

NA 568 - Winter 2022

Particle Filtering

Maani Ghaffari

January 18, 2022



Nonlinear Dynamic Systems Excited by Noise

- Nonlinear process model:

$$x_k = f(u_k, x_{k-1}, w_k)$$

- Nonlinear measurement model:

$$z_k = h(x_k, v_k)$$



Q. How to map belief (a probability distribution) through a nonlinear function?

Key ideas:

- ▶ Linearization via Taylor expansion
→ Extended Kalman Filter (EKF)
- ▶ Unscented Transform (deterministic sampling)
→ Unscented Kalman Filter (UKF)
- ▶ **Monte Carlo methods (random sampling)**
→ **Sequential Monte Carlo methods (Particle Filters)**

Sequential Monte Carlo methods

Sequential Monte Carlo (SMC) methods are a set of simulation-based methods for computing posterior distributions.

- ▶ Observations arrive sequentially in time and we wish to perform online inference;
- ▶ The posterior distribution is updated as data become available (recursive Bayesian estimation/learning);
- ▶ SMC methods are used when dealing with non-Gaussian, high-dimensionality, and nonlinearity where often obtaining an analytical solution is not possible.

Remark

SMC methods can be used for inferring both filtering and smoothing posterior distributions.

The Dirac delta function, for $x \in \mathbb{R}$, is defined by the properties

$$\delta(x) = \begin{cases} \infty & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x) dx = 1$$

For any smooth function f and $a \in \mathbb{R}$, we have:

$$\int_{-\infty}^{\infty} \delta(x - a) f(x) dx = f(a)$$

which is a Lebesgue integral with respect to the measure δ (thought as a point mass). This can be generalized to \mathbb{R}^n or any set with the similar idea to define δ measure or unit mass concentrated at a point.

Suppose we can simulate n independent and identically distributed (i.i.d.) random samples (particles), $\{x_{0:k}^i\}_{i=1}^n$ according to $p(x_{0:k}|z_{1:k})$. An empirical estimate of this distribution is given by

$$p_n(x_{0:k} - x_{0:k}^{1:n}|z_{1:k}) = \frac{1}{n} \sum_{i=1}^n \delta(x_{0:k} - x_{0:k}^i)$$

Suppose we can simulate n independent and identically distributed (i.i.d.) random samples (particles), $\{x_{0:k}^i\}_{i=1}^n$ according to $p(x_{0:k}|z_{1:k})$. An empirical estimate of this distribution is given by

$$p_n(x_{0:k} - x_{0:k}^{1:n}|z_{1:k}) = \frac{1}{n} \sum_{i=1}^n \delta(x_{0:k} - x_{0:k}^i)$$

Then, the following integral can be computed:

$$I_n(f) = \int f(x_{0:k}) p_n(x_{0:k} - x_{0:k}^{1:n}|z_{1:k}) = \frac{1}{n} \sum_{i=1}^n f(x_{0:k}^i)$$

- ▶ This estimate is unbiased (why?);
- ▶ if the posterior variance is bounded, i.e.,
 $\sigma_f^2 := \mathbb{E}_{p(x_{0:k}|z_{1:k})}[f^2(x_{0:k})] - I^2(f) < \infty$;
- ▶ then $\mathbb{V}[I_n(f)] = \frac{\sigma_f^2}{n}$ (sample variance);
- ▶ and from the law of large numbers, $n \rightarrow \infty$, almost surely,
 $I_n(f) \rightarrow I(f)$;
- ▶ moreover, if $\sigma_f^2 < \infty$, then a central limit theorem holds; that is $n \rightarrow \infty$, $\sqrt{n}(I_n(f) - I(f))$ converges in distribution to $\mathcal{N}(0, \sigma_f^2)$.

- ▶ We can easily estimate any quantity $I(f)$;
- ▶ the rate of convergence is independent of the integrand dimension;
- ▶ any deterministic numerical integration method has a rate of convergence that decreases as the dimension of the integrand increases;

- ▶ We can easily estimate any quantity $I(f)$;
- ▶ the rate of convergence is independent of the integrand dimension;
- ▶ any deterministic numerical integration method has a rate of convergence that decreases as the dimension of the integrand increases;
- ▶ in practice, it is usually impossible to sample efficiently from the posterior distribution $p(x_{0:k}|z_{1:k})$:(

- ▶ We introduce an *importance sampling distribution* (also called *proposal distribution*), $\pi(x_{0:k}|z_{1:k})$;
- ▶ we also assume the support of $\pi(x_{0:k}|z_{1:k})$ includes the support of $p(x_{0:k}|z_{1:k})$; we get:

