Introduction to Particle Filters

Rui-Yang Zhang

Contents

Chapter 1

Introduction

Consider the task of drawing samples from a known distribution π, supported on $\mathbb{R}^n$. This is a common task in statistics, and especially in Bayesian inference, as we would want to compute certain quantities (such as the posterior mean of a parameter of interest) that are dependent on integrals with respect to (w.r.t.) a probability distribution (expectation of the posterior distribution). These integrals are often of high dimensions and of complex form, which stops us from computing them directly by hand and turning to numerical approximations for help. Such complicated integrals are often called **intractable integrals**.

One fundamental idea of doing numerical approximations of such integrals (integrals that could be written as an expectation) is to use the Monte Carlo methods. Given a random variable X with probability measure π that admits a density (i.e. we have $\pi(dx) = \pi(x)dx$), we may wish to compute the integral of the form

$$\int f(x)\pi(dx) = \int f(x)d\pi(x) = \int f(x)\pi(x)dx = \mathbb{E}_X[f(X)] \tag{1}$$

where f is some arbitrary function. Note that the first equality above is a mere notation change - these two forms of integration w.r.t. a probability measure would be used interchangeably in the rest of this note. Assuming that we could obtain independent and identically distributed (i.i.d.) samples $X_1, X_2, \ldots, X_N$ of the target random variable/probability distribution X, we can approximate the above quantity by the empirical mean, which is

$$\mathbb{E}_X[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i). \tag{2}$$

This method of numerical integration is known as the **Monte Carlo method**, and the approximation is known as the **Monte Carlo approximations**.

As a conceptual remark, the Monte Carlo method is actually doing an approximation to the target distribution π using empirical distribution from the samples. Formally, given N i.i.d. samples $X_1, X_2, \ldots, X_N$ of the target random variable/probability distribution X, the Monte Carlo estimate of π is provided by

$$\pi(dx) \approx \sum_{i=1}^{N} \frac{1}{N}\delta_{X_i}(dx) \tag{3}$$

where δ_{X_i} is the Dirac delta function, and $\delta_{X_i}(dx) = 1$ when $x = X_i$ and $\delta_{X_i}(dx) = 0$ otherwise. Intuitively, we can think about this as approximating a density function using a histogram, as illustrated in Figure 1. This also allows us to derive the Monte Carlo approximation of the integral, as stated by Equation (2), by combining Equations (3) and (1), which is given by

$$\begin{aligned}\mathbb{E}_X[f(X)] &= \int f(x)\pi(dx) \approx \int f(x)\sum_{i=1}^{N}\frac{1}{N}\delta_{x_i}(dx)\\ &= \sum_{i=1}^{N}\frac{1}{N}\int f(x)\delta_{x_i}(dx) = \sum_{i=1}^{N}\frac{1}{N}f(x_i).\end{aligned}$$

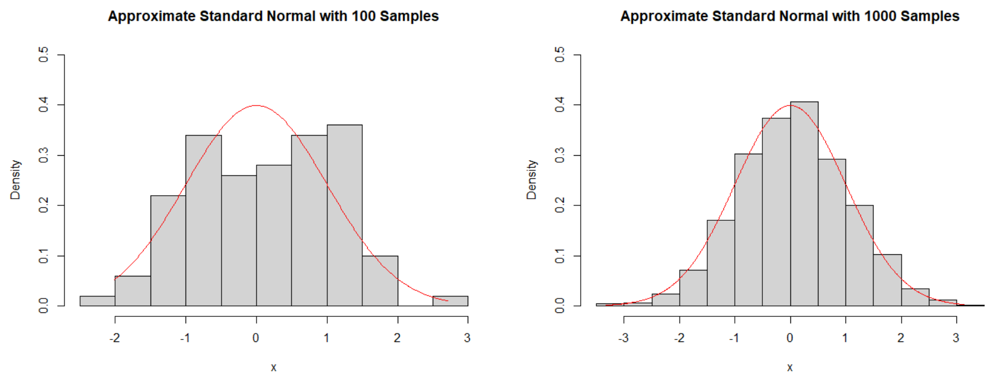

Figure 1: Simulations of approximating a standard normal using Monte Carlo samples. Sample sizes are 100 (left) and 1000 (right).

The Monte Carlo approximation is justified by the **law of large numbers** which states that given i.i.d. samples of a random variable with finite mean, the sample mean will converge, as the number of samples tends to infinity, to the true mean of the random variable. The actual modes of convergences are almost surely (for the strong law of large numbers) and in probability (for the weak law of large numbers). In addition, the above case involves the expectation of a function of a random variable. Its Monte Carlo approximation is justified using the uniform law of large numbers, as long as the function is continuous and is bound above by another function with finite mean. Further details and proofs of such theoretical results can be found in standard measure-theoretic probability theory textbooks such as Williams (1991).

Another benefit of the Monte Carlo method is that we could use the central limit theorem to provide a confidence interval for the approximation when the variance of the random variable (or the function of the random variable) is finite. Although the actual confidence interval would require the knowledge of true variance - which is often not accessible for numerical integration tasks - we could replace it with an empirical estimate of variance.

The Monte Carlo method is the fundamental principle that the rest of this report is based on. To actually conduct this method in practice would require us to have samples (and i.i.d. ones) of the target probability distribution. This challenge is the key question that the rest of this report (and multiple decades of academic research by thousands of researchers) is trying to tackle.

In this chapter, we will introduce Sequential Monte Carlo, and describe some of the common types of Sequential Monte Carlo algorithms with their respective properties. We will introduce some elementary sampling techniques in Chapter 2, such as rejection sampling and importance sampling. Then, we will introduce the hidden Markov model in Chapter 3 to set up the notations as well as the foundations for particle filters. The sequential importance sampling, a first attempt to approximate the hidden Markov models is provided in Chapter 4, and some of its problems are addressed in Chapter 5, which leads us to the full particle filters in Chapter 6.

The content of this note is mostly introductory, so exact references will be omitted. Standard textbooks such as Chopin and Papaspiliopoulos (2020), Liu (2001), and Robert and Casella (1999) would cover all of the material with more detailed references.

Chapter 2

Elementary Sampling Techniques

We will start the chapter with some elementary sampling techniques. Some of those would be developed and extended to sample from more complicated distributions.

2.1 Inverse CDF

Consider a continuous random variable X with an invertible (on the support[1]) cumulative distribution function (CDF) $F(x) = \mathbb{P}(X \leq x)$. This is a very simple setup. Such random variables include a normal distribution with mean μ and variance σ^2 or an exponential distribution with mean μ. We first have the following result.

Proposition 2.1. *For a continuous random variable X with invertible (on the support) cumulative distribution function F, the random variable $F(X)$ follows a Uniform$[0,1]$ distribution.*

Proof. For any $u \in [0,1]$ and x in the support of X, we have

$$\mathbb{P}(F(X) \leq u) = \mathbb{P}(X \leq F^{-1}(u)) = F(F^{-1}(u)) = u$$

as required for $F(X)$ to follow the uniform distribution on $[0,1]$. □

This result implies that if we can get the CDF of the target distribution, and we can invert it, then sampling from that distribution can be achieved by first sample from the Uniform$[0,1]$ distribution[2] then passing the sample through the inverse of the cumulative distribution F^{-1} will yield a sample from the target distribution. This method is known as the **inverse CDF** method for obvious reasons, and the procedure is listed below which generates a list of samples $\{X_i\}_i$.

Algorithm 1 Inverse CDF Algorithm

```
Require: Cumulative distribution function $F$ of the target random variable that is invertible.
1: for $i = 1, 2, \cdots$ do
2:     Draw $U \sim$ Uniform$[0,1]$.
3:     Set $X_i = F^{-1}(U)$.
4: end for
```

2.2 Rejection Sampling

Common distributions like an exponential distribution and a normal distribution can be sampled this way, but it is quite restrictive. Next, we will introduce the **rejection sampling** method.

[1]The support of a random variable/function is the subset of the domain where the random variable takes a non-zero value.

