Aalto University
School of Electrical
Engineering

ELEC-E8740 — Bootstrap Particle Filtering

Simo Särkkä

Aalto University

November 27, 2020

Contents

1. Intended Learning Outcomes and Recap
2. Idea of Particle Filter
3. Resampling and Particle Filter Algorithm
4. Summary

Intended Learning Outcomes

After this lecture, you will be able to:

- describe the basic idea of particle filtering,
- explain the three steps in particle filtering: simulation, weighting, resampling,
- identify the differences between Kalman filtering and particle filtering.

Recap: Extended Kalman Filter

- Model approximation:

$$\mathbf{x}_n = \mathbf{f}(\mathbf{x}_{n-1}) + \mathbf{q}_n \approx \mathbf{f}(\hat{\mathbf{x}}_{n-1|n-1}) + \mathbf{F}_x(\mathbf{x}_{n-1} - \hat{\mathbf{x}}_{n-1|n-1}) + \mathbf{q}_n$$
$$\mathbf{y}_n = \mathbf{g}(\mathbf{x}_n) + \mathbf{r}_n \approx \mathbf{g}(\hat{\mathbf{x}}_{n|n-1}) + \mathbf{G}_x(\mathbf{x}_n - \hat{\mathbf{x}}_{n|n-1}) + \mathbf{r}_n$$

- Prediction:

$$\begin{aligned}
\hat{\mathbf{x}}_{n|n-1} &= \mathbf{f}(\hat{\mathbf{x}}_{n-1|n-1}), \\
\mathbf{P}_{n|n-1} &= \mathbf{F}_x \mathbf{P}_{n-1|n-1} \mathbf{F}_x^\mathsf{T} + \mathbf{Q}_n,
\end{aligned}$$

- Measurement update:

$$\begin{aligned}
\mathbf{K}_n &= \mathbf{P}_{n|n-1} \mathbf{G}_x^\mathsf{T} (\mathbf{G}_x \mathbf{P}_{n|n-1} \mathbf{G}_x^\mathsf{T} + \mathbf{R}_n)^{-1}, \\
\hat{\mathbf{x}}_{n|n} &= \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n(\mathbf{y}_n - \mathbf{g}(\hat{\mathbf{x}}_{n|n-1})), \\
\mathbf{P}_{n|n} &= \mathbf{P}_{n|n-1} - \mathbf{K}_n(\mathbf{G}_x \mathbf{P}_{n|n-1} \mathbf{G}_x^\mathsf{T} + \mathbf{R}_n)\mathbf{K}_n^\mathsf{T}.
\end{aligned}$$

Recap: Unscented Kalman Filter

- Uses a nonlinear transformation of deterministic sampling points
- Prediction:
  - Calculate the sigma-points using $\hat{\mathbf{x}}_{n-1|n-1}$ and $\mathbf{P}_{n-1|n-1}$
  - Propagate the sigma-points $\mathbf{x}_n^j = \mathbf{f}(\mathbf{x}_{n-1}^j)$
  - Calculate the mean and covariance $\hat{\mathbf{x}}_{n|n-1}$, $\mathbf{P}_{n|n-1}$
- Measurement update:
  - Calculate the sigma-points using $\hat{\mathbf{x}}_{n|n-1}$ and $\mathbf{P}_{n|n-1}$
  - Propagate the sigma-points $\mathbf{y}_n^j = \mathbf{g}(\mathbf{x}_n^j)$
  - Calculate the mean and covariance $\mathrm{E}\{\mathbf{y}_n \mid \mathbf{y}_{1:n-1}\}$, $\mathrm{Cov}\{\mathbf{y}_n \mid \mathbf{y}_{1:n-1}\}$, $\mathrm{Cov}\{\mathbf{x}_n, \mathbf{y}_n \mid \mathbf{y}_{1:n-1}\}$
  - Perform the Kalman filter measurement update:

$$\begin{aligned}
\mathbf{K}_n &= \mathrm{Cov}\{\mathbf{x}_n, \mathbf{y}_n \mid \mathbf{y}_{1:n-1}\}\,\mathrm{Cov}\{\mathbf{y}_n \mid \mathbf{y}_{1:n-1}\}^{-1}, \\
\hat{\mathbf{x}}_{n|n} &= \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n(\mathbf{y}_n - \mathrm{E}\{\mathbf{y}_n \mid \mathbf{y}_{1:n-1}\}), \\
\mathbf{P}_{n|n} &= \mathbf{P}_{n|n-1} - \mathbf{K}_n\,\mathrm{Cov}\{\mathbf{y}_n \mid \mathbf{y}_{1:n-1}\}\mathbf{K}_n^{\mathsf{T}}.
\end{aligned}$$

