

Lecture 6: Particle Filtering

Simo Särkkä

February 13, 2019

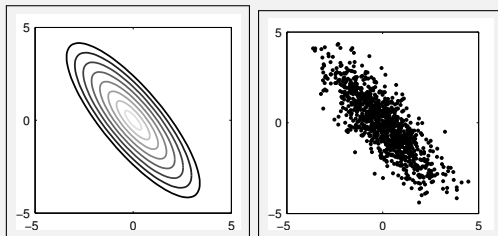
Learning Outcomes

- 1 Summary of the Last Lecture
- 2 Principle of Particle Filter
- 3 Monte Carlo Integration and Importance Sampling
- 4 Sequential Importance Sampling and Resampling
- 5 Particle Filter Properties
- 6 Summary and Demonstration

Summary of the Last Lecture

- **Unscented transform** (UT) approximates transformations of Gaussian variables by propagating **sigma points** through the non-linearity.
- In UT the **mean and covariance** are approximated as **linear combination** of the sigma points.
- The **unscented Kalman filter** uses unscented transform for computing the approximate means and covariance in non-linear filtering problems.
- **A non-linear transformation** can also be approximated with **Gaussian moment matching**.
- **Gaussian filter** is based on matching the moments with numerical integration \Rightarrow many kinds of Kalman filters.
- **Gauss-Hermite Kalman filter (GHKF)** and **Cubature Kalman filter (CKF)** are examples of them.

Particle Filtering: Principle



- Particle filter uses **Monte Carlo approximation** instead of **Gaussian approximation** of the filtering distribution.
- More specifically, particle filter uses **importance sampling** for propagating the Monte Carlo samples in time.
- Animation: Kalman vs. Particle Filtering:
 - ▶ [Kalman filter animation](#)
 - ▶ [Particle filter animation](#)

Particle Filtering: Principle (cont.)

- Mathematically, particle filter forms a **weighted sample (or particle) presentation** $(\mathbf{x}^{(i)}, w^{(i)})$ of the filtering distribution:

$$p(\mathbf{x}) \approx \sum_{i=1}^N w^{(i)} \delta(\mathbf{x} - \mathbf{x}^{(i)}).$$

- E.g., **mean and covariance** can then be approximated as

$$\mathbb{E}[\mathbf{x}] \approx \sum_{i=1}^N w^{(i)} \mathbf{x}^{(i)} = \mathbf{m}_{PF}$$

$$\text{Cov}[\mathbf{x}] \approx \sum_{i=1}^N w^{(i)} (\mathbf{x}^{(i)} - \mathbf{m}_{PF}) (\mathbf{x}^{(i)} - \mathbf{m}_{PF})^{\top}$$

- Other statistics** can be approximated analogously.
- Particle filter samples from the Bayesian filtering equations with **sequential importance sampling**.

Monte Carlo Integration

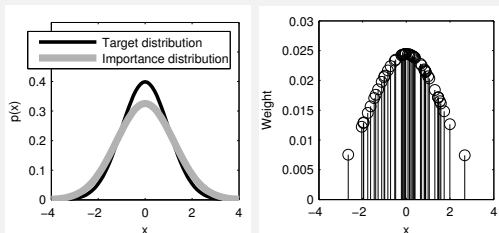
- In **Bayesian inference** we often want to compute posterior expectations of the form

$$\mathbb{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] = \int \mathbf{g}(\mathbf{x}) p(\mathbf{x} \mid \mathbf{y}_{1:T}) d\mathbf{x}$$

- For example, **posterior mean** and **posterior covariance** are such expectations.
- **Monte Carlo**: draw N independent random samples from $\mathbf{x}^{(i)} \sim p(\mathbf{x} \mid \mathbf{y}_{1:T})$ and estimate the expectation as

$$\mathbb{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \frac{1}{N} \sum_{i=1}^N \mathbf{g}(\mathbf{x}^{(i)}).$$

Importance Sampling: Basic Version [1/2]



- In practice, we rarely can directly draw samples from the distribution $p(\mathbf{x} \mid \mathbf{y}_{1:T})$.
- In **importance sampling (IS)**, we draw samples from an **importance distribution** $\mathbf{x}^{(i)} \sim \pi(\mathbf{x} \mid \mathbf{y}_{1:T})$ and compute weights $\tilde{w}^{(i)}$ such that

$$E[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \sum_{i=1}^N \tilde{w}^{(i)} \mathbf{g}(\mathbf{x}^{(i)})$$

Importance Sampling: Basic Version [2/2]

- Importance sampling is based on the identity

$$\begin{aligned} \mathbb{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] &= \int \mathbf{g}(\mathbf{x}) p(\mathbf{x} \mid \mathbf{y}_{1:T}) d\mathbf{x} \\ &= \int \left[\mathbf{g}(\mathbf{x}) \frac{p(\mathbf{x} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x} \mid \mathbf{y}_{1:T})} \right] \pi(\mathbf{x} \mid \mathbf{y}_{1:T}) d\mathbf{x} \end{aligned}$$

