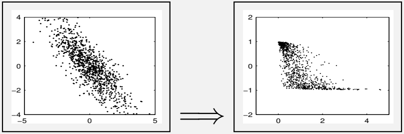
Lecture 6: Particle Filtering

Simo Särkkä

March 9, 2016

Contents

Particle Filtering: Principle



- Animation: Kalman vs. Particle Filtering:
  - Kalman filter animation
  - Particle filter animation
- The idea is to form a weighted particle presentation $(\mathbf{x}^{(i)}, w^{(i)})$ of the posterior distribution:

$$p(\mathbf{x}) \approx \sum_i w^{(i)} \, \delta(\mathbf{x} - \mathbf{x}^{(i)}).$$

- Approximates Bayesian optimal filtering equations with importance sampling.
- Particle filtering = Sequential importance sampling, with additional resampling step.

Monte Carlo Integration

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

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

- 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

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

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

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

Importance Sampling: Basic Version [2/2]

- Importance sampling is based on the identity

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

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

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

- 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 $p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})$ – often not possible.
- However, it turns out that we get a valid algorithm if we define unnormalized importance weights as

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

and then normalize them:

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

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

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

Importance Sampling: Algorithm

Importance Sampling

- Draw N samples from the importance distribution:

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

- 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

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

- 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* $\{(w_k^{(i)}, \mathbf{x}_k^{(i)}) \; : \; i = 1, \ldots, N\}$ such that

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

- 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:

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

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

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

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

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

- Expression for the importance weights can be written as

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

- Thus the weights satisfy the recursion

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

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 $\{\mathbf{x}_k^{(i)} \mid i = 1, \ldots, N\}$.
- 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 Kth step.
- In adaptive resampling, we resample when the effective number of samples is too low (say, $N/10$):

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

- In theory, biased, but in practice works very well and is often used.

Sequential Importance Resampling: Algorithm

Sequential Importance Resampling

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

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

- Calculate new weights

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

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

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

- Calculate new weights

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

and normalize them to sum to unity.

- Perform resampling.

Sequential Importance Resampling: Optimal Importace 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

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

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

Sequential Importance Resampling: Importace Distribution via Kalman Filtering

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

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

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

Rao-Blackwellized Particle Filter: Idea

- Rao-Blackwellized particle filtering (RBPF) is concerned with conditionally Gaussian models:

$$
\begin{aligned}
p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{A}_{k-1}(\mathbf{u}_{k-1})\,\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}(\mathbf{u}_{k-1})) \\
p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{u}_k) &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k(\mathbf{u}_k)\,\mathbf{x}_k, \mathbf{R}_k(\mathbf{u}_k)) \\
p(\mathbf{u}_k \mid \mathbf{u}_{k-1}) &= (\text{any given form}),
\end{aligned}
$$

  where
  - $\mathbf{x}_k$ is the state
  - $\mathbf{y}_k$ is the measurement
  - $\mathbf{u}_k$ is an arbitrary latent variable
- Given the latent variables $\mathbf{u}_{1:T}$ the model is Gaussian.
- The RBPF uses SIR for the latent variables and computes the conditionally Gaussian part in closed form with Kalman filter.

Rao-Blackwellized Particle Filter: Derivation [1/3]

- The full posterior at step k can be factored as

$$p(\mathbf{u}_{0:k}, \mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) = p(\mathbf{x}_{0:k} \mid \mathbf{u}_{0:k}, \mathbf{y}_{1:k}) \, p(\mathbf{u}_{0:k} \mid \mathbf{y}_{1:k})$$

- The first term is Gaussian and computable with Kalman filter and RTS smoother
- For the second term we get the following recursion:

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

Rao-Blackwellized Particle Filter: Derivation [2/3]

- Let's take a look at the terms in

$$p(\mathbf{y}_k \mid \mathbf{u}_{0:k}, \mathbf{y}_{1:k-1})\, p(\mathbf{u}_k \mid \mathbf{u}_{k-1})\, p(\mathbf{u}_{0:k-1} \mid \mathbf{y}_{1:k-1})$$

- The first term can be computed by running Kalman filter with fixed $\mathbf{u}_{0:k}$ over the measurement sequence.
- The second term is just the dynamic model.
- The third term is the posterior from the previous step.