Discrete-Time Nonlinear State-Space Model

- Discrete-time nonlinear state-space model:

$$\mathbf{x}_n = \mathbf{f}(\mathbf{x}_{n-1}) + \mathbf{q}_n$$
$$\mathbf{y}_n = \mathbf{g}(\mathbf{x}_n) + \mathbf{r}_n$$

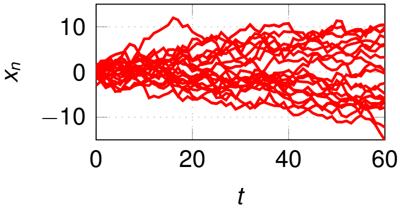
- Process noise: $\mathbf{q}_n \sim p(\mathbf{q}_n)$
- Measurement noise: $\mathbf{r}_n \sim p(\mathbf{r}_n)$
- Initial state: $\mathbf{x}_0 \sim p(\mathbf{x}_0)$

This is a stochastic process, each realization of the state sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ is different

Example: Random Walk Process (1/2)

- Dynamic model:

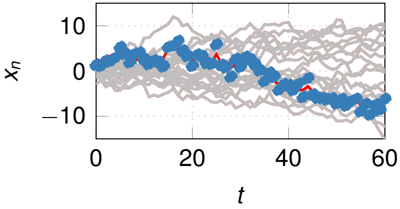
$$
\begin{aligned}
x_n &= x_{n-1} + q_n \\
x_0 &\sim \mathcal{N}(0, 1) \\
q_n &\sim \mathcal{N}(0, 1)
\end{aligned}
$$



Example: Random Walk Process (2/2)

- Only one realization of the process is observed
- Measurement model:

$$y_n = x_n + r_n$$
$$r_n \sim \mathcal{N}(0, 1)$$



Particle Filtering: Idea

Prediction

- Given: Simulated states $\mathbf{x}_{n-1}^j$ ($j = 1, \ldots, J$)
- Simulate from t_{n-1} to t_n to obtain $\mathbf{x}_n^j$ ($j = 1, \ldots, J$)

Measurement Update

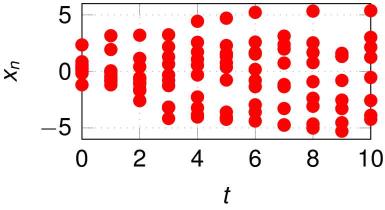
- Evaluate how well $\mathbf{x}_n^j$ explains $\mathbf{y}_n$ ($j = 1, \ldots, J$)
- Assign a weight w_n^j to $\mathbf{x}_n^j$ ($j = 1, \ldots, J$)

Prediction: Simulation

- Intuitive way: Use the dynamic model to simulate one time step

$$\mathbf{x}_n = \mathbf{f}(\mathbf{x}_{n-1}) + \mathbf{q}_n$$

- Two step procedure:
  1. Sample $\mathbf{q}_n^j \sim p(\mathbf{q}_n)$,
  2. Calculate $\mathbf{x}_n^j = \mathbf{f}(\mathbf{x}_{n-1}^j) + \mathbf{q}_n^j$.



Measurement Update: Importance Weights

- Weights w_n^j indicate the relevance of each sample
- Importance weights:
  - High weight w_n^j: Explains $\mathbf{y}_n$ well
  - Low weight w_n^j: Explains $\mathbf{y}_n$ poorly
  - Should sum to one:

$$\sum_{j=1}^{J} w_n^j = 1,$$

- Cost function gives low values for good estimates of $\mathbf{x}_n$

Measurement Update: Likelihood (1/2)

- Measurement model:

$$
\begin{aligned}
\mathbf{y}_n &= \mathbf{g}(\mathbf{x}_n) + \mathbf{r}_n \\
\mathbf{r}_n &\sim p(\mathbf{r}_n)
\end{aligned}
$$

