Nonlinear system identification using sequential Monte Carlo methods

Part 2 – Sequential Monte Carlo, the particle filter

Thomas Schöhn
Division of Systems and Control
Department of Information Technology
Uppsala University.
Email: thomas.schon@it.uu.se,
www: user.it.uu.se/~thosc112

Outline – Part 2

Aim: Derive the bootstrap particle filter and provide some of its properties.

1. Summary of Part 1
2. The Monte Carlo idea
3. Importance sampling
4. Derive the bootstrap particle filter
5. Sequential Monte Carlo
   a) General problem formulation
   b) The underlying pdf
   c) Convergence results

Summary of Part 1 (I/II)

The two identification strategies we are concerned with are:

- **Marginalization** Deal with the states by marginalizing (integrating) them out.
- **Data augmentation** Deal with the states by treating them as auxiliary variables to be estimated along with the parameters.

Summary of Part 1 (II/II)

The measurement update
\[ p(x_t | y_{1:t}) = \frac{g(y_t | x_t) p(x_t | y_{1:t-1})}{p(y_t | y_{1:t-1})}, \]

and time update
\[ p(x_t | y_{1:t-1}) = \int f(x_{t+1} | x_t) p(x_{t-1} | y_{1:t-1}) \, dx_{t-1}. \]

The joint smoothing pdf \( p(x_{1:T} | y_{1:T}) \) is given by
\[
p(x_{1:T} | y_{1:T}) = p(x_T | y_{1:T}) \prod_{t=1}^{T-1} \frac{f(x_{t+1} | x_t) p(x_t | y_{1:t})}{p(x_{t+1} | y_{1:t})}.
\]
**Why do we need Monte Carlo methods?**

In solving inference problems we are typically faced with various integration problems, which tend to be intractable and live in large dimensional spaces.

For example **expectation** arising in obtaining a point estimate. A commonly used point estimate is the conditional mean

\[
\hat{x}_{t|t} = \mathbb{E}[x_t | y_{1:t}] = \int x_t p(x_t | y_{1:t}) dx_t.
\]

Monte Carlo methods provides **computational solutions** where the distributions of interest are approximated by a large number of $N$ random samples called particles.

Monte Carlo methods respects the model and the expressions we are trying to approximate.

---

**The Monte Carlo idea (I/II)**

(Very) restrictive assumption: Assume that we have $N$ samples \( \{x^i\}_{i=1}^N \) from the target density \( \pi(x) \),

\[
\hat{\pi}(x) = \sum_{i=1}^N \frac{1}{N} \delta_{x^i}(x)
\]

Allows for the following approximation of the integral,

\[
\mathbb{E}[\varphi(x)] = \int \varphi(x) \pi(x) dx \approx \int \varphi(x) \sum_{i=1}^N \frac{1}{N} \delta_{x^i}(x) dx = \frac{1}{N} \sum_{i=1}^N \varphi(x^i)
\]

\[
\int + \delta \rightarrow \sum
\]

---

**The Monte Carlo idea (II/II)**

The integral

\[
I(\varphi(x)) \triangleq \mathbb{E}[\varphi(x)] = \int \varphi(x) \pi(x) dx.
\]

is approximated by

\[
\hat{I}_N(\varphi(x)) = \frac{1}{N} \sum_{i=1}^N \varphi(x^i).
\]

The strong law of large numbers tells us that

\[
\hat{I}_N(\varphi(x)) \xrightarrow{a.s.} I(\varphi(x)), \quad N \rightarrow \infty,
\]

and the central limit theorem state that

\[
\frac{\sqrt{N} \left( \hat{I}_N(\varphi(x)) - I(\varphi(x)) \right)}{\sigma_{\varphi}} \xrightarrow{d} \mathcal{N}(0, 1), \quad N \rightarrow \infty.
\]

---

**The Monte Carlo idea – toy illustration**

\[
\pi(x) = 0.3N(x | 2, 2) + 0.7N(x | 9, 19)
\]

5 000 samples

50 000 samples

**Obvious problem:** In general we are **not** able to directly sample from the density we are interested in.
Importance sampling – problem and idea

Importance sampling can be used to evaluate integrals of the form

\[ I(\varphi(x)) = E[\varphi(x)] = \int \varphi(x)\pi(x)dx, \]

where it is hard to generate samples from the target density \( \pi(x) \).

