Particle Filtering: An Overview

Brian G. Booth

SFU Machine Learning Reading Group

March 5, 2014

What's in a name?

Particle Filtering, a.k.a:

- Sequential Monte Carlo (SMC)
- Bootstrap filtering
- Condensation
- Interacting Particle Approximations
- Survival of the Fittest
- Genetic Algorithms(?)

Introduction to Particle Filtering

To the videos...

Outline

1. Introduction: What is particle filtering?

Outline

1. Introduction: What is particle filtering?
2. Particle Filters and Hidden Markov Models

Outline

1. Introduction: What is particle filtering?
2. Particle Filters and Hidden Markov Models
3. Sampling Particles

Outline

1. Introduction: What is particle filtering?
2. Particle Filters and Hidden Markov Models
3. Sampling Particles
4. Re-sampling Particles

Outline

1. Introduction: What is particle filtering?
2. Particle Filters and Hidden Markov Models
3. Sampling Particles
4. Re-sampling Particles
5. Particle Filters & Genetic Algorithms

Outline

Setting Up Particle Filtering

Notations:

- Hidden States $\mathbf{x} \in \mathcal{X}$
- Observations $\mathbf{z} \in \mathcal{Z}$

Setting Up Particle Filtering

Notations:

- Hidden States $\mathbf{x} \in \mathcal{X}$
- Observations $\mathbf{z} \in \mathcal{Z}$

Given:

- Initial state distribution $p(\mathbf{x}_0)$
- Transition distribution $f(\mathbf{x}_t|\mathbf{x}_{t-1})$
- Observation (i.e. scoring) distribution $g(\mathbf{z}_t|\mathbf{x}_t)$

Setting Up Particle Filtering

Notations:

- Hidden States $\mathbf{x} \in \mathcal{X}$
- Observations $\mathbf{z} \in \mathcal{Z}$

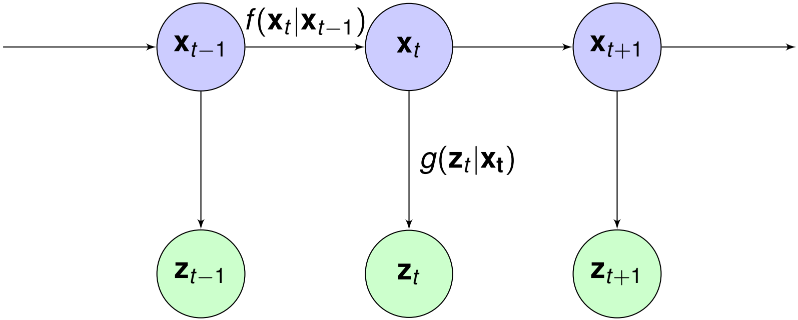
Given:

- Initial state distribution $p(\mathbf{x}_0)$
- Transition distribution $f(\mathbf{x}_t|\mathbf{x}_{t-1})$
- Observation (i.e. scoring) distribution $g(\mathbf{z}_t|\mathbf{x}_t)$

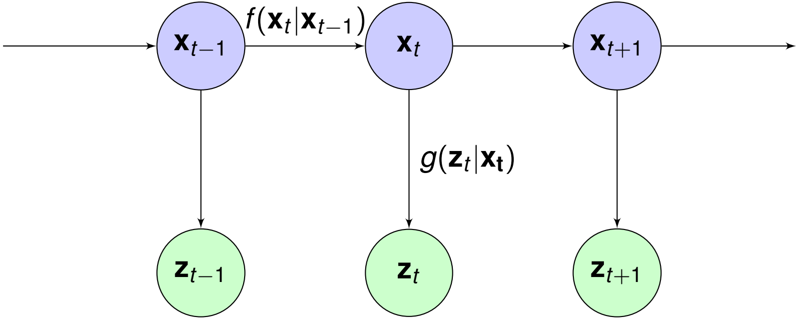
Find:

- Full trajectory distribution $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$

Assuming a Hidden Markov Model



Assuming a Hidden Markov Model



- States (in blue) are hidden
- Observations (in green) are known
- Relationships are “roughly” known

Here comes Bayes Rule...

