



STK4051/STK9051

Computational Statistics

Riccardo De Bin

`debin@math.uio.no`

Outline of the lecture

- Approximate Simulation
 - Sampling Importance Resampling Algorithm
 - Adaptive Importance, Bridge, and Path Sampling
 - Sequential Monte Carlo
 - Sequential Importance Sampling for Markov Processes
 - General Sequential Importance Sampling
 - Weight Degeneracy, Rejuvenation, and Effective Sample Size
 - Sequential Importance Sampling for Hidden Markov Models
 - Particle Filters

Approximate Simulation: Introduction

Before the Easter break:

- exact methods, the sample is from f ;
- in many cases an approximate method is easier or even the only feasible choice;
- approximation is not a critical flaw of these methods,
 - ▶ the degree of approximation can be controlled by user-specified parameters in the algorithms.

Sampling Importance Resampling Algorithm

The *sampling importance resampling* (SIR) algorithm simulates realizations approximately from some target density f .

Based on the importance sampling approach:

- $\mathbf{X} = (X_1, \dots, X_p)$ denotes a random vector with density f
- g is the density of a multivariate envelope for f ;
- *standardized importance weights* to correct sampling probabilities,

$$w(\mathbf{x}_i) = \frac{f(\mathbf{x}_i)/g(\mathbf{x}_i)}{\sum_{i=1}^m f(\mathbf{x}_i)/g(\mathbf{x}_i)}$$

- $\mathbf{x}_1, \dots, \mathbf{x}_m$ independently drawn from g .
- useful to standardize the weights so they sum to 1;
- when we know f up to a proportionality constant c .

SIR: algorithm

We may view importance sampling as approximating f by the discrete distribution having mass $w(\mathbf{x}_i)$ on each $\mathbf{x}_i$, $i = 1, \dots, m$.

The **SIR algorithm** is therefore:

1. sample candidates $\mathbf{Y}_1, \dots, \mathbf{Y}_m$ independently from g ;
2. calculate $w(\mathbf{Y}_1), \dots, w(\mathbf{Y}_m)$;
3. resample $\mathbf{X}_1, \dots, \mathbf{X}_n$ from $\mathbf{Y}_1, \dots, \mathbf{Y}_m$ with replacement with probabilities $w(\mathbf{Y}_1), \dots, w(\mathbf{Y}_m)$.

Note:

- a random variable $\mathbf{X}$ drawn with the SIR algorithm has distribution that converges to f as $m \rightarrow \infty$;
- SIR is sensitive to the choice of g ,
 - it should have heavier tails than f .

SIR: SIR vs rejection sampling

Although both SIR and rejection sampling similarly

- rely on the ratio of target to envelope,

they differ in an important way:

- **Rejection sampling** is perfect,
 - ▶ the distribution of a generated draw is exactly f ;
 - ▶ requires a random number of draws to get a sample of size n ;
- The **SIR algorithm**
 - ▶ uses a pre-set number of draws to generate an n -sample;
 - ▶ permits a random degree of approximation to f in the distribution of the sampled points.

SIR: remarks

Sample size:

- important to consider the relative sizes of
 - ▶ the initial sample m ;
 - ▶ the resample n ;
- in principle, $n/m \rightarrow 0$ for distributional convergence;
- ideally:
 - ▶ largest possible m for faster convergence;
 - ▶ competing desire for a larger n , for more precise inference;
- in practice, the maximum tolerable ratio n/m depends on the quality of the envelope
 - ▶ the book suggests $n/m \leq 1/10$.

SIR: remarks

Envelope g :

- the algorithm is sensitive to the choice of g ,
- the support of g must include the entire support of f ;
- g should have heavier tails than f ;
- more generally, g should be chosen to ensure that $f(\mathbf{x})/g(\mathbf{x})$ never grows too large;
- when the distribution of weights is highly skewed, use to a different envelope or a different sampling strategy.

SIR: Slash distribution example

The random variable Y has a *slash distribution* if $Y = X/U$ where $X \sim N(0, 1)$ and $U \sim \text{Unif}(0, 1)$ independently.