Note that:

\[ \int \varphi(x)\pi(x)dx = \int \varphi(x)\frac{\pi(x)}{q(x)}q(x)dx. \]

**Idea:** Chose the proposal density \( q(x) \) such that it is easy to generate samples from it and compensate for the mismatch between the target and the proposal.

---

Importance sampling (IS)

We are free to choose the proposal density as long as \( \gamma(x) > 0 \Rightarrow q(x) > 0. \)

1. Draw \( N \) samples \( x^i \sim q(x), \ i = 1, \ldots, N. \)
2. Insert the Monte Carlo approximation

\[ \hat{\pi}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i}(x) \]

into

\[ \pi(x) = \frac{w(x)q(x)}{Z} \quad \text{and} \quad Z = \int w(x)q(x)dx. \]

---

Algorithm 1 Importance sampler (IS)

1. Sample \( x^i \sim q(x) \).
2. Compute the weights \( w(x^i) = \gamma(x^i)/q(x^i) \).
3. Normalize the weights \( w^i = w(x^i)/\sum_j w(x^j) \).

Each step is carried out for \( i = 1, \ldots, N. \)

The convergence of the resulting approximation \( \hat{\pi}(x) = \sum_{i=1}^{N} w^i\delta_{x^i}(x) \) is since long well established.

---

The importance of a good proposal density

Sampling from a user-chosen proposal distribution \( q \) is corrected for by the weights, which accounts for the discrepancy between the proposal \( q \) and the target \( \pi \).

Lesson learned: It is important to be careful in selecting the proposal density.
Outline – Part 2

**Aim:** Derive the bootstrap particle filter and provide some of its properties.

1. Summary of Part 1
2. The Monte Carlo idea
3. Importance sampling
4. Derive the bootstrap particle filter
5. Sequential Monte Carlo
   a) General problem formulation
   b) The underlying pdf
   c) Convergence results

Using IS for our purposes

Recall that the nonlinear filtering problem amounts to computing the filter PDF \( p(x_t | y_{1:t}) \) when the model is given by

\[
\begin{align*}
  x_{t+1} | x_t & \sim f(x_{t+1} | x_t), \\
  y_t | x_t & \sim g(y_t | x_t), \\
  x_1 & \sim \mu(x_1).
\end{align*}
\]

We have showed that the solution is

\[
p(x_t | y_{1:t}) = \frac{g(y_t | x_t)p(x_t | y_{1:t-1})}{p(y_t | y_{1:t-1})},
\]

\[
p(x_t | y_{1:t-1}) = \int f(x_t | x_{t-1})p(x_{t-1} | y_{1:t-1})dx_{t-1}.
\]

**Relevant idea:** Try to solve this using importance sampling!!

Finding a proposal (I/II)

Assume (in an “induction-like” fashion) that at time \( t - 1 \) have

\[
\hat{p}(x_{t-1} | y_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^i \delta_{x_{t-1}^i} (x_{t-1}),
\]

allowing us to approximate the integral for \( p(x_t | y_{1:t-1}) \),

\[
\hat{p}(x_t | y_{1:t-1}) = \int f(x_t | x_{t-1}) \sum_{i=1}^{N} w_{t-1}^i \delta_{x_{t-1}^i} (x_{t-1})dx_{t-1} = \sum_{i=1}^{N} w_{t-1}^i f(x_t | x_{t-1}^i).
\]

Hence, \( p(x_t | y_{1:t-1}) \) is approximated by a mixture distribution, where there is one mixture component for each of the \( N \) particles.

Finding a proposal (I/II)

Inserting

\[
\hat{p}(x_t | y_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^i f(x_t | x_{t-1}^i)
\]

into

\[
p(x_t | y_{1:t}) = \frac{g(y_t | x_t)p(x_t | y_{1:t-1})}{p(y_t | y_{1:t-1})},
\]

results in

\[
p(x_t | y_{1:t}) \approx \frac{g(y_t | x_t)}{p(y_t | y_{1:t-1})} \sum_{i=1}^{N} w_{t-1}^i f(x_t | x_{t-1}^i).
\]

The task is now to approximate (1) using importance sampling.
**Importance sampling reminder**

**Algorithm 2 Importance sampler**

1. Sample $x^i \sim r(x)$.
2. Compute the weights $\nu(x^i) = \gamma(x^i)/r(x^i)$.
3. Normalize the weights $\nu^i = \nu(x^i)/\sum_{j=1}^{N} \nu(x^j)$.

