to score about $s = \alpha/\sigma$ standard deviations away from the target, we want the ratio of the second best to first best particles’ scores to be significantly less than $\sqrt{1 + 2c/s^2}$ (where $c = \log(N^2/\varepsilon)$).

Practically speaking, $N$ will typically lie between 10 and 1000, and we will take $\varepsilon = .01$. This puts $c = \log(N^2/\varepsilon)$ in a range of 10 to 20. To be safe, we thus want $k \ll \sqrt{1 + 20/s^2}$. If our best result is about 1 standard deviation off, this means the next best result only has to be about 4 deviations off. However, if we tighten the constraint so that even the best expected result is 5 deviations off, then the next best result needs to stay within about 6.7 deviations!

A paradox arises here. In most situations, if we expect even our best result to be quite far off, the next best result will likely be a lot worse, unless our samples are tightly clustered. This means particle degeneracy is virtually guaranteed. Thus, using constraint functions where even the highest scoring particles are expected to land in the tails of the distribution is inadvisable.
We can rewrite the above inequality to get a heuristic for selecting $\sigma$:

$$\sigma \gg \alpha \sqrt{\frac{k^2 - 1}{2 \log(N^2/\varepsilon)}} \approx \frac{\alpha}{2} \sqrt{\frac{k^2 - 1}{2 + \log N}}$$

with the substitution $\varepsilon \approx e^{-4}$. Note that this expression decreases very gradually in $N$, so once particle degeneracy occurs, it is impractical to attempt to address it only by increasing the number of particles. Rather, $\sigma$ should be increased or the model itself should be modified in order to decrease $\alpha$ and $k$.

### 3.4 Global constraints

A common special case exists in procedural modeling where we have a single global constraint $g(Y|X_T)$ that can be evaluated on partially run programs. This occurs most often when trying to control spatial attributes of procedural model, for example, having it either fill in or avoid a region of two or three dimensional space. Also refer back to Figure 3.2, where we use an image-similarity function on the output of procedural model that constructively builds up itself up one geometric primitive at a time. In that case, we can easily evaluate the constraint on partially completed models to get an estimate of whether the model is “off too a good start.”

This situation arises in any case where the global constraint function permits greedy estimation, so that $g(Y|X_T) < g(Y|X'_T)$ implies $g(Y|X_U) < g(Y|X'_U)$ for all $U \subseteq T$ (i.e., models that are better in the end are also better when partially completed). Applying SMC when this assumption does not hold leads to final samples that approximate the distribution poorly since they will have been “misguided” early on.

We have to change our resampling step slightly to accommodate the fact that we now have what amounts to a single, global observation $g(y|x_T)$ rather than a collection of observations $g_s(y_s|x_T)$. We present a variation of Nicolas Chopin’s static SMC algorithm to accommodate this change [1]. Given two partial completions of a program $x_T$ and $x_{T'}$, where $T \subseteq T'$, let
\( U = T' - T \). We have the new update equation

\[
p(x_T'|y) = \frac{p(x_U, x_T'|y)}{p(x_T|y)} \cdot p(x_T|y) \\
= \frac{p(x_U, x_T, y)}{p(x_T, y)} \cdot p(x_T|y) \\
= \frac{p(y|x_U, x_T)p(x_U, x_T)}{p(y|x_T)p(x_T)} \cdot p(x_T|y) \\
= \frac{g(y|x_T')}{g(y|x_T)} \cdot p(x_U|x_T)p(x_T|y).
\]

Assume we have particles approximating \( p(x_T|y) \). As before, \( p(x_U|x_T) \) is just the result of running our program forward until it has sampled all the variables with indices in \( U \). Thus we see that the only modification necessary to update our particles from approximating \( p(x_U|x_T)p(x_T|y) \) to \( p(x_T'|y) \) is a resampling step with importance weights

\[
\tilde{w}^{(i)} \propto \frac{g(y|x_T^{(i)})}{g(y|x_T')}, \quad \sum_{i=1}^{N} \tilde{w}^{(i)} = 1.
\]
Appendix A

Convergence of basic SMC

A.1 Framework

We will closely follow the framework of Crisan and Doucet [2].

