CSC 665-1: Advanced Topics in Probabilistic Graphical Models

Particle Filtering

Instructor: Prof. Jason Pacheco

(Slides adapted from Prof. Erik Sudderth)
Importance sampling and likelihood weighting

Sequential Monte Carlo: Particle Filters
Monte Carlo Estimators

\[ \mu \triangleq \mathbb{E}[f] = \int f(x)p(x) \, dx \approx \frac{1}{L} \sum_{\ell=1}^{L} f(x^{(\ell)}) \triangleq \hat{f}_L \]

- Expectation estimated from empirical distribution of \( L \) samples:
  \[ \hat{p}_L(x) = \frac{1}{L} \sum_{\ell=1}^{L} \delta_{x^{(\ell)}}(x) \quad x^{(\ell)} \sim p(x) \quad \text{Practical challenge: Must draw samples!} \]
  
- For any \( L \) this estimator, a random variable, is unbiased:
  \[ \mathbb{E}[\hat{f}_L] = \frac{1}{L} \sum_{\ell=1}^{L} \mathbb{E}[f(x^{(\ell)})] = \mathbb{E}[f] \]

- Guarantees about estimator quality as number of samples \( L \) grows:
  \[ \text{Var}[\hat{f}_L] = \frac{1}{L} \text{Var}[f] = \frac{1}{L} \mathbb{E}[(f(x) - \mu)^2] \quad \text{Pr} \left( \lim_{L \to \infty} \hat{f}_L = \mu \right) = 1 \]
Importance Sampling

**Target Distribution:**

\[ p(x) = \frac{1}{Z} p^*(x) \]

**Proposal Distribution:**

\[ q(x) = \frac{1}{Z'} q^*(x) \quad q(x) > 0 \text{ where } p(x) > 0 \]

\[ \mathbb{E}[f] = \int f(x)p(x) \, dx = \int f(x)w(x)q(x) \, dx \quad w(x) = \frac{p(x)}{q(x)} \]

- Estimate target moments via *importance weighted* samples:

\[ \hat{f}_L = \sum_{\ell=1}^L w_\ell f(x^{(\ell)}) \]

\[ w_\ell = \frac{w^*(x^{(\ell)})}{\sum_{m=1}^L w^*(x^{(m)})} \]

\[ w^*(x) = \frac{p^*(x)}{q^*(x)} \]

Assumes we can evaluate un-normalized densities, and sample \( x^{(\ell)} \sim q(x) \)

- Estimator is *asymptotically unbiased*, and *minimum-variance proposal distribution* is

\[ \hat{q}(x) \propto |f(x)|p(x) \]

For evaluation of \( f(x) \), this is more efficient than sampling from target \( p(x) \)!
Importance Sampling

Target Distribution: \[ p(x) = \frac{1}{Z} p^*(x) \]
Proposal Distribution: \[ q(x) = \frac{1}{Z'} q^*(x) \]
\[ q(x) > 0 \text{ where } p(x) > 0 \]

\[ \mathbb{E}[f] = \int f(x) p(x) \, dx = \int f(x) w(x) q(x) \, dx \]
\[ w(x) = \frac{p(x)}{q(x)} \]

*Optimal proposal can be derived via Jensen’s inequality:*

\[ \text{Var}_q[f(x)w(x)] = \mathbb{E}_q[f^2(x)w^2(x)] - \mu^2 \]
\[ \mathbb{E}_q[f^2(x)w^2(x)] \geq \left( \mathbb{E}_q[|f(x)|w(x)] \right)^2 = \left( \int |f(x)|p(x) \, dx \right)^2 \]

*Estimator is asymptotically unbiased, and minimum-variance proposal distribution is*

\[ \hat{q}(x) \propto |f(x)|p(x) \]

*For evaluation of \( f(x) \), this is more efficient than sampling from target \( p(x) \)!*
Selecting Proposal Distributions

Kernel or Parzen window estimators interpolate to predict density:

\[ \hat{p}(x) = \sum_{\ell=1}^{L} w^{(\ell)} N(x; x^{(\ell)}, \Lambda) \]

\[ w^{(\ell)} \propto \frac{p(x^{(\ell)})}{q(x^{(\ell)})} \]
Selecting Proposal Distributions

- For a toy one-dimensional, heavy-tailed target distribution:

\[ \hat{f}_L \]

\[ \text{Samples (L)} \]

Gaussian Proposal

Cauchy (Student’s-t) Proposal

Empirical variance of weights may not predict estimator variance!