Remember our goal:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$$

Here comes Bayes Rule...

Remember our goal:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$$

Apply Bayes Rule:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{p(\mathbf{x}_{0:n}, \mathbf{z}_{1:n})}{p(\mathbf{z}_{1:n})}$$

Here comes Bayes Rule...

Remember our goal:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$$

Apply Bayes Rule:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{p(\mathbf{x}_{0:n}, \mathbf{z}_{1:n})}{p(\mathbf{z}_{1:n})}$$

The joint probabilities can be written in conditional form:

$$\begin{aligned} p(\mathbf{x}_{0:n}, \mathbf{z}_{1:n}) =& p(\mathbf{x}_{0:n-1}, \mathbf{z}_{1:n-1}) f(\mathbf{x}_n|\mathbf{x}_{n-1}) g(\mathbf{z}_n|\mathbf{x}_n) \\ p(\mathbf{z}_{1:n}) =& p(\mathbf{z}_n|\mathbf{z}_{1:n-1}) p(\mathbf{z}_{1:n-1}) \end{aligned}$$

Bayes Rule just keeps giving

Plugging in known $f(\cdot)$ and $g(\cdot)$:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}, \mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})p(\mathbf{z}_{1:n-1})}$$

Bayes Rule just keeps giving

Plugging in known $f(\cdot)$ and $g(\cdot)$:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}, \mathbf{z}_{1:n-1}) \frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})p(\mathbf{z}_{1:n-1})}$$

Replace $p(\mathbf{x}_{0:n-1}, \mathbf{z}_{1:n-1})$ with it's conditional:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1}) \frac{p(\mathbf{z}_{1:n-1})f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})p(\mathbf{z}_{1:n-1})}$$

Bayes Rule just keeps giving

Plugging in known $f(\cdot)$ and $g(\cdot)$:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}, \mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})p(\mathbf{z}_{1:n-1})}$$

Replace $p(\mathbf{x}_{0:n-1}, \mathbf{z}_{1:n-1})$ with it's conditional:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{p(\mathbf{z}_{1:n-1})f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})p(\mathbf{z}_{1:n-1})}$$

The $p(\mathbf{z}_{1:n-1})$ distributions cancel out...

Wrapping up Bayes Rule

Thus:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})}$$

Wrapping up Bayes Rule

Thus:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})}$$

We know from normalization rules that:

$$p(\mathbf{z}_n|\mathbf{z}_{1:n-1}) = \int p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)d\mathbf{x}_{n-1:n}$$

Wrapping up Bayes Rule

Thus:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{p(\mathbf{z}_n|\mathbf{z}_{1:n-1})}$$

We know from normalization rules that:

$$p(\mathbf{z}_n|\mathbf{z}_{1:n-1}) = \int p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)d\mathbf{x}_{n-1:n}$$

Let C_n be the integral above. Then...

Things just got Recursive

...we get:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{C_n}$$

Things just got Recursive

...we get:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{C_n}$$

- Our state distribution is recursive!

Things just got Recursive

...we get:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{C_n}$$

- Our state distribution is recursive!
- Recall: we have the base case $p(\mathbf{x}_0)$

Things just got Recursive

...we get:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{C_n}$$

- Our state distribution is recursive!
- Recall: we have the base case $p(\mathbf{x}_0)$
- Distributions $f(\cdot)$ and $g(\cdot)$ are also known

Things just got Recursive

...we get:

$$p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = p(\mathbf{x}_{0:n-1}|\mathbf{z}_{1:n-1})\frac{f(\mathbf{x}_n|\mathbf{x}_{n-1})g(\mathbf{z}_n|\mathbf{x}_n)}{C_n}$$

- Our state distribution is recursive!
- Recall: we have the base case $p(\mathbf{x}_0)$
- Distributions $f(\cdot)$ and $g(\cdot)$ are also known
- We have everything we need to solve this problem!