First, let $(E, d)$ be a metric space and let $\{a_t\}, \{b_t\}$ be sequences of continuous functions $a_t, b_t : E \to E$. Define $k_t$ and $k_{1:t}$ to be

$$k_t = a_t \circ b_t, \quad k_{1:t} = k_t \circ k_{t-1} \circ \cdots \circ k_1$$

where “$\circ$” denotes function composition, so $a_t \circ b_t(e) = a_t(b_t(e))$. Since $k_t$ and $k_{1:t}$ are composed of continuous functions, they are also continuous.

In the SMC case, $E$ will be a space of probability density functions (themselves defined on some space $S$), $b_t$ will map from $\pi_{t-1|t-1}$ to $\pi_{t|t-1}$, and $a_t$ will map from $\pi_{t|t-1}$ to $\pi_{t|t}$. Thus $k_t$ will map from $\pi_{t-1|t-1}$ to $\pi_{t|t}$ and $k_{1:t}$ will map all the way from $\pi_{0|0}$ to $\pi_{t|t}$. Carefully note that in this case $a_t, b_t, k_t$, and $k_{1:t}$ are operators mapping from functions to functions.

Since $k_{1:t}$ represents the optimal filtering operator, we need to insert functions $c^N : E \to E$ in order to account for SMC’s sampling step. Write $k_t^N = c^N \circ a_t \circ c^N \circ b_t$ and $k_{1:t}^N = k_t^N \circ \cdots \circ k_1^N$. We require that for all $e, e_N \in E$ with $e_N \to e$, we must have $c^N(e_N) \to e$ under the given metric $d$ (note that this is quite different from assuming that $c^N$ is continuous). Since any
continuous function \( f : E \to E \) will satisfy \( f(e_N) \to f(e) \) by definition, we have

\[
\begin{align*}
e^N & \to e \\
\implies b_t(e_N) & \to b_t(e) \\
\implies c^N(b_t(e_N)) & \to b_t(e) \\
\implies a_t(c^N(b_t(e_N))) & \to a_t(b_t(e)) \\
\implies c^N(a_t(c^N(b_t(e_N)))) & \to a_t(b_t(e)) \\
\implies k_t^N(e_N) & \to k_t(e).
\end{align*}
\]

By induction, we thus see that \( k_t^N(e_N) \to k_{1,t}(e) \). In the SMC setting, if \( u \) is the density for our original set of particles and \( u^N \) is the empirical distribution provided by the particles, this fact implies that

\[
\pi^N_{1,t} = k_t^N(u^N) \to k_{1,t}(u) = \pi_{1,t}
\]

as desired! All that remains is to show that the above assumptions make sense in the context of SMC.

## A.2 Applied to SMC

First, we must define the space \( E \) and a form of convergence in \( E \) for the context of SMC. The notion of convergence we will focus on for probability densities is weak convergence. Given some space \( S \), a probability density function \( q : S \to [0, \infty] \), and a continuous bounded function \( \varphi : S \to \mathbb{R} \), we will use the shorthand

\[
(q, \varphi) = \int_S \varphi(x)q(x)dx = E[\varphi(X)] \text{ for } X \sim q.
\]

We say that the sequence of densities \( u_N \) converges weakly to \( q \) if \( \lim_{N \to \infty}(q_N, \varphi) = (q, \varphi) \) for all continuous bounded functions \( \varphi : S \to \mathbb{R} \). Weak convergence is the same as convergence in distribution; if we have an equivalent sequence of distribution functions \( F_N \) and a limit function \( F \), convergence in distribution means that \( \lim_{N \to \infty} F_N(x) = F(x) \) for all \( x \).
Now we need to specify \( b_t, a_t, \) and \( c^N \). We define \( b_t \) to be the mapping such that

\[
b_t(q)(x_t) = \int_S K(x_t|x_{t-1}) u(x_{t-1}) dx_{t-1}.
\]