- $\mathbf{r}_n$ is a random variable $\Rightarrow$ $\mathbf{y}_n$ is a random variable too
- $\mathbf{y}_n$ must have a probability density function (pdf)
- Given $\mathbf{x}_n$, the pdf for $\mathbf{y}_n$ is the same as for $\mathbf{r}_n$ but shifted by $\mathbf{g}(\mathbf{x}_n)$
- The pdf for $\mathbf{y}_n$ given $\mathbf{x}_n$ is called the likelihood

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

Measurement Update: Likelihood (2/2)

- The likelihood is a suitable measure for the importance weights w_n^j
- The non-normalized weights are then:

$$\tilde{w}_n^j = p(\mathbf{y}_n \mid \mathbf{x}_n^j).$$

- Normalization:

$$w_n^j = \frac{\tilde{w}_n^j}{\sum_{i=1}^{J} \tilde{w}_n^i}.$$

Example: Gaussian Likelihood (1/2)

- Measurement model:

$$\mathbf{y}_n = \mathbf{g}(\mathbf{x}_n) + \mathbf{r}_n$$

- The measurement noise is often (assumed) Gaussian:

$$p(\mathbf{r}_n) = \mathcal{N}(\mathbf{r}_n; 0, \mathbf{R}_n)$$

- Then, the likelihood is Gaussian too:

$$p(\mathbf{y}_n \mid \mathbf{x}_n) = \mathcal{N}(\mathbf{y}_n; \mathbf{g}(\mathbf{x}_n), \mathbf{R}_n).$$

Example: Gaussian Likelihood (2/2)

- Example: Scalar case with

$$y_n = \mathbf{g}(x_n) + r_n$$
$$r_n \sim \mathcal{N}(0, \sigma_r^2)$$

Point Estimates

- Moments of the state can be calculated using weighted sums of the weighted samples
- Mean:

$$\hat{\mathbf{x}}_{n|n} = \sum_{j=1}^{J} w_n^j \mathbf{x}_n^j$$

- Covariance:

$$\mathbf{P}_{n|n} = \sum_{j=1}^{J} w_n^j (\mathbf{x}_n^j - \hat{\mathbf{x}}_{n|n})(\mathbf{x}_n^j - \hat{\mathbf{x}}_{n|n})^\mathsf{T}.$$

Summary: Sequential Sampling and Weighing

- Initialization: Sample J particles:

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

- Prediction: Sample $\mathbf{q}_n^j$ and propagate particles:

$$\mathbf{q}_n^j \sim p(\mathbf{q}_n)$$
$$\mathbf{x}_n^j = \mathbf{f}(\mathbf{x}_n^j) + \mathbf{q}_n^j$$

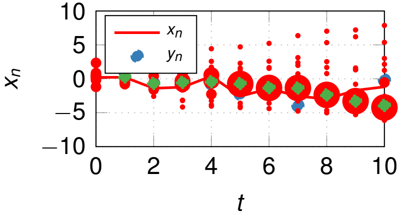
- Measurement update: Calculate and normalize the particle weights:

$$\tilde{w}_n^j = p(\mathbf{y}_n \mid \mathbf{x}_n^j)$$
$$w_n^j = \frac{w_n^j}{\sum_{i=1}^J \tilde{w}_n^i}$$

Example: Random Walk Process

- State-space model:

$$x_n = x_{n-1} + q_n$$
$$y_n = x_n + r_n$$



Resampling (1/2)

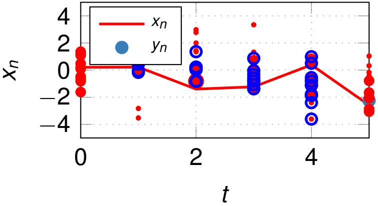
- Problem: The particles diverge after a few samples
- Resampling:
  - Remove samples with low weights
  - Replicate samples with high weights
  - Samples should be represented proportional to their weight:

$$\lfloor w_n^j J \rceil$$

- Equivalent interpretation

$$\Pr\{\tilde{\mathbf{x}}_n^i = \mathbf{x}_n^j\} = w_n^j,$$

Resampling (2/2)



Bootstrap Particle Filter

Algorithm 1 Bootstrap Particle Filter (Gaussian Noises)