[2]which can be easily done by the random number generation functions on modern day computers.

We still assume the target random variable X is continuous, and we denote its probability density function as f. Consider we can find another (easy to sample from, say via inverse CDF) random variable Y with probability density function g such that there exists a constant $M > 0$ such that $Mg(x) \geq f(x)$ on the support of X, then we can draw samples from X using samples from Y in the following way.

Algorithm 2 Rejection Sampling Algorithm

```
Require: Target random variable $X$ with density $f$. Proposal random variable $Y$ with density
    $g$. Constant $M > 0$ such that $Mg(x) \geq f(x)$ on the support of $X$.
1: for $i = 1, 2, \cdots$ do
2:     Draw $y \sim Y$.
3:     Draw $U \sim \text{Uniform}[0, 1]$.
4:     if $U < f(y)/(Mg(y))$ then
5:         Set $X_i = y$.
6:     else
7:         return to Step 2
8:     end if
9: end for
```

First, we will prove that the above algorithm is doing the right thing.

Proposition 2.2. *The output of Algorithm 2 are indeed samples from the target distribution.*

Proof. For a sample X_i to be accepted, it must be first drawn from Y and then accepted with probability $f(X_i)/(Mg(X_i))$. This means we have

$$\mathbb{P}(X_i = x, X_i \text{ is accepted}) = \mathbb{P}(Y = x)\frac{f(x)}{Mg(x)} = g(x) \cdot \frac{f(x)}{Mg(x)} = \frac{f(x)}{M}$$

and

$$\mathbb{P}(X_i \text{ is accepted}) = \int_x \mathbb{P}(X_i = x, X_i \text{ is accepted})dx = \int_x \frac{f(x)}{M}dx = \frac{1}{M}.$$

So, using the Bayes formula, we have

$$\mathbb{P}(X_i = x \mid X_i \text{ is accepted}) = \frac{\mathbb{P}(X_i = x, X_i \text{ is accepted})}{\mathbb{P}(X_i \text{ is accepted})} = f(x)$$

as desired. □

Rejection sampling will draw (possibly) multiple samples from the proposal distribution to get a sample from the target distribution. The number of draws one needs to get a sample is clearly highly linked to the value of M. This relationship is made precise in the following result.

Proposition 2.3. *The number of draws from the proposal distribution until one draw is accepted follows a geometric distribution with a mean of M.*

The above result is easy to see as each draw is independent, and the success probability is $1/M$ for each draw, and the rest follows from the definition of a geometric distribution.

Therefore, we should pick the proposal distribution such that it is easy to sample from, but more importantly it is uniformly (i.e. across values in the support) close to the target distribution, so we would have a small M. It is often not very easy to choose a good proposal distribution that satisfies both these properties, especially when the distributions are of high dimensions and complicated form which prevents direct visualisation.

2.3 Importance Sampling

As an improvement of the rejection sampling, we will introduce the **importance sampling** algorithm. In rejection sampling, the rejected samples are simply omitted, but they still could be recycled and be somewhat useful - maybe not as a full sample from the target but as a part sample.

Consider the setup where we have a continuous target random variable X with density f and a proposal random variable Y with density g. Assuming we further have the condition that $f(x) > 0 \implies g(x) > 0$[3], we have

$$\int f(x)dx = \int \frac{f(x)}{g(x)} g(x)dx =: \int w(x)g(x)dx$$

where we define $w(x) = f(x)/g(x)$. This means, assuming we have $h(x)f(x) > 0 \implies g(x) > 0$, we can make the following statement

$$\begin{aligned}\mathbb{E}_X[h(X)] &= \int h(x)f(x)dx = \int h(x)\frac{f(x)}{g(x)}g(x)dx \\ &= \int h(x)w(x)g(x)dx = \mathbb{E}_Y[h(Y)w(Y)] \approx \frac{1}{N}\sum_{i=1}^{N} h(Y_i)w(Y_i)\end{aligned}$$

where $\{Y_i\}_{i=1}^N$ are i.i.d. samples of Y. This allows us to keep all the drawn samples from the proposal random variable, and reweigh them accordingly to make sure they are all samples from the target random variable. This is obviously a more versatile sampling algorithm than the other two described above. A formal description of the rejection sampling is included below.

Algorithm 3 Importance Sampling Algorithm

```
Require: Target random variable $X$ with density $f$. Proposal random variable $Y$ with density
    $g$. Function $h$ of the target integral. Require $h(x)f(x) > 0 \implies g(x) > 0$. Sample size $N$.
1: for $i = 1, 2, \cdots, N$ do
2:     Draw $X_i \sim Y$.
3:     Compute weight $w_i = f(X_i)/g(X_i)$.
4: end for
5: return approximate $\frac{1}{N}\sum_{i=1}^{N} w_i h(X_i)$ of $\mathbb{E}_X[f(X)]$.
```

Using the interpretation of the Monte Carlo method as approximations to the target distribution, the importance sampling of Algorithm 3 outputs an approximation

$$f(x) \approx \sum_{i=1}^{N} \frac{w_i}{N}\delta_{X_i}(x).$$

2.3.1 Proposal Distribution Assessments

One would like to make statements about the quality of the proposal random variable (and therefore the output) from the importance sampling algorithm to help us decide which proposal random variable we should use. Unlike in the case of rejection sampling where the quality of the proposal random variable can be directly reflected by the simple quantity of the constant M, things are slightly more complicated here.

One metric people can use to assess the quality of the estimator is by computing the **effective sample size**. This concept is very commonly used in numerical experiments to compare the

[3]in the case of f, g being probability measures, we would require f to be absolutely continuous with respect to g, which means that for any measurable subset A of the space f, g is defined on, we have $f(A) > 0 \implies g(A) > 0$.

quality and effectiveness of various algorithms. In the context of importance sampling, the effective sample size of the output of Algorithm 3 would be defined as

$$\text{ESS} = \frac{\left[\sum_{i=1}^{N} w_i\right]^2}{\sum_{i=1}^{N} w_i^2}.$$

It is not hard to notice that the value of ESS ranges between 1 and N. In the case where the proposed distribution is identical to the target distribution, each weight would be equal to 1, and we would get an ESS of N. In the case where the proposed distribution is very different from the target, we would expect most of the weights to be very small, and the ESS would be small too. In short, the larger the ESS the better. Intuitively, ESS denotes the value of the obtained samples in terms of independent samples from the target distribution. This should somewhat justify the use of ESS as a simple (and rough) indicator of the quality of the estimator.

Another way to assess the quality of the estimator is by looking at its variance. We know from the concept of the Monte Carlo method that such estimators are unbiased. If we denote the true integral to be I, and the importance sampling estimator with N samples to be $\tilde{I}_N$, then we have $\mathbb{E}[\tilde{I}_N] = I$ and for each sample $w(X_i)h(X_i)$, we have

$$\text{Var}_Y[w(X_i)h(X_i)] = \mathbb{E}_Y[w^2(X_i)h^2(X_i)] - I^2 = \int g(x)\frac{f^2(x)}{g^2(x)}h^2(x)dx - I^2.$$

Using the Jensen's inequality[4], we have

$$\mathbb{E}_Y[w^2(X_i)h^2(X_i)] \geq \mathbb{E}_Y[w(X_i)|h(X_i)|]^2 = \left(\int g(x)\frac{f(x)}{g(x)}|h(x)|dx\right)^2 = \left(\int f(x)|h(x)|dx\right)^2$$

by noticing $k(x) = x^2$ is a convex function. Therefore, the choice of g that minimises the variance would match the lower bound, and satisfy the identity

$$\int g(x)\frac{f^2(x)}{g^2(x)}h^2(x)dx = \left(\int f(x)|h(x)|dx\right)^2$$

which gives us the **optimal proposal density** g_{opt} defined as

$$g_{\text{opt}}(x) = \frac{|h(x)|f(x)}{\int |h(x)|f(x)dx}. \tag{4}$$

This result is merely a theoretical one, as obtaining this optimal density would require us to compute $\int |h(x)|f(x)dx$ which is as hard as the original task. This makes the problem a bit cyclical. The value of this result is mostly to set a goal for our selection of proposal density.