Choose a proposal density of the same form as $p(x_t | y_{1:t})$,

$$r(x_t | y_{1:t}) = \sum_{i=1}^{N} \nu^i_{t-1} r(x_t | x^i_{t-1}, y_t),$$

where both the mixture components $r(x_t | x^i_{t-1}, y_t)$ and the mixture weights $\nu^i_{t-1}$ are design choices.

**Sampling from the proposal**

Two step procedure to sample from mixture proposal $r(x_t | y_{1:t})$.

Notation: The ancestor index $a^i_t$ denotes the index of the mixture component selected for the $i^{th}$ component at time $t$.

1. Select one of the components (resampling),

$$\mathbb{P}(a^i_t = j) = \nu^i_{t-1}, \quad j = 1, \ldots, N.$$

2. Generate a sample from that component,

$$x^i_t \sim r(x_t | x^a^i_{t-1}, y_t).$$

Repeat this $N$ times.

**Resampling (I/II)**

The particle $\bar{x}_{t-1} = x^a^i_{t-1}$ is referred to as the ancestor particle of $x^i_t$, since $x^i_t$ is generated conditionally on $\bar{x}_{t-1}$.

This also explains why $a^i_t$ is referred to as the ancestor index, since it indexes the ancestor of particle $x^i_t$ at time $t - 1$.

**Resampling (II/II)**

Resampling is the procedure that (randomly) turns a weighted set of samples $\{x^i_{t-1}, w^i_{t-1}\}_{i=1}^{N}$ into an unweighted set of samples $\{\bar{x}_{t-1}, 1/N\}_{i=1}^{N}$ according to

$$\mathbb{P}(a^i_t = j) = \nu^i_{t-1}, \quad \bar{x}_{t-1} = x^{a^i_t}_{t-1}.$$
Next step – computing the weights

Algorithm 3 Importance sampler
1. Sample $x_t^i \sim r(x)$. 
2. Compute the weights $w(x^i_t) = \gamma(x^i_t)/r(x^i_t)$. 
3. Normalize the weights $w^t = w(x^i_t)/\sum_{j=1}^N w(x^j_t)$.

Compute the weights

$$w_t(x^i_t) = \frac{g(y_t | x^i_t) \tilde{p}(x_t | y_{t-1})}{r(x_t | y_{t-1})} = \frac{g(y_t | x^i_t) \sum_{j=1}^N w^j_{t-1} f(x^j_t | x^j_{t-1})}{\sum_{j=1}^N \nu^j_{t-1} r(x^j_t | x^j_{t-1}, y_t)}$$

Computational complexity: $O(N^2)$!

Pragmatic solution to quadratic complexity

A pragmatic solution to the quadratic computational complexity is to use the freedom available in the proposal density,

$$r(x_t | y_{1:t}) = \sum_{j=1}^N w^j_{t-1} f(x^j_t | x^j_{t-1})$$

Resulting weight computation

$$w_t(x^i_t) = g(y_t | x^i_t)$$. 

Result – A first particle filter

Algorithm 4 Bootstrap particle filter (for $i = 1, \ldots, N$

1. Initialization ($t = 1$):
   (a) Sample $x_1^i \sim \mu(x_1)$.
   (b) Compute the weights $\tilde{w}_1^i = g(y_1 | x_1^i)$ and normalize, 
       $w_1^i = \tilde{w}_1^i / \sum_{j=1}^N \tilde{w}_1^j$.
2. for $t = 2$ to $T$ do
   (a) Resample $\{x_{t-1}^i, w_{t-1}^i\}$ resulting in equally weighted particles $\{x_{t-1}^i, 1/N\}$.
   (b) Propagate by sampling $x_t^i \sim f(x_t | x_{t-1}^i)$.
   (c) Weight by computing $\tilde{w}_t^i = g(y_t | x_t^i)$ and normalize 
       $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$.