What's stopping us?

In some cases, we can get $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$ analytically:

- e.g. Kalman filters
  [$p(\mathbf{x}_0)$ is Gaussian, $f(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $g(\mathbf{z}_t|\mathbf{x}_t)$ are linear]

What's stopping us?

In some cases, we can get $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$ analytically:

- e.g. Kalman filters
  [$p(\mathbf{x}_0)$ is Gaussian, $f(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $g(\mathbf{z}_t|\mathbf{x}_t)$ are linear]

In most cases, we can't:

- Integration in C_n intractable
- Often, $p(\mathbf{x}_0)$ not known, estimated with simpler distribution.

What's stopping us?

In some cases, we can get $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$ analytically:

- e.g. Kalman filters
  [$p(\mathbf{x}_0)$ is Gaussian, $f(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $g(\mathbf{z}_t|\mathbf{x}_t)$ are linear]

In most cases, we can't:

- Integration in C_n intractable
- Often, $p(\mathbf{x}_0)$ not known, estimated with simpler distribution.

The Solution:

- Estimate $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$ from samples (i.e. particles).

Outline

Outline

Monte Carlo Estimates

Traditional Monte Carlo:

- Generate N i.i.d. samples $\{x_{0:n}^1, \cdots, x_{0:n}^N\}$ from $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$

Monte Carlo Estimates

Traditional Monte Carlo:

- Generate N i.i.d. samples $\{x_{0:n}^1, \cdots, x_{0:n}^N\}$ from $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$
- Obtain discrete approximation of $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$:

$$\hat{p}(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_{0:n}^i)$$

Monte Carlo Estimates

Traditional Monte Carlo:

- Generate N i.i.d. samples $\{x_{0:n}^1, \cdots, x_{0:n}^N\}$ from $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$
- Obtain discrete approximation of $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$:

$$\hat{p}(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_{0:n}^i)$$

- As $N \to \infty$, $\hat{p}(\cdot) \to p(\cdot)$

Monte Carlo Estimates

Traditional Monte Carlo:

- Generate N i.i.d. samples $\{x_{0:n}^1, \cdots, x_{0:n}^N\}$ from $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$
- Obtain discrete approximation of $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$:

$$\hat{p}(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta(x_{0:n}^i)$$

- As $N \to \infty$, $\hat{p}(\cdot) \to p(\cdot)$

Issues:

- $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$ only known "recursively"

Monte Carlo Estimates

Traditional Monte Carlo:

- Generate N i.i.d. samples $\{x_{0:n}^1, \cdots, x_{0:n}^N\}$ from $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$
- Obtain discrete approximation of $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$:

$$\hat{p}(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_{0:n}^i)$$

- As $N \to \infty$, $\hat{p}(\cdot) \to p(\cdot)$

Issues:

- $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$ only known "recursively"
- Base case $p(\mathbf{x}_0)$ may not be exactly known

Importance Sampling (IS)

Sampling $p(\mathbf{x}_0)$ indirectly:

- Sample from a simple distribution $\pi(\mathbf{x}_0)$ (e.g. Uniform, Gaussian)

Importance Sampling (IS)

Sampling $p(\mathbf{x}_0)$ indirectly:

- Sample from a simple distribution $\pi(\mathbf{x}_0)$ (e.g. Uniform, Gaussian)
- Weight each sample by sampling error:

$$w(\mathbf{x}_0^i) = \frac{p(\mathbf{x}_0^i)}{\pi(\mathbf{x}_0^i)}$$

Importance Sampling (IS)

Sampling $p(\mathbf{x}_0)$ indirectly:

