Partially observed Markov process (POMP) models: Filtering and likelihood evaluation

Edward Ionides
University of Michigan, Department of Statistics

Lecture 2 at Wharton Statistics Department
Wednesday 26th April, 2017

Slides are online at
`http://dept.stat.lsa.umich.edu/~ionides/talks/upenn`

A perspective from 2001

The following six issues identified by Bjørnstad and Grenfell (*Science*, 2001) are not solved by classical time series methodology. They require consideration of **nonlinear mechanistic models** as statistical tools for biological systems:

1. Combining measurement noise and process noise.
2. Including covariates in mechanistically plausible ways.
3. Continuous time models.
4. Modeling and estimating interactions in coupled systems.
5. Dealing with unobserved variables.
6. Modeling spatial-temporal dynamics.

Partially observed Markov process (POMP) models

- Data $y_1^*, \ldots, y_N^*$ are collected at times $t_1 < \cdots < t_N$.
- A **partially observed Markov process (POMP)** model consists of
  1. a **latent Markov process** $\{X(t), t \geq t_0\}$
  2. an **observable process** $Y_1, \ldots, Y_N$
  3. an **unknown parameter vector** θ.
- We suppose Y_n given $X(t_n)$ is conditionally independent of the rest of the latent and observable processes.
- POMPs are also called hidden Markov models or state space models.
- General nonlinear non-Gaussian POMP models can capture all the desiderata of Bjørnstad and Grenfell (2001). Numerous other applications include rocket science, economics, geophysical systems...

Partially observed Markov process (POMP) models

- Data $y_1^*, \ldots, y_N^*$ are collected at times $t_1 < \cdots < t_N$.
- A **partially observed Markov process (POMP)** model consists of
  1. a **latent Markov process** $\{X(t), t \geq t_0\}$
  2. an **observable process** $Y_1, \ldots, Y_N$
  3. an **unknown parameter vector** θ.
- We suppose Y_n given $X(t_n)$ is conditionally independent of the rest of the latent and observable processes.
- POMPs are also called hidden Markov models or state space models.
- General nonlinear non-Gaussian POMP models can capture all the desiderata of Bjørnstad and Grenfell (2001). Numerous other applications include rocket science, economics, geophysical systems...
- **Is it computationally feasible to carry out effective inference for general POMP models?**

Partially observed Markov process (POMP) models

- Data $y_1^*, \ldots, y_N^*$ are collected at times $t_1 < \cdots < t_N$.
- A **partially observed Markov process (POMP)** model consists of
  1. a **latent Markov process** $\{X(t), t \geq t_0\}$
  2. an **observable process** $Y_1, \ldots, Y_N$
  3. an **unknown parameter vector** θ.
- We suppose Y_n given $X(t_n)$ is conditionally independent of the rest of the latent and observable processes.
- POMPs are also called hidden Markov models or state space models.
- General nonlinear non-Gaussian POMP models can capture all the desiderata of Bjørnstad and Grenfell (2001). Numerous other applications include rocket science, economics, geophysical systems...
- **Is it computationally feasible to carry out effective inference for general POMP models?**
- **Are there any theoretical or practical reasons not to use standard parametric inference techniques, if they are computationally feasible?**

On stationarity as an assumption

- Recall that a stochastic process $\{X(t), t \geq t_0\}$ is **stationary** if $X(t_1), X(t_2), \ldots, X(t_k)$ has the same distribution as $X(t_1 + s), X(t_2 + s), \ldots, X(t_k + s)$ for all $t_1, \ldots, t_k$ and $s > 0$.
- Much asymptotic theory for Markov processes requires stationarity.
- Much asymptotic theory for statistical inference requires stationarity.
- We study dynamic systems not well modeled by stationary processes.
  - A model that conditions on time-varying covariates is non-stationary. We're often interested in covariates: what is the effect of $Z(t)$ on the dynamic system?
  - For disease transmission, population size and birth rates vary; vaccination and other interventions vary.