Then

\[
b_t(\pi_{t-1|t-1})(x_t) = \int_S K(x_t|x_{t-1}) \pi_{t-1|t-1}(x_{t-1}) dx_{t-1} = \pi_{t|t-1}(x_t).
\]

Using the notation

\[
(K\varphi)(x_{t-1}) = \int_S \varphi(x_t) K(x_t|x_{t-1}) dx_t
\]

(“what is the expected value of the function \( \varphi \) after mutating from starting position \( x_{t-1} \) using the kernel \( K \)?”) we see that

\[
(b_t(q), \varphi) = \int_S \varphi(x_t) \left[ \int_S K(x_t|x_{t-1}) u(x_{t-1}) dx_{t-1} \right] dx_t
\]

\[
= \int_S \left[ \int_S \varphi(x_t) K(x_t|x_{t-1}) \right] u(x_{t-1}) dx_{t-1}
\]

\[
= (q, K\varphi).
\]

Given this fact, instead of requiring \( b_t \) to be continuous we can substitute the stronger assumption that if \( \varphi : S \to \mathbb{R} \) is continuous and bounded, so is \( K\varphi \). This works since it implies that for any densities with \( q_N \to q \),

\[
\lim_{N \to \infty} (b_t(q_N), \varphi) = \lim_{N \to \infty} (q_N, K\varphi) = (q, K\varphi) = (b_t(q), \varphi)
\]

and thus \( b_t(q_N) \to b_t(q) \) (as \( \varphi \) was arbitrary), so \( b_t \) is continuous.

We define \( a_t \) to be the mapping such that

\[
a_t(q)(x_t) = \frac{g(y_t|x_t)q(x_t)}{\int_S g(y_t|x'_t)q(x'_t) dx'_t},
\]

thus \( \pi_{t|t} = a_t(\pi_{t|t-1}) = (a_t \circ b_t)(\pi_{t-1|t-1}) \). Note that if we write \( g_t = g(y_t|\cdot) \) then

\[
(a_t(q), \varphi) = \int_S \varphi(x_t) \left[ \frac{g_t(x_t)q(x_t)}{\int_S g(y_t|x'_t)q(x'_t) dx'_t} \right] dx_t = \frac{\int_S [\varphi(x_t) g_t(x_t)] q(x_t) dx_t}{\int_S g(y_t|x'_t)q(x'_t) dx'_t} = \frac{(q, \varphi g_t)}{(q, g_t)}.
\]
We can ensure $a_t$ is continuous by assuming that $g_t$ is a continuous, bounded, and strictly positive function. In this case, for any densities with $q_N \rightarrow q$ and any continuous bounded function $\varphi$ we will have

$$
\lim_{N \rightarrow \infty} (a_t(q_N), \varphi) = \lim_{N \rightarrow \infty} \frac{(q_N, \varphi g_t)}{(q_N, g_t)} = \frac{(q, \varphi g_t)}{(q, g_t)} = (a_t(q), \varphi).
$$

Thus $a_t(q_N) \rightarrow a_t(q)$, so $a_t$ is continuous.

### A.3 The sampling operator $c^N$

Finally, we need to define $c^N$. First of all, note that $c^N(q)$ is not constant given $N$ and $q$, since it maps $q$ to an empirical density approximation of $q$, which involves taking $N$ independent samples from $q$. Thus we have

$$
c^N(q) = \frac{1}{N} \sum_{i=1}^{N} \delta_{V_i}
$$

where the $V_i$ are identical independent random variables with common density $q$. This means that given a function $\varphi : S \rightarrow \mathbb{R}$ and densities $q_N, q$ with $q_n \rightarrow q$ weakly, we will prove that

$$
P[c^N(q_N) \rightarrow q] = P \left[ \lim_{N \rightarrow \infty} (c^N(q_N), \varphi) = (q, \varphi) \right] = 1
$$

rather than a deterministic result. Thus our final convergence result will hold almost surely (with probability 1).

