Lecture 2 - Solving inference problems using SMC

UPPSALA UNIVERSITET

Thomas Schön
e-mail: thomas.schon@it.uu.se

Division of Systems and Control
Department of Information Technology
Uppsala University

Summary – Lecture 1 (I/III)

A **model** is a compact and interpretable representation of the data that is observed. In this course, **model = PDF**!

Introduced the nonlinear state space model (SSM)

$$\begin{aligned} x_{t+1} \mid x_t &\sim f_{\theta,t}(x_{t+1} \mid x_t, u_t), \\ y_t \mid x_t &\sim g_{\theta,t}(y_t \mid x_t, u_t), \\ x_1 &\sim \mu_{\theta,t}(x_1), \quad (\theta \sim p(\theta)). \end{aligned}$$

Summary – Lecture 1 (II/III)

State inference referes to the problem of finding information about the state(s) $x_{k:l}$ based on the available measurements $y_{1:t}$.

We showed the **key strategies** for state filtering and state smoothing. These strategies forms the foundation for what we will do today.

The state filtering problem: Compute the filtering pdf $p(x_t \mid y_{1:t})$.

Solved via the forward computations

$$p(x_t \mid y_{1:t}) = \frac{\overbrace{g(y_t \mid x_t)}^{\text{measurement}} \overbrace{p(x_t \mid y_{1:t-1})}^{\text{prediction pdf}}}{p(y_t \mid y_{1:t-1})},$$

$$p(x_t \mid y_{1:t-1}) = \int \underbrace{f(x_t \mid x_{t-1})}_{\text{dynamics}} \underbrace{p(x_{t-1} \mid y_{1:t-1})}_{\text{filtering pdf}} \mathrm{d}x_{t-1},$$

Summary – Lecture 1 (III/III)

We also derived the backward computations, where the **backward kernel** is key

$$p(x_t \mid x_{t+1}, y_{1:t}) = \frac{f(x_{t+1} \mid x_t) p(x_t \mid y_{1:t})}{p(x_{t+1} \mid y_{1:t})}.$$

Monte Carlo methods provide computational solutions to otherwise intractable problems relying on random sampling.

Importance sampling: Offers a way of evaluating integrals of the form

$$\int \varphi(z)\pi(z)\mathrm{d}z.$$

Outline – Lecture 2

1. Derive a first working particle filter (most common SMC sampler)
2. Particle filtering examples
   a) An LGSS model
   c) A nonlinear SSM
   b) Indoor positioning
3. Targeting the JSD and particle degeneracy
4. Ancestor indices and an equivalen SMC formulation
5. Particle smoothers (very brief)
6. Some recent work

Using IS for our purposes

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

$$\begin{aligned} x_{t+1} \mid x_t &\sim f(x_{t+1} \mid x_t), \\ y_t \mid x_t &\sim g(y_t \mid x_t), \\ x_1 &\sim \mu(x_1). \end{aligned}$$

We have showed that the solution is

$$p(x_t \mid y_{1:t}) = \frac{g(y_t \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})},$$

$$p(x_t \mid y_{1:t-1}) = \int f(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1})\mathrm{d}x_{t-1}.$$

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

Importance sampling (IS)

Algorithm 1 Importance sampler (IS)

1. Sample $z^i \sim q(z)$.
2. Compute the weights $\widetilde{w}^i = \widetilde{\pi}(z^i)/q(z^i)$.
3. Normalize the weights $w^i = \widetilde{w}^i / \sum_{j=1}^N \widetilde{w}^j$.

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

Resampling (I/II)

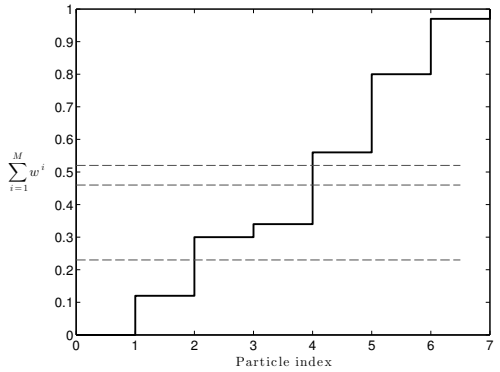
Resampling is the procedure that (randomly) **turns a weighted** set of samples $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N$ **into an unweighted** set of samples $\{\widetilde{x}_{t-1}^i, 1/N\}_{i=1}^N$ according to

$$\mathbb{P}\left(\widetilde{x}_{t-1} = x_{t-1}^i \mid \{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N\right) = w_{t-1}^i,$$

Resampling can be implemented in several ways and many algorithms exists.