- Sample from a simple distribution $\pi(\mathbf{x}_0)$ (e.g. Uniform, Gaussian)
- Weight each sample by sampling error:

$$w(\mathbf{x}_0^i) = \frac{p(\mathbf{x}_0^i)}{\pi(\mathbf{x}_0^i)}$$

- Obtain *weighted* discrete approximation of $p(\mathbf{x}_0)$:

$$\hat{p}(\mathbf{x}_0) = \frac{1}{\sum_{i=1}^N w^i} \sum_{i=1}^N w^i \delta(x_0^i)$$

Importance Sampling (IS)

Sampling $p(\mathbf{x}_0)$ indirectly:

- Sample from a simple distribution $\pi(\mathbf{x}_0)$ (e.g. Uniform, Gaussian)
- Weight each sample by sampling error:

$$w(\mathbf{x}_0^i) = \frac{p(\mathbf{x}_0^i)}{\pi(\mathbf{x}_0^i)}$$

- Obtain *weighted* discrete approximation of $p(\mathbf{x}_0)$:

$$\hat{p}(\mathbf{x}_0) = \frac{1}{\sum_{i=1}^{N} w^i} \sum_{i=1}^{N} w^i \delta(x_0^i)$$

- Known as Importance Sampling (IS)

Importance Sampling (IS)

Sampling $p(\mathbf{x}_0)$ indirectly:

- Sample from a simple distribution $\pi(\mathbf{x}_0)$ (e.g. Uniform, Gaussian)
- Weight each sample by sampling error:

$$w(\mathbf{x}_0^i) = \frac{p(\mathbf{x}_0^i)}{\pi(\mathbf{x}_0^i)}$$

- Obtain *weighted* discrete approximation of $p(\mathbf{x}_0)$:

$$\hat{p}(\mathbf{x}_0) = \frac{1}{\sum_{i=1}^{N} w^i} \sum_{i=1}^{N} w^i \delta(x_0^i)$$

- Known as Importance Sampling (IS)

How do we know the sampling error in our case?

Sequential Importance Sampling (SIS)

How do we know sampling error in our case?

- We have our observations! $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$

Sequential Importance Sampling (SIS)

How do we know sampling error in our case?

- We have our observations! $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$
- As our sampled states $\mathbf{x}_t^i$ change across time, measure sampling error using

$$w_t^i = g(\mathbf{z}_t | \mathbf{x}_t)$$

Sequential Importance Sampling (SIS)

How do we know sampling error in our case?

- We have our observations! $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$
- As our sampled states $\mathbf{x}_t^i$ change across time, measure sampling error using

$$w_t^i = g(\mathbf{z}_t|\mathbf{x}_t)$$

- Obtain *weighted* discrete approximation of $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$:

$$\hat{p}(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{1}{\sum_{i=1}^{N} w^i} \sum_{i=1}^{N} w^i \delta(x_{0:n}^i)$$

Sequential Importance Sampling (SIS)

How do we know sampling error in our case?

- We have our observations! $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$
- As our sampled states $\mathbf{x}_t^i$ change across time, measure sampling error using

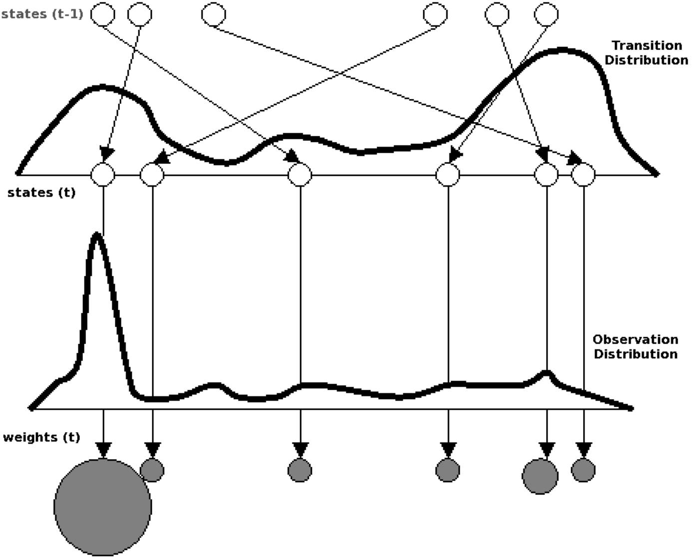
$$w_t^i = g(\mathbf{z}_t|\mathbf{x}_t)$$