Goal:

- use the slash distribution as envelope to generate standard normal variates;
- use the standard normal as envelope to generate slash variates.

The slash density function is

$$f(y) = \begin{cases} \frac{1 - \exp\{-y^2/2\}}{y^2\sqrt{2\pi}}, & y \neq 0, \\ \frac{1}{2\sqrt{2\pi}}, & y = 0, \end{cases}$$

i.e., it has very heavy tails.

SIR: Slash distribution example

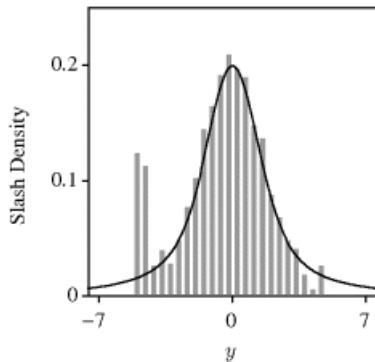
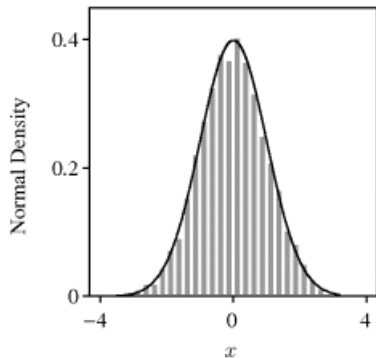
Note:

- a slash distribution is fine for generating draws from a standard normal distribution using SIR;
- not the other way around, because the envelope's tails are far lighter than the target's.

The next figure shows the results,

- using $m = 100000$ and $n = 5000$;
- left panel: histogram of approx. draws from a standard normal density obtained via SIR with a slash distribution envelope;
- right panel: histogram of approximate draws from a slash density obtained via SIR using a normal envelope;
- the solid lines show the target densities.

SIR: Slash distribution example



(Givens & Hoeting, 2013, figure 6.7)

SIR: Adaptive Importance, Bridge, and Path Sampling

Sometimes one can only specify only a very poor g :

- when the target density has support nearly limited to a lower-dimensional space or surface due to strong dependencies between variables not well understood by the analyst;
- one may wish to conduct importance sampling for a variety of related problems, but no single envelope may be suitable for all the target densities of interest.

Bridge sampling:

- a single envelope is inadequate for several densities;
- employs an unnormalized density that is between them.

SIR: Adaptive Importance, Bridge, and Path Sampling

Adaptive importance sampling:

- an initial sample of size m_1 taken from an initial envelope e_1 ;
- this sample is weighted (and possibly resampled) to obtain an initial estimate of quantities of interest or an initial view of f ;
- based on this, the envelope is improved, yielding e_2 ;
- and so on;
- it is most efficient to use the draws from all the steps, along with their weights, to formulate suitable inference;
- alternatively, quick envelope refinement during several initial steps, withholding the majority of simulation effort to the final stage and limiting inference to this final sample for simplicity.

SIR: SIR vs Importance Sampling

Assume

- $Y_1, \dots, Y_m$ i.i.d. from g ;
- $X_1, \dots, X_n$ resampled from $\{Y_1, \dots, Y_m\}$, $w(Y_i) = \frac{f(Y_i)}{g(Y_i)}$;
- there are two possible estimates of $\mu = E_f[h(X)]$:

$$\hat{\mu}_{SIR} = \frac{1}{n} \sum_{i=1}^n h(X_i) \quad \text{and} \quad \hat{\mu}_{IS} = \sum_{i=1}^m w(Y_i) Y_i$$

- it can be shown that $E[\hat{\mu}_{SIR}] = E[\hat{\mu}_{IS}]$;
- and $\text{Var}[\hat{\mu}_{SIR}] \geq \text{Var}[\hat{\mu}_{IS}]$;
- i.e., that $\text{MSE}(\hat{\mu}_{SIR}) \geq \text{MSE}(\hat{\mu}_{IS})$.