- Always (asymptotically) unbiased, but variance of estimator can be enormous unless weight function bounded above:

\[
\mathbb{E}_q[\hat{f}_L] = \mathbb{E}_p[f] \\
\text{Var}_q[\hat{f}_L] = \frac{1}{L} \text{Var}_q[f(x)w(x)] \\
w(x) = \frac{p(x)}{q(x)}
\]
CS242: Lecture 6B Outline

- Importance sampling and likelihood weighting
- Sequential Monte Carlo: Particle Filters
Non-linear State Space Models

- State dynamics and measurements given by potentially complex *nonlinear functions*
- Noise sampled from *non-Gaussian* distributions
- Usually no closed form for messages or marginals
Sequential Importance Sampling

• Suppose interested in some complex, global function of state:

\[ \mathbb{E}[f] = \int f(x)p(x \mid y) \, dx \approx \sum_{\ell=1}^{L} w_\ell f(x^{(\ell)}) \quad w_\ell \propto \frac{p(x^{(\ell)} \mid y)}{q(x^{(\ell)} \mid y)} \quad x^{(\ell)} \sim q(x \mid y) \]

• Could use Markov structure to construct efficient proposal:

\[ q(x \mid y) = q(x_0) \prod_{t=1}^{T} q(x_t \mid x_{t-1}, y_t) \]

\[ q(x_t \mid x_{t-1}, y_t) \approx p(x_t \mid x_{t-1}, y) \]

Weights will become degenerate, with most approaching zero
Particle Resampling

\[ p(x_t \mid y_\bar{t}) \approx \sum_{\ell=1}^{L} \omega_t^{(\ell)} \delta_{x_t^{(\ell)}}(x_t) \quad \rightarrow \quad p(x_t \mid y_\bar{t}) \approx \sum_{\ell=1}^{L} \frac{1}{L} \delta_{x_t^{(\ell)}}(x_t) \]

where \( y_\bar{t} = \{y_1, \ldots, y_t\} \)

Resample with replacement produces random discrete distribution with same mean as original distribution

While remaining unbiased, resampling avoids degeneracies in which most weights go to zero
Particle Filtering Algorithms

- Represent state estimates using a set of samples
- Propagate over time using sequential importance sampling with resampling

Sample-based density estimate

Weight by observation likelihood

Resample & propagate by dynamics
Particle Filters: The Movie

(M. Isard, 1996)
BP for State-Space Models

\[
m_{t-1,t}(x_t) \propto p(x_t \mid y_{t-1}) \quad \text{where} \quad y_{t-1} = \{y_1, \ldots, y_t\}
\]

Prediction (Integral/Sum step of BP):

\[
m_{t-1,t}(x_t) \propto \int p(x_t \mid x_{t-1}) q_{t-1}(x_{t-1}) \, dx_{t-1}
\]

Inference (Product step of BP):

\[
q_t(x_t) = \frac{1}{Z_t} m_{t-1,t}(x_t) p(y_t \mid x_t)
\]
Particle Filter: Measurement Update

- **Incoming message:** A set of $L$ weighted particles

$$m_{t-1,t}(x_t) \approx \sum_{\ell=1}^{L} w_{t-1,t}^{(\ell)} \delta(x_t, x_t^{(\ell)})$$

$$\sum_{\ell=1}^{L} w_{t-1,t}^{(\ell)} = 1$$

- **Bayes’ Rule:** Posterior at particles proportional to prior times likelihood

$$q_{t}(x_t) \propto m_{t-1,t}(x_t) p(y_t \mid x_t) \propto \sum_{\ell=1}^{L} w_{t-1,t}^{(\ell)} p(y_t \mid x_t^{(\ell)}) \delta(x_t, x_t^{(\ell)})$$

$$q_{t}(x_t) = \sum_{\ell=1}^{L} w_{t}^{(\ell)} \delta(x_t, x_t^{(\ell)})$$

$$w_{t}^{(\ell)} = \frac{w_{t-1,t}^{(\ell)} p(y_t \mid x_t^{(\ell)})}{\sum_{m=1}^{L} w_{t-1,t}^{(m)} p(y_t \mid x_t^{(m)})}$$