2.3.2 Unnormalised Distributions

A practical question to think about is we may only obtain an unnormalised version of the densities. This is common in Bayesian inference, for example, where the distributions of interest would be posterior distributions and the normalising constants are unknown. We will extend the importance sampling algorithm to cases where only unnormalised versions of the proposal and the target distribution are known.

Consider we have target X with density f, but we only know f_u defined by $f_u(x) = Kf(x)$ for unknown constant $K > 0$. Similarly, we assume we only know the unnormalised density g_u of the proposal Y with true density g such that $g_u(x) = Jg(x)$ for unknown constant $J > 0$. Importance sampling can still be conducted according to the following identity:

$$\mathbb{E}_X[h(X)] = \int h(x)f(x)dx = \frac{\int h(x)\frac{f(x)}{g(x)}g(x)dx}{\int \frac{f(x)}{g(x)}g(x)dx} = \frac{\int h(x)\frac{f_u(x)}{g_u(x)}g(x)dx}{\int \frac{f_u(x)}{g_u(x)}g(x)dx}.$$

[4] Jensen's inequality states that, for any convex function k, we have $\mathbb{E}[k(X)] \geq k(\mathbb{E}[X])$. A real-valued function k is **convex** if for any two points x, y in its domain, and $\alpha \in [0, 1]$, we have $k(\alpha x + (1-\alpha)y) \leq \alpha k(x) + (1-\alpha)k(y)$.

So, we can compute the weight using $w(x) = f_u(x)/g_u(x)$, and we just need to normalise it when computing the final estimate using a Monte Carlo approximation of the denominator above, which is

$$\mathbb{E}_X[h(X)] \approx \frac{\frac{1}{N}\sum_{i=1}^N h(Y_i)w(Y_i)}{\frac{1}{N}\sum_{i=1}^N w(Y_i)} = \sum_{i=1}^N \frac{w(Y_i)}{\sum_{j=1}^N w(Y_j)} h(Y_i) =: \sum_{i=1}^N W_i h(Y_i)$$

where we define the **normalised weight** $W_i := w(Y_i)/\sum_j w(Y_j)$. This version of the importance sampling is called the **auto-normalised importance sampling**. A formal algorithm is listed below, and its properties are identical to the standard importance sampling algorithm. The ESS of the auto-normalised would simply become

$$\text{ESS} = \frac{1}{\sum_{i=1}^N (W_i)^2}$$

by definition.

Algorithm 4 Auto-Normalised Importance Sampling Algorithm

```
Require: Target random variable $X$ with unnormalised density $f_u$. Proposal random variable $Y$
    with unnormalised density $g_u$. Function $h$ of the target integral. Require $h(x)f_u(x) > 0 \implies$
    $g_u(x) > 0$. Sample size $N$.
1: for $i = 1, 2, \cdots, N$ do
2:     Draw $X_i \sim Y$.
3:     Compute weight $w_i = f_u(X_i)/g_u(X_i)$.
4: end for
5: For each $i$, define the normalised weight $W_i = w_i/\sum_{j=1}^N w_j$.
6: return approximate $\sum_{i=1}^N W_i h(X_i)$ of $\mathbb{E}_X[f(X)]$.
```

Finally, using the interpretation of the Monte Carlo method as approximations to the target distribution, the auto-normalised importance sampling of Algorithm 4 outputs an approximation

$$f(x) \approx \sum_{i=1}^N W_i \delta_{X_i}(x).$$

Chapter 3

Hidden Markov Models

In this chapter, we will consider a class of models called **hidden Markov models** (HMM), also known as **state space models** (SSM). Hidden Markov models are extremely versatile and are able to capture time-dependent events where there is a true (Markovian) process - often called the signal process - that we cannot observe and a noisy process - often called the observation process - that depends on the true process which we can observe.

3.1 Examples of Hidden Markov Models

For example, if we wish to track a moving plane using radar stations, we can only make observations of the location of the moving plane using radar stations, say the distance and the angle, which need to be translated into the Euclidean coordinate system to reflect the 'true' location. This conversion needs to be done at each observation time. Formulated using an HMM, the signal process is the Euclidean location of the plane, while the observation process is the polar location of the plane according to the radar station. This is illustrated graphically in Figure 2.

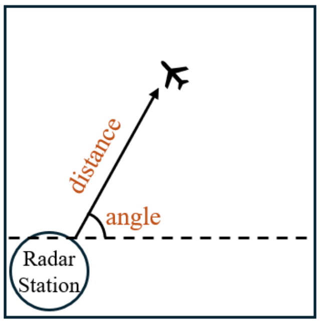

Figure 2: Tracking of plane movements using radar stations.

Another example of events that could be modelled by an HMM is the progression of infectious diseases in a closed (i.e. assuming no birth and death has happened) population. Using the framework of a compartmental model, we divide the population into disjoint compartments and model the change in the number of people in each compartment over time. For example, the SIR model divides the population into **S**uspectible (currently uninfected but can be infected), **I**nfectious (currently infected), and **R**emoved (immune to infection or dead after infection). Infection in this model happens with a fixed probability when a susceptible individual meets an infectious individual. The model dynamics assume that the population is well-mixed so each individual has an equal chance of being in contact with everyone else, and an equal chance of getting infected and an equal chance of getting removed. This leads to the following system of differential equations that can be used to model, in continuous time, the change in the number in each compartment. Assuming the population size is N, and we use S_t, I_t, R_t to denote the number of people in each of these three compartments at time t, we then have

$$\begin{cases} \frac{d}{dt}S_t &= -\beta \frac{I_t}{N} S_t \\ \frac{d}{dt}I_t &= \beta \frac{I_t}{N} S_t - \gamma I_t \\ \frac{d}{dt}R_t &= \gamma I_t \\ N &= S_t + I_t + R_t \end{cases}$$

where β is the infection rate constant and γ is the recovery rate constant. So far, we have described a physical model for the true underlying process. In reality, we would only be able to access noisy and partial versions of S_t, I_t, R_t through surveys or tests, which serve as the observation process. Such models of infectious diseases would be the key model of interest in this report, and would be further discussed in the following chapters.

3.2 Definition of Hidden Markov Models

It should be clear at this stage that hidden Markov models are useful modelling tools in many application areas, and are capable of incorporating expert knowledge about processes with observed data. We will now define an HMM formally.

Consider two discrete-time stochastic processes $\{X_t\}_t$ and $\{Y_t\}_t$. We let $\{X_t\}_t$ denote the signal process, and let $\{Y_t\}_t$ denote the observation process. We assume that each Y_t only depends on X_t, and the signal process is Markovian so that each X_t only depends on X_{t-1}. This relationship can be captured in the following illustration where the pointed arrows represent the conditional dependencies.

$$\begin{tikzcd} \cdots \arrow[r, "P"] & X_{t-1} \arrow[r, "P"] \arrow[d, "g"] & X_t \arrow[r, "P"] \arrow[d, "g"] & X_{t+1} \arrow[r, "P"] \arrow[d, "g"] & \cdots & \text{(signal)} \\ \cdots & Y_{t-1} & Y_t & Y_{t+1} & \cdots & \text{(observation)} \end{tikzcd}$$

We would describe the relationship between states using transition kernels and conditional distributions as labelled in the illustration. To simplify the notation, we would use $X_{a:b}$ to denote $(X_a, X_{a+1}, \dots, X_b)$. Also, we would use capital letters to denote random variables while using lowercase letters to denote their realisations. We have

$$\begin{aligned} X_0 &\sim \pi_0 \\ X_t \mid (x_{0:t-1}, y_{1:t-1}) &\sim P(dx_t|x_{t-1}) \\ Y_t \mid (x_{0:t}, y_{1:t-1}) &\sim g(y_t|x_t)d\nu(y_t) \end{aligned}$$

where π_0 is the initial distribution, P is the transition kernel of the signal process, and g is the conditional distribution of the observation process given the signal process. Note that P is a kernel and it does not need to admit a density, whereas g admits a density with respect to a reference measure ν which is usually the Lebesgue (or counting) measure.