On stationarity as an assumption

- Recall that a stochastic process $\{X(t), t \geq t_0\}$ is **stationary** if $X(t_1), X(t_2), \ldots, X(t_k)$ has the same distribution as $X(t_1 + s), X(t_2 + s), \ldots, X(t_k + s)$ for all $t_1, \ldots, t_k$ and $s > 0$.
- Much asymptotic theory for Markov processes requires stationarity.
- Much asymptotic theory for statistical inference requires stationarity.
- We study dynamic systems not well modeled by stationary processes.
  - A model that conditions on time-varying covariates is non-stationary. We're often interested in covariates: what is the effect of $Z(t)$ on the dynamic system?
  - For disease transmission, population size and birth rates vary; vaccination and other interventions vary.
- **Our methods must apply to non-stationary models.**

On stationarity as an assumption

- Recall that a stochastic process $\{X(t), t \geq t_0\}$ is **stationary** if $X(t_1), X(t_2), \ldots, X(t_k)$ has the same distribution as $X(t_1 + s), X(t_2 + s), \ldots, X(t_k + s)$ for all $t_1, \ldots, t_k$ and $s > 0$.
- Much asymptotic theory for Markov processes requires stationarity.
- Much asymptotic theory for statistical inference requires stationarity.
- We study dynamic systems not well modeled by stationary processes.
  - A model that conditions on time-varying covariates is non-stationary. We're often interested in covariates: what is the effect of $Z(t)$ on the dynamic system?
  - For disease transmission, population size and birth rates vary; vaccination and other interventions vary.
- **Our methods must apply to non-stationary models.**
- We will not demand full asymptotic justification in the limit as the data grow.
- We assess empirical behavior on models resembling the data.
- Asymptotics guides methodology: mathematical theory is a useful heuristic.

Remembering not to forget initial conditions

- The **initial value** is $X(t_0)$.
- For Markov chain asymptotic theory, $X(t_0)$ is typically unimportant.
- For stationary models, $X(t_0)$ is drawn from a stationary distribution.

Remembering not to forget initial conditions

- The **initial value** is $X(t_0)$.
- For Markov chain asymptotic theory, $X(t_0)$ is typically unimportant.
- For stationary models, $X(t_0)$ is drawn from a stationary distribution.
- A different perspective: For deterministic dynamic models, such as ordinary differential equations (ODEs) the initial value perfectly determines the whole future of the process.

Remembering not to forget initial conditions

- The **initial value** is $X(t_0)$.
- For Markov chain asymptotic theory, $X(t_0)$ is typically unimportant.
- For stationary models, $X(t_0)$ is drawn from a stationary distribution.
- A different perspective: For deterministic dynamic models, such as ordinary differential equations (ODEs) the initial value perfectly determines the whole future of the process.
- We work with situations where a deterministic model is not appropriate as a statistical description of the data. Nevertheless, the obvious role of initial values in the deterministic case is a wake-up call.

Remembering not to forget initial conditions

- The **initial value** is $X(t_0)$.
- For Markov chain asymptotic theory, $X(t_0)$ is typically unimportant.
- For stationary models, $X(t_0)$ is drawn from a stationary distribution.
- A different perspective: For deterministic dynamic models, such as ordinary differential equations (ODEs) the initial value perfectly determines the whole future of the process.
- We work with situations where a deterministic model is not appropriate as a statistical description of the data. Nevertheless, the obvious role of initial values in the deterministic case is a wake-up call.
- Often, we can separate the parameter vector θ into a **regular parameter** (RP) θ_{RP} and an **initial value parameter** (IVP) θ_{IVP}. An RP is a parameter that plays a role in the dynamics or the observation process, and an IVP determines only $X(t_0)$.

Remembering not to forget initial conditions

- The **initial value** is $X(t_0)$.
- For Markov chain asymptotic theory, $X(t_0)$ is typically unimportant.
- For stationary models, $X(t_0)$ is drawn from a stationary distribution.
- A different perspective: For deterministic dynamic models, such as ordinary differential equations (ODEs) the initial value perfectly determines the whole future of the process.
- We work with situations where a deterministic model is not appropriate as a statistical description of the data. Nevertheless, the obvious role of initial values in the deterministic case is a wake-up call.
- Often, we can separate the parameter vector θ into a **regular parameter** (RP) θ_{RP} and an **initial value parameter** (IVP) θ_{IVP}. An RP is a parameter that plays a role in the dynamics or the observation process, and an IVP determines only $X(t_0)$.
- A canonical approach is to write the parameter space as $\Theta = \Theta_{RP} \times \Theta_{IVP}$ where $\Theta_{IVP} = \mathbb{X}$. Given $\theta = (\theta_{RP}, \theta_{IVP})$, model $X(t_0)$ as a point mass at θ_{IVP}.