Rather than going through all these algorithms, let us illustrate the meaning of resampling, resulting in one resampling algorithm.

Resampling (II/II)



Illustrating how resampling works (using 7 particles).

1. Compute the cumulative sum of the weights.
2. Generate $u \sim \mathcal{U}[0,1]$.

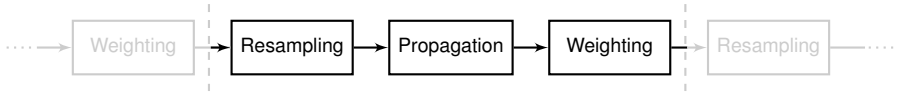
Three new samples are generated in the figure above, corresponding to sample $2, 4$ and 4.

Result: A first particle filter

Algorithm 2 Bootstrap particle filter

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

The structure of the particle filter



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

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

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

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

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

An LGSS example (I/II)

"Whenever you are working on a nonlinear inference method, always make sure that it solves the linear special case first."

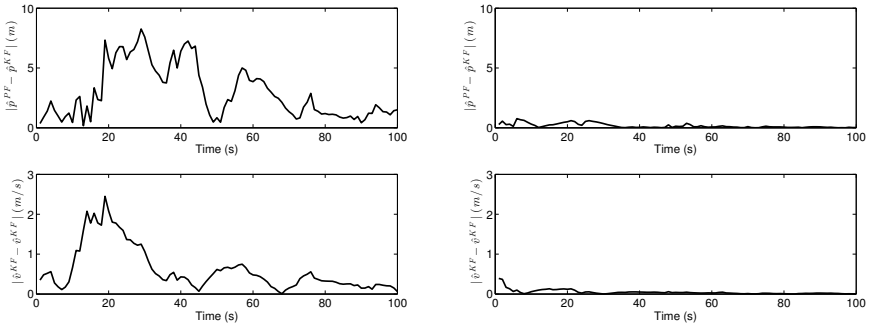
Consider the following LGSS model (simple one dimensional positioning example)

$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \\ a_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & T_s & T_s^2/2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + \begin{pmatrix} T_s^3/6 \\ T_s^2/2T_s \end{pmatrix} v_t, \qquad v_t \sim \mathcal{N}(0, Q),$$

$$y_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + e_t, \qquad e_t \sim \mathcal{N}(0, R).$$

The KF provides the true filtering density, which implies that we can compare the PF to the truth in this case.

An LGSS example (II/II)



Using 200 particles. Using 20 000 particles.

The PF estimate converge as the number of particles tends to infinity.

Xiao-Li Hu, Thomas B. Schön and Lennart Ljung. **A Basic Convergence Result for Particle Filtering**. *IEEE Transactions on Signal Processing*, 56(4):1337-1348, April 2008.

D. Crisan and A. Doucet, **A survey of convergence results on particle filtering methods for practitioners**, *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736-746, 2002.

A nonlinear example (I/II)

Consider the following SSM (standard example in PF literature)

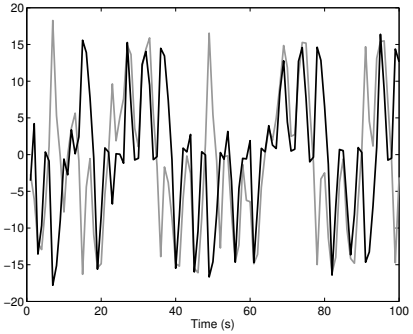
$$x_{t+1} = \frac{x_t}{2} + \frac{25x_t}{1+x_t^2} + 8\cos(1.2t) + v_t, \qquad v_t \sim \mathcal{N}(0, 0.5),$$

$$y_t = \frac{x_t^2}{20} + e_t, \qquad e_t \sim \mathcal{N}(0, 0.5).$$

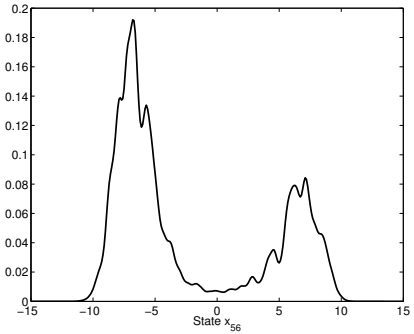
What it tricky with this model?

