Lecture 1 - Probabilistic dynamical models and strategies for state inference

UPPSALA UNIVERSITET

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

Division of Systems and Control
Department of Information Technology
Uppsala University

The aim of this module

The **aim of this module** is to provide an introduction to the theory and application of sequential Monte Carlo (SMC) methods for nonlinear state inference problems in dynamical systems.

The **SMC** methods we will focus on are,

- Particle filters,
- Particle smoothers (very briefly).

After this module you should be able to derive the particle filter (and other SMC) so that you can start implementing (and deriving) your own particle filter (and other SMC) algorithms to solve problems.

Model – data – inference algorithm

In solving problems we have to make assumptions and a **model** will to a large extent capture many of these assumptions.

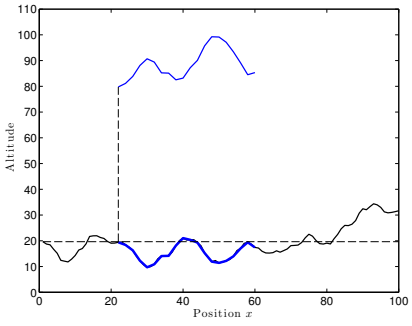
A **model** is a compact and interpretable representation of the data that is observed.

Using models to solve problems requires three key ingredients;

1. **Data:** Measurements from the system we are interested in.
2. **Model:** We use probabilistic models, allowing us to employ probability theory to represent and systematically work with the uncertainty that is inherent in most data.
3. **Inference algorithm:** The key topic of this course is the particle filter.

Particle filter – introductory example (I/III)

Consider a toy 1D localization problem.



Dynamic model:

$$x_{t+1} = x_t + u_t + v_t,$$

where x_t denotes position, u_t denotes velocity (known), $v_t \sim \mathcal{N}(0, 5)$ denotes an unknown disturbance.

Measurements:

$$y_t = h(x_t) + e_t.$$

where $h(\cdot)$ denotes the world model (here the terrain height) and $e_t \sim \mathcal{N}(0, 1)$ denotes an unknown disturbance.

Particle filter – introductory example (II/III)

Task: Find the state x_t based on a set of measurements $y_{1:t} \triangleq \{y_1, \ldots, y_t\}$. Do this by computing the filter PDF $p(x_t \mid y_{1:t})$.

The particle filter (PF) maintains an approximation according to

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

where each sample x_t^i is referred to as a **particle**.

For intuition: Think of each particle as one simulation of the system state (in this example the horizontal position) and only keep the good ones.

Particle filter – introductory example (III/III)

Highlights two **key capabilities** of the PF:

1. Automatically handles an unknown and dynamically changing number of hypotheses.
2. Work with nonlinear/non-Gaussian models.

Hinting at the potential – state estimation (I/II)

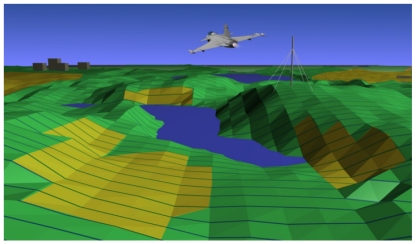
Fighter aircraft navigation using particle filters together with Saab.

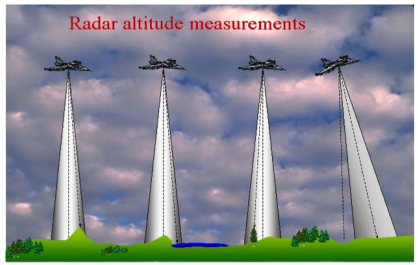
The task is to find the aircraft position using information from several sensors:

- Inertial sensors
- Radar
- Terrain elevation database

This **sensor fusion** problem requires a nonlinear state estimation problem to be solved, where we want to compute $p(x_t \mid y_{1:t})$.

Hinting at the potential – state estimation (II/II)





Key theory that allowed us to do this

- Particle filter (derived in this module)
- Rao-Blackwellized particle filter (combination of the Kalman filter and the particle filter)

Details of this particular example are provided in

Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. **Marginalized Particle Filters for Mixed Linear/Nonlinear State-Space Models**. *IEEE Transactions on Signal Processing*, 53(7):2279-2289, July 2005.

Important message!

Given the computational tools we have today it can be rewarding to resist the linear Gaussian convenience!!