Sequential Monte Carlo: Introduction

When the target density f is high dimensional,

- algorithms seen so far are inefficient;
- can be difficult to implement,
 - ▶ challenging to find a good high-dimensional envelope that closely approximates the target with sufficiently heavy tails but little waste;
- Sequential Monte Carlo methods address the problem by
 - ▶ splitting the high-dimensional task into a sequence of simpler steps;
 - ▶ each of which updates the previous one.

Sequential Monte Carlo: Setting

Assume:

- $\mathbf{X}_{1:t} = (X_1, \dots, X_t)$ is a discrete time stochastic process;
- X_t is the observation at time t ;
- $\mathbf{X}_{1:t}$ is the entire history of the sequence;
- f_t is the density of $\mathbf{X}_{1:t}$.

Sequential importance sampling:

- it would be very inefficient to start the SIR approach from scratch at each time t ;
- develop a strategy that will enable us to append the simulated $\mathbf{X}_t$ to the $\mathbf{X}_{1:t-1}$ previously simulated;
- and to adjust the previous importance weights in order to estimate the expected value of the quantity of interest.

Sequential Monte Carlo: Introduction

This incremental updating is important for on-line estimation:

- estimation in real time;
- object tracking to estimate the current location of an object (e.g., a missile) from noisy observations made by remote sensors (e.g., radar);
- at any time, the true location is estimated from the current observation and the sequence of past observations (i.e., the estimated past trajectory).

Sequential Monte Carlo: Sequential IS for Markov Processes

Assume the simple case that $\mathbf{X}_{1:t}$ is a Markov process,

- $\mathbf{X}_t$ depends only on $\mathbf{X}_{t-1}$ rather than the whole $\mathbf{X}_{1:t-1}$;
- the target density $f_t(\mathbf{x}_{1:t})$ may be expressed as

$$f_t(\mathbf{x}_{1:t}) = f_1(x_1) \prod_{i=2}^t f_i(x_i|x_{i-1})$$

- also assume the Markov property for the envelope,

$$g_t(\mathbf{x}_{1:t}) = g_1(x_1) \prod_{i=2}^t g_i(x_i|x_{i-1})$$

- Then the importance weights are

$$w(\mathbf{x}_{1:t}) = \frac{f_t(\mathbf{x}_{1:t})}{g_t(\mathbf{x}_{1:t})} = \frac{f_1(x_1)}{g_1(x_1)} \prod_{i=2}^t \frac{f_i(x_i|x_{i-1})}{g_i(x_i|x_{i-1})} = w(\mathbf{x}_{1:t-1}) \underbrace{\frac{f_t(x_t|x_{t-1})}{g_t(x_t|x_{t-1})}}_{u_t}.$$

Sequential Monte Carlo: Sequential IS for Markov Processes

This means that, instead of

- sampling at each time t from $g_t(\mathbf{x}_{1:t-1})$;
- re-weight each $\mathbf{x}_{1:t}$ value by $w_t = f_t(\mathbf{x}_{1:t})/g_t(\mathbf{x}_{1:t})$;

we take advantage of the Markov property by

- sampling only the next component X_t ;
- appending the value to $\mathbf{x}_{1:t-1}$;
- adjusting w_{t-1} using the multiplicative factor $u_t = \frac{f_t(x_t|x_{t-1})}{g_t(x_t|x_{t-1})}$.

Sequential Monte Carlo: Sequential IS for Markov Processes

The algorithm is:

1. sample $X_1 \sim g_1(\cdot)$; let $w_1 = u_1 = f_1(x_1)/g_1(x_1)$; set $t = 2$;
2. sample $X_t|x_{t-1} \sim g_t(x_t|x_{t-1})$;
3. append x_t to $\mathbf{x}_{1:t-1}$, obtaining $\mathbf{x}_{1:t}$;
4. let $u_t = f_t(x_t|x_{t-1})/g_t(x_t|x_{t-1})$;
5. let $w_t = w_{t-1}u_t$, the importance weight for $\mathbf{x}_{1:t}$;
6. increment t and return to step 2.