SMC structure

The structure is the same for all SMC algorithms. For the bootstrap PF we have, 

Resampling: $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N \rightarrow \{\bar{x}_{t-1}^i, 1/N\}_{i=1}^N$.

Propagation: $x_t^i \sim f(x_t | \bar{x}_{t-1}^i)$.

Weighting: $\tilde{w}_t^i = W_t(x_t^i) = g(y_t | x_t^i)$ and normalize.

The result is a new weighted set of particles $\{x_t^i, w_t^i\}_{i=1}^N$. 

Result – A first particle filter

Algorithm 4 Bootstrap particle filter (for $i = 1, \ldots, N$

1. Initialization ($t = 1$):
   (a) Sample $x_1^i \sim \mu(x_1)$.
   (b) Compute the weights $\tilde{w}_1^i = g(y_1 | x_1^i)$ and normalize, 
       $w_1^i = \tilde{w}_1^i / \sum_{j=1}^N \tilde{w}_1^j$.
2. for $t = 2$ to $T$ do
   (a) Resample $\{x_{t-1}^i, w_{t-1}^i\}$ resulting in equally weighted particles $\{x_{t-1}^i, 1/N\}$.
   (b) Propagate by sampling $x_t^i \sim f(x_t | x_{t-1}^i)$.
   (c) Weight by computing $\tilde{w}_t^i = g(y_t | x_t^i)$ and normalize 
       $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$.
Important “design” considerations

1. Adaptive resampling – only resample “when needed”.
2. Be careful when selecting the importance density.
3. Exploit analytically tractable sub-structures (Rao-Blackwellization).
4. ...

Application – indoor localization (I/II)

Aim: Compute the position of a person moving around indoors using sensors (inertial, magnetometer and radio) located in an ID badge and a map.

PDF for an office environment, the bright areas are rooms and corridors (i.e. walkable space).

The sensors (IMU and radio) and the DSP are mounted inside an ID badge.

Application – indoor localization (II/II)

Targeting the JSD instead

Our derivation of the PF is rather non-standard. The reason I like it is that it clearly shows why the resampling step is needed and where the need for the resampling step comes from.

The more standard way of deriving the PF is by targeting the sequence of joint smoothing densities (JSD) \( \{p(x_{1:t} | y_{1:t})\}_{t \geq 1} \).

Enlightening derivation as well!! Shows that

\[ SMC = SIS + \text{Resampling} \]
Can you see any problems with the algorithm producing approximations of the JSD according to

\[ p(x_{1:t} \mid y_{1:t}) = \sum_{i=1}^{N} w_i^t \delta_{x_{1:t}^i}(x_{1:t}) \]

The resampling step remove particles with small weights and duplicate particles with large weights.

This results in **path degeneracy**, which we explain using a simple example.

**Left plot**: At each point in time all particles are plotted using a black dot and each particle is connected with its ancestor using a black line.

**Right plot**: The grey dots represents the \( p(x_t \mid y_{1:t}) \) at each point in time. The black lines shows the particle trajectories \( x_{1:25}^{i=1} \) at time \( t = 25 \).

The right plot corresponds to the left plot with all trajectories that are not resampled removed (all particles are still visualized using gray dots).

This implies that if we are interested in the smoothing distribution

\[ p(x_{1:T} \mid y_{1:T}) \]

or some of its marginals we are **forced** to use different algorithms, which leads us to particle smoothers. **Backward simulation is key** here (and elsewhere!).
**Outline – Part 2**

**Aim:** Derive the bootstrap particle filter and provide some of its properties.

1. Summary of Part 1
2. The Monte Carlo idea
3. Importance sampling
4. Derive the bootstrap particle filter
5. Sequential Monte Carlo
   a) General problem formulation
   b) The underlying pdf
   c) Convergence results

---

**SMC – (abstract) problem formulation**

The distribution of interest, $\pi(x)$ is called **target distribution**.

**Problem formulation:** Sample sequentially from a sequence of target distributions $\{\pi_t(x_{1:t})\}_{t \geq 1}$ of increasing dimension, where

$$\pi_t(x_{1:t}) = \frac{\gamma_t(x_{1:t})}{Z_t},$$

such that $\gamma_t(x_t) : X^t \to \mathbb{R}^+$ is known pointwise and $Z_t = \int \pi(x_{1:t}) dx_{1:t}$ is computationally challenging.