Dynamical systems are everywhere

Outline of the module

L1 Probabilistic dynamical models and strategies for state inference
- a) Module infrastructure
- b) Probabilistic modeling of dynamical systems
- c) Strategies for state inference in nonlinear systems
- d) Importance sampling

L2 Solving inference problems using SMC
- a) Derive a first working particle filter (most common SMC sampler)
- b) Particle filtering examples
- c) Particle smoothers (very brief)
- d) General SMC
- e) Some recent work
- (f) Outlook and conclusion

Outline – Lecture 1

1. Module infrastructure
2. Probabilistic modeling of dynamical systems
   - a) Nonlinear state space model (SSM)
   - b) Linear Gaussian state space (LGSS) model
3. Strategies for state inference
   - a) Forward computations
   - b) Backward computations
4. Importance sampling

Module infrastructure

1. The white board will be used, i.e. the slides do not cover everything.
2. Slides and homework available from the course home page

   `www.mattiasvillani.com/teaching/bayeslearn2/`
3. The following sections of the manuscript are relevant for this module: 1.1, 2.1-2.2, 3.1-3.4, 4.1, **4.3**-4.4, **5.1-5.4**, 8.1.

Basics – probability

Let a and b be continuous random variables.

- Conditional probability:

$$p(a,b) = p(a \mid b)p(b)$$

- Marginalization (integrate out a variable)

$$p(a) = \int p(a,b)\mathrm{d}b$$

- Bayes' rule:

$$p(a \mid b) = \frac{p(b \mid a)p(a)}{p(b)}$$

The Markov property: $p(x_{t+1} \mid x_1, \ldots, x_t) = p(x_{t+1} \mid x_t)$.

SSM – an example

The stochastic volatility model is one possibility when it comes to modelling the variation in econometric and financial problems.

$$\begin{aligned} x_{t+1} &= \phi x_t + \sigma v_t, & v_t &\sim \mathcal{N}(0,1), \\ y_t &= \beta \exp(x_t/2) e_t, & e_t &\sim \mathcal{N}(0,1). \end{aligned}$$

We can also express this using PDFs only,

$$\begin{aligned} x_{t+1} \mid x_t &\sim f(x_{t+1} \mid x_t) = \mathcal{N}(x_t; \phi x_t, \sigma^2), \\ y_t \mid x_t &\sim g(y_t \mid x_t) = \mathcal{N}(y_t; 0, \beta^2 \exp(x_t)). \end{aligned}$$

See Johan Dahlin's licentiate thesis for more examples of nonlinear SSMs and new developments when it comes to SMC.

1. Representing an SSM using pdf's

Definition (State space model (SSM))

A state space model (SSM) consists of a Markov process $\{x_t\}_{t\geq 1}$ and a measurement process $\{y_t\}_{t\geq 1}$, related according to

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

where $x_t \in \mathbb{R}^{n_x}$ denotes the state, $u_t \in \mathbb{R}^{n_u}$ denotes a known deterministic input signal, $y_t \in \mathbb{R}^{n_y}$ denotes the observed measurement and $\theta \in \Theta \subseteq \mathbb{R}^{n_\theta}$ denotes any unknown (static) parameters.

2. Representing SSM using difference equations

In engineering literature, the SSM is often written in terms of a difference equation and an accompanying measurement equation,

$$\begin{aligned} x_{t+1} &= a_{\theta,t}(x_t, u_t) + v_{\theta,t}, \\ y_t &= c_{\theta,t}(x_t, u_t) + e_{\theta,t}, \end{aligned}$$

3. Representing SSM using a graphical model (I/II)

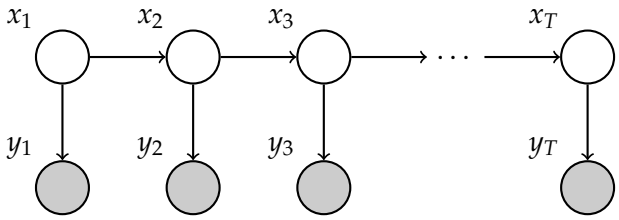


Figure: Graphical model for the SSM. Each stochastic variable is encoded using a node, where the nodes that are filled (gray) corresponds to variables that are observed and nodes that are not filled (white) are latent variables. The arrows pointing to a certain node encodes which variables the corresponding node are conditioned upon.

The SSM is an instance of a (directed) graphical model called **Bayesian network**, or **belief network**.

3. Representing SSM using a graphical model (II/II)

A Bayesian network directly describes how the joint distribution of all the involved variables (here $p(x_{1:T}, y_{1:T})$) is decomposed into a product of factors,

$$p(x_{1:T}, y_{1:T}) = \prod_{t=1}^{T} p(x_t \mid \mathrm{pa}(x_t)) \prod_{t=1}^{T} p(y_t \mid \mathrm{pa}(y_t)),$$

where $\mathrm{pa}(x_t)$ denotes the set of parents to x_t.

$$p(x_{t:T}, y_{1:T}) = \mu(x_1) \prod_{t=1}^{T-1} f_{\theta,t}(x_{t+1} \mid x_t) \prod_{t=1}^{T} g_{\theta,t}(y_t \mid x_t).$$

Graphical models offers a powerful framework for modeling, inference and learning,

Bishop, C. M. (2006). **Pattern Recognition and Machine Learning**. Springer.
Koller, D. and Friedman, N. (2009). **Probabilistic Graphical Models: Principles and Techniques**. MIT Press.