Note:

- it is not necessary to standardize the weights at each cycle;
- one can simulate n sequences **in parallel**,
 - ▶ $\hat{E}_f[h(\mathbf{X}_{1:t})] = \sum_{i=1}^n w_t^{(i)} \mathbf{X}_{1:t}^{(i)} / \sum_{i=1}^n w_t^{(i)}.$

Sequential Monte Carlo: Simple Markov Process example

Let $X_t|X_{t-1} \sim f_t$ denote a Markov process where

$$f_t(x_t|x_{t-1}) \propto |\cos(X_t - X_{t-1})| \exp \left\{ -\frac{1}{4}(X_t - X_{t-1})^2 \right\}$$

We are interested in estimating σ , using

$$\left[\frac{\sum_{i=1}^n w_t^{(i)} (x_t^{(i)} - \hat{\mu}_t)^2}{1 - \sum_{i=1}^n (w_t^{(i)})^2} \right]^{1/2}, \text{ with } \hat{\mu}_t = \frac{\sum_{i=1}^n w_t^{(i)} x_t^{(i)}}{\sum_{i=1}^n w_t^{(i)}}.$$

Using sequential importance sampling, at stage t ,

- sample from a normal envelope $X_t^{(i)}|x_{t-1}^{(i)} \sim N(x_{t-1}^{(i)}, 1.5^2)$;
- due to the Markov property, the weight updates are

$$u_t^{(i)} \propto \frac{|\cos(x_t - x_{t-1})| \exp \left\{ -\frac{1}{4}(x_t - x_{t-1})^2 \right\}}{\phi(x_t^{(i)}; x_{t-1}^{(i)}, 1.5^2)}.$$

Sequential Monte Carlo: Simple Markov Process example

Thus to update the $(t - 1)$ -th sample,

- $X_t^{(i)}$ are drawn;
- appended to form $\mathbf{x}_{1:t}$;
- with the weights updated according to $w_t^{(i)} = w_{t-1}^{(i)} u_t^{(i)}$.

Results:

- at $t = 100$ we find $\hat{\sigma}_t = 13.4$;
- with $X_t | x_{t-1} \sim N(x_{t-1}, 4)$,
 - ▶ no sequential importance sampling is needed;
 - ▶ $\hat{\sigma}_t = 9.9$;
- the cosine term adds variance to the distribution f_t .

Sequential Monte Carlo: General Sequential Importance Sampling

Sequential Importance Sampling can be generalised to situations in which the Markov property does not apply:

- the target density is

$$f_t(\mathbf{x}_{1:t}) = f_1(x_1)f_2(x_2|x_1)f_3(x_3|\mathbf{x}_{1:2}) \dots f_t(x_t|\mathbf{x}_{1:t-1});$$

- the envelope

$$g_t(\mathbf{x}_{1:t}) = g_1(x_1)g_2(x_2|x_1)g_3(x_3|\mathbf{x}_{1:2}) \dots g_t(x_t|\mathbf{x}_{1:t-1});$$

- the important weights

$$w_t(\mathbf{x}_{1:t}) = \frac{f_1(x_1)f_2(x_2|x_1)f_3(x_3|\mathbf{x}_{1:2}) \dots f_t(x_t|\mathbf{x}_{1:t-1})}{g_1(x_1)g_2(x_2|x_1)g_3(x_3|\mathbf{x}_{1:2}) \dots g_t(x_t|\mathbf{x}_{1:t-1})};$$

- and the recursive updating expression

$$w_t(\mathbf{x}_{1:t}) = w_t(\mathbf{x}_{1:t-1}) \frac{f_t(x_t|\mathbf{x}_{1:t-1})}{g_t(x_t|\mathbf{x}_{1:t-1})}.$$

Sequential Monte Carlo: Weight Degeneracy

As t increases, the weights develop an undesirable degeneracy,

- the total weight will be concentrated in only a few sample sequences.

Degeneracy of the weights:

- degrades estimation performance;
- measured by the effective sample size;
- related to their variability,
 - ▶ as weights are increasingly concentrated on a few samples with the remaining ≈ 0 , the variance of the weights increases.