So far we have seen that this formulation includes nonlinear SSMs, $\pi_t(x_{1:t}) = p(x_{1:t} | y_{1:t})$, $\gamma_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$, $Z_t = p(y_{1:t})$, but **importantly** it is more general than that.

---

**Generality of SMC**

The sequence of target distributions $\{\pi_t(x_{1:t})\}_{t \geq 1}$ can be constructed in many different ways!

Two concrete examples:

1. When variables are **not** defined on product spaces, $\pi : X \to \mathbb{R}^+$ we can introduce an artificial sequence of (auxiliary) distributions, where we are only interested in one of the marginals.
   

2. Inference in probabilistic graphical models (PGM) is possible via such a sequence of auxiliary distributions. SMC provide consistent estimates and an unbiased estimate of the partition (normalization) constant (also for **loopy** PGMs!).


---

**Sequential Monte Carlo – particle filter**

The sequence of target distributions $\{\pi_t(x_{1:t})\}_{t \geq 1}$ of increasing dimension, where

$$\pi_t(x_{1:t}) = \frac{\gamma_t(x_{1:t})}{Z_t},$$

such that $\gamma_t(x_t) : X^t \to \mathbb{R}^+$ is known pointwise and $Z_t = \int \pi(x_{1:t}) dx_{1:t}$ is computationally challenging.

So far we have seen that this formulation includes nonlinear SSMs, $\pi_t(x_{1:t}) = p(x_{1:t} | y_{1:t})$, $\gamma_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$, $Z_t = p(y_{1:t})$, but **importantly** it is more general than that.

**SMC = resampling + sequential importance sampling**

1. **Resampling:** $P(a_i^t = j) = \bar{w}_{t-1}^i / \sum_i \bar{w}_{t-1}^i$.
2. **Propagation:** $x_t^j \sim f_{\theta}(x_t | x_{1:t-1}^a)$ and $x_{1:t}^j = \{x_{1:t-1}^a, x_t^j\}$.
3. **Weighting:** $\bar{w}_t^i = W_t(x_t^i) = g_{\theta}(y_t | x_t^i)$.

The ancestor indices $\{a_i^t\}_{i=1}^N$ are very **useful** auxiliary variables! They make the stochasticity of the resampling step explicit.
Sequential Monte Carlo – particle filter

Let
\[ x_t \equiv \{x_1^t, \ldots, x_N^t\}, \quad a_t \equiv \{a_1^t, \ldots, a_N^t\} \]
denote all particles and ancestor indices generated at time \( t \).

The SMC algorithm generates a single realization of a collection of random variables
\[ \{x_{1:T}, a_{2:T}\} \in X^{NT} \times \{1, \ldots, N\}^{N(T-1)} \]
distributed according to
\[
\psi(x_{1:T}, a_{2:T}) \triangleq N \prod_{i=1}^{N} q_1(x_1^i) \prod_{t=2}^{T} N \prod_{i=1}^{N} M_t(a_t^i, x_t^i),
\]
where
\[
M_t(a_t, x_t) = \frac{w_t^{a_t}}{\sum_t w_t^{a_t}} f_t(x_t | x_{1:t-1}^{a_t}).
\]

Convergence results in one slide...

Let \( \varphi : X \mapsto \mathbb{R} \) be some test function of interest. The expectation
\[
E[\varphi(x_t) | y_{1:t}] = \int \varphi(x_t) p(x_t | y_{1:t}) \, dx_t,
\]
can be estimated by the particle filter
\[
\hat{\varphi}_N^t \triangleq \sum_{i=1}^{N} w_t^i \varphi(x_t^i).
\]
The CLT governing the convergence of this estimator states
\[
\sqrt{N} \left( \hat{\varphi}_N^t - E[\varphi(x_t) | y_{1:t}] \right) \overset{d}{\to} N(0, \sigma_t^2(\varphi)).
\]

The likelihood estimate \( \hat{p}(y_{1:t}) = \prod_{s=1}^{t} \left\{ \frac{1}{N} \sum_{i=1}^{N} \bar{w}_s^i \right\} \) from the PF is unbiased, \( E[\hat{p}(y_{1:t})] = p(y_{1:t}) \) for any value of \( N \) and there are CLTs available as well.