Rao-Blackwellized Particle Filter: Derivation [3/3]

- We can form the importance distribution recursively:

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

- We then get the following recursion for the weights:

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

- Given the marginal posterior for $\mathbf{u}_{0:k}$ we can recover the Gaussian part $\mathbf{x}_{0:k}$ with Kalman filter and RTS smoother.
- The optimal importance distribution takes the form

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

Rao-Blackwellized Particle Filter

- Perform Kalman filter predictions for each of the Kalman filter means and covariances in the particles $i = 1, \ldots, N$ conditional on the previously drawn latent variable values $\mathbf{u}_{k-1}^{(i)}$

$$\begin{aligned} \mathbf{m}_k^{-(i)} &= \mathbf{A}_{k-1}(\mathbf{u}_{k-1}^{(i)})\, \mathbf{m}_{k-1}^{(i)} \\ \mathbf{P}_k^{-(i)} &= \mathbf{A}_{k-1}(\mathbf{u}_{k-1}^{(i)})\, \mathbf{P}_{k-1}^{(i)}\, \mathbf{A}_{k-1}^T(\mathbf{u}_{k-1}^{(i)}) + \mathbf{Q}_{k-1}(\mathbf{u}_{k-1}^{(i)}). \end{aligned}$$

- Draw new latent variables $\mathbf{u}_k^{(i)}$ for each particle in $i = 1, \ldots, N$ from the corresponding importance distributions

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

Rao-Blackwellized Particle Filter: Algorithm [2/3]

Rao-Blackwellized Particle Filter (cont.)

- Calculate new weights as follows:

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

  where the likelihood term is the marginal measurement likelihood of the Kalman filter:

$$\begin{aligned} & p(\mathbf{y}_k \mid \mathbf{u}_{0:k}^{(i)}, \mathbf{y}_{1:k-1}) \\ & = \mathrm{N}\left(\mathbf{y}_k \,\middle|\, \mathbf{H}_k(\mathbf{u}_k^{(i)}) \, \mathbf{m}_k^{-(i)}, \mathbf{H}_k(\mathbf{u}_k^{(i)}) \, \mathbf{P}_k^{-(i)} \, \mathbf{H}_k^T(\mathbf{u}_k^{(i)}) + \mathbf{R}_k(\mathbf{u}_k^{(i)})\right). \end{aligned}$$

- Then normalize the weights to sum to unity.

Rao-Blackwellized Particle Filter: Algorithm [3/3]

Rao-Blackwellized Particle Filter (cont.)

- Perform Kalman filter updates for each of the particles conditional on the drawn latent variables $\mathbf{u}_k^{(i)}$

$$
\begin{aligned}
\mathbf{v}_k^{(i)} &= \mathbf{y}_k - \mathbf{H}_k(\mathbf{u}_k^{(i)})\,\mathbf{m}_k^- \\
\mathbf{S}_k^{(i)} &= \mathbf{H}_k(\mathbf{u}_k^{(i)})\,\mathbf{P}_k^{-(i)}\,\mathbf{H}_k^T(\mathbf{u}_k^{(i)}) + \mathbf{R}_k(\mathbf{u}_k^{(i)}) \\
\mathbf{K}_k^{(i)} &= \mathbf{P}_k^{-(i)}\,\mathbf{H}_k^T(\mathbf{u}_k^{(i)})\,\mathbf{S}_k^{-1} \\
\mathbf{m}_k^{(i)} &= \mathbf{m}_k^{-(i)} + \mathbf{K}_k^{(i)}\,\mathbf{v}_k^{(i)} \\
\mathbf{P}_k^{(i)} &= \mathbf{P}_k^{-(i)} - \mathbf{K}_k^{(i)}\,\mathbf{S}_k^{(i)}\,[\mathbf{K}_k^{(i)}]^T.
\end{aligned}
$$

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

Rao-Blackwellized Particle Filter: Properties

- The Rao-Blackwellized particle filter produces a set of weighted samples $\{w_k^{(i)}, \mathbf{u}_k^{(i)}, \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)} \;:\; i = 1, \ldots, N\}$
- The expectation of a function $\mathbf{g}(\cdot)$ can be approximated as

$$\mathrm{E}[\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} \int \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k^{(i)}) \; \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}) \, \mathrm{d}\mathbf{x}_k.$$

- Approximation of the filtering distribution is

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

- It is possible to do approximate Rao-Blackwellization by replacing the Kalman filter with a Gaussian filter.

Rao-Blackwellization of Static Parameters

- Rao-Blackwellization can sometimes be used in models of the form

$$\begin{aligned} \mathbf{x}_k &\sim p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, \mathbf{u}) \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \,|\, \mathbf{x}_k, \mathbf{u}) \\ \mathbf{u} &\sim p(\mathbf{u}), \end{aligned}$$

  where vector $\mathbf{u}$ contains the unknown static parameters.
- Possible if the posterior distribution of parameters $\mathbf{u}$ depends only on some sufficient statistics $\mathbf{T}_k$:

$$\mathbf{T}_k = \mathbf{T}_k(\mathbf{x}_{1:k}, \mathbf{y}_{1:k})$$

- We also need to have a recursion rule for the sufficient statistics.
- Can be extended to time-varying parameters.

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.
- Superset of other filtering methods – Kalman filter is a Rao-Blackwellized particle filter with one particle.

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 and efficient Rao-Blackwellized filters quite tricky to implement.
- Very hard to find programming errors (i.e., to debug).

Summary

- 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.
- In Rao-Blackwellized particle filters a part of the state is sampled and part is integrated in closed form with Kalman filter.

Particle Filter: Demo

- 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*