The LGSS model

Definition (Linear Gaussian State Space (LGSS) model)

The time invariant linear Gaussian state space (LGSS) model is defined by

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + v_t, \\ y_t &= Cx_t + Du_t + e_t, \end{aligned}$$

where $x_t \in \mathbb{R}^{n_x}$ denotes the state, $u_t \in \mathbb{R}^{n_u}$ denotes the known input signal and $y_t \in \mathbb{R}^{n_y}$ denotes the observed measurement. The initial state and the noise are distributed according to

$$\begin{pmatrix} x_1 \\ v_t \\ e_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} P_1 & 0 & 0 \\ 0 & Q & S \\ 0 & S^T & R \end{pmatrix} \right).$$

Notation for Gaussian (Normal) variables

The pdf of a Gaussian variable is denoted $\mathcal{N}(x \mid \mu, \Sigma)$, i.e.,

$$\mathcal{N}(x \mid \mu, \Sigma) \triangleq \frac{1}{(2\pi)^{n/2}\sqrt{\det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

See the appendix of the lecture notes for basic theorems needed in manipulating Gaussian variables.

State inference

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 will represent this information using **PDFs**.

Name	Probability density function
Filtering	$p(x_t \mid y_{1:t})$
Prediction	$p(x_{t+1} \mid y_{1:t})$
k-step prediction	$p(x_{t+k} \mid y_{1:t})$
Joint smoothing	$p(x_{1:T} \mid y_{1:T})$
Marginal smoothing	$p(x_t \mid y_{1:T})$, $t \leq T$
Fixed-lag smoothing	$p(x_{t-l+1:t} \mid y_{1:t})$, $l > 0$
Fixed-interval smoothing	$p(x_{r:t} \mid y_{1:T})$, $r < t \leq T$

Notation $y_{1:t} \triangleq \{y_1, y_2, \ldots, y_t\}$.

The nonlinear filtering problem

State filtering problem: Find x_t based on $\{u_{1:T}, y_{1:T}\}$ when the model is given by,

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

Strategy: Compute the filter PDF $p(x_t \mid y_{1:t})$.

Forward computations

Summarizing this development, we have **measurement update**

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

and **time update**

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

Backward computations

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

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

Forward and backward computations

Two key strategies are used:

1. Forward filtering and backward **smoothing** (FFBSm): Provides the smoothing density (exact or approximate).
2. Forward filtering and backward **simulation** (FFBSi): Provides samples distributed (exactly or approximately) according to the smoothing density.

Example – linear smoothing

Basic results of Gaussian variables provides the backward kernel

$$p(x_t \mid x_{t+1}, y_{1:t}) = \mathcal{N}\left(x_t \mid J_t x_{t+1} - J_t \widehat{x}_{t+1|t} + \widehat{x}_{t|t}, P_{t|t} - J_t A P_{t|t}\right),$$

where $J_t = P_{t|t} A^{\mathsf{T}} (A P_{t|t} A^{\mathsf{T}} + Q)^{-1}$. Using this together with $p(x_{t+1} \mid y_{1:T}) = \mathcal{N}(x_{t+1} \mid \widehat{x}_{t+1|T}, P_{t+1|T})$ results in

$$p(x_t \mid y_{1:T}) = \mathcal{N}\left(x_t \mid \widehat{x}_{t|T}, P_{t|T}\right),$$

where

$$\begin{aligned}
\widehat{x}_{t|T} &= \widehat{x}_{t|t} + J_t \left(\widehat{x}_{t+1|T} - \widehat{x}_{t+1|t}\right), \\
P_{t|T} &= P_{t|t} + J_t \left(P_{t+1|T} - P_{t+1|t}\right) J_t^{\mathsf{T}}, \\
J_t &= P_{t|t} A^{\mathsf{T}} (A P_{t|t} A^{\mathsf{T}} + Q)^{-1} = P_{t|t} A^{\mathsf{T}} P_{t+1|t}^{-1}.
\end{aligned}$$

Exact backward simulation in LGSS models

Algorithm 1 Backwards simulator (LGSS)

1. **Initialise:** Run the Kalman filter and store $\widehat{x}_{t|t}, P_{t|t}$ for $t = 1, \ldots, T$.
2. Draw $\widetilde{x}_T \sim \mathcal{N}(\widehat{x}_{T|T}, P_{T|T})$
3. **For** $j = 1$ **to** M **do:**
4. **For** $t = T-1$ **to** 1 **do:**

(a) Draw $\widetilde{x}_t \sim \mathcal{N}(\mu_t, L_t)$, where

$$\begin{aligned}
\mu_t &= \hat{x}_{t|t} + P_{t|t} A^{\mathsf{T}} (A P_{t|t} A^{\mathsf{T}} + Q)^{-1} (\widetilde{x}_{t+1}^j - A \hat{x}_{t|t}), \\
L_t &= P_{t|t} - P_{t|t} A^{\mathsf{T}} (A P_{t|t} A^{\mathsf{T}} + Q)^{-1} A P_{t|t}.
\end{aligned}$$

(b) Store the sample $\widetilde{x}_{t:T}^j = (\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j)$.