- Thus we can form a Monte Carlo approximation as follows:

$$\mathbb{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \frac{1}{N} \sum_{i=1}^N \frac{p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})} \mathbf{g}(\mathbf{x}^{(i)})$$

- That is, the importance weights can be defined as

$$\tilde{w}^{(i)} = \frac{1}{N} \frac{p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}$$

Importance Sampling: Weight Normalization

- The problem is that we need to evaluate the **normalization constant** of $p(\mathbf{x}^{(i)} | \mathbf{y}_{1:T})$ – often not possible.
- However, it turns out that we get a valid algorithm if we define **unnormalized importance weights** as

$$w^{*(i)} = \frac{p(\mathbf{y}_{1:T} | \mathbf{x}^{(i)}) p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)} | \mathbf{y}_{1:T})}$$

and then **normalize** them:

$$w^{(i)} = \frac{w^{*(i)}}{\sum_j w^{*(j)}}$$

- The (weight-normalized) importance sampling approximation is then

$$\mathbb{E}[\mathbf{g}(\mathbf{x}) | \mathbf{y}_{1:T}] \approx \sum_{i=1}^N w^{(i)} \mathbf{g}(\mathbf{x}^{(i)})$$

Importance Sampling

- Draw N samples from the **importance distribution**:

$$\mathbf{x}^{(i)} \sim \pi(\mathbf{x} \mid \mathbf{y}_{1:T}), \quad i = 1, \dots, N.$$

- Compute the **unnormalized weights** by

$$w^{*(i)} = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}^{(i)}) p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})},$$

and the **normalized weights** by

$$w^{(i)} = \frac{w^{*(i)}}{\sum_{j=1}^N w^{*(j)}}.$$

Importance Sampling: Properties

- The **approximation** to the **posterior expectation** of $\mathbf{g}(\mathbf{x})$ is

$$\mathbb{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \sum_{i=1}^N w^{(i)} \mathbf{g}(\mathbf{x}^{(i)}).$$

- The **posterior probability density approximation** can be formally written as

$$p(\mathbf{x} \mid \mathbf{y}_{1:T}) \approx \sum_{i=1}^N w^{(i)} \delta(\mathbf{x} - \mathbf{x}^{(i)}),$$

where $\delta(\cdot)$ is the Dirac delta function.

- The **efficiency** depends on the choice of the **importance distribution**.

Sequential Importance Sampling: Idea

- Sequential Importance Sampling (SIS) is concerned with models

$$\mathbf{x}_k \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$$

$$\mathbf{y}_k \sim p(\mathbf{y}_k \mid \mathbf{x}_k)$$

- The SIS algorithm uses a weighted set of **particles** $\{(w_k^{(i)}, \mathbf{x}_k^{(i)}) : i = 1, \dots, N\}$ such that

$$\mathbb{E}[\mathbf{g}(\mathbf{x}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{i=1}^N w_k^{(i)} \mathbf{g}(\mathbf{x}_k^{(i)}).$$

- Or equivalently

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^N w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}),$$

where $\delta(\cdot)$ is the Dirac delta function.

- Uses **importance sampling sequentially**.

Sequential Importance Sampling: Derivation [1/2]

- Let's consider the **full posterior** distribution of states $\mathbf{x}_{0:k}$ given the measurements $\mathbf{y}_{1:k}$.
- We get the following **recursion** for the **posterior distribution**:

$$\begin{aligned} p(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) &\propto p(\mathbf{y}_k | \mathbf{x}_{0:k}, \mathbf{y}_{1:k-1}) p(\mathbf{x}_{0:k} | \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k-1}) p(\mathbf{x}_{0:k-1} | \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{0:k-1} | \mathbf{y}_{1:k-1}). \end{aligned}$$

- We could now construct an **importance distribution** $\mathbf{x}_{0:k}^{(i)} \sim \pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k})$ and compute the corresponding (normalized) **importance weights** as

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)}) p(\mathbf{x}_{0:k-1}^{(i)} | \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k}^{(i)} | \mathbf{y}_{1:k})}.$$

Sequential Importance Sampling: Derivation [2/2]

- Let's form the **importance distribution recursively** as follows:

$$\pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) = \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \pi(\mathbf{x}_{0:k-1} | \mathbf{y}_{1:k-1})$$

- Expression for the **importance weights** can be written as

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \underbrace{\frac{p(\mathbf{x}_{0:k-1}^{(i)} | \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k-1}^{(i)} | \mathbf{y}_{1:k-1})}}_{\propto w_{k-1}^{(i)}}$$

- Thus the weights satisfy the **recursion**

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} w_{k-1}^{(i)}$$

Sequential Importance Sampling: Algorithm

Sequential Importance Sampling

- **Initialization:** Draw N samples $\mathbf{x}_0^{(i)}$ from the prior

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$$

and set $w_0^{(i)} = 1/N$.