The best (only?) way of really understanding something is to implement it yourself.

A nonlinear example (II/II)



True state (gray) and PF conditional mean estimate (black).



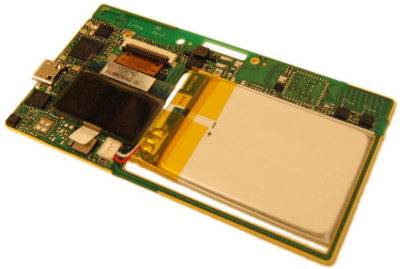
PF estimate of the filtering pdf $\widehat{p}(x_{56} \mid y_{1:56})$.

Another indication that the conditional mean point estimate is dangerous.

Application example – indoor positioning (I/III)

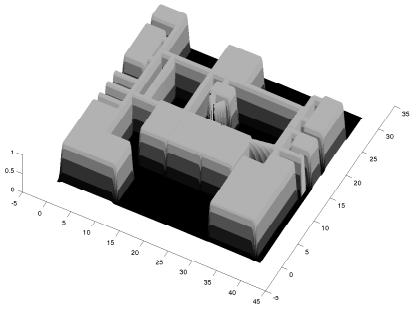
Aim: Compute the position of a person moving around indoors using sensors located in an ID badge and a map.

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

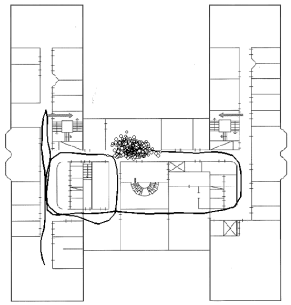


The inside of the ID badge.

Application example – indoor positioning (II/III) 17(46)



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



An estimated trajectory and the particle cloud visualized at a particular instance.

Application example – indoor positioning (III/III) 18(46)

Show movies

This is work by Johan Kihlberg and Simon Tegelid in their MSc thesis entitled **Map aided indoor positioning**.

This work will also be presented at **Reglermötet** here in Linköping in the beginning of June.

Outline – Lecture 2 19(46)

1. Derive a first working particle filter (most common SMC sampler)
2. Particle filtering examples
   a) An LGSS model
   b) A nonlinear SSM
   c) Indoor positioning
3. **Targeting the JSD and particle degeneracy**
4. Ancestor indices and an equivalen SMC formulation
5. Particle smoothers (very brief)
6. Some recent work

Targeting the JSD instead 20(46)

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)** $\{p(x_{1:t} \mid y_{1:t})\}_{t \geq 1}$ (see lecture notes for details).

SMC algorithm

Algorithm 3 Sequential Monte Carlo (SMC)

1. Initialise by sampling $x_1^i \sim Q_1(x_1)$, setting $\widetilde{w}_1^i = W_1(x_1^i)$, normalizing and setting $t = 1$.
2. **for** $t = 1$ **to** T **do**
   (a) **Resample:** $\{x_{1:t}^i, w_t^i\}_{i=1}^N \rightarrow \{\widetilde{x}_{1:t}^i, 1/N\}_{i=1}^N$.
   (b) **Propagate:** propose new samples, $x_{t+1}^i \sim q_t(x_{t+1} \mid \widetilde{x}_{1:t}^i)$ and set $x_{1:t+1}^i = \{x_{1:t}^i, \widetilde{x}_{t+1}^i\}$.
   (c) **Weight:** $\widetilde{w}_{t+1}^i = W_t(x_{t+1}^i, \widetilde{x}_{1:t}^i)$ and normalize.

There is a myriad of possible design choices available, some of them carrying their own name. The important thing is to understand the basic principles in action; the rest is just design choices.

Problems?

Can you see any problems with the algorithm producing approximations of the JSDs according to

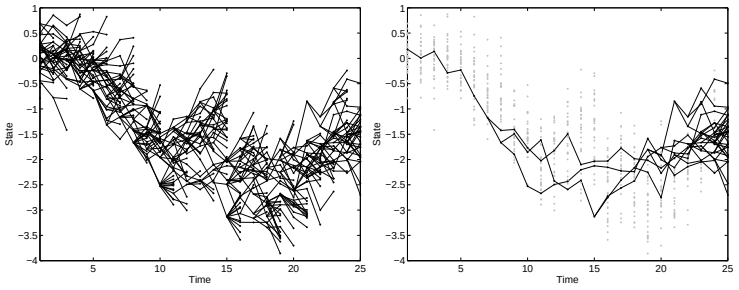
$$p(x_{1:t} \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_{1:t}^i}(x_{1:t})$$

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

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

Illustration of particle degeneracy (I/II)

Illustration of particle degeneracy (II/II)



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 $\{x_{1:25}^i\}_{i=1}^{30}$ 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).