Sequential Monte Carlo: Weight Degeneracy

About the last point,

- we know that
$$\text{Var}[Y] = E[\text{Var}[Y|Z]] + \text{Var}[E[Y|Z]] \geq \text{Var}[E[Y|Z]]$$
- using $Y = w_t$ and $Z = \mathbf{X}_{1:t-1}$ (w_{t-1} given by $\mathbf{x}_{1:t-1}$),

$$E[w_t | \mathbf{X}_{1:t-1}] = w_{t-1} \underbrace{E \left[\frac{f_t(X_t | X_{t-1})}{g_t(X_t | X_{t-1})} \middle| \mathbf{X}_{1:t-1} \right]}_{=1} = w_{t-1}$$

implying that

$$\text{Var}[w_t] \geq \text{Var}[w_{t-1}]$$

i.e., the variance **increases** at each time-step.

- Consequence:
 - ▶ only a few samples will dominate the others;
 - ▶ the variability of the estimate will increase.

Sequential Monte Carlo: Effective Sample Size

To measure the level of degeneracy, it is useful to introduce the concept of **effective sample size**, $\hat{N}(g, f)$:

- measures how much the n weighted samples in IS are worth in terms of samples drawn from f ,
- the larger the better;
- for unstandardised weights, $\hat{N}(g, f) = \frac{n}{1 + \widehat{Var}[w^*(X)]}$;
- for standardised weights, $\hat{N}(g, f) = \frac{1}{\sum_{i=1}^n w(x^{(i)})^2}$:

$$= n \quad \text{if } w_i = 1/n \text{ for all } i;$$

$$= n - z \quad \text{if } w_i = 0 \text{ for } i \leq z \text{ and } w_i = 1/(n - z) \text{ for } i > z;$$

$$= 1 \quad \text{if } w_i = 0 \text{ for all } i \neq j \text{ and } w_j = 1;$$

Sequential Monte Carlo: Rejuvenation

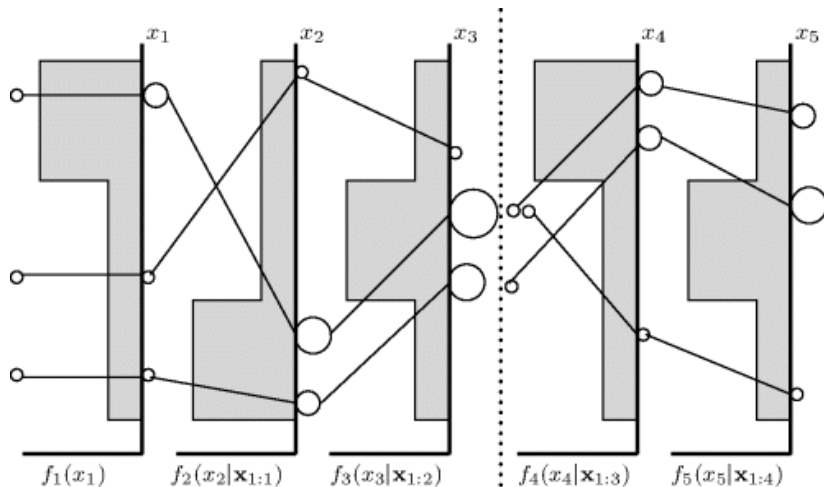
In the case of sequential importance sampling,

- we need to monitor the importance weight degeneracy;
- the effective sample size will decrease as time progresses;
- when $\hat{N}(g, f)$ falls below some threshold,
 - ▶ the collection of $\mathbf{X}_{1:t-1}^{(i)}, i = 1, \dots, n$, should be **rejuvenated**.

The simplest rejuvenation approach is:

- resample the sequences with replacement with $p = w_t(\mathbf{x}_{1:t}^{(i)})$;
- reset all weights to $1/n$;
- called sequential importance sampling with resampling;
- closely related to the notion of particle filters.

Sequential Monte Carlo: Weight Degeneracy and Rejuvenation



(Givens & Hoeting, 2013, figure 6.8)

Sequential Monte Carlo: High-Dimensional Distribution example

Sequential importance sampling can also be used for sampling from a p -dimensional target distribution f_p for fixed p ,

- importance weighted sample from an high-dimensional distribution one dimension at a time;
- by sampling from the univariate conditional densities;
- after p steps we obtain the $\mathbf{X}_{1:p}^{(i)}$ and their corresponding weights needed to approximate a sample from f_p .