- Obtain *weighted* discrete approximation of $p(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$:

$$\hat{p}(\mathbf{x}_{0:n}|\mathbf{z}_{1:n}) = \frac{1}{\sum_{i=1}^{N} w^i} \sum_{i=1}^{N} w^i \delta(x_{0:n}^i)$$

- Known as Sequential Importance Sampling (SIS)

SIS in Image Form



Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. "particles" $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$

Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. "particles" $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)

Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. "particles" $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
    - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
    - Let $w^i = w_0^i = \frac{1}{N}$, for all particles.

Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. “particles” $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all particles.
2. Propagate particles to new state using $f(\mathbf{x}_t|\mathbf{x}_{t-1})$

Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. "particles" $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all particles.
2. Propagate particles to new state using $f(\mathbf{x}_t|\mathbf{x}_{t-1})$
3. Given observation $\mathbf{z}_t$, determine particles' weight $w_t^i = g(\mathbf{z}_t|\mathbf{x}_t)$

Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. "particles" $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all particles.
2. Propagate particles to new state using $f(\mathbf{x}_t|\mathbf{x}_{t-1})$
3. Given observation $\mathbf{z}_t$, determine particles' weight $w_t^i = g(\mathbf{z}_t|\mathbf{x}_t)$
4. Update weight $w^i = w_{t-1}^i w_t^i$.

Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. “particles” $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all particles.
2. Propagate particles to new state using $f(\mathbf{x}_t|\mathbf{x}_{t-1})$
3. Given observation $\mathbf{z}_t$, determine particles’ weight $w_t^i = g(\mathbf{z}_t|\mathbf{x}_t)$
4. Update weight $w^i = w_{t-1}^i w_t^i$.
5. Repeat steps 2-4 n times.

Particle Filtering using SIS

Particle Filtering in 6 Basic Steps:

1. Generate N i.i.d. "particles" $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all particles.
2. Propagate particles to new state using $f(\mathbf{x}_t|\mathbf{x}_{t-1})$
3. Given observation $\mathbf{z}_t$, determine particles' weight $w_t^i = g(\mathbf{z}_t|\mathbf{x}_t)$
4. Update weight $w^i = w_{t-1}^i w_t^i$.
5. Repeat steps 2-4 n times.
6. Obtain *weighted* discrete approximation $\hat{p}(\mathbf{x}_{0:n}|\mathbf{z}_{1:n})$.

Use SIS with Caution

The "gotchas":

- Doesn't work well in high-dimensions (need large N).

Use SIS with Caution

The “gotchas”:

- Doesn’t work well in high-dimensions (need large *N*).
- Degeneracy: After a few timesteps, all but one particle will have negligible weight

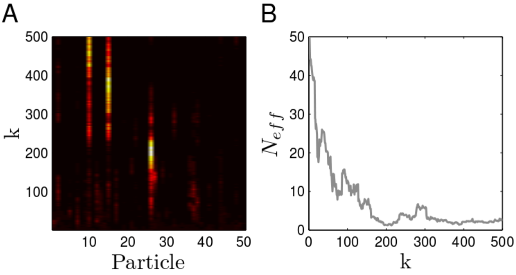


Figure 2: (A) The weights of all 50 particles (x-axis) at each time step k (y-axis). Hotter colors represent larger weights. (B) The effective sample size N_{eff} as a function of time step k.