Implication of the particle degeneracy problem

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**.

Resampling – again

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

$$\mathbb{P}\left(\widetilde{x}_{t-1} = x_{t-1}^i \mid \{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N\right) = w_{t-1}^i,$$

Using the **auxiliary variables** offered by the **ancestor indices** $\{a_t^i\}_{i=1}^N$ we can keep track of exactly what happens in each resampling step. This is important and valuable information.

Example – ancestral indices and ancestral paths

Evolution of three particles for $t = 1, 2, 3$.

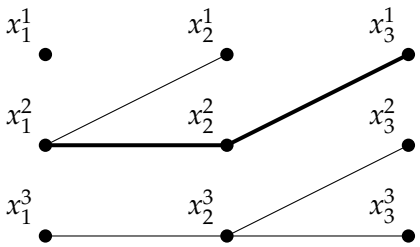


Figure: Evolution of a particle system. The ancestral path of x_3^1, i.e. $x_{1:3}^1$, is shown as a thick line.

Example – ancestral indices and ancestral paths

At time $t = 1$, particle x_1^2 is resampled twice and particle x_1^3 is resampled once. Hence, at time $t = 2$, the **ancestor indices** are

$$a_2^1 = 2,\ a_2^2 = 2 \text{ and } a_2^3 = 3.$$

Similarly, at time $t = 3$, the **ancestor indices** are given by

$$a_3^1 = 2,\ a_3^2 = 3 \text{ and } a_3^3 = 3.$$

The **ancestral path** of particle x_3^1, which we denote $x_{1:3}^1$, is shown as a thick line in the figure. This path is given recursively from the ancestor indices,

$$x_{1:3}^1 = (x_1^{a_2^{a_3^1}}, x_2^{a_3^1}, x_3^1) = (x_1^2, x_2^2, x_3^1).$$

SMC – equivalent formulation

Introduce a sequence of proposal kernels on the product space $\{1, \ldots, N\} \times \mathsf{X}$,

$$M_t(a_t, x_t) = w_{t-1}^{a_t} q_t(x_t \mid x_{1:t-1}^{a_t}).$$

Algorithm 4 Sequential Monte Carlo (SMC)

1. Initialise by sampling $x_1^i \sim q_1(x_1)$, setting $\widetilde{w}_1^i = W_1(x_1^i)$, normalizing and setting $t = 1$.
2. **for** $t = 2$ **to** T **do**
   (a) Draw $\{a_t^i, x_t^i\} \sim M_t(a_t, x_t)$.
   (b) Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$
   (c) Compute $\widetilde{w}_t^i = W_t(x_{1:t}^i)$ and normalize.
3. **end**

PF as a generator of random variables

The PF can be interpreted as a mechanism for generating a realization from the $2NT - N$ random variables $X_{1:T}, A_{2:T}$ described by the pdf

$$\psi(\mathbf{X}_{1:T}, \mathbf{A}_{2:T}) =$$

$$\underbrace{\left\{\prod_{i=1}^{N} q_1(X_1^i \mid y_1)\right\}}_{\text{Initialization}} \left\{\prod_{t=2}^{T}\prod_{i=1}^{N} \underbrace{\mathcal{C}(A_t^i \mid \mathbf{w}_{t-1})}_{\text{Resampling}} \underbrace{q_t\left(X_t^i \mid x_{1:t-1}^{a_t^i}, y_{1:t}\right)}_{\text{Propagation}}\right\}$$

which is defined on the space $\mathsf{X}^{TN} \times \{1, \ldots, N\}^{(T-1)N}$.

The PF is a **combination of two sampling operations**.

PF as a generator of random variables

The fact that the PF is a combination of two sampling operations can be taken one step further.

