NA 568 - Winter 2024

Particle Filtering

Minghan Zhu and Maani Ghaffari

January 24, 2024

UNIVERSITY OF MICHIGAN

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
  $\longrightarrow$ Extended Kalman Filter (EKF)
- Unscented Transform (deterministic sampling)
  $\longrightarrow$ Unscented Kalman Filter (UKF)
- **Monte Carlo methods (random sampling)**
  $\longrightarrow$ **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

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

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

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

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.

We can also denote $\delta_a(x) = \delta(x-a)$.

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

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}(x)$$

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

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}(x)$$

Then, the following integral can be computed:

$$\mathrm{I}_n(f) = \int f(x)p_n(x)\mathrm{d}x = \frac{1}{n}\sum_{i=1}^{n} f(x_i),$$

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

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}(x)$$

Then, the following integral can be computed:

$$\mathrm{I}_n(f) = \int f(x) p_n(x) \mathrm{d}x = \frac{1}{n} \sum_{i=1}^{n} f(x_i),$$

which is an approximation of $\mathbb{E}[f(X)]$ under the true distribution $p(x)$:

$$\mathbb{E}[f(X)] \equiv \mathrm{I}(f) = \int f(x) p(x) \mathrm{d}x \approx \mathrm{I}_n(f)$$

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

- We can easily estimate any quantity $\mathrm{I}(f)$;
- 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.

Monte Carlo Sampling

- We can easily estimate any quantity $\mathrm{I}(f)$;
- 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.
- What if it is hard to sample from $p(x)$?

We can sample from a distribution $q(x)$ on which we know how to sample.

$$\mathbb{E}[f(X)] = \int f(x)p(x)\mathrm{d}x = \int f(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x$$

$$\approx \frac{1}{n}\sum_{i=1}^{N} f(x_i)\frac{p(x_i)}{q(x_i)}$$

where x_i's are sampled from $q(x)$ distribution.

We can sample from a distribution $q(x)$ on which we know how to sample.

$$\mathbb{E}[f(X)] = \int f(x)p(x)\mathrm{d}x = \int f(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x$$

$$\approx \frac{1}{n}\sum_{i=1}^{N} f(x_i)\frac{p(x_i)}{q(x_i)}$$

where x_i's are sampled from $q(x)$ distribution.

Terminologies:

- Target distribution $p(x)$;
- Importance or proposal distribution $q(x)$;
- Likelihood ratio $p(x)/q(x)$.

We assume $f(x)p(x) = 0$ wherever $q(x) = 0$.

Perfect Monte Carlo Sampling for Time Series

We generalize X above to a time series $X_{0:k}$.

Perfect Monte Carlo Sampling for Time Series

We generalize X above to a time series $X_{0:k}$.

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

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

Perfect Monte Carlo Sampling for Time Series

We generalize X above to a time series $X_{0:k}$.

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

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

Then, the following integral can be computed:

$$\mathrm{I}_n(f) = \int f(x_{0:k}) p_n(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k} = \frac{1}{n}\sum_{i=1}^{n} f(x^i_{0:k}) \approx \mathbb{E}[f(X_{0:k})]$$

Perfect Monte Carlo Sampling for Time Series

We generalize X above to a time series $X_{0:k}$.

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

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

Then, the following integral can be computed:

$$\mathrm{I}_n(f) = \int f(x_{0:k})p_n(x_{0:k}|z_{1:k})\mathrm{d}x_{0:k} = \frac{1}{n}\sum_{i=1}^{n} f(x^i_{0:k}) \approx \mathbb{E}[f(X_{0:k})]$$

However, we do not know $p(x_{0:k}|z_{1:k})$!

- 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:

$$\mathrm{I}(f) = \frac{\int f(x_{0:k}) w(x_{0:k}) \pi(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}}{\int w(x_{0:k}) \pi(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}}$$
$$= \frac{\int f(x_{0:k}) p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}}{\int p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}} = \int f(x_{0:k}) p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}$$

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 $\pi(x_{0:k}|z_{1:k})$:

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

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!