More POMP model notation

- Let $0:N$ denote the sequence $0, 1, \ldots, N$.
- For inference, we are most interested in the value of the latent process at the observation times.
- We write $X_n = X(t_n)$ and $X_{0:N} = (X_0, X_1, \ldots, X_N)$.
- The joint density of $X_{0:N}$ and $Y_{1:N}$ is denoted $f_{X_{0:N}Y_{1:N}}(x_{0:N}, y_{1:N}\,;\theta)$.
- The **one-step transition density** is $f_{X_n|X_{n-1}}(x_n \,|\, x_{n-1}\,;\theta)$.
  - $f_{X_n|X_{n-1}}$ can depend on n, so we don't assume time-homogeneity or stationarity.
  - In particular, if our model conditions on a sequence of covariates $z_{0:N}$ this setup allows $f_{X_n|X_{n-1}}$ to depend on z_n.
- The **measurement density** is $f_{Y_n|X_n}(y_n \,|\, x_n\,;\theta)$.

[Avoidance of] discussion of measure-theoretic details

- Let $X(t)$ take values in $\mathbb{X}$ and Y_n take values in $\mathbb{Y}$.
- We have supposed $X_{0:N}$ and $Y_{1:N}$ have a joint density $f_{X_{0:N}Y_{1:N}}$ with respect to some suitable measure on $\mathbb{X}^{N+1} \times \mathbb{Y}^N$. This implies the existence of all marginal and conditional densities.
- If the measure is discrete, the densities are often called probability mass functions.

Three basic POMP identities

The **likelihood function** $\ell(\theta) = f_{Y_{1:N}}(y^*_{1:N}\,;\theta)$ satisfies

$$[L1] \quad \ell = \prod_{n=1}^{N} \int f_{Y_n|X_n}(y^*_n \,|\, x_n)\, f_{X_n|Y_{1:n-1}}(x_n \,|\, y^*_{1:n-1})\, dx_n,$$

suppressing θ and letting $1:0$ be the empty set.

The **prediction distribution** $f_{X_{n+1}|Y_{1:n}}(x_n \,|\, y^*_{1:n})$ satisfies

$$[L2] \quad f_{X_{n+1}|Y_{1:n}}(x_n \,|\, y^*_{1:n}) = \int f_{X_{n+1}|X_n}(x_{n+1} \,|\, x_n)\, f_{X_n|Y_{1:n}}(x_n \,|\, y^*_{1:n})\, dx_n.$$

The **filter distribution** $f_{X_n|Y_{1:n}}(x_n \,|\, y^*_{1:n})$ satisfies

$$[L3] \quad f_{X_n|Y_{1:n}}(x_n \,|\, y^*_{1:n}) = \frac{f_{X_n|Y_{1:n-1}}(x_n \,|\, y^*_{1:n-1})\, f_{Y_n|X_n}(y^*_n \,|\, x_n)}{\int f_{X_n|Y_{1:n-1}}(\tilde{x}_n \,|\, y^*_{1:n-1})\, f_{Y_n|X_n}(y^*_n \,|\, \tilde{x}_n)\, d\tilde{x}_n}.$$

Recursive solution of $[L2]$ and $[L3]$ enables evaluation of $[L1]$.

Exercise: derive the basic POMP identities from the POMP definition.

Why are the filtering and prediction recursions useful?

- There are many factorizations of the likelihood. What is special about the factorization used by the basic POMP identities?

Why are the filtering and prediction recursions useful?

- There are many factorizations of the likelihood. What is special about the factorization used by the basic POMP identities?
- In general, obtaining the likelihood for a model with latent variable $X_{0:N}$ involves integration over $\mathbb{X}^{N+1}$ using

$$\ell = \int f_{X_{0:N}Y_{1:N}}(x_{0:N}, y^*_{1:N})\, dx_0\, dx_1 \ldots dx_N.$$

Why are the filtering and prediction recursions useful?

- There are many factorizations of the likelihood. What is special about the factorization used by the basic POMP identities?
- In general, obtaining the likelihood for a model with latent variable $X_{0:N}$ involves integration over $\mathbb{X}^{N+1}$ using

$$\ell = \int f_{X_{0:N}Y_{1:N}}(x_{0:N}, y^*_{1:N})\, dx_0\, dx_1 \ldots dx_N.$$

- The basic POMP identities reduce this high-dimensional integral over $\mathbb{X}^{N+1}$ to a sequence of integrals over $\mathbb{X}$ with simplifications arising from the POMP structure.