There are four main tasks associated with a hidden Markov model like the one above: **predicting**, **filtering**, **smoothing**, and **parameter estimation**. In this section, we will give a brief description of them. Further studies will appear in later chapters.

The transition kernel P and the conditional distribution g usually depend on some parameters, and we denote the full vector of parameters by θ. The parameter dependency would not be made explicit most of the time to make the notation clean. In a Bayesian framework, we can think about an HMM as a Bayesian inference problem: π_0 is the prior distribution of the signal process, and as we make further observations $y_{1:t}$, we update our belief. The likelihood functions, denoted in general by p, are

$$p(x_{0:t}) = \pi_0(dx_0) \prod_{i=1}^{t} P(dx_i|x_{i-1}) \qquad p(y_{1:t}|x_{0:t}) = \prod_{i=1}^{t} g(y_i|x_i).$$

We can use the Bayes formula to get the posterior distribution of the signal process $X_{0:t}$ after observing $y_{1:t}$, which is given by

$$p(x_{0:t}|y_{1:t}) = \frac{p(x_{0:t}, y_{1:t})}{p(y_{1:t})} = \frac{p(x_{0:t})p(y_{1:t}|x_{0:t})}{\int p(y_{1:t}|x_{0:t})dx_{0:t}}.$$

The distribution $p(x_{0:t}|y_{1:t})$ above is called the **smoothing distribution**, and the task of finding it is called **(complete) smoothing**. Roughly speaking, this is the task of learning the distribution of the full trajectory of the signal process given all the available data.

Sometimes, we may be only interested in knowing the distribution of the current state in the signal process instead of the whole trajectory. We wish to find the conditional distribution of X_t given observations $y_{1:t}$, which is

$$\begin{aligned} p(x_t|y_{1:t}) &= \frac{p(x_t, y_{1:t})}{p(y_{1:t})} \\ &= \frac{g(y_t|x_t)p(x_t|y_{1:t-1})p(y_{1:t-1})}{p(y_t|y_{1:t-1})p(y_{1:t-1})} \\ &= \frac{g(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \\ &= g(y_t|x_t)\frac{\int P(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}}{\int g(y_t|x_t)p(x_t|y_{1:t-1})dx_t}. \end{aligned}$$

This distribution $p(x_t|y_{1:t})$ is called the **filtering** distribution, and the task of finding it is called **filtering**. Notice that on the right-hand side of the above equation, the integral in the denominator replies on knowing $p(x_t|y_{1:t-1})$, which is the full integral in the numerator - which depends on $p(x_{t-1}|y_{1:t-1})$. This indicates a highly iterative structure of the filtering distribution - the filtering distribution at time t depends on that at time $t-1$, which depends on that at time $t-2$, etc. Furthermore, the numerator of the right-hand side of the equation above is the **prediction distribution** $p(x_t|y_{1:t-1})$, which is essentially making the prediction of the next state in the signal distribution given all observations. The denominator $p(y_{1:t})$ of the right-hand side of the equation above is the **likelihood** of observing the data, which depends on the parameters θ. The likelihood would allow us to estimate θ, say using maximum likelihood estimation.

Therefore, we have described all four main tasks associated with an HMM. They are summarised below.

- **(predict)** Find $p(x_t|y_{1:t-1})$
- **(filter)** Find $p(x_t|y_{1:t})$
- **(smooth)** Find $p(x_{0:t}|y_{1:t})$
- **(parameter estimation)** Estimate θ using likelihood $p(y_{1:t})$

Chapter 4

Sequential Importance Sampling

In this chapter, we will directly apply some of the sampling techniques of Chapter 2 to solve some of the tasks described in Chapter 3. The focus would be put on solving the filtering problem in particular, as it is the easiest to solve and serves as a foundation for solving the rest of the problem. This will be a starting point for studying a hidden Markov model, which will be extended in the next Chapter.

4.1 A First Attempt

To solve the filtering problem of HMM, we could apply some techniques from importance sampling and exploit the conditional structure of HMM. We will use the same setup as in Chapter 3, and we will put the following illustration again to indicate the HMM we are interested in.

$$\begin{tikzcd}
\cdots \arrow[r, "P"] & X_{t-1} \arrow[r, "P"] \arrow[d, "g"] & X_t \arrow[r, "P"] \arrow[d, "g"] & X_{t+1} \arrow[r, "P"] \arrow[d, "g"] & \cdots & \text{(signal)} \\
\cdots & Y_{t-1} & Y_t & Y_{t+1} & \cdots & \text{(observation)}
\end{tikzcd}$$

The things that we assume we know about the above HMM, other than its structure, are the initial distribution π_0 of X_0, the transition kernel $P(dx_t|x_{t-1})$ of the signal process, and the conditional distribution $g(y_t|x_t)$ of the observation given the signal. The dependency on parameter vector θ is omitted to simplify the notation.

Recall that in Chapter 2, we introduce the (auto-normalised) importance sampling where we draw N samples from an (unbiased) proposal distribution g_u that is supported on a set which includes the support of the (unbiased) target distribution f_u, and we compute the (auto-normalised) weight of each sample to obtain a Monte Carlo approximation to the target density f. We will use the same algorithm here.

Starting from π_0, we will draw N samples (which we will also call **particles**) from it, denote them as x_0^i and compute their respective (auto-normalised) weight $W_0^{(i)}$ for $i = 1, 2, \ldots, N$. Note that if we can sample directly from π_0, the weights would be $1/N$ each. If not, we will use the auto-normalised weight as stated in Algorithm 4.

We wish to extend our samples from X_0 to samples from X_1, which is a step called the **propagation step**. This is relatively easy as we know the exact transition kernel P. For each sample x_0^i, we will draw its propagated sample $x_1^i \sim P(\cdot|x_0^i)$. The weight for each sample will remain unchanged at this step, i.e. $\tilde{w}_1^i = W_0^i$.

The HMM would make observations (y_t), and we would adjust the weights of the particles accordingly - a step called the **assimilation step**. Due to the fact that

$$p(x_1|y_1) \propto g(y_1|x_1)p(x_1),$$

we should account for the $g(y_1|x_1)$ factor into our samples from X_1. This gives us the assimilated unnormalised weight

$$w_1^{(i)} = g(y_1|x_1^{(i)})\tilde{w}_1^i$$

and the assimilated normalised weight

$$W_1^{(i)} = w_1^{(i)} / \sum_{j=1}^{N} w_1^{(j)}.$$

The propagation and assimilation steps transform the original particle-weight pair $(x_0^{(i)}, W_0^{(i)})_i$ of X_0 to the updated particle-weight pair $(x_1^{(i)}, W_1^{(i)})_i$ of X_1 given the observation y_1. Note that the pairs will form Monte Carlo approximations, i.e.

$$\pi_0(dx) \approx \sum_{i=1}^{N} W_0^{(i)} \delta_{x_0^{(i)}}(dx) \qquad\qquad p(dx_1|y_1) \approx \sum_{i=1}^{N} W_1^{(i)} \delta_{x_1^{(i)}}(dx_1).$$

Repeating the propagation and assimilation steps recursively until we have run out of observations would give us a vanilla version of the **sequential importance sampling** algorithm, and a formal version of the algorithm is listed below as Algorithm 5.