Combine the two sampling operations into on sampling operation using $M_t(a_t, x_t)$

$$\psi(\mathbf{X}_{1:T}, \mathbf{A}_{2:T}) = \left\{\prod_{i=1}^{N} q_1(X_1^i \mid y_1)\right\} \prod_{t=2}^{T}\prod_{i=1}^{N} M_t\left(A_t^i, X_t^i \mid \mathbf{w}_{t-1}, x_{1:t-1}^{a_t^i}, y_{1:t}\right)$$

Outline – Lecture 2

1. Derive a first working particle filter (most common SMC sampler)
2. Particle filtering examples
   a) An LGSS model
   b) A nonlinear SSM
   c) Indoor positioning
3. Targeting the JSD and particle degeneracy
4. Ancestor indices and an equivalen SMC formulation
5. **Particle smoothers (very brief)**
6. Some recent work

Recall – backward computations from lecture 1

By marginalizing

$$p(x_{1:T} \mid y_{1:T}) = p(x_T \mid y_{1:T}) \prod_{t=1}^{T-1} \frac{f(x_{t+1} \mid x_t) p(x_t \mid y_{1:t})}{p(x_{t+1} \mid y_{1:t})}.$$

w. r. t. $x_{1:t-1}$ and $x_{t+1:T}$ we obtain the following expression for the marginal smoothing pdf

$$p(x_t \mid y_{1:T}) = p(x_t \mid y_{1:t}) \int \frac{f(x_{t+1} \mid x_t) p(x_{t+1} \mid y_{1:T})}{p(x_{t+1} \mid y_{1:t})} \mathrm{d}x_{t+1}.$$

A first particle smoother

Algorithm 5 Forward filtering/Backward smoothing (FFBSm)

1. Run the PF and store the particles and their weights $\{(w_t^i, x_t^i)\}_{i=1}^N$ for $t = 1, \ldots, T$.
2. Initialise the smoothed weights $w_{T|T}^i = w_T^i$, for $i = 1, \ldots, N$.
3. **for** $t = T-1$ **to** 1 **do**
   (a) Compute the smoothed weights

$$w_{t|T}^i = w_t^i \sum_{k=1}^N w_{t+1|T}^i \frac{f(x_{t+1}^k \mid x_t^i)}{v_t^k}, \qquad v_t^k = \sum_{i=1}^N w_t^i f(x_{t+1}^k \mid x_t^i).$$

The resulting approximation $\widehat{p}(x_t \mid y_{1:T}) = \sum_{i=1}^N w_{t|T}^i \delta_{x_t^i}(x_t)$ does not suffer from degeneracy.

Recall – backwards simulation from lecture 1

$$p(x_{1:T} \mid y_{1:T}) = \left(\prod_{t=1}^T p(x_t \mid x_{t+1}, y_{1:t}) \right) p(x_T \mid y_{1:T}),$$

$$p(x_t \mid x_{t+1}, y_{1:t}) = \frac{f(x_{t+1} \mid x_t) p(x_t \mid y_{1:t})}{p(x_{t+1} \mid y_{1:t})}.$$

Strategy: Run a forward filter and generate samples backwards in time:

Sample $\widetilde{x}_T \sim p(x_T \mid y_{1:T})$
$\vdots$
Sample $\widetilde{x}_t \sim p(x_t \mid \widetilde{x}_{t+1}, y_{1:t})$
$\vdots$
Sample $\widetilde{x}_1 \sim p(x_1 \mid \widetilde{x}_2, y_1)$

The resulting sample is then by construction $\widetilde{x}_{1:T} \sim p(x_{1:T} \mid y_{1:T})$.

Sampling from the backward kernel

Key idea: Generate a backward trajectory by sampling from the following approximation

$$\widehat{p}(x_t \mid \widetilde{x}_{t+1}^j, y_{1:t}) = \sum_{i=1}^N \underbrace{\frac{w_t^i f(\widetilde{x}_{t+1}^j \mid x_t^i)}{\sum_{k=1}^N w_t^k f(\widetilde{x}_{t+1}^j \mid x_t^k)}}_{\triangleq \widetilde{w}_{t|T}^{i,j}} \delta_{x_t^i}(x_t)$$

of the backward kernel. This amounts to making use of the backwards simulation strategy

$$\widetilde{x}_T \sim \widehat{p}^N(x_T \mid y_{1:T}),$$
$$\widetilde{x}_t \sim \widehat{p}^N(x_t \mid \widetilde{x}_{t+1}, y_{1:T}),$$

to sample an entire backward trajectory $\widetilde{x}_{1:T}$.

Still computationally very costly!