Outline

Outline

Addressing Degeneracy

Detecting degeneracy:

- At each timestep t, measure effective sample size

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w^i)^2}$$

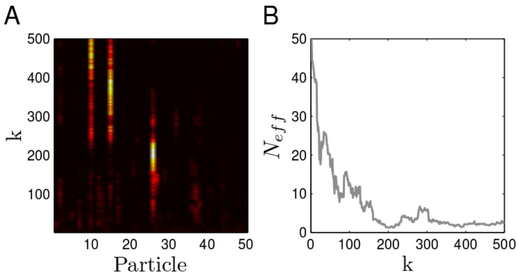


Figure 2: (A) The weights of all 50 particles (x-axis) at each time step k (y-axis). Hotter colors represent larger weights. (B) The effective sample size N_{eff} as a function of time step k.

Resampling Particles

If $N_{eff} < \tau$ at timestep t:

- Sample N new particles $\{x_t^1, \cdots, x_t^N\}$ from $\hat{p}(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$
- Reset weights: $w^i = \frac{1}{N}$

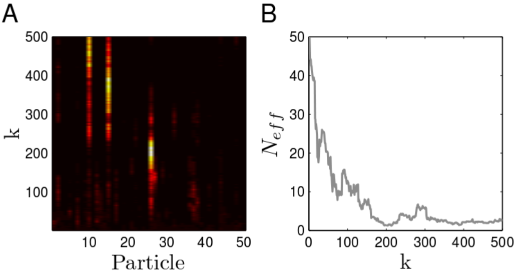


Figure 2: (A) The weights of all 50 particles (x-axis) at each time step k (y-axis). Hotter colors represent larger weights. (B) The effective sample size N_{eff} as a function of time step k.

Sequential Importance Resampling (SIR)

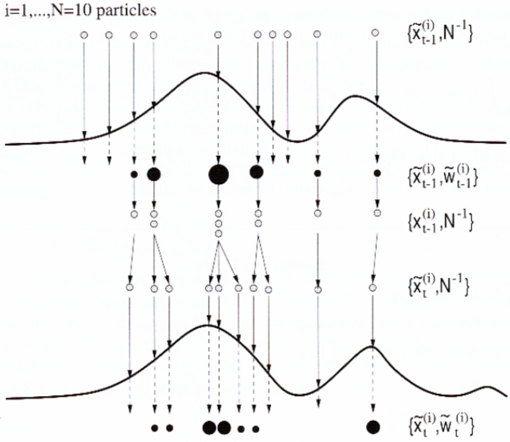
unweighted measure

compute importance weights $\Rightarrow p(x_{t-1}|z_{1:t-1})$

resampling

move particles

predict $p(x_t|z_{1:t-1})$



Pros and Cons of SIR Particle Filtering

Pros:

- Estimation of full state PDFs
- No assumptions on distributions
- Parallelizable

Pros and Cons of SIR Particle Filtering

Pros:

- Estimation of full state PDFs
- No assumptions on distributions
- Parallelizable

Cons:

- Degeneracy possible
- Good estimates may need large N
- Computationally expensive

Outline

1. Introduction: What is particle filtering?
2. Particle Filters and Hidden Markov Models
3. Sampling Particles
4. Re-sampling Particles
5. Particle Filters & Genetic Algorithms

Outline

1. Introduction: What is particle filtering?
2. Particle Filters and Hidden Markov Models
3. Sampling Particles
4. Re-sampling Particles
5. Particle Filters & Genetic Algorithms

More General Particle Filtering

Consider optimization problems:

$$x^* = \arg\min_x E(x)$$

Often solved with Euler-Lagrange

- Introduce artificial timestep t
- Compute $\frac{\partial x}{\partial t}$
- Update using gradient descent $x_t = x_{t-1} - \delta t \frac{\partial x}{\partial t}$

Looks similar...

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all solutions.