Algorithm 5 Vanilla Sequential Importance Sampling Algorithm

```
Require: Number of particles $N$. Observations $y_{0:T}$. Transition kernel $P(dx_t|x_{t-1})$ of the signal process. Conditional distribution $g(y_t|x_t)$ of the observation given the signal. Initial distribution $\pi_0$ of $X_0$.
1: Draw samples $x_0^i \sim \pi_0(dx_0)$ and assign weights $W_0^{(i)} = 1/N$ for $i = 1, 2, \dots, N$.
2: for $t = 1, 2, \cdots, T$ do
3:     (propagation) Draw samples $x_t^i \sim P(dx_t|x_{t-1}^i)$ and assign weights $\tilde{w}_t^i = W_{t-1}^i$ for $i = 1, 2, \dots, N$.
4:     (assimilation) Update weights $w_t^{(i)} = g(y_t|x_t^{(i)})\tilde{w}_t^i$ for $i = 1, 2, \dots, N$. Normalise weights by $W_t^{(i)} = w_t^{(i)} / \sum_{j=1}^{N} w_t^{(j)}$ for $i = 1, 2, \dots, N$.
5: end for
6: return approximate $\sum_{i=1}^{N} W_T^{(i)} \delta_{x_T^{(i)}}(dx_T)$ of the filtering distribution $p(x_T|y_{0:T})$.
```

An important remark at this stage is that the prediction distribution can be approximated as a by-product of the above algorithm. At the propagation step, the particle-weight pairs $(x_t^i, \tilde{w}_t^i)_i$ can serve as Monte Carlo samples for the prediction distribution $p(x_t|y_{1:t-1})$.

4.2 The General Version

The sequential importance sampling, as the name may suggest, is doing importance samplings sequentially. Let us first make the importance sampling explicit.

At time 1, we have the particle-weight pair $(x_0^{(i)}, W_0^{(i)})$ that yields a distribution $p_0(\cdot)$ which approximates π_0, and an observation y_1. The target distribution at this point is $p(x_1|y_1)$. The proposal distribution is $P(x_1|x_0)p_0(x_0)$ as we propagate $x_0^{(i)}$ using the transition kernel P. This gives us

$$\frac{p(x_1|y_1)}{P(x_1|x_0)p_0(x_0)} \propto \frac{g(y_1|x_1)P(x_1|x_0)\pi(x_0)}{P(x_1|x_0)p_0(x_0)} = g(y_1|x_1)\frac{\pi(x_0)}{p_0(x_0)} = g(y_1|x_1)W_0$$

so we get the weight update rule of $w_1^{(i)} = g(y_1|x_1^{(i)})W_0^i$. The weights are then normalised since our distributions are unnormalised to yield $W_1^{(i)} = w_1^{(i)} / \sum_j w_1^{(j)}$.

At time 2, we have the particle-weight pair $(x_1^{(i)}, W_1^{(i)})$ that yields a distribution $p_1(\cdot|y_1)$ which approximates $p(x_1|y_1)$, and an observation y_2. The target distribution at this point is $p(x_2|y_{1:2})$, and the proposal distribution is $P(x_2|x_1)p_1(x_1|y_1)$ as we propagate $x_1^{(i)}$ using the transition kernel P. This gives us

$$\frac{p(x_2|y_{1:2})}{P(x_2|x_1)p_1(x_1|y_1)} \propto \frac{g(y_2|x_2)p(x_2|y_1)}{P(x_2|x_1)p_1(x_1|y_1)} = g(y_2|x_2)\frac{p(x_1|y_1)}{p_1(x_1|y_1)} = g(y_2|x_2)W_1$$

so we get the weight update rule of $w_2^{(i)} = g(y_2|x_2^{(i)})W_1^i$. The weights are then normalised since our distributions are unnormalised to yield $W_2^{(i)} = w_2^{(i)}/\sum_j w_2^{(j)}$. The same logic and derivation are used to establish the rest of the update rules.

Note that at each of the above steps, we are choosing the proposal distribution simply by propagating the previous target distribution using the transition kernel P. This is certainly not the optimal choice compared to the optimal proposal density of Equation (4) outlined in Chapter 2. This motivates using a more general formulation of the sequential importance sampling which uses an arbitrary transition kernel Q that admits density to propagate the particles. At time 1, for example, we have the particle-weight pair $(x_0^{(i)}, W_0^{(i)})$ that yields a distribution $p_0(\cdot)$ which approximates π_0, and an observation y_1. The target distribution at this point is $p(x_1|y_1)$. The proposal distribution becomes $Q(x_1|x_0)p_0(x_0)$ as we propagate $x_0^{(i)}$ using the transition kernel Q. This gives us

$$\frac{p(x_1|y_1)}{Q(x_1|x_0)p_0(x_0)} \propto \frac{g(y_1|x_1)P(x_1|x_0)\pi(x_0)}{Q(x_1|x_0)p_0(x_0)} = g(y_1|x_1)\frac{P(x_1|x_0)}{Q(x_1|x_0)}\frac{\pi(x_0)}{p_0(x_0)} = \frac{P(x_1|x_0)}{Q(x_1|x_0)}g(y_1|x_1)W_0$$

which means the weight update rule becomes

$$w_1^{(i)} = g(y_1|x_1^{(i)})\frac{P(x_1|x_0)}{Q(x_1|x_0)}W_0^i \qquad\qquad W_1^{(i)} = \frac{w_1^{(i)}}{\sum_j w_1^{(j)}}.$$

The same applies to all the other iterations. It is not hard to notice that we recover the vanilla sequential importance sampling by setting $Q = P$. A formal version of the general sequential importance sampling is listed below as Algorithm 6.

Algorithm 6 General Sequential Importance Sampling Algorithm

```
Require: Number of particles $N$. Observations $y_{0:T}$. Transition kernel $P(dx_t|x_{t-1})$ of the signal
    process. Conditional distribution $g(y_t|x_t)$ of the observation given the signal. Propagation
    kernel $Q(dx_t|x_{t-1})$. Initial distribution $\pi_0$ of $X_0$.
1: Draw samples $x_0^i \sim \pi_0(dx_0)$ and assign weights $W_0^{(i)} = 1/N$ for $i = 1, 2, \ldots, N$.
2: for $t = 1, 2, \cdots, T$ do
3:     (propagation) Draw samples $x_t^i \sim Q(dx_t|x_{t-1}^i)$ and assign weights $\tilde{w}_t^i = W_{t-1}^i$ for $i =$
    $1, 2, \ldots, N$.
4:     (assimilation) Update weights
           $$w_t^{(i)} = g(y_t|x_t^{(i)})\frac{P(x_t^{(i)}|x_{t-1}^{(i)})}{Q(x_t^{(i)}|x_{t-1}^{(i)})}\tilde{w}_t^i$$
    for $i = 1, 2, \ldots, N$. Normalise weights by $W_t^{(i)} = w_t^{(i)}/\sum_{j=1}^N w_t^{(j)}$ for $i = 1, 2, \ldots, N$.
5: end for
6: return approximate $\sum_{i=1}^N W_T^{(i)}\delta_{x_T^{(i)}}(dx_T)$ of the filtering distribution $p(x_T|y_{0:T})$.
```

We could certainly obtain some theoretical optimal transition kernel using the optimal proposal density result of Equation (4), which is hardly attainable in practice.

Chapter 5

Particle Degeneracy and Resampling

In this Chapter, we will implement the sequential importance sampling on a simple example to illustrate how it works concretely, and highlight one particular practical problem - particle degeneracy - in Section 5.1, as well as how one could remedy this problem using resampling in Section 5.3.

5.1 Particle Degeneracy

Let us apply sequential importance sampling on a toy example of a linear Gaussian model defined by

$$\begin{aligned} X_0 &\sim N(0,1) \\ X_t &= \alpha X_{t-1} + \sigma U_t \\ Y_t &= X_t + \sigma V_t \end{aligned} \tag{5}$$

for $t = 1, 2, \ldots$ where U_t, V_t are i.i.d. standard Gaussians, α, σ are constants. The filtering of the above linear Gaussian model could in fact be solved using a special tool called the Kalman filter, which we will not use here. Instead, we will apply the vanilla sequential importance sampling of Algorithm 5 to illustrate concretely how this algorithm works.