Variance of importance weights **increases** with each update
Particle Filter: Sample Propagation

- **State Posterior Estimate:** A set of $L$ weighted particles

\[ q_t(x_t) = \sum_{\ell=1}^{L} w_{t}^{(\ell)} \delta(x_t, x_t^{(\ell)}) \quad \sum_{\ell=1}^{L} w_t^{(\ell)} = 1 \]

- **Prediction:** Sample next state conditioned on current particles

\[ m_{t,t+1}(x_{t+1}) = \sum_{\ell=1}^{L} w_{t,t+1}^{(\ell)} \delta(x_{t+1}, x_t^{(\ell)}) \quad x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid x_t^{(\ell)}) \quad w_{t,t+1}^{(\ell)} = w_t^{(\ell)} \]

Assumption for now: Can exactly simulate temporal dynamics
Particle Filter: Resampling

- **State Posterior Estimate:**
  \[ q_t(x_t) = \sum_{\ell=1}^{L} w_t^{(\ell)} \delta(x_t, x_t^{(\ell)}) \]

- **Prediction:** Sample next state conditioned on randomly chosen particles
  \[ m_{t,t+1}(x_{t+1}) = \sum_{\ell=1}^{L} w_{t,t+1}^{(\ell)} \delta(x_{t+1}, x_{t+1}^{(\ell)}) \]

  Resampling with replacement preserves expectations, but increases the variance of subsequent estimators

\[ \tilde{x}_t^{(\ell)} \sim q_t(x_t) \]
\[ x_{t+1} \sim p(x_{t+1} \mid \tilde{x}_t^{(\ell)}) \]
\[ w_{t,t+1}^{(\ell)} = 1/L \]
Particle Filter: Resampling

- **Effective Sample Size:**
  
  \[ L_{\text{eff}} = \left( \sum_{\ell=1}^{L} \left( w^{(\ell)} \right)^2 \right)^{-1} \]

  \[ 1 \leq L_{\text{eff}} \leq L \]

- **State Posterior Estimate:**
  
  \[ q_t(x_t) = \sum_{\ell=1}^{L} w_t^{(\ell)} \delta(x_t, x_t^{(\ell)}) \]

- **Prediction:** Sample next state conditioned on randomly chosen particles
  
  \[ m_{t,t+1}(x_{t+1}) = \sum_{\ell=1}^{L} w_{t,t+1}^{(\ell)} \delta(x_{t+1}, x_{t+1}^{(\ell)}) \]

  Resampling with replacement preserves expectations, but increases the variance of subsequent estimators

  \[ \tilde{x}_t^{(\ell)} \sim q_t(x_t) \]
  
  \[ x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid \tilde{x}_t^{(\ell)}) \]
  
  \[ w_{t,t+1}^{(\ell)} = 1/L \]
Particle Filtering Algorithms

- Represent state estimates using a set of samples
- Propagate over time using sequential importance sampling with resampling

Sample-based density estimate
Weight by observation likelihood
Resample & propagate by dynamics
Bootstrap Particle Filter Summary

- Represent state estimates using a set of samples
- Propagate over time using sequential importance sampling with resampling

Assume sample-based approximation of incoming message:

\[ m_{t-1,t}(x_t) = p(x_t \mid y_{t-1}, \ldots, y_1) \approx \sum_{\ell=1}^{L} \frac{1}{L} \delta_{x_t^{(\ell)}}(x_t) \]