- ▶ We introduce an *importance sampling distribution* (also called *proposal distribution*), $\pi(x_{0:k}|z_{1:k})$;
- ▶ we also assume the support of $\pi(x_{0:k}|z_{1:k})$ includes the support of $p(x_{0:k}|z_{1:k})$; we get:

$$\begin{aligned} I(f) &= \frac{\int f(x_{0:k})w(x_{0:k})\pi(x_{0:k}|z_{1:k})dx_{0:k}}{\int w(x_{0:k})\pi(x_{0:k}|z_{1:k})dx_{0:k}} \\ &= \frac{\int f(x_{0:k})p(x_{0:k}|z_{1:k})dx_{0:k}}{\int p(x_{0:k}|z_{1:k})dx_{0:k}} = \int f(x_{0:k})p(x_{0:k}|z_{1:k})dx_{0:k} \end{aligned}$$

where $w(x_{0:k})$ is known importance weight:

$$w(x_{0:k}) = \frac{p(x_{0:k}|z_{1:k})}{\pi(x_{0:k}|z_{1:k})}$$

Drawing n i.i.d. particles according to $\pi(x_{0:k}|z_{1:k})$:

$$\hat{I}_n(f) = \frac{\frac{1}{n} \sum_{i=1}^n f(x_{0:k}^i) w(x_{0:k}^i)}{\frac{1}{n} \sum_{i=1}^n w(x_{0:k}^i)} = \sum_{i=1}^n f(x_{0:k}^i) \tilde{w}_k^i$$

where the *normalized importance weights* are given by

$$\tilde{w}_k^i = \frac{w(x_{0:k}^i)}{\sum_{i=1}^n w(x_{0:k}^i)}$$

but this is not adequate for recursive estimation!

Sequential Importance Sampling (SIS)

Let us factor the importance sampling distribution as follows:

$$\begin{aligned}\pi(x_{0:k}|z_{1:k}) &= \pi(x_{0:k-1}|z_{1:k-1})\pi(x_k|x_{0:k-1}, z_{1:k}) \\ &= \pi(x_0) \prod_{j=1}^k \pi(x_j|x_{0:j-1}, z_{1:j})\end{aligned}$$

Sequential Importance Sampling (SIS)

Let us factor the importance sampling distribution as follows:

$$\begin{aligned}\pi(x_{0:k}|z_{1:k}) &= \pi(x_{0:k-1}|z_{1:k-1})\pi(x_k|x_{0:k-1}, z_{1:k}) \\ &= \pi(x_0) \prod_{j=1}^k \pi(x_j|x_{0:j-1}, z_{1:j})\end{aligned}$$

then:

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{0:k-1}^i, z_{1:k})} \quad (w(x_{0:k}) = \frac{p(x_{0:k}|z_{1:k})}{\pi(x_{0:k}|z_{1:k})})$$

$$\text{Recall: } p(x_{0:k}^i|z_{1:k}) = p(x_{0:k-1}^i|z_{1:k-1}) \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{p(z_k|z_{1:k-1})}$$

Sequential Importance Sampling (SIS)

Remark

In filtering, we often set $\pi(x_k | x_{0:k-1}, z_{1:k}) = \pi(x_k | x_{k-1}, z_k)$ so that the importance sampling distribution only depends on x_{k-1} and z_k .

Remark

A **special case** is when the importance sampling distribution is chosen to be the prior distribution, or in robotics the robot motion model $p(x_k|x_{k-1}, u_k)$. Note that u_k is deterministic and the motion model does not depend on the measurement z_k .

Then the weights can be computed using:

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i p(z_k|x_k^i)$$

Algorithm 1 sis-particle-filter

Require: particles $\mathcal{X}_{k-1} = \{x_{k-1}^i, w_{k-1}^i\}_{i=1}^n$, measurement z_k ;

1: $\mathcal{X}_k \leftarrow \emptyset$

2: **for** each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**

3: $x_k^i \sim \pi(x_k^i | x_{k-1}^i, z_k)$

▷ sample from proposal distribution

4: $w_k^i \leftarrow w_{k-1}^i \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i)}{\pi(x_k^i | x_{k-1}^i, z_k)}$

▷ update importance weights

5: $\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i\}$

▷ add i -th weighted sample to the new set

6: **return** \mathcal{X}_k

- ▶ As time increases, the distribution of the of the weights, \tilde{w}_k^i becomes more and more skewed, in practice, reducing to one particle with non-zero weight after a few iterations;
- ▶ the *resampling* idea was introduced to fix this problem.