We will use $N = 10$ samples, and we will generate observations using Equation 5 for 5 time steps. The transition kernel is

$$P(x_t|x_{t-1}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x_t - \alpha x_{t-1}}{\sigma}\right)^2\right)$$

which is the density of $N(\alpha x_{t-1}, \sigma^2)$. The conditional distribution is

$$g(y_t|x_t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_t - x_t}{\sigma}\right)^2\right)$$

which is the density of $N(x_t, \sigma^2)$. The initial distribution π_0 will be the standard normal distribution. For the linear Gaussian model of Equation (5), we set $\alpha = 0.9$ and $\sigma = 1$, and we will look at five timesteps.

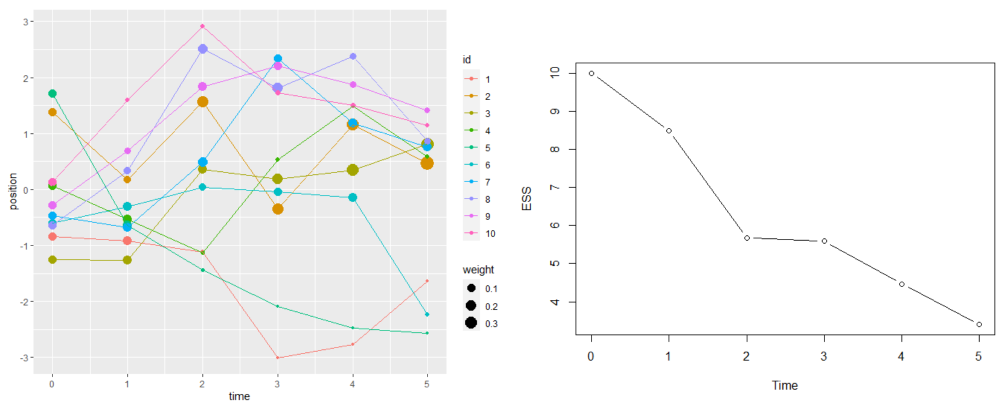

Figure 3: Filtering of a Linear Gaussian Model using Vanilla Sequential Importance Sampling. (Left) the trajectory of the particles over time with sizes representing weights. (Right) the effective sample size of the particles over time.

The result can be found in Figure 3. Notice that as time progresses, the weights are increasingly dominated by a few particles, and the effective sample size drops significantly, representing a drastic deterioration of the sample quality. The phenomenon of a small number of particles holding a majority of the weights is frequently called the **particle degeneracy**, or the **particle collapse**. This is quite a severe problem as illustrated in Figure 3, since it took only five timesteps for the ESS to drop from 10 to 3.

5.2 Particle Weights and Informative Observations

Here, we try to provide an intuition on why the ESS of particle weights decay as they did in the case of Figure 3. For importance sampling, as we have derived in Chapter 2, the quality of the proposal distribution is closely linked to how close it is to the target distribution - the closer, the better. The assessment we used to quantify (in an ad hoc manner) the quality of the proposal is also via ESS. In the case of the linear Gaussian example, we notice that at each iteration, the proposal distribution is just the propagation of the previous filter distribution, and the target distribution is that multiplies the information from the observation (up to the normalising constant). Therefore, the gap between the proposal and the target will deviate greatly if the information from the observation is significant, as one would hypothese.

To verify our hypothesis, at least using some experiments, we adjust the linear Gaussian example above by considering

$$\begin{aligned} X_0 &\sim N(0,1) \\ X_t &= \alpha X_{t-1} + \sigma U_t \\ Y_t &= \beta X_t + \sigma V_t \end{aligned} \tag{6}$$

for $t = 1, 2, \ldots$ where U_t, V_t are i.i.d. standard Gaussians, α, σ, β are constants. This added β indicates the (linear) dependency between the observation and the signal. Notice that, for very big β, the value of Y_t would be dominated by the influence of the signal X_t, while for very small β, the value of Y_t would be dominated by the influence of the noise σV_t - when everything else remains unchanged. Figure 4 lists three graphs of the ESS (each starting at the same seed to make the comparison fair) for varying values of β: $\beta = 1/3, 1, 3$.

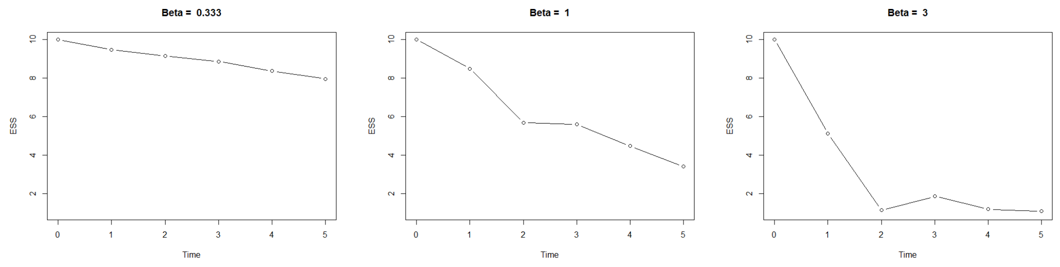

Figure 4: Effective sample sizes (ESS) over time for sequential importance sampling on linear Gaussian models with 10 particles. The signal and observation dependency coefficient β varies: (left) $\beta = 1/3$, (middle) $\beta = 1$, (right) $\beta = 3$.

The results align with our hypothesis. Therefore, as the observations are increasingly informative about our signals, the vanilla sequential importance sampling performs decreasingly well. In which case, the general sequential importance sampling of Algorithm 6 with a conditional distribution g informed propagation kernel would be preferred.

5.3 Resampling

A remedy to the problem of particle degeneracy is **resampling**. If at each iteration, we would resample/bootstrap the particles according to their weights, then we would get a collection of particles with balanced weights, which would yield a slower decay of ESS. Intuitively, we are removing particles with small weights and replicating particles with large weights. To ensure the particles still have diversity, we would do the resampling before propagation so that the randomness of the propagation step would recover some diversity. The key problem in resampling is which type of resampling is performed.

There are many types of resamplings - multinomial, residual, stratified, and systematic, to name a few. A more in-depth discussion on various resampling techniques and their comparisons can be found in Chapter 9 of Chopin and Papaspiliopoulos (2020). There, the authors recommend using **systematic resampling**, which we will discuss below.

Given N particles with normalised weights $(W_i)_{i=1}^N$, we wish to generate a N-vector of indices $(A_i)_{i=1}^N$ representing the resampled particles. The indices can be repeated. The starting point is to use an inverse CDF method to draw N samples from a multinomial distribution with weights according to $(W_i)_i$. We consider the cumulative weights

$$C_0 = 0 \qquad C_n := \sum_{i=1}^{n} W_i$$

for $n = 1, 2, \ldots, N$. Then, the inverse CDF methods would yield $(A_i)_i$ using N samples $(U_i)_i$ from Uniform$[0, 1]$ by setting

$$A_m = n \Longleftrightarrow C_{n-1} \leq U_i \leq C_n.$$

The correctness of this method is supported by the inverse CDF method justification outlined in Chapter 2.

Building up on this, we would wish to use a better method such that the variance of the drawn samples is reduced. This can be achieved by using a quasi-Monte Carlo approach and the common random number.

Quasi-Monte Carlo uses similar ideas from Monte Carlo, where we still wish to use empirical distribution built by samples to approximate the target distribution. The difference is that Monte Carlo draws the samples randomly, whereas quasi-Monte Carlo uses evenly spaced out (and deterministic!) points to represent the samples. **Common random number** reduces the variance among samples directly by noticing a lot of the samples are drawn using a set of randomly generated values. If the same value is used for all of the randomly generated values, the variance of the samples would then be much smaller.