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:

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

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

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.

Sequential Importance Sampling (SIS)

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*.*

Recall that:

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

Then the weights can be computed using:

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

SIS Particle Filter Algorithm

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: $\mathcal{X}_k \leftarrow \varnothing$
2: **for** each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**
3: $\quad x_k^i \sim \pi(x_k^i | x_{k-1}^i, z_k)$ ▷ sample from proposal distribution
4: $\quad w_k^i \leftarrow w_k^i \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i)}{\pi(x_k^i | x_{k-1}^i, z_k)}$ ▷ update importance weights
5: $\quad \mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i\}$ ▷ add i-th weighted sample to the new set
6: **return** $\mathcal{X}_k$

Degeneracy Problem

- 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): $\forall i \in \{1:n\}$, $\tilde{w}_k^i = \frac{1}{n} \Longrightarrow n_{\text{eff}} = n$;
  2. the entire distribution mass is placed in one particle (singular): $\forall i \in \{1:j-1, j+1:n\}$, $\tilde{w}_k^i = 0$ and $\tilde{w}_k^j = 1 \Longrightarrow n_{\text{eff}} = 1$;

Sample Impoverishment

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. $n/3$);

$\mathcal{X}_k \leftarrow \varnothing$

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

  draw $x_k^i \sim \pi(x_k^i | x_{k-1}^i, z_k)$ ▷ sample from proposal distribution

  $w_k^i \leftarrow \tilde{w}^i \frac{p(z_k | x_k^i) p(x_k^i | x_{k-1}^i)}{\pi(x_k^i | x_{k-1}^i, z_k)}$ ▷ update importance weights

$w_\mathrm{total} \leftarrow \sum_{i=1}^n w_k^i$ ▷ compute total weight to normalize importance weights

$\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i / w_\mathrm{total}\}_{i=1}^n$ ▷ add weighted samples to the new set

$n_\mathrm{eff} \leftarrow 1 / \sum_{i=1}^n (\tilde{w}_k^i)^2$ ▷ compute effective sample size

if $n_\mathrm{eff} < n_\mathrm{t}$ **then**

  $\mathcal{X}_k \leftarrow$ resample using $\mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights

return $\mathcal{X}_k$

A Basic Particle Filter Algorithm in Robotics

Algorithm 3 `particle-filter`

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

$\mathcal{X}_k \leftarrow \varnothing$

for each $x^i_{k-1} \in \mathcal{X}_{k-1}$ **do**

  draw $x^i_k \sim p(x_k | x^i_{k-1}, u_k)$ ▷ sample from motion model

  $w^i_k \leftarrow \tilde{w}^i_{k-1} p(z_k | x^i_k)$ ▷ update importance weights

$w_{\text{total}} \leftarrow \sum_{i=1}^n w^i_k$ ▷ compute total weight to normalize importance weights

$\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x^i_k, w^i_k / w_{\text{total}}\}^n_{i=1}$ ▷ add weighted samples to the new set

$n_{\text{eff}} \leftarrow 1 / \sum_{i=1}^n (\tilde{w}^i_k)^2$ ▷ compute effective sample size

if $n_{\text{eff}} < n_{\text{t}}$ **then**

  $\mathcal{X}_k \leftarrow$ resample using $\mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights

return $\mathcal{X}_k$

Resampling

- 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).