Particle smoother – backwards simulation

1. At time T draw among the particles according to

$$\mathbb{P}\left(\widetilde{x}_T = x_T^i\right) = w_T^i.$$

2. At time $t = T-1, \ldots, 1$ draw among the particles according to

$$\mathbb{P}\left(\widetilde{x}_t = x_t^i\right) = \widetilde{w}_{t|T}^i,$$

Repeat M times, resulting in

$$\widehat{p}^M(x_{1:T} \mid y_{1:T}) = \frac{1}{M}\sum_{j=1}^{M} \delta_{\widetilde{x}_{1:T}^j}(x_{1:T})$$

Particle based FFBSi

Algorithm 6 Particle based FFBSi

1. Run a PF to generare $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$.
2. **Initialization ($t = T$):**
   (a) Sample independently $\{b_T(j)\}_{j=1}^M \sim \mathcal{C}\left(\{w_T^i\}_{i=1}^N\right)$.
   (b) Set $\widetilde{x}_T^j = x_T^{b_T(j)}$ for $j = 1, \ldots, M$.
3. **for** $t = T-1$ **to** 1 **do**
4. **for** $j = 1$ **to** M **do**
   (a) Compute $\widetilde{w}_{t|T}^{i,j} \propto w_t^i f(\widetilde{x}_{t+1}^j \mid x_t^i)$and normalize.
   (b) Sample independently $b_t(j) \sim \mathcal{C}\left(\{\widetilde{w}_{t|T}^{i,j}\}_{i=1}^N\right)$
   (c) Set $\widetilde{x}_t^j = x_t^{b_t(j)}$ and $\widetilde{x}_{1:T}^j = \{\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j\}$.

Particle based BSi – example (I/II)

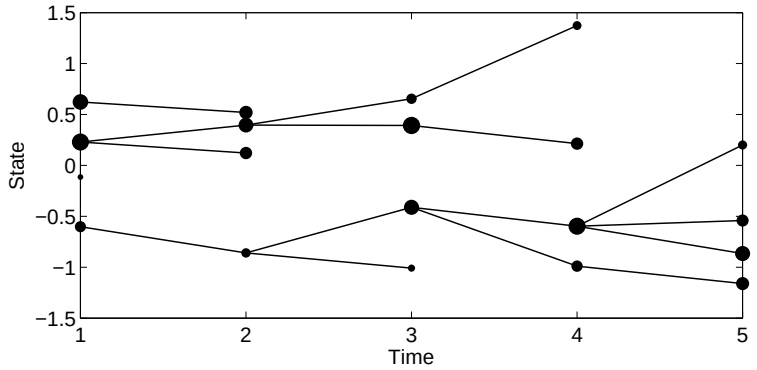
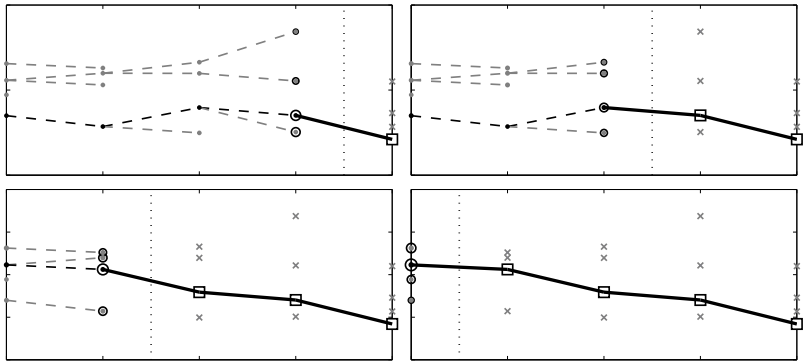


Illustration of a particle system $\{x_t^i, w_t^i\}_{i=1}^4$ generated using $N = 4$ particles during $T = 5$ time steps. The size of the particle visualize the corresponding weight.

Particle based BSi – example (II/II)



The backward smoothing weights $\widetilde{w}_{t|T}^i$ are visualized using circles.

Sampling from the backward kernel - in words

The BSi samples a trajectory from the empirical joint smoothing density by first choosing a particle x_T^m with probability proportional to w_T^m and then generating the **ancestral path** $b_{1:T}$ according to the backward kernel.