Example of the book:

- $f_t(x_{1:t}) = k_t \exp\{-||x_{1:t}||^3/3\}$, for some constants k_t ,
 - ▶ $t = 1, \dots, p$;
 - ▶ $|| \cdot ||$ is the Euclidean norm;
 - ▶ $X_{1:p}$ is the random variable having density f_p ;
- although $f_t(x_{1:t})$ is a smooth unimodal density, no easy method for sampling from it directly.

Sequential Monte Carlo: High-Dimensional Distribution example

Use $\phi(x)$ as envelope. The t -th conditional density can be written as $f_t(x_t|x_{1:t-1}) = f_t(x_{1:t})/f_{t-1}(x_{1:t-1})$.

The sequential importance sampling strategy is given by:

1. let $t = 1$; sample n points $X_1^{(1)}, \dots, X_1^{(n)}$ from $N(0, 1)$ and calculate the initial weights $w_1^{(i)} = \exp\{-|x_1^{(i)}|^{3/2}\}/\phi(x_1^{(i)})$;
2. for $t > 1$ sample n points $X_t^{(i)}$ from $N(0, 1)$. Append these to the $\mathbf{X}_{1:t-1}$, yielding $\mathbf{X}_{1:t}$;
3. calculate the weight adjustment factors $u_t^{(i)} = \frac{f_t(x_t^{(i)}|\mathbf{x}_{1:t-1}^{(i)})}{\phi(x_t^{(i)})}$;
4. set $w_t^{(i)} = w_{t-1}^{(i)} u_t^{(i)}$ and standardize the new weights;
5. calculate $\hat{N}(\phi, f_t)$; if $\hat{N}(\phi, f_t) < \alpha n$ then resample $\mathbf{X}_{1:t}^{(i)}$ with replacement with $p = w_t^{(i)}$ and: (i) discard $\mathbf{X}_{1:t}^{(i)}$ in favour of this new sample; (ii) reset all the weights to $1/n$;
6. increment t and return to step 2 until $t = p$.

Sequential Monte Carlo: High-Dimensional Distribution example

When $p = 50$, $n = 5000$, and $\alpha = 0.2$,

- over the 50 stages, step 5 was triggered 8 times;
- the median $\hat{N}(\phi, f_t)$ during the process was about 2000;
 - ▶ drawing 5000 points from a $N_{50}(\mathbf{0}, \mathbf{I}_{50})$ envelope and one-shot importance sampling, effective sample size of about 1.13;
 - ▶ $n = 250,000$ only increases the effective sample size to 1.95.

Note:

- High-dimensional space is too vast to pick many p -dimensional random draws with high density using the one-step approach;
- points must be generated sequentially one dimension by one.

Sequential Monte Carlo: Sequential IS for Hidden Markov Models

Consider a hidden Markov process:

- $X_0, X_1, X_2, \dots$ is a Markov sequence of unobservable r.v. indexed by time t ;
- the distribution of $X_t|x_{1:t-1}$ depends only on x_{t-1} ;
- there is an observable sequence of r.v. $Y_0, Y_1, Y_2, \dots$
- Y_t is dependent on the process state at the same time, X_t ;
- $Y_t \sim p_y(y_t|x_t)$ and $X_t \sim p_x(x_t|x_{t-1})$, for $t = 1, 2, \dots$

Aim:

- use the observed $\mathbf{y}_{1:t}$ as data to estimate the states $\mathbf{x}_{1:t}$;
- the target distribution is $f_t(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$.