- ▶ Measure of degeneracy using the effective sample size:

$$n_{\text{eff}} = \frac{1}{\sum_{i=1}^n (\tilde{w}_k^i)^2} \quad 1 < n_{\text{eff}} < n$$

- ▶ two extreme cases:

1 all particles have the same weights (uniform): $\forall i \in \{1 : n\}$,

$$\tilde{w}_k^i = \frac{1}{n} \implies n_{\text{eff}} = n;$$

2 the entire distribution mass is placed in one particle (singular):

$$\forall i \in \{1 : j-1, j+1 : n\}, \tilde{w}_k^i = 0 \text{ and } \tilde{w}_k^j = 1 \implies n_{\text{eff}} = 1;$$

Although resampling step reduces the effect of degeneracy, it introduces a new problem known as *sample impoverishment*.

- ▶ It limits the parallel implementation of the algorithm since all particles must be combined;
- ▶ the particles with high weights are selected many times, leading to the loss of diversity.

Generic Particle Filter Algorithm

Algorithm 2 generic-particle-filter

Require: particles $\mathcal{X}_{k-1} = \{x_{k-1}^i, \tilde{w}_{k-1}^i\}_{i=1}^n$, measurement z_k , resampling threshold n_t (e.g. $n/3$);

- 1: $\mathcal{X}_k \leftarrow \emptyset$
 - 2: **for** each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**
 - 3: draw $x_k^i \sim \pi(x_k^i | x_{k-1}^i, z_k)$ ▷ sample from proposal distribution
 - 4: $w_k^i \leftarrow \tilde{w}^i \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i)}{\pi(x_k^i | x_{k-1}^i, z_k)}$ ▷ update importance weights
 - 5: $w_{\text{total}} \leftarrow \sum_{i=1}^n w_k^i$ ▷ compute total weight to normalize importance weights
 - 6: $\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i / w_{\text{total}}\}_{i=1}^n$ ▷ add weighted samples to the new set
 - 7: $n_{\text{eff}} \leftarrow 1 / \sum_{i=1}^n (\tilde{w}_k^i)^2$ ▷ compute effective sample size
 - 8: **if** $n_{\text{eff}} < n_t$ **then**
 - 9: $\mathcal{X}_k \leftarrow \text{resample using } \mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights
 - 10: **return** \mathcal{X}_k
-

A Basic Particle Filter Algorithm in Robotics

Algorithm 3 particle-filter

Require: particles $\mathcal{X}_{k-1} = \{x_{k-1}^i, \tilde{w}_{k-1}^i\}_{i=1}^n$, action u_k , measurement z_k , resampling threshold n_t (e.g. $n/3$);

- 1: $\mathcal{X}_k \leftarrow \emptyset$
 - 2: **for** each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**
 - 3: **draw** $x_k^i \sim p(x_k | x_{k-1}^i, u_k)$ ▷ sample from motion model
 - 4: $w_k^i \leftarrow \tilde{w}_{k-1}^i p(z_k | x_k^i)$ ▷ update importance weights
 - 5: $w_{\text{total}} \leftarrow \sum_{i=1}^n w_k^i$ ▷ compute total weight to normalize importance weights
 - 6: $\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i / w_{\text{total}}\}_{i=1}^n$ ▷ add weighted samples to the new set
 - 7: $n_{\text{eff}} \leftarrow 1 / \sum_{i=1}^n (\tilde{w}_k^i)^2$ ▷ compute effective sample size
 - 8: **if** $n_{\text{eff}} < n_t$ **then**
 - 9: $\mathcal{X}_k \leftarrow \text{resample using } \mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights
 - 10: **return** \mathcal{X}_k
-

- ▶ Resampling eliminates particles with low weights and multiplies particles with high weights;
- ▶ the particles with high weights are selected many times, leading to the loss of diversity (i.e., loss of alternative hypotheses).