- **Prediction:** Draw N new samples $\mathbf{x}_k^{(i)}$ from importance distributions

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})$$

- **Update:** Calculate new weights according to

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} | \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})}$$

Sequential Importance Sampling: Degeneracy

- The problem in SIS is that the algorithm is **degenerate**
- It can be shown that the **variance** of the **weights increases at every step**
- It means that we will always converge to **single non-zero weight** $w^{(i)} = 1$ and the rest being zero – not very useful algorithm.
- **Solution:** resampling!

Sequential Importance Resampling: Resampling Step

- Sequential Importance Resampling (SIR) algorithm adds the following resampling step to SIS algorithm:

Resampling

- Interpret each weight $w_k^{(i)}$ as the probability of obtaining the sample index i in the set $\{\mathbf{x}_k^{(i)} \mid i = 1, \dots, N\}$.
 - Draw N samples from that discrete distribution and replace the old sample set with this new one.
 - Set all weights to the constant value $w_k^{(i)} = 1/N$.
- There are many algorithms for implementing this – **stratified resampling** is optimal in terms of variance.

Sequential Importance Resampling: Effective Number of Particles

- A simple way to do resampling is at every step – but every **resampling** operation **increases variance**.
- We can also resample at, say, **every K th step**.
- In **adaptive resampling**, we resample when the effective number of samples is too low (say, $N/10$):

$$n_{\text{eff}} \approx \frac{1}{\sum_{i=1}^N \left(w_k^{(i)}\right)^2},$$

Sequential Importance Resampling

- Draw samples $\mathbf{x}_k^{(i)}$ from the importance distribution:

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}), \quad i = 1, \dots, N.$$

- Calculate new weights

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})}, \quad i = 1, \dots, N,$$

and normalize them to sum to unity.

- If the effective number of particles is too low, perform resampling.

Sequential Importance Resampling: Bootstrap filter

- In **bootstrap filter** we use the **dynamic model** as the importance distribution

$$\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}) = p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})$$

and resample at every step:

Bootstrap Filter

- Draw samples $\mathbf{x}_k^{(i)}$ from the dynamic model:

$$\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}), \quad i = 1, \dots, N.$$

- Calculate new weights

$$w_k^{(i)} \propto p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}), \quad i = 1, \dots, N,$$

and normalize them to sum to unity.

- Perform resampling.

Sequential Importance Resampling: Optimal Importance Distribution

- The **optimal importance** distribution is

$$\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}) = p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)$$

- Then the weight update reduces to

$$w_k^{(i)} \propto w_{k-1}^{(i)} p(\mathbf{y}_k \mid \mathbf{x}_{k-1}^{(i)}), \quad i = 1, \dots, N.$$

- The optimal importance distribution can be used, for example, when the state space is finite.

Sequential Importance Resampling: Importance Distribution via Kalman Filtering

- We can also form a **Gaussian approximation** to the optimal importance distribution:

$$p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k) \approx N(\mathbf{x}_k^{(i)} \mid \tilde{\mathbf{m}}_k^{(i)}, \tilde{\mathbf{P}}_k^{(i)}).$$

by using a single prediction and update steps of a Gaussian filter starting from a singular distribution at $\mathbf{x}_{k-1}^{(i)}$.

- We can also replace above with the result of a Gaussian filter $N(\mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)})$ started from a random initial mean.
- A very common way seems to be to use the previous sample as the mean: $N(\mathbf{x}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)})$.
- A particle filter with UKF proposal has been given name **unscented particle filter (UPF)** – you can invent new PFs easily this way.

Particle Filter: Advantages

- No restrictions in model – can be applied to **non-Gaussian models, hierarchical models etc.**
- **Global** approximation.
- Approaches the **exact solution**, when the number of samples goes to infinity.
- In its **basic** form, very **easy to implement**.

Particle Filter: Disadvantages

- Computational requirements much higher than of the Kalman filters.
- Problems with nearly noise-free models, especially with accurate dynamic models.
- Good importance distributions quite tricky to implement.
- Very hard to find programming errors (i.e., to debug).

- **Particle filters** can be used for approximate filtering in general probabilistic state-space models.
- **Particle filters** use **weighted set of samples** (particles) for approximating the filtering distributions.
- **Sequential importance resampling (SIR)** is the general framework and **bootstrap filter** is a simple special case of it.
- **EKF, UKF and other Gaussian filters** can be used for forming good **importance distributions**.
- The **optimal importance distribution** is the minimum variance importance distribution.

- The discretized pendulum model:

$$\begin{pmatrix} x_k^1 \\ x_k^2 \end{pmatrix} = \underbrace{\begin{pmatrix} x_{k-1}^1 + x_{k-1}^2 \Delta t \\ x_{k-1}^2 - g \sin(x_{k-1}^1) \Delta t \end{pmatrix}}_{\mathbf{f}(\mathbf{x}_{k-1})} + \begin{pmatrix} 0 \\ q_{k-1} \end{pmatrix}$$
$$y_k = \underbrace{\sin(x_k^1)}_{\mathbf{h}(\mathbf{x}_k)} + r_k,$$

- \Rightarrow *Matlab demonstration*