Sequential Monte Carlo: Sequential IS for Hidden Markov Models

There is a recursive relationship between f_t and f_{t-1} ,

$$\begin{aligned} f(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) &= \frac{f(\mathbf{x}_{1:t}, y_t|\mathbf{y}_{1:t-1})}{f(y_t|\mathbf{y}_{1:t-1})} \\ &= \frac{f(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})p_x(x_t|x_{t-1})p_y(y_t|x_t)}{f(y_t|\mathbf{y}_{1:t-1})} \\ &\propto f(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})p_x(x_t|x_{t-1})p_y(y_t|x_t) \end{aligned}$$

Adopting the envelope $g_t(x_t|x_{t-1}) = p_x(x_t|x_{t-1})$,

$$\begin{aligned} w_t &= \frac{f(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{g(\mathbf{x}_{1:t})} \\ &\propto \frac{f(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})p_x(x_t|x_{t-1})p_y(y_t|x_t)}{p_{x_1}(x_1) \prod_{s=2}^t p_x(x_s|x_{s-1})} \\ &= \frac{f(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})}{g(\mathbf{x}_{1:t-1})} \frac{p_x(x_t|x_{t-1})p_y(y_t|x_t)}{p_x(x_t|x_{t-1})} = w_{t-1}p_y(y_t|x_t). \end{aligned}$$

Sequential Monte Carlo: Terrain Navigation example

An airplane flying over uneven terrain can use information about the ground elevation to estimate its location $X_t = (X_{1t}, X_{2t})$:

- as the plane follows its flight path, sequential elevation measurements $m(x_t)$ come from a map database;
- simultaneously, an inertial navigation system provides an estimated travel direction and distance, d_t ;
- at each time point the previously estimated location of the plane is updated using both types of new information.

Sequential Monte Carlo: Terrain Navigation example

The hidden Markov model for terrain navigation is

$$\mathbf{Y}_t = m(\mathbf{x}_t) + \delta_t \quad \text{and} \quad \mathbf{X}_t = \mathbf{x}_{t-1} + \mathbf{d}_t + \boldsymbol{\epsilon}_t$$

where

- $\boldsymbol{\epsilon}_t$ and δ_t are independent random error processes representing the error in drift and elevation measurement;
- Y_t is the observed elevation;
- d_t is considered as known.

Sequential Monte Carlo: Terrain Navigation example

Suppose that random error in location ϵ_t has a distribution characterized by $\epsilon_t = \mathbf{R}_t^T \mathbf{Z}_t$, where

$$\mathbf{R}_t = \begin{pmatrix} -x_{1t} & x_{2t} \\ -x_{2t} & x_{1t} \end{pmatrix} \text{ and } \mathbf{Z}_t \sim N_2 \left(\mathbf{0}, q^2 \begin{pmatrix} 1 & 0 \\ 0 & k^2 \end{pmatrix} \right),$$

with $q = 400$ and $k = 1/2$.

The $g_t(\epsilon_t)$ is the importance sampling envelope $g_t(x_t|x_{t-1})$:

- complicated formulation to handle the geography;
- conceptually can be a $N_2(\mathbf{0}, \mathbf{I}_2)$.

Sequential Monte Carlo: Terrain Navigation example

Sequential importance sampling algorithm for this problem:

1. initialize at $t = 0$. Draw n starting points $\mathbf{X}_0^{(i)}, i = 1, \dots, n$;
2. receive observed elevation data Y_t ;
3. calculate the weight update factor $u_t^{(i)} = \phi(y_t; m(\mathbf{x}_t^{(i)}), \sigma^2)$;
4. Update the weights according to $w_t^{(i)} = w_{t-1}^{(i)} u_t^{(i)}$; if $t = 0$, then $w_{t-1}^{(i)} = 1/n$; normalize the weights so they sum to 1;
5. the current estimate of the true location is $\hat{\mathbf{x}} = \sum_{i=1}^n w_t^{(i)} \mathbf{x}_t^{(i)}$;
6. if $\hat{N}_t(g_t, f_t) = 1 / \sum_{i=1}^n (w_t^{(i)})^2 < 0.3n$, then rejuvenate (apply substeps a–c); otherwise, proceed straight to step 7.
 - a resample $\mathbf{x}_{t,\text{new}}^{(i)}$ from $\mathbf{x}_t^{(i)}$ with replacement with $p = w_t^{(i)}$;
 - b replace the current sample of $\mathbf{x}_t^{(i)}$ with the new draws $\mathbf{x}_{t,\text{new}}^{(i)}$;
 - c reset the weights $w_t^{(i)}$ to $1/n$ for all i ;
7. sample a set of location errors $\epsilon_{t+1} \sim g_{t+1}(\epsilon)$;
8. advance the set of locations, $\mathbf{x}_{t+1}^{(i)} = \mathbf{x}_t^{(i)} + \mathbf{d}_{t+1}^{(i)} + \epsilon_{t+1}$;
9. increment t and return to step 2.