Why are the filtering and prediction recursions useful?

- There are many factorizations of the likelihood. What is special about the factorization used by the basic POMP identities?
- In general, obtaining the likelihood for a model with latent variable $X_{0:N}$ involves integration over $\mathbb{X}^{N+1}$ using

$$\ell = \int f_{X_{0:N}Y_{1:N}}(x_{0:N}, y^*_{1:N})\, dx_0\, dx_1 \ldots dx_N.$$

- The basic POMP identities reduce this high-dimensional integral over $\mathbb{X}^{N+1}$ to a sequence of integrals over $\mathbb{X}$ with simplifications arising from the POMP structure.
- If this sequence of integrals is a **stable** recursion — meaning that small numerical errors in evaluating one step have decreasing consequences at later steps — then a good numerical integral for the prediction and filtering identities should recursively enable a good numerical evaluation of the likelihood.

Linear Gaussian POMP models

- A linear Gaussian POMP model has the form $X_0 \sim N[\mu_0, U_0]$ and

$$\begin{aligned} X_n &= A_n X_{n-1} + \epsilon_n, & \epsilon_n &\sim N[\mu_n, U_n] \\ Y_n &= B_n X_n + \eta_n, & \eta_n &\sim N[\nu_n, V_n] \end{aligned}$$

- The basic POMP identities have a closed form solution in the linear Gaussian case — the celebrated **Kalman filter** (Kalman, 1960).
- Applying this solution to a linear Gaussian approximation of a nonlinear model results in the **extended Kalman filter**. This was used in the Apollo space program (Grewal and Andrews, 2010) and many other applications since.
- For good performance on complex models, the extended Kalman filter is problematic. A step toward the development of the Google self-driving car was the discovery that a Monte Carlo filter can out-perform an extended Kalman filter for the Simultaneous Localization and Mapping problem (Montemerlo and Thrun, 2007).

Finite-state POMP models

- When $\mathbb{X}$ is finite, $X(t)$ is a finite state Markov chain.
- In this case, the integrals in the basic POMP identities become finite sums.
- This has led to numerical methods used in speech recognition, genetics, and diverse other applications involving modest-sized finite state POMP models (Rabiner, 1989).

Monte Carlo approximations to intractable integrals

- A **Monte Carlo approximation** to a density $f_X(x)$ is a collection of random variables $X_1^{MC}, \ldots, X_J^{MC}$ such that, for all suitable $h(x)$,
$$\int h(x)\, f_X(x)\, dx \approx \frac{1}{J} \sum_{j=1}^{J} h\left(X_j^{MC}\right).$$
- We will not focus on formalizations of $\approx$.
- We write this as $X_{1:J}^{MC} \overset{MC}{\sim} f_X(x)$.
- We call each X_j^{MC} a **particle** and $X_{1:J}^{MC}$ a **swarm**.
- $X_{1:J}^{MC}$ do not have to be independent.
- It is sufficient for each particle X_j^{MC} to have marginal distribution approximating $f_X(x)$.
- A **weighted swarm** $\left(X_{1:J}^{MC}, W_{1:J}\right) \overset{MC}{\sim} f_X(x)$ has
$$\int h(x)\, f_X(x)\, dx \approx \frac{1}{\sum_{k=1}^{J} W_k} \sum_{j=1}^{J} W_j\, h\left(X_j^{MC}\right).$$

Sequential Monte Carlo (SMC) for POMP models

- Sequential Monte Carlo (SMC) is a class of algorithms widely used for numerical solution of the basic POMP identities.
- SMC constructs a swarm of particles recursively approximating solutions to the basic POMP identities:

  $X^F_{n,1:J} \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$, the filter distribution.

  $X^P_{n,1:J} \overset{MC}{\sim} f_{X_n|Y_{1:n-1}}(x_n|y^*_{1:n-1})$, the prediction distribution.

The SMC prediction recursion

Given a swarm approximating the filter distribution at time t_n, moving each particle independently according to the POMP transition density gives a swarm approximating the prediction distribution at time t_{n+1}.

The SMC prediction recursion

Given a swarm approximating the filter distribution at time t_n, moving each particle independently according to the POMP transition density gives a swarm approximating the prediction distribution at time t_{n+1}.