In this case, the randomness occurs from the N draws from Uniform$[0,1]$. This can be improved by first dividing the $[0,1]$ interval into N equally sized smaller intervals $[(n-1)/N, n/N]$ for $n = 1,2,\dots,N$; and second drawing a uniform random variables in each of the small intervals. This guarantees the drawn samples are evenly spaced out to represent a more complete picture of the target distribution - ultimately improving the quality of the approximation. Next, instead of drawing each of the uniform random variables from the small intervals independently, we will draw only one, say from $[0,1/N]$, and extend this to the other intervals. To be more precise, the N draws from the Uniform$[0,1]$, denoted by $(U_i)_i$, will now be obtained as

$$U_1 \sim \text{Uniform}[0,1/N], \qquad U_n = U_1 + (n-1)/N$$

for $n = 2,3,\dots,N$. The draws are also ordered, as they are all increasing, so we would denote them by $(U_{(i)})_i$ to highlight that.

Combining them yields the following algorithm of systematic resampling.

Algorithm 7 Systematic Resampling

```
Require: Normalised weights $(W_i)_{i=1}^N$.
 1: Draw $U \sim$ Uniform$[0,1]$.
 2: Compute the cumulative weights $v_i = N\sum_{m=1}^i W_m$ for $i = 1,2,\dots,N$.
 3: Set $s \leftarrow U$.
 4: Set $m \leftarrow 1$.
 5: for $i = 1,2,\cdots,N$ do
 6:     while $v_i < s$ do
 7:         Set $m \leftarrow m+1$.
 8:     end while
 9:     Set $A_i \leftarrow m$.
10:     Set $s \leftarrow s+1$.
11: end for
12: return Indices $(A_i)_{i=1}^N$.
```

The outputted indices $(A_i)_i$ would be used to decide which particles are used to do propagation and assimilation. Using the same example in Section 5.1 but with resampling added at each sequential importance sampling iteration, we have the following result as shown in Figure 5. It is not hard to notice that the ESS deteriorates much more slowly than before in Figure 3.

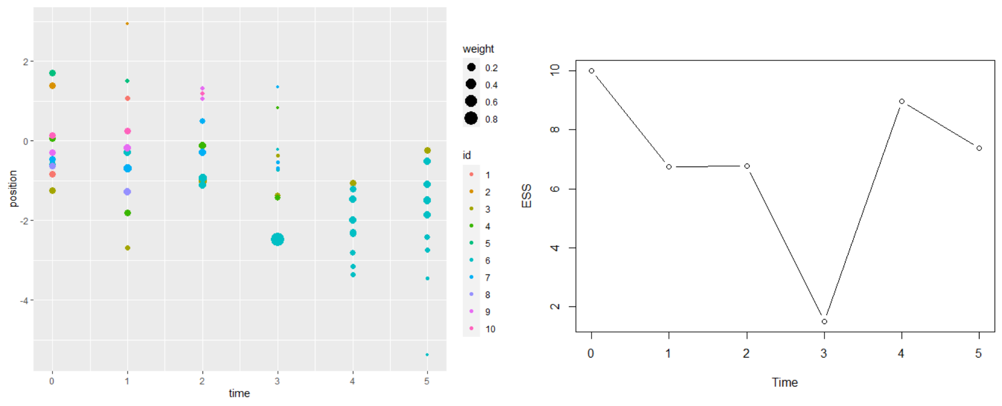

Figure 5: Filtering of a Linear Gaussian Model using Vanilla Sequential Importance Sampling with Systematic Resampling. (Left) the trajectory of the particles over time with sizes representing weights. The colour of the particles denotes the original index of the particle at $t = 0$. (Right) the effective sample size of the particles over time.

Chapter 6

Particle Filters: Bootstrap, Guided, and Auxiliary

In this chapter, we will combine the ideas derived from the previous chapters to introduce **particle filters**, also known as **sequential Monte Carlo** (SMC). There are three fundamental types of particle filters - bootstrap particle filter in Section 6.1, guided particle filter in Section 6.2, and auxiliary particle filter in Section 6.3.

6.1 Bootstrap Particle Filter

The **bootstrap particle filter**, also known as **sequential importance resampling**, is exactly the vanilla sequential importance sampling algorithm of Algorithm 5 with resampling at the start (before propagation) of each iteration. We will use by default the systematic resampling of Algorithm 7 in this report unless stated otherwise.

Resampling is beneficial as it stochastically removes particles with low weights and replicates particles with high weights, allowing future states of the processes to be better explored. However, resampling would always increase the Monte Carlo error of the approximation to the filtering distribution by introducing more variance. Therefore, a common ad hoc practice, which we would adopt here too, is to only trigger the resampling when the variance of particle weights is too high, assessed using ESS. We would set a threshold ESS_{min}, usually at $N/2$ where N is the number of particles, and trigger the resampling step whenever the ESS of the weights is lower than the threshold. This version of resampling is called **adaptive resampling**.

Combining the two, we have the following formal description of the bootstrap particle filter as Algorithm 8.

6.2 Guided Particle Filter

Just like we have developed the general sequential importance sampling of Algorithm 6 as a generalised of the vanilla sequential importance sampling of Algorithm 5 to incorporate the possibility of using more optimal proposal distribution, we can do the same to generalised the bootstrap particle filter. This gives us the **guided particle filter**, as formally described in Algorithm 9 below.

The reason why this algorithm is called the 'guided' algorithm is that, ideally, the propagation kernel Q is going to match the target filter distribution better, i.e. guiding our particles to move in the directions that give a higher weight. Recall that the bootstrap particle filter of Algorithm 8 completely ignores the observation when propagating the particles forward, here a good choice of propagation kernel Q should take the observation into account and direct the particles accordingly.

Algorithm 8 Bootstrap Particle Filter

```
Require: Number of particles $N$. Observations $y_{0:T}$. Transition kernel $P(dx_t|x_{t-1})$ of the signal
    process. Conditional distribution $g(y_t|x_t)$ of the observation given the signal. Initial distribu-
    tion $\pi_0$ of $X_0$. Resampling algorithm resample (e.g. systematic resampling of Algorithm 7).
    Effective sample size threshold $\text{ESS}_{\min}$, with default being $N/2$.
 1: Draw samples $x_0^i \sim \pi_0(dx_0)$ and assign weights $W_0^{(i)} = 1/N$ for $i = 1, 2, \ldots, N$.
 2: Set index vector $A_0 = 1 : N$.
 3: for $t = 1, 2, \cdots, T$ do
 4:     if $\text{ESS}(W_{t-1}) < \text{ESS}_{\min}$ then
 5:         (resample) Set index vector $A_t = \text{resample}(W_{t-1})$.
 6:         Assign weights $\tilde{w}_t^i = 1$ for $i = 1, 2, \ldots, N$.
 7:     else
 8:         Set index vector $A_t = A_{t-1}$.
 9:         Assign weights $\tilde{w}_t^i = W_{t-1}^{A_t^i}$ for $i = 1, 2, \ldots, N$.
10:     end if
11:     (propagation) Draw samples $x_t^i \sim P(dx_t|x_{t-1}^{A_t^i})$.
12:     (assimilation) Update weights $w_t^{(i)} = g(y_t|x_t^{(i)})\tilde{w}_t^i$ for $i = 1, 2, \ldots, N$. Normalise weights
    by $W_t^{(i)} = w_t^{(i)} / \sum_{j=1}^N w_t^{(j)}$ for $i = 1, 2, \ldots, N$.
13: end for
14: return approximate $\sum_{i=1}^N W_T^{(i)} \delta_{x_T^{(i)}}(dx_T)$ of the filtering distribution $p(x_T|y_{0:T})$.
```