Account for observation via importance weights:

\[ p(x_t \mid y_t, y_{t-1}, \ldots, y_1) \approx \sum_{\ell=1}^{L} w_t^{(\ell)} \delta_{x_t^{(\ell)}}(x_t) \quad w_t^{(\ell)} \propto p(y_t \mid x_t^{(\ell)}) \]

Sample from forward dynamics distribution of next state:

\[ m_{t,t+1}(x_{t+1}) \approx \sum_{m=1}^{L} \frac{1}{L} \delta_{x_{t+1}^{(m)}}(x_{t+1}) \quad x_{t+1}^{(m)} \sim \sum_{\ell=1}^{L} w_t^{(\ell)} p(x_{t+1} \mid x_t^{(\ell)}) \]
Bootstrap Particle Filter Summary

1. Propagation

2. Weighting

3. Resampling

[ Source: Cappe ]
Toy Nonlinear Model

Nonlinear dynamics and observation model...

\[
x_t = \frac{x_{t-1}}{2} + 25 \frac{x_{t-1}}{1 + x_{t-1}^2} + 8 \cos(1.2t) + u_t
\]

\[
y_t = \frac{x_t^2}{20} + u_t
\]

Dynamics

...filter equations lack closed form.

Measurement

Gaussian Noise

\[ N(0, \sigma^2) \]
What is the probability that a state sequence, sampled from the prior model, is consistent with all observations?
A More General Particle Filter

- Assume sample-based approximation of previous state’s marginal:
  \[ p(x_{t-1} | y_{t-1}, \ldots, y_1) \approx \sum_{\ell=1}^{L} \frac{1}{L} \delta_{x_t^{(\ell)}}(x_{t-1}) \]

- Sample from a proposal distribution \( q \):
  \[ x_t^{(\ell)} \sim q(x_t | x_{t-1}^{(\ell)}, y_t) \approx p(x_t | x_{t-1}^{(\ell)}, y_t) \]

- Account for observation and proposal via importance weights:
  \[ w_t^{(\ell)} \propto \frac{p(x_t^{(\ell)} | x_{t-1}^{(\ell)})p(y_t | x_t^{(\ell)})}{q(x_t^{(\ell)} | x_{t-1}^{(\ell)}, y_t)} \]

- Resample to avoid particle degeneracy:
  \[ p(x_t | y_t, \ldots, y_1) \approx \sum_{\ell=1}^{L} \frac{1}{L} \delta_{x_t^{(\ell)}}(x_t) \quad x_t^{(\ell)} \sim \sum_{m=1}^{L} w_t^{(m)} \delta_{x_t^{(m)}}(x_t) \]
Switching State-Space Model

Discrete switching state:

\[ z_t \mid z_{t-1} \sim \text{Cat}(\pi(z_{t-1})) \]

With stochastic transition matrix \( \pi \)

Switching state selects dynamics:

\[ x_t \mid x_{t-1} \sim \mathcal{N}(A_{zt} x_{t-1}, \Sigma_{zt}) \]

(e.g. Nonlinear Gaussian)

Colors indicate 3 writing modes

[ Video: Isard & Blake, ICCV 1998. ]
Example: Particle Filters for SLAM

Simultaneous Localization & Mapping (FastSLAM, Montemerlo 2003)

\[
p(x_t, m \mid z_{1:t}, u_{1:t})
\]
\[
x_t = \text{State of the robot at time } t
\]
\[
m = \text{Map of the environment}
\]
\[
z_{1:t} = \text{Sensor inputs from time } 1 \text{ to } t
\]
\[
u_{1:t} = \text{Control inputs from time } 1 \text{ to } t
\]

Raw odometry (controls)

True trajectory (GPS)

Inferred trajectory & landmarks
Define shorthand notation: \[ y_{1}^{t-1} \triangleq \{ y_1, \ldots, y_{t-1} \} \]

Filtering

Compute \( p(x_t \mid y_1^t) \) at each time \( t \)

Smoothing

Compute full posterior marginal \( p(x_t \mid y_1^T) \) at each time \( t \)
Define shorthand notation: \( y_{1}^{t-1} \triangleq \{ y_{1}, \ldots, y_{t-1} \} \)