This results from a basic property of Monte Carlo approximations:

- If $X_{1:J}^{MC} \overset{MC}{\sim} f_X(x)$ and $Y_j^{MC} | X_j^{MC} \overset{MC}{\sim} f_{Y|X}(y \,|\, X_j^{MC})$ for each j, we expect $(X_{1:J}^{MC}, Y_{1:J}^{MC}) \overset{MC}{\sim} f_{XY}(x, y)$ and so $Y_{1:J}^{MC} \overset{MC}{\sim} f_Y(y)$.

The SMC prediction recursion

Given a swarm approximating the filter distribution at time t_n, moving each particle independently according to the POMP transition density gives a swarm approximating the prediction distribution at time t_{n+1}.

This results from a basic property of Monte Carlo approximations:

- If $X_{1:J}^{MC} \overset{MC}{\sim} f_X(x)$ and $Y_j^{MC} | X_j^{MC} \overset{MC}{\sim} f_{Y|X}(y \,|\, X_j^{MC})$ for each j, we expect $(X_{1:J}^{MC}, Y_{1:J}^{MC}) \overset{MC}{\sim} f_{XY}(x, y)$ and so $Y_{1:J}^{MC} \overset{MC}{\sim} f_Y(y)$.
- Consequently, if $X_{n,1:J}^F \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y_{1:n}^*)$ and $X_{n+1,j}^P | X_{n,j}^F \overset{MC}{\sim} f_{X_{n+1}|X_n}(x_{n+1} \,|\, X_{n,j}^F)$ then the prediction identity gives $X_{n+1,1:J}^P \overset{MC}{\sim} f_{X_{n+1}|Y_{1:n}}(x_{n+1}|y_{1:n}^*)$.

The SMC filter recursion, Part I. Filter weights

Given a swarm approximating the prediction distribution at time t_n, we get a swarm approximating the filter distribution by giving each particle a weight proportional to its density evaluated at the data point y_n^*.

The SMC filter recursion, Part I. Filter weights

Given a swarm approximating the prediction distribution at time t_n, we get a swarm approximating the filter distribution by giving each particle a weight proportional to its density evaluated at the data point y_n^*.

This follows from a basic property of Monte Carlo approximations: conditioning can be implemented by weighting.

- If $X_{1:J}^{MC} \overset{MC}{\sim} f_X(x)$, we can construct a weighted swarm $\left(X_{1:J}^{MC}, W_{1:J}\right) \overset{MC}{\sim} f_{X|Y}(x \,|\, y^*)$ by setting $W_j = f_{Y|X}(y^* \,|\, X_j^{MC})$.

The SMC filter recursion, Part I. Filter weights

Given a swarm approximating the prediction distribution at time t_n, we get a swarm approximating the filter distribution by giving each particle a weight proportional to its density evaluated at the data point y_n^*.

This follows from a basic property of Monte Carlo approximations: conditioning can be implemented by weighting.

- If $X_{1:J}^{MC} \overset{MC}{\sim} f_X(x)$, we can construct a weighted swarm $\left(X_{1:J}^{MC}, W_{1:J}\right) \overset{MC}{\sim} f_{X|Y}(x \,|\, y^*)$ by setting $W_j = f_{Y|X}(y^* \,|\, X_j^{MC})$.
- The filter identity then gives $\left(X_{n,1:J}^{P}, W_{n,1:J}\right) \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y_{1:n}^*)$ for $W_{n,j} = f_{Y_n|X_n}(y_n^* \,|\, X_{n,j}^P)$.

Resampling a weighted swarm

- Suppose we have a weighted swarm, $\left(X_{1:J}^{MC}, W_{1:J}\right) \overset{MC}{\sim} f_X(x)$.
- Let $R_{1:J}$ be a random sequence of non-negative integers with $\sum_j R_j = J$ and $\mathbb{E}[R_j] = \frac{W_j}{\sum_k W_k} \times J$.
- Let $X_{1:J}^{RE}$ be an unweighted swarm with R_j copies of X_j^{MC}.
- The **resampled swarm** satisfies $X_{1:J}^{RE} \overset{MC}{\sim} f_X(x)$.
- Multinomial resampling gives one possible construction of $R_{1:J}$, with

$$R_{1:J} \sim \text{Multinomial}\left(J\,,\, \frac{W_{1:J}}{\sum_k W_k}\right).$$

- However, it is usually better to resample at least $\lfloor JW_j \{\sum_k W_k\}^{-1} \rfloor$ copies of X_j^{MC}, and randomize sampling only for the fractional expectation, $\mathbb{E}[R_j] - \lfloor \mathbb{E}[R_j] \rfloor$. One such procedure is called **systematic resampling**.