Algorithm 9 Guided Particle Filter

```
Require: Number of particles $N$. Observations $y_{0:T}$. Transition kernel $P(dx_t|x_{t-1})$ of the signal
    process. Conditional distribution $g(y_t|x_t)$ of the observation given the signal. Propagation
    kernel $Q(dx_t|x_{t-1})$. Initial distribution $\pi_0$ of $X_0$. Resampling algorithm resample (e.g. sys-
    tematic resampling of Algorithm 7). Effective sample size threshold $\text{ESS}_{\min}$, with the default
    being $N/2$.
 1: Draw samples $x_0^i \sim \pi_0(dx_0)$ and assign weights $W_0^{(i)} = 1/N$ for $i = 1, 2, \ldots, N$.
 2: Set index vector $A_0 = 1 : N$.
 3: for $t = 1, 2, \cdots, T$ do
 4:     if $\text{ESS}(W_{t-1}) < \text{ESS}_{\min}$ then
 5:         (resample) Set index vector $A_t = \text{resample}(W_{t-1})$.
 6:         Assign weights $\tilde{w}_t^i = 1$ for $i = 1, 2, \ldots, N$.
 7:     else
 8:         Set index vector $A_t = A_{t-1}$.
 9:         Assign weights $\tilde{w}_t^i = W_{t-1}^{A_t^i}$ for $i = 1, 2, \ldots, N$.
10:     end if
11:     (propagation) Draw samples $x_t^i \sim Q(dx_t|x_{t-1}^{A_t^i})$.
12:     (assimilation) Update weights

            $$w_t^{(i)} = g(y_t|x_t^{(i)}) \frac{P(x_t^{(i)}|x_{t-1}^{A_t^i})}{Q(x_t^{(i)}|x_{t-1}^{A_t^i})} \tilde{w}_t^i$$

    for $i = 1, 2, \ldots, N$. Normalise weights by $W_t^{(i)} = w_t^{(i)} / \sum_{j=1}^N w_t^{(j)}$ for $i = 1, 2, \ldots, N$.
13: end for
14: return approximate $\sum_{i=1}^N W_T^{(i)} \delta_{x_T^{(i)}}(dx_T)$ of the filtering distribution $p(x_T|y_{0:T})$.
```

The local optimal propagation density exists too which we will state below without proof, although it is, again, hard to achieve in practice. Attempts to approximate the optimal density have been made, and we will delay such discussions to later chapters.

Proposition 6.1 (Local Optimality of Guided Particle Filters)**.** *Using the guided particle filter of Algorithm 9, the propagation kernel Q that minimises the variance is provided by*

$$Q_{opt}(dx_t|x_{t-1}) = \frac{g(y_t|x_t)P(dx_t|x_{t-1})}{\int g(y_t|x)P(dx|x_{t-1})}.$$

6.3 Auxiliary Particle Filter

The guided particle filter generalised the choice of propagation kernel from the bootstrap particle filter. The **auxiliary particle filter** makes another generalisation to the guided particle filter, in which the distribution used for the resampling is generalised too.

The weights used in the resampling of the two particle filters described so far are all directly about the particles. For particles $X_t^{(i)}$, we have the weight $W_t^{(i)}$. The auxiliary particle filter introduces a time-dependent auxiliary function η_t defined on the space of signal process states $\mathcal{X}$ and $\eta_t : \mathcal{X} \to \mathbb{R}^+$. This function helps us to change the weights into

$$\bar{w}_t^{(i)} = W_t^{(i)} \eta(X_t^{(i)})$$

which we then normalise to get

$$\bar{W}_t^{(i)} = \frac{\bar{w}_t^{(i)}}{\sum_{j=1}^N \bar{w}_t^{(j)}}.$$

The effect of this weight is to change the underlying particle weight distribution that we use to resample. If $\eta = 1$ and does nothing, then we recover the original resampling.

This change will influence the assimilation weight update too, in that we would need to multiply an extra ratio $W_t^{(i)}/\bar{W}_t^{(i)}$ to account for the above change. This gives us the following description of the auxiliary particle filter as Algorithm 10.

There exists a theoretical optimal choice of the auxiliary function η, as stated by the following result.

Proposition 6.2 (Local Optimality of Auxiliary Particle Filter)**.** *Using the auxiliary particle filter of Algorithm 10, at time $t-1$, assuming the auxiliary functions η_s for $s < t-1$ are fixed. For simplicity, we set $ESS_{\min} = N$. The auxiliary function η_{t-1} that minimised the variance at iteration t is*

$$\eta_{t-1}^{opt}(x_{t-1}) = \sqrt{\int Q(dx_t|x_{t-1}) \left[\frac{g(y_t|x_t)P(dx_t|x_{t-1})}{Q(dx_t|x_{t-1})}\right]^2}.$$

As a remark, if we pick the propagation kernel Q as the optimal one as outlined in Proposition 6.1, the optimal auxiliary function would become

$$\eta_{t-1}^{\text{opt}}(x_{t-1}) = \int P(dx_t|x_{t-1})g(y_t|x_t)$$

which is the density of y_t condition on $X_t = x_{t-1}$.

It has also been observed that, at least numerically, the performance of the auxiliary particle filter deviates minimally from that of the guided particle filter.

Algorithm 10 Auxiliary Particle Filter

```
Require: Number of particles $N$. Observations $y_{0:T}$. Transition kernel $P(dx_t|x_{t-1})$ of the signal
    process. Conditional distribution $g(y_t|x_t)$ of the observation given the signal. Propagation
    kernel $Q(dx_t|x_{t-1})$. Initial distribution $\pi_0$ of $X_0$. Resampling algorithm resample (e.g. sys-
    tematic resampling of Algorithm 7). Auxiliary function $\eta_t$. Effective sample size threshold
    $\text{ESS}_{\min}$, with the default being $N/2$.
 1: Draw samples $x_0^i \sim \pi_0(dx_0)$ and assign weights $W_0^{(i)} = 1/N$ for $i = 1, 2, \ldots, N$.
 2: Update weights $\bar{w}_0^{(i)} \leftarrow W_0^{(i)} \eta_0(X_0^{(i)})$ for $i = 1, 2, \ldots, N$.
 3: Normalise weights by $\bar{W}_0^{(i)} \leftarrow \bar{w}_0^{(i)} / \sum_{j=1}^N \bar{w}_0^{(j)}$ for $i = 1, 2, \ldots, N$.
 4: Set index vector $A_0 = 1 : N$.
 5: for $t = 1, 2, \cdots, T$ do
 6:     if $\text{ESS}(W_{t-1}) < \text{ESS}_{\min}$ then
 7:         (resample) Set index vector $A_t = \text{resample}(W_{t-1})$.
 8:         Assign weights $\tilde{w}_t^i = W_{t-1}^{(i)} / \bar{W}_{t-1}^{(i)}$ for $i = 1, 2, \ldots, N$.
 9:     else
10:         Set index vector $A_t = A_{t-1}$.
11:         Assign weights $\tilde{w}_t^i = W_{t-1}^{A_t^i}$ for $i = 1, 2, \ldots, N$.
12:     end if
13:     (propagation) Draw samples $x_t^i \sim Q(dx_t|x_{t-1}^{A_t^i})$.
14:     (assimilation) Update weights

            $$w_t^{(i)} = g(y_t|x_t^{(i)}) \frac{P(x_t^{(i)}|x_{t-1}^{A_t^i})}{Q(x_t^{(i)}|x_{t-1}^{A_t^i})} \tilde{w}_t^i$$

        for $i = 1, 2, \ldots, N$. Normalise weights by $W_t^{(i)} = w_t^{(i)} / \sum_{j=1}^N w_t^{(j)}$ for $i = 1, 2, \ldots, N$.
15:     Assign weights $\bar{w}_t^i = W_t^{(i)} \eta_t(x_t^{(i)})$ for $i = 1, 2, \ldots, N$.
16:     Normalise weights by $\bar{W}_t^{(i)} \leftarrow \bar{w}_t^{(i)} / \sum_{j=1}^N \bar{w}_t^{(j)}$ for $i = 1, 2, \ldots, N$.
17: end for
18: return approximate $\sum_{i=1}^N W_T^{(i)} \delta_{x_T^{(i)}}(dx_T)$ of the filtering distribution $p(x_T|y_{0:T})$.
```

Bibliography

Chopin, N. and Papaspiliopoulos, O. (2020). *An introduction to sequential Monte Carlo*, Vol. 4, Springer.

Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*, Vol. 10, Springer.

Robert, C. P. and Casella, G. (1999). *Monte Carlo statistical methods*, Vol. 2, Springer.

Williams, D. (1991). *Probability with martingales*, Cambridge university press.