A Resampling Algorithm

Algorithm 4 `low-variance-resampling`

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

$w_\mathsf{c} \leftarrow$ compute the vector of cumulative sum of the weights using $\{\tilde{w}_k^i\}_{i=1}^n$ ▷ w_c is the Cumulative Distribution Function (CDF)

$r \leftarrow \texttt{rand}(0, n^{-1})$ ▷ draw a uniform random number between 0 and n^{-1}

$j \leftarrow 1$ ▷ dummy index to climb the CDF and select particles

for all $i \in \{1 : n\}$ **do**

    $u \leftarrow r + (i-1)n^{-1}$ ▷ move along the CDF

    while $u > w_\mathsf{c}^j$ **do**

        $j \leftarrow j + 1$

    $x_k^i \leftarrow x_k^j$ ▷ replicate the survived particle

    $\tilde{w}_k^i \leftarrow n^{-1}$ ▷ set the weight to n^{-1} (uniform distribution)

return $\mathcal{X}_k$

A Resampling Algorithm

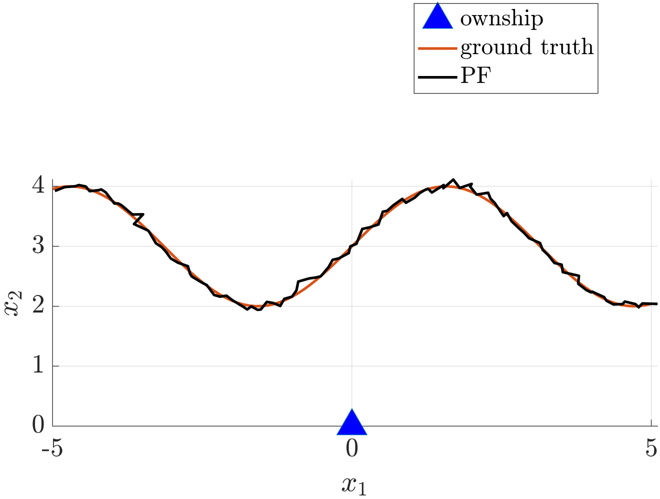
A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to relative noisy range and bearing measurements of the target position at any time step.

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

$$z_k = h(x_k) + v_k = \left[\begin{array}{c} \sqrt{x_k[[1]^2] + \mathbf{x}_k^{[2]^2}} \\ \mathrm{atan2}(\mathbf{x}_k^{[1]}, \mathbf{x}_k^{[2]}) \end{array} \right] + v_k$$

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

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.

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

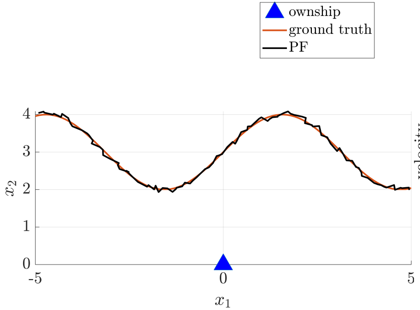
$$F_k = \begin{bmatrix} I & \Delta t I \\ 0 & I \end{bmatrix} \qquad \Delta t : \text{sampling time}$$

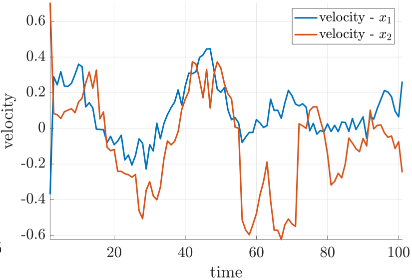
$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{{x_k^1}^2 + {x_k^2}^2} \\ \mathrm{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k$$

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

Example: PF Target Tracking, Constant Velocity Motion Model

See `pf_single_target_cv.m` for code.





SMC and PF Summary

- 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.

Readings (Particle Filtering)

- Probabilistic Robotics: Ch. 4
- State Estimation for Robotics: Ch. 4 (4.2.8)
- Lecture Notes for Mobile Robotics: Ch. 8
- Sequential Monte Carlo Methods in Practice: Ch. 1

Next Time

- Friday: Hands-on Lecture on Bayes Filter. Walk you through implementing the Bayes filter in Python for the grid world localization problem (Markov Localization).

- Monday: Rigid Body Motion - MG

- Readings:
  - State Estimation for Robotics: Ch. 6
  - Lecture Notes for Mobile Robotics: Ch. 3