Sequential Monte Carlo: Terrain Navigation example

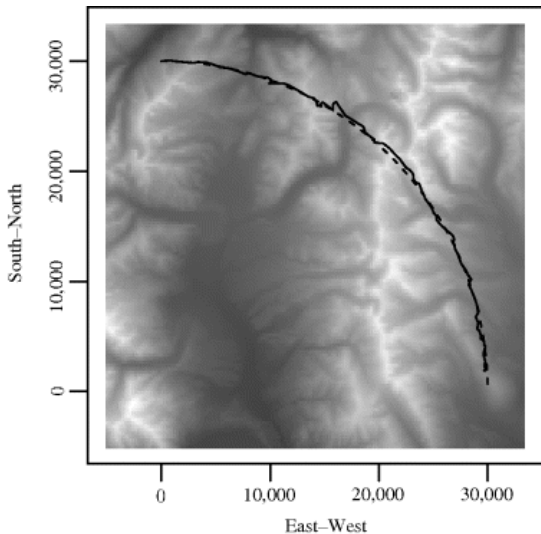
In this algorithm,

- each sequence $\mathbf{x}_{1:t}^{(i)}$ represents one possible airplane path;
- these paths have corresponding importance weights $w_t^{(i)}$.

Results in the next figure,

- the dashed line is the true flight path;
- the solid line is the estimated path calculated as in step 5 above, using the elevation data y_t ;
- very good tracking of the true location;
- results are dependent on n and the magnitudes of noise.

Sequential Monte Carlo: Terrain Navigation example



(Givens & Hoeting, 2013, figure 6.9)

Sequential Monte Carlo: Particle Filters

Particle filters are sequential Monte Carlo strategies that emphasize the need for preventing degeneracy:

- developed within the hidden Markov models framework;
- at time t , the current sample of sequences is viewed as a collection of weighted particles;
 - ▶ particles with high weights enjoy increasing influence;
 - ▶ underweighted particles are filtered away as t increases;
- can be seen as
 - ▶ generalizations of sequential importance sampling;
 - ▶ specific strategies for sequential importance sampling with resampling;
- adopting a particle filter mindset, resample at each t ;
- an algorithm described as a particle filter is often characterized by a stronger focus on resampling or adjustment of the new draws between or within cycles to prevent degeneracy.

Sequential Monte Carlo: Particle Filters

Resampling alone does not prevent degeneracy:

- low-weight sequences are likely to be discarded;
- high-weight sequences are replicated rather than diversified;
- particle filters favour tactics like perturbing or smoothing,
 - ▶ supplement the resampling step by moving samples according to a Markov chain transition kernel;
 - ▶ smooth the resampling via weighted smoothed bootstrap;
 - ▶ employ an adaptive form of bridge sampling;
- the simplest particle filter is the bootstrap filter:
 - ▶ relies on a simple multinomial importance-weighted resample;
 - ▶ rather than waiting for the weights to degenerate excessively;
 - ▶ sequences are resampled with replacement with probability proportional to their current weights, then $\text{weights} = 1/n$;

Sequential Monte Carlo: Terrain Navigation, Continued

The bootstrap filter is easy to implement in the previous example:

- always resample the current collection of paths at each t ;
- we replace step 6 with
 - 6 regardless of $\hat{N}_t(g_t, f_t)$, carry out substeps (a), (b), and (c).

Result:

- the estimated flight path is qualitatively indistinguishable from that previously estimated.

References |

GIVENS, G. H. & HOETING, J. A. (2013). *Computational Statistics*. Wiley.