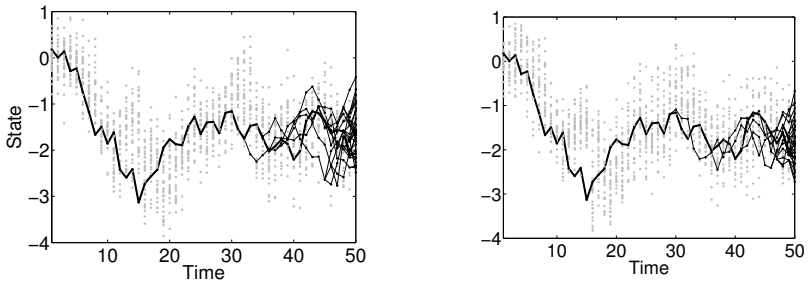
The resulting trajectory is

$$x_{1:T}^\star = \{x_1^{b_1}, \ldots, x_T^{b_T}\}.$$

The computational cost is $\mathcal{O}(N^2)$, which is too expensive to be practical.

Sampling from the backward kernel - in pictures

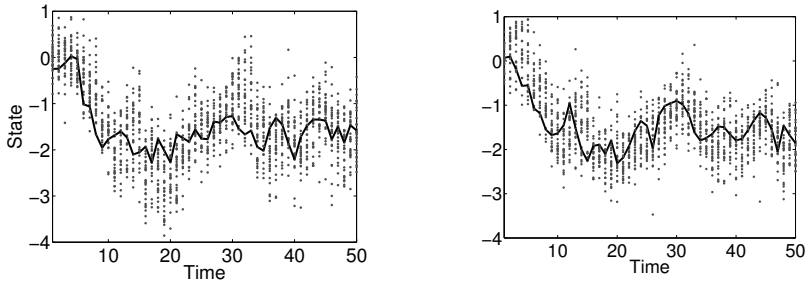
Recall that the **main problem** with using the PF to sample from $p(x_{1:T} \mid y_{1:T})$ is the particle degeneracy problem.



In the figures we show two samples from $p(x_{1:T} \mid y_{1:T})$ generated using the PF. The diversity is completely gone from time ≈ 30 and backwards.

Sampling from the backward kernel - in pictures

The backward simulator **resolves the particle degeneracy problem**.



Note that in the above samples there is no loss of diversity.

Towards a practical particle smoother

To obtain a practical algorithm we have to reduce the computational complexity, how?!

Key insight: We do not have to compute all the smoothing weights $\{\widetilde{w}_{t|T}^{i,j}\}$ to be able to sample from the empirical backward kernel.

How: Use rejection sampling to **sample the indices**! This means that we should not have to compute N indices just to draw one sample.

For details, see Chapter 8 and/or

Fredrik Lindsten and Thomas B. Schön. **Backward simulation methods for Monte Carlo statistical inference**. *Foundations and Trends in Machine Learning*, 6(1):1-143, 2013.

Recent work: BNP dynamical models

New result: We have been able to construct and learn a Gaussian process (GP) state space model

$$
\begin{aligned}
f(x_t) &\sim \mathcal{GP}(m_{\theta_x}(x_t), k_{\theta_x}(x_t, x_t')), \\
x_{t+1} \mid f_t &\sim \mathcal{N}(x_{t+1} \mid f_t, Q), \\
y_t \mid x_t &\sim p(y_t \mid x_t, \theta_y).
\end{aligned}
$$

Key idea: Marginalize out the function f.

Problem: Renders the model non-Markovian. **Solution:** PG-AS

Roger Frigola, Fredrik Lindsten, Thomas B. Schön and Carl E. Rasmussen, **Bayesian inference and learning in Gaussian process state-space models with particle MCMC**. In *Advances in Neural Information Processing Systems (NIPS) 26*, Lake Tahoe, NV, USA, December 2013.

Roger Frigola, Fredrik Lindsten, Thomas B. Schön and Carl E. Rasmussen. **Identification of Gaussian Process state-space models with particle stochastic approximation EM**. In *Proceedings of the 18th World Congress of the International Federation of Automatic Control (IFAC)*, Cape Town, South Africa, August 2014. (accepted for publication).

Recent work: Inference in graphical models

SMC samplers are used to approximate a sequence of probability distributions on a sequence of probability spaces.

Constructing an artificial sequence of intermediate target distributions for an SMC sampler is a powerful (and **quite possibly underutilized**) idea.

Key idea: Perform and make use of a **sequential decomposition** of the graphical model.

See talk at the Newton institute two weeks ago:

`http://www.newton.ac.uk/programmes/MCM/seminars/2014042510401.html`