To proceed, we define a sequence of random variables $Z_N = |(c^N(q_N), \varphi) - (q_N, \varphi)|$. Then we have

$$
\lim_{N \rightarrow \infty} Z_N = 0 \\
\iff \lim_{N \rightarrow \infty} |(c^N(q_N), \varphi) - (q_N, \varphi)| = 0 \\
\iff \lim_{N \rightarrow \infty} (c^N(q_N), \varphi) = \lim_{N \rightarrow \infty} (q_N, \varphi) = (q, \varphi)
$$
Thus our desired result is equivalent to having

$$P\left[ \lim_{N \to \infty} Z_N = 0 \right] = 1$$

$$\iff \forall \varepsilon > 0, \; P[Z_N \geq \varepsilon \text{ holds for infinitely many } N] = 0.$$ 

The Borel-Cantelli lemma tells us that the above statement will be true if

$$\sum_{N=1}^{\infty} P[Z_N \geq \varepsilon] < \infty$$

for each $\varepsilon > 0$. By Markov’s inequality, we know that

$$P[Z_N \geq \varepsilon] = P[Z_N^4 \geq \varepsilon^4] \leq \frac{E[Z_N^4]}{\varepsilon^4}$$

so it is sufficient to show once that $\sum_{N=1}^{\infty} E[Z_N^4] < \infty$. This simply requires some arithmetic:

$$E[Z_N^4] = E\left[ ((c^N(q_N), \varphi) - (q, \varphi))^4 \right]$$

$$= E\left[ \left( \int \left( \frac{1}{N} \sum_{i=1}^{N} \delta_{V_i}(x) \right) \varphi(x) dx - (q, \varphi) \right)^4 \right]$$

$$= \frac{1}{N^4} E\left[ \left( \sum_{i=1}^{N} \varphi(V_i) - (q, \varphi) \right)^4 \right]$$

Note that $E[\varphi(V_i)] = (q, \varphi)$ since each $V_i$ is distributed according to $q$. Thus if we let $\theta_i = \varphi(V_i) - (q, \varphi)$, we have $E[\theta_i] = 0$. Also note that if we let $||\varphi|| = \max_{x \in S} \varphi(x)$, we have

$$E[\theta_i^m] \leq E[(2||\varphi||)^m] = 2^m ||\varphi||^m.$$ 

When we expand the product

$$E\left[ \left( \sum_{i=1}^{N} \theta_i \right)^4 \right]$$
using the independence of the $\theta_i$ we get terms of the forms

$$E[\theta_i^4], \ E[\theta_i^3]E[\theta_i], \ E[\theta_i^2]^2, \ E[\theta_i^2]E[\theta_i]^2, \ E[\theta_i]^4.$$  

Because $E[\theta_i] = 0$, the only terms that survive are $E[\theta_i^4]$ and $E[\theta_i^2]^2$. Therefore

$$\frac{1}{N^4} E \left[ \left( \sum_{i=1}^{N} \varphi(V_i) - (q, \varphi) \right)^4 \right] = \frac{1}{N^4} \sum_{i=1}^{N} E[\theta_i^4] + \frac{4!/(2!2!)}{N^4} \sum_{i,j=1, i\neq j}^{N} E[\theta_i^2]E[\theta_j^2]$$

$$\leq \frac{1}{N^4} \sum_{i=1}^{N} 2^4 \|\varphi\|^4 + \frac{6}{N^4} \sum_{i=1}^{N} 2^4 \|\varphi\|^4$$  

$$= \frac{2^4 \|\varphi\|^4 N + 6 \cdot 2^4 \|\varphi\|^4 (N^2 - N)}{N^4}$$

$$\leq \frac{96 \|\varphi\|^4}{N^2}.$$  

Thus

$$\sum_{N=1}^{\infty} E[Z_N^4] \leq 96 \|\varphi\|^4 \sum_{N=1}^{\infty} \frac{1}{N^2} < \infty$$

as desired.

We have now shown that

$$P \left[ \lim_{N \to \infty} (c^N(q_N), \varphi) = (q, \varphi) \right] = 1.$$  

This concludes the proof of convergence.
Bibliography