The SMC filter recursion, Part II. Resampling

- We can set $X^F_{1:J}$ to be an unweighted resampling of $\left(X^P_{n,1:J}, W_{n,1:J}\right)$. Since $\left(X^P_{n,1:J}, W_{n,1:J}\right) \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$, this gives $X^F_{n,1:J} \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$.

The SMC filter recursion, Part II. Resampling

- We can set $X^F_{1:J}$ to be an unweighted resampling of $\left(X^P_{n,1:J}, W_{n,1:J}\right)$. Since $\left(X^P_{n,1:J}, W_{n,1:J}\right) \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$, this gives $X^F_{n,1:J} \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$.
- Why do we carry out resampling? We have a legitimate Monte Carlo approximation both before and afterwards. The resampling adds additional Monte Carlo variability, which is bad. What advantage does it offer?

The SMC filter recursion, Part II. Resampling

- We can set $X^F_{1:J}$ to be an unweighted resampling of $\left(X^P_{n,1:J}, W_{n,1:J}\right)$. Since $\left(X^P_{n,1:J}, W_{n,1:J}\right) \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$, this gives $X^F_{n,1:J} \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$.
- Why do we carry out resampling? We have a legitimate Monte Carlo approximation both before and afterwards. The resampling adds additional Monte Carlo variability, which is bad. What advantage does it offer?
- Without resampling, weights get more and more imbalanced. Eventually, all particles except one have negligible weight.

The SMC filter recursion, Part II. Resampling

- We can set $X^F_{1:J}$ to be an unweighted resampling of $\left(X^P_{n,1:J}, W_{n,1:J}\right)$. Since $\left(X^P_{n,1:J}, W_{n,1:J}\right) \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$, this gives $X^F_{n,1:J} \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$.
- Why do we carry out resampling? We have a legitimate Monte Carlo approximation both before and afterwards. The resampling adds additional Monte Carlo variability, which is bad. What advantage does it offer?
- Without resampling, weights get more and more imbalanced. Eventually, all particles except one have negligible weight.
- With resampling, more particles are available to explore the area most consistent with the data.

The SMC filter recursion, Part II. Resampling

- We can set $X^F_{1:J}$ to be an unweighted resampling of $\left(X^P_{n,1:J}, W_{n,1:J}\right)$. Since $\left(X^P_{n,1:J}, W_{n,1:J}\right) \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$, this gives $X^F_{n,1:J} \overset{MC}{\sim} f_{X_n|Y_{1:n}}(x_n|y^*_{1:n})$.
- Why do we carry out resampling? We have a legitimate Monte Carlo approximation both before and afterwards. The resampling adds additional Monte Carlo variability, which is bad. What advantage does it offer?
- Without resampling, weights get more and more imbalanced. Eventually, all particles except one have negligible weight.
- With resampling, more particles are available to explore the area most consistent with the data.
- Resampling is not a panacea. Recursive numerical integration relies for success on **stability**. A small error at one step should have diminishing consequences later.

Sequential Monte Carlo (SMC): a "vanilla" particle filter

input: simulator for $f_{X_n|X_{n-1}}(x_n \,|\, x_{n-1}\,;\theta)$; simulator for $f_{X_0}(x_0\,;\theta)$;
evaluator for $f_{Y_n|X_n}(y_n \,|\, x_n\,;\theta)$;
parameter, θ; data, $y^*_{1:N}$; number of particles, J.