5. **End**
6. **End**

Exact backward simulation in LGSS models

Consider

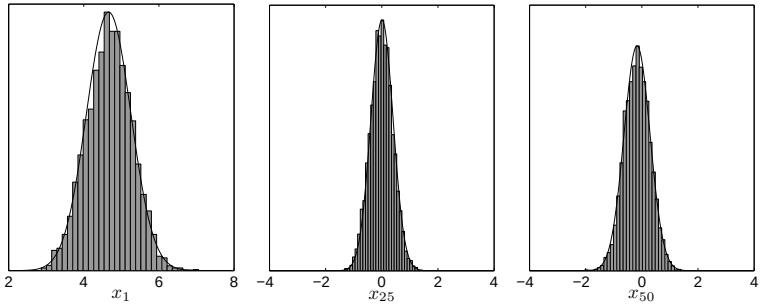
$$x_{t+1} = 0.9x_t + v_t, \qquad v_t \sim \mathcal{N}(0, 0.1),$$
$$y_t = x_t + e_t, \qquad e_t \sim \mathcal{N}(0, 1),$$

$x_1 \sim \mathcal{N}(0, 10)$ and generate $T = 50$ samples $y_{1:T}$.

Use FFBSi to sample 5 000 backwards trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$ from the JSD $p(x_{1:T} \mid y_{1:T})$.

Exact backward simulation in LGSS models

Histograms of $\{\widetilde{x}_t^j\}_{j=1}^M$ for $t = 1$, $t = 25$ and $t = 50$ (from left to right). The true marginal smoothing densities $p(x_t \mid y_{1:T})$ are shown using a solid black curve.



As expected they agree!

Why backward simulation?

Backward simulation is the **key strategy** underlying several of the best particle smoothers that are available today.

For a detailed introduction to backward simulation see:

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.

Outline – Lecture 1

1. Module infrastructure
2. Probabilistic modeling of dynamical systems
   a) Nonlinear state space model (SSM)
   b) Linear Gaussian state space (LGSS) model
3. Strategies for state inference
   a) Forward computations
   b) Backward computations
4. **Importance sampling**

Importance sampling – the problem

Importance sampling offers a solution to the problem of evaluating integrals of the form

$$I(\varphi) = \mathrm{E}\left[\varphi(z)\right] = \int \varphi(z)\pi(z)\mathrm{d}z,$$

when it is (potentially) hard to sample from the target density $\pi(z)$.

Importance sampling (IS)

Algorithm 2 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 $i = 1, \ldots, N$.

Importance sampling – alternative interpretation

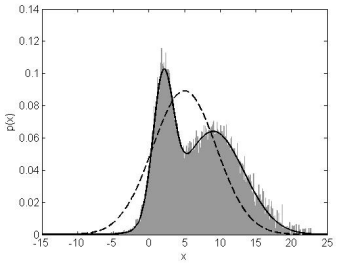
IS does not provide samples from the target density, but the samples $\{z^i\}_{i=1}^M$ together with the normalized weights $\{w^i\}_{i=1}^M$ provides an **empirical approximation** of the target density,

$$\widehat{\pi}(z) = \sum_{i=1}^M w^i \delta_{z^i}(z).$$

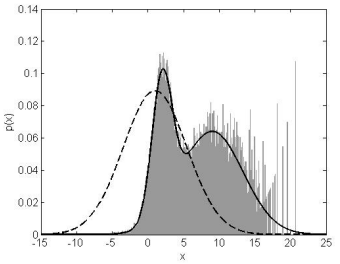
When this approximation is inserted into $I(\varphi(z)) = \int \varphi(z)\pi(z)\mathrm{d}z$ the resulting estimate is

$$\widehat{I}^M(\varphi(z)) = \sum_{i=1}^M w^i \varphi(z^i).$$

The importance of a good proposal density



$q_1(x) = \mathcal{N}(5, 20)$ (dashed curve)



$q_2(x) = \mathcal{N}(1, 20)$ (dashed curve)

50 000 samples used in both simulations.

Lesson learned: It is important to be careful in selecting the importance density.

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

$$\begin{aligned} p(x_t \mid y_{1:t}) &= \frac{g(y_t \mid x_t) p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}, \\ p(x_t \mid y_{1:t-1}) &= \int f(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) \mathrm{d}x_{t-1}. \end{aligned}$$

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