Algorithm 4 low-variance-resampling

Require: particles $\mathcal{X}_k = \{x_k^i, \tilde{w}_k^i\}_{i=1}^n$;

- 1: $w_c \leftarrow$ compute the vector of cumulative sum of the weights using $\{\tilde{w}_k^i\}_{i=1}^n$
 $\triangleright w_c$ is the Cumulative Distribution Function (CDF)
 - 2: $r \leftarrow \text{rand}(0, n^{-1})$ \triangleright draw a uniform random number between 0 and n^{-1}
 - 3: $j \leftarrow 1$ \triangleright dummy index to climb the CDF and select particles
 - 4: **for** all $i \in \{1 : n\}$ **do**
 - 5: $u \leftarrow r + (i - 1)n^{-1}$ \triangleright move along the CDF
 - 6: **while** $u > w_c^j$ **do**
 - 7: $j \leftarrow j + 1$
 - 8: $x_k^i \leftarrow x_k^j$ \triangleright replicate the survived particle
 - 9: $\tilde{w}_k^i \leftarrow n^{-1}$ \triangleright set the weight to n^{-1} (uniform distribution)
 - 10: **return** \mathcal{X}_k
-

A Resampling Algorithm

Example: PF Target Tracking

A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to relative noisy range and bearing measurements of the target position at any time step.

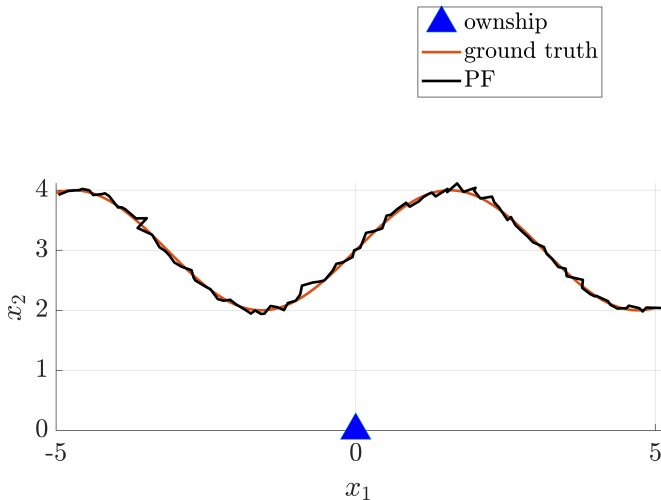
$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{x_k[[1]^2] + \mathbf{x}_k^{[2]^2}} \\ \text{atan2}(\mathbf{x}_k^{[1]}, \mathbf{x}_k^{[2]}) \end{bmatrix} + v_k$$

$$Q_k = 0.1 \, I_2, \, R_k = \text{diag}(0.05^2, 0.01^2)$$

Example: PF Target Tracking

See `pf_single_target.m` for code.



Example: PF Target Tracking, Constant Velocity Motion Model

There is no knowledge of the target motion, but this time, we assume a constant velocity random walk motion model and estimate the target velocity along with the position.

$$x_k = f(u_k, x_{k-1}) + w_k = F_k x_{k-1} + w_k$$

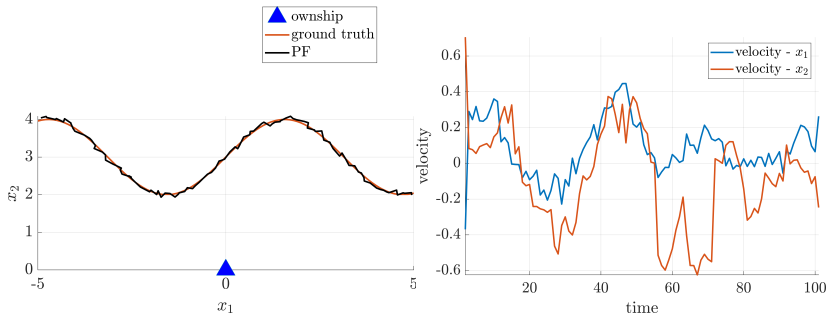
$$F_k = \begin{bmatrix} I & \Delta t I \\ 0 & I \end{bmatrix} \quad \Delta t : \text{sampling time}$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{x_k^1{}^2 + x_k^2{}^2} \\ \text{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k$$

$$Q_k = \text{diag}(0.1^2, 0.1^2, 0.01^2, 0.01^2), R_k = \text{diag}(0.05^2, 0.01^2)$$

Example: PF Target Tracking, Constant Velocity Motion Model

See `pf_single_target_cv.m` for code.



- ▶ SMC methods can solve complex nonlinear, non-Gaussian online estimation problems. For example, dealing with global uncertainty in robot localization and solving the “kidnapped robot” problem.
- ▶ The algorithms are applicable to a very large class of models and are often straightforward to implement.
- ▶ The price to pay for this simplicity is inefficiency in some application domains.

- ▶ Probabilistic Robotics: Ch. 4
- ▶ State Estimation for Robotics: Ch. 4
- ▶ Sequential Monte Carlo Methods in Practice: Ch. 1