---

initialize filter particles: simulate $X^F_{0,j} \sim f_{X_0}(\,\cdot\,;\theta)$ for j in $1:J$.

for n in $1:N$ **do**

simulate for prediction: $X^P_{n,j} \sim f_{X_n|X_{n-1}}\big(\,\cdot \,|\, X^F_{n-1,j};\theta\big)$ for j in $1:J$.

evaluate weights: $w(n,j) = f_{Y_n|X_n}(y^*_n \,|\, X^P_{n,j}\,;\theta)$ for j in $1:J$.

normalize weights: $\tilde{w}(n,j) = w(n,j)/\sum_{m=1}^{J} w(n,m)$.

apply systematic resampling: $k_{1:J}$ with $\mathbb{P}[k_j = m] = \tilde{w}(n,m)$.

resample: set $X^F_{n,j} = X^P_{n,k_j}$ for j in $1:J$.

conditional log likelihood: $\hat{\ell}_{n|1:n-1} = \log\big(J^{-1}\,\sum_{m=1}^{J} w(n,m)\big)$.

end

---

output: log likelihood estimate, $\hat{\ell}(\theta) = \sum_{n=1}^{N} \hat{\ell}_{n|1:n-1}$;
filter sample, $X^F_{n,1:J}$, for n in $1:N$.

Systematic resampling

input: Weights, $\tilde{w}_{1:J}$, normalized so that $\sum_{j=1}^{J} \tilde{w}_j = 1$.

---

construct cumulative sum: $c_j = \sum_{m=1}^{j} \tilde{w}_m$, for j in $1:J$.
uniform initial sampling point: $U_1 \sim \text{Uniform}(0, J^{-1})$.
evenly spaced sampling points: $U_j = U_1 + (j-1)J^{-1}$, for j in $2:J$.
initialize: set $p = 1$.
for j in $1:J$ **do**
    while $U_j > c_p$ **do**
        step to the next resampling index: set $p = p + 1$.
    end
    assign resampling index: set $k_j = p$.
end

---

output: resampling indices, $k_{1:J}$.

Some history of sequential Monte Carlo (SMC)

- SMC grew in the 1990s, simultaneously called **particle filtering**, **bootstrap filtering**, **Monte Carlo filtering**, **sequential importance sampling with resampling**, and **the condensation algorithm**.
- Used in physics and chemistry since the 1950s. Poorly marketed as **Poor man's Monte Carlo** (Hammersley and Morton, 1954).
- In modern theory, SMC and MCMC have similar asymptotic guarantees.
- SMC provides an alternative to MCMC for many computations.
- To simulate the 3D structure of a molecule with 1000 atoms, MCMC transitions adjust the position of all 1000 atoms; SMC builds up the molecule one atom at a time.
- For dynamic systems, SMC is preferred.

An evolutionary analogy for the SMC algorithm

- SMC can be viewed as Darwinian natural selection on the swarm of particles. The fittest particles (measured by consistency with the data) are over-represented in the next generation (meaning the next loop through the prediction and filtering recursion).
- The weighting and resampling procedure propagates particles consistent with data; inconsistent particles are "pruned" or "killed."
- Propagated particles are "mutated" according to the stochastic dynamic system, adding variation to the swarm.
- The "genome" of the particle is its state. The Markov property ensures that this genome is heritable: the future trajectory of the particle depends only on its current state.
- We have the ingredients for Darwinian evolution: mutation and selection based on a heritable characteristic.
- The natural selection is done in such a way that SMC approximates an ideal nonlinear filter.

When and why does SMC fail?

- We expect evolution to be good at finding fitness improvements locally in genetic space.
- We do not expect evolution (or evolutionary optimization algorithms, or any other known methods) to be good at finding globally optimal solutions to complex problems.
  What is the globally "optimal" animal?
- The theory for SMC (and more generally, for MCMC, simulated annealing, etc) typically assures global convergence given sufficient computer time.
- Practical interpretation of this theory requires care! When appropriate, practitioners should interpret global convergence results as indicators of good local behavior.

Particle depletion

- Evolution since the most recent common ancestor (MRCA) defines the 'local' neighborhoods which an evolutionary search can hope to explore.
- When selection is strong, or the offspring distribution is highly skewed, the MRCA can be only a few generations back even for a large number of particles (say, $J = 10^5$).
- The unpleasant phenomenon of the proximity of the MRCA and its consequences for global search is known as **particle depletion**.

References

Bjørnstad, O. N. and Grenfell, B. T. (2001). Noisy clockwork: Time series analysis of population fluctuations in animals. *Science*, 293:638–643.

Grewal, M. S. and Andrews, A. P. (2010). Applications of Kalman filtering in aerospace 1960 to the present. *IEEE Control Systems*, 30(3):69–78.

Hammersley, J. M. and Morton, K. W. (1954). Poor man's Monte Carlo. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 16:23–38.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45.

Montemerlo, M. and Thrun, S. (2007). *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*. Springer.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–285.