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all solutions.
2. Propagate solutions using gradient descent $f(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathbf{x}_{t-1} - \delta t \frac{\partial \mathbf{x}}{\partial t}$

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all solutions.
2. Propagate solutions using gradient descent $f(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathbf{x}_{t-1} - \delta t \frac{\partial \mathbf{x}}{\partial t}$
3. Given observation $E(x_t^i)$, determine solutions' weight $w_t^i = e^{-E(x_t^i)}$

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all solutions.
2. Propagate solutions using gradient descent
   $f(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathbf{x}_{t-1} - \delta t \frac{\partial \mathbf{x}}{\partial t}$
3. Given observation $E(x_t^i)$, determine solutions' weight
   $w_t^i = e^{-E(x_t^i)}$
4. Update weight $w^i = w_{t-1}^i w_t^i$.

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all solutions.
2. Propagate solutions using gradient descent $f(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathbf{x}_{t-1} - \delta t \frac{\partial \mathbf{x}}{\partial t}$
3. Given observation $E(x_t^i)$, determine solutions' weight $w_t^i = e^{-E(x_t^i)}$
4. Update weight $w^i = w_{t-1}^i w_t^i$.
5. Check for degeneracy.

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all solutions.
2. Propagate solutions using gradient descent
   $f(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathbf{x}_{t-1} - \delta t \frac{\partial \mathbf{x}}{\partial t}$
3. Given observation $E(x_t^i)$, determine solutions' weight
   $w_t^i = e^{-E(x_t^i)}$
4. Update weight $w^i = w_{t-1}^i w_t^i$.
5. Check for degeneracy.
6. Repeat steps 2-5 until convergence.

Particle Filtering Optimization

Particle Filtering Optimization in 7 Basic Steps:

1. Generate N i.i.d. solutions $\{x_0^1, \cdots, x_0^N\}$ from $\hat{p}(\mathbf{x}_0)$
   - $\hat{p}(\mathbf{x}_0)$ may be very simple (e.g. Uniform, Gaussian)
   - Let $w^i = w_0^i = \frac{1}{N}$, for all solutions.
2. Propagate solutions using gradient descent
   $f(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathbf{x}_{t-1} - \delta t \frac{\partial \mathbf{x}}{\partial t}$
3. Given observation $E(x_t^i)$, determine solutions' weight
   $w_t^i = e^{-E(x_t^i)}$
4. Update weight $w^i = w_{t-1}^i w_t^i$.
5. Check for degeneracy.
6. Repeat steps 2-5 until convergence.
7. Select x_n^i with largest weight w^i as x^*.

Particle Filtering = Genetic Algorithms?

So what's going on?

- Generating a PDF of $e^{-E(x)}$ using particle filtering.
- Largest-weighted particle is estimate of MLE of PDF.

Particle Filtering = Genetic Algorithms?

So what's going on?

- Generating a PDF of $e^{-E(x)}$ using particle filtering.
- Largest-weighted particle is estimate of MLE of PDF.

But this is what genetic algorithms does...

Particle Filtering = Genetic Algorithms?

So what's going on?

- Generating a PDF of $e^{-E(x)}$ using particle filtering.
- Largest-weighted particle is estimate of MLE of PDF.

But this is what genetic algorithms does...

- Generate sample solutions (i.e. population)
- Score the solutions using $E(x)$ (i.e. fitness rating)
- Resample based on those scores
  (i.e. survival of the fittest)
- Repeat until scores no longer improve
  (i.e. population stabilizes)

Particle Filtering = Genetic Algorithms?

So what's going on?

- Generating a PDF of $e^{-E(x)}$ using particle filtering.
- Largest-weighted particle is estimate of MLE of PDF.

But this is what genetic algorithms does...

- Generate sample solutions (i.e. population)
- Score the solutions using $E(x)$ (i.e. fitness rating)
- Resample based on those scores (i.e. survival of the fittest)
- Repeat until scores no longer improve (i.e. population stabilizes)

Particle filtering is for more than HMM problems.