Initialize: $\mathbf{x}_0^j \sim \mathcal{N}(\mathbf{m}_0, \mathbf{P}_0)$ $(j = 1, \ldots, J)$
for $n = 1, 2, \ldots$ **do**
  for $j = 1, 2, \ldots, J$ **do**
    Sample: $\mathbf{q}_n^j \sim \mathcal{N}(0, \mathbf{Q})$
    Propagate the state: $\mathbf{x}_n^j = \mathbf{f}(\mathbf{x}_{n-1}^j) + \mathbf{q}_n^j$
    Calculate the weights: $\tilde{w}_n^j = \mathcal{N}(\mathbf{y}_n; \mathbf{g}(\mathbf{x}_n^j), \mathbf{R}_n)$
  end for
  Normalize the importance weights $(j = 1, \ldots, J)$

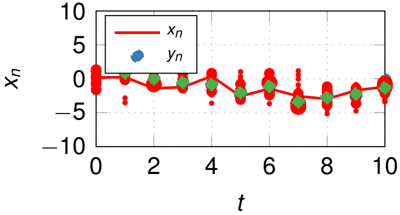
$$w_n^j = \frac{\tilde{w}_n^j}{\sum_{i=1}^J \tilde{w}_n^i}$$

  Calculate the mean $\hat{\mathbf{x}}_{n|n}$ and covariance $\mathbf{P}_{n|n}$
  Resample such that $\Pr\{\tilde{\mathbf{x}}_n^i = \mathbf{x}_n^j\} = w_n^j$
end for

Example: Random Walk

- State-space model:

$$x_n = x_{n-1} + q_n$$
$$y_n = x_n + r_n$$



Example: Object Tracking (1/3)

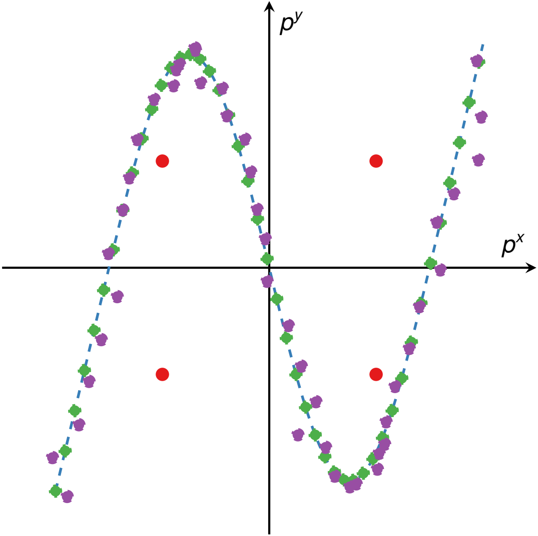
- Quasi-constant turn model:

$$\begin{bmatrix} \dot{p}^x(t) \\ \dot{p}^y(t) \\ \dot{v}(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\varphi(t)) \\ v(t)\sin(\varphi(t)) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{w}(t)$$

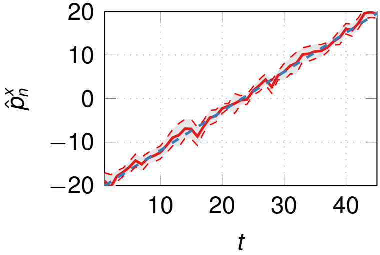
- Range (distance) measurements:

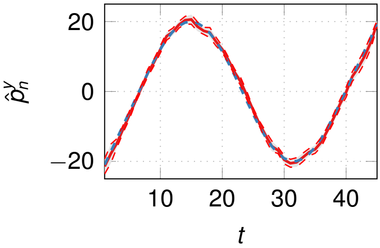
$$\mathbf{y}_n = \begin{bmatrix} |\mathbf{p}_n - \mathbf{p}_1^s| \\ |\mathbf{p}_n - \mathbf{p}_2^s| \\ \vdots \\ |\mathbf{p}_n - \mathbf{p}_K^s| \end{bmatrix} + \mathbf{r}_n$$

Example: Object Tracking (2/3)



Example: Object Tracking (3/3)





Summary

- The particle filter uses a set of random samples to estimate the state
- During prediction, the samples are simulated from t_{n-1} to t_n
- The bootstrap particle filter uses the dynamic model to simulate the samples
- The measurement update evaluates the likelihood to assign an importance weight to each sample
- Resampling is used to mitigate particle degeneracy
- Particle filtering is a universal approach equally applicable to linear and nonlinear system
- It can be shown that particle filters are asymptotically ($J \to \infty$) optimal in many cases