If estimates at time \( t \) are not needed immediately, then better smoothed estimates are possible by incorporating future observations.
A Note On Smoothing

- Each resampling step discards states and they cannot subsequently restored
- Resampling introduces dependence across trajectories (common ancestors)
- Smoothed marginal estimates are generally poor
- Backwards simulation improves estimates of smoothed trajectories
Particle Filter Smoothing

Smoothing distribution factorizes as,

\[ p(x_T^T \mid y_1^T) = p(x_T \mid y_1^T) \prod_{t=1}^{T-1} p(x_t \mid x_{t+1}, y_1^T) \]

\[ = p(x_T \mid y_1^T) \prod_{t=1}^{T-1} p(x_t \mid x_{t+1}, y_t^T) \]

Suggests an algorithm to sample from \( p(x_T^T \mid y_1^T) \):

1. Compute and store filter marginals, \( p(x_t \mid y_t^1) \) for \( t=1, \ldots, T \)
2. Sample final state from full posterior marginal, \( x_T \sim p(x_T \mid y_1^T) \)
3. Sample in reverse for \( t=(T-1), (T-2), \ldots, 2, 1 \) from, \( x_t \sim p(x_t \mid x_{t+1}, y_t^1) \)

Use resampling idea to sample from current particle trajectories in reverse
Particle Filter Smoothing

Reverse conditional given by def’n of conditional prob.:

\[
p(x_t \mid x_{t+1}, y_1^t) = \frac{p(x_{t+1} \mid x_t)p(x_t \mid y_1^t)}{p(x_{t+1} \mid y_1^t)}
\]

\[\propto p(x_{t+1} \mid x_t)p(x_t \mid y_1^t)\]

Forward pass sample-based filter marginal estimates:

\[
p(x_t \mid y_1^t) \approx \sum_{\ell=1}^{L} w_t^{(\ell)} \delta(x_t - x_t^{(\ell)})
\]

Thus particle estimate of reverse prediction is:

\[
p(x_t \mid x_{t+1}, y_1^T) \approx \sum_{\ell=1}^{L} \rho_t^{(\ell)}(x_{t+1}) \delta(x_t - x_t^{(\ell)}) \quad \text{where} \quad \rho_t^{(i)}(x_{t+1}) = \frac{w_t^{(i)} p(x_{t+1} \mid x_t^{(i)})}{\sum_{i=1}^{L} w_t^{(i)} p(x_{t+1} \mid x_t^{(i)})}
\]
Algorithm 5 Particle Smoother

for $t = 0$ to $T$ \Comment{Forward Pass Filter}

Run Particle filter, storing at each time step the particles and weights $\{x_t^{(i)}, \omega_t^{(i)}\}_{1 \leq i \leq L}$

end for

Choose $\tilde{x}_T = x_T^{(i)}$ with probability $\omega_T^{(i)}$.

for $t = T - 1$ to $1$ \Comment{Backward Pass Smoother}

Calculate $\rho_t^{(i)} \propto \omega_t^{(i)} p(\tilde{x}_{t+1} | x_t^{(i)})$ for $i = 1, \ldots, L$ and normalize the modified weights.

Choose $\tilde{x}_t = x_t^{(i)}$ with probability $\rho_t^{(i)}$.

end for
Particle Smoothing Example

Smoothing trajectories for $T=100$. True states (*).

Kernel density estimates based on smoothed trajectories. True states (*).
Additional Particle Filter Topics

- Auxiliary particle filter – bias samples towards those more likely to “survive”

- Rao-Blackwell PF – analytically marginalize tractable sub-components of the state (e.g. linear Gaussian terms)

- MCMC PF – apply MC kernel with correct target $p(x_1^t | y_1^t)$ to sample trajectory prior to the resampling step

- Other smoothing topics:
  - Generalized two-filter smoothing
  - MC approximation of posterior marginals $p(x_t | y_1^T)$

- Maximum a posteriori (MAP) particle filter

- Maximum likelihood parameter estimation using PF