Computer Science

CSC535: Probabilistic Graphical Models

Monte Carlo Methods

Prof. Jason Pacheco

Some material from:
Prof. Erik Sudderth & Prof. Kobus Barnard

Outline

- Monte Carlo Estimation
- Sequential Monte Carlo
- Markov Chain Monte Carlo

Outline

- Monte Carlo Estimation
- Sequential Monte Carlo
- Markov Chain Monte Carlo

Motivation for Monte Carlo Methods

- Real problems are typically complex and high dimensional.

- Suppose that we *could* generate samples from a distribution that is proportional to one we are interested in.

- Typically we want posterior samples,

$$p(z \mid \mathcal{D}) = \frac{p(z)p(\mathcal{D} \mid z)}{p(\mathcal{D})} \propto \widetilde{p}(z)$$

Unnormalized posterior

Don't know marginal likelihood / normalizer

- Typically, $\widetilde{p}(z)$ is easier to evaluate (though not always)

Motivation for Monte Carlo Methods

- Generally, Z lives in a very high dimensional space.
- Generally, regions of high $\tilde{p}(z)$ is very little of that space.
- IE, the probability mass is very localized.
- Watching samples from $\tilde{p}(z)$ should provide a good maximum (one of our inference problems)

Motivation for Monte Carlo Methods

- Now consider computing the expectation of a function $f(z)$ over $p(z)$.

- Recall that this looks like $E_{p(z)}[f] = \int_z f(z)p(z)dz$

- How can we approximate or estimate E[f]?

A bad plan…

Discretize the space where z lives into L blocks

Then compute $E_{p(z)}[f] \cong \frac{1}{L}\sum_{l=1}^{L} p(z)f(z)$

Scales poorly with dimension of Z

A better plan…

Given independant samples $z^{(l)}$ from $\tilde{p}(z)$

Estimate $E_{p(z)}[f] \cong \frac{1}{L}\sum_{l=1}^{L} f(z)$

Challenges for Monte Carlo Methods

- In real problems sampling $p(z)$ is very difficult
- Typically don't know normalization, so need to use $\widetilde{p}(z)$ instead
- Even if we **can** sample $p(z)$, it can be hard to know if/when they are "good" and if we have enough (e.g. to approximate E[f] well)
- Sometimes evaluating $\widetilde{p}(z)$ can also be hard

Inference (and related) Tasks

- Simulation: $x \sim p(x) = \frac{1}{Z} f(x)$
- Compute expectations: $\mathbb{E}[\phi(x)] = \int p(x)\phi(x)\, dx$
- Optimization: $x^* = \arg\max_x f(x)$
- Compute normalizer / marginal likelihood: $Z = \int f(x)\, dx$

Inference (and related) Tasks

- Simulation: $x \sim p(x) = \frac{1}{Z} f(x)$
- Compute expectations: $\mathbb{E}[\phi(x)] = \int p(x)\phi(x)\,dx$
- Optimization: $x^* = \arg\max_x f(x)$
- Compute normalizer / marginal likelihood: $Z = \int f(x)\,dx$

Basic Sampling (so far…)

- Uniform sampling (everything builds on this)
- Sampling from simple discrete distributions
  - Multinomial / categorical
  - Binomial / Bernoulli
  - Etc.
- Sampling for selected continuous distributions (e.g., Gaussian)
  - At least, Matlab and Numpy / Scipy know how to do it.
- Ancestral sampling

Sampling Continuous RVs

Recall that the CDF is the integral of the PDF and (left) tail probability,

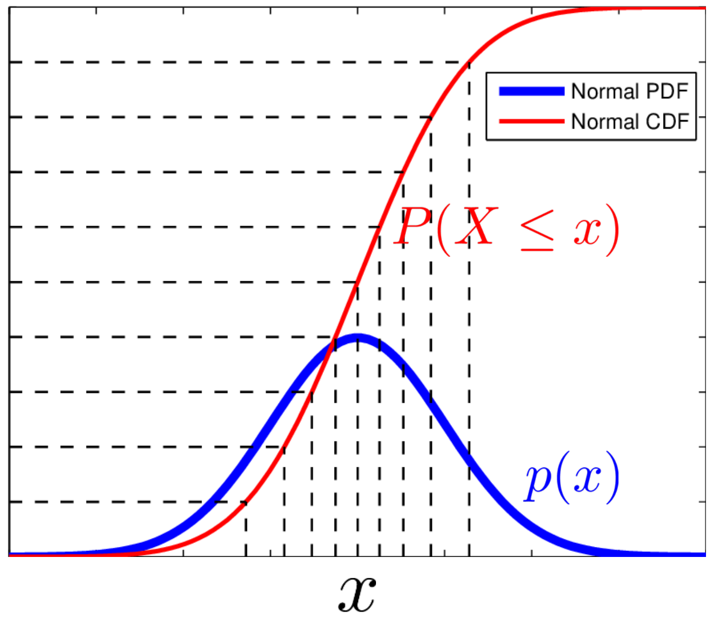
$$P(X \leq x) = \int_{-\infty}^{x} p(X = t)\, dt$$

Observation 1 Equally spaced intervals of CDF correspond to regions of equal event probability

Observation 2 The same events have unequal regions under PDF

Question Given samples $\{x_i\}_{i=1}^N \sim p(x)$ what is the probability distribution of the CDF values,

$$\{P(X \leq x_i)\}_{i=1}^N \sim ???$$



Sampling Continuous RVs

This property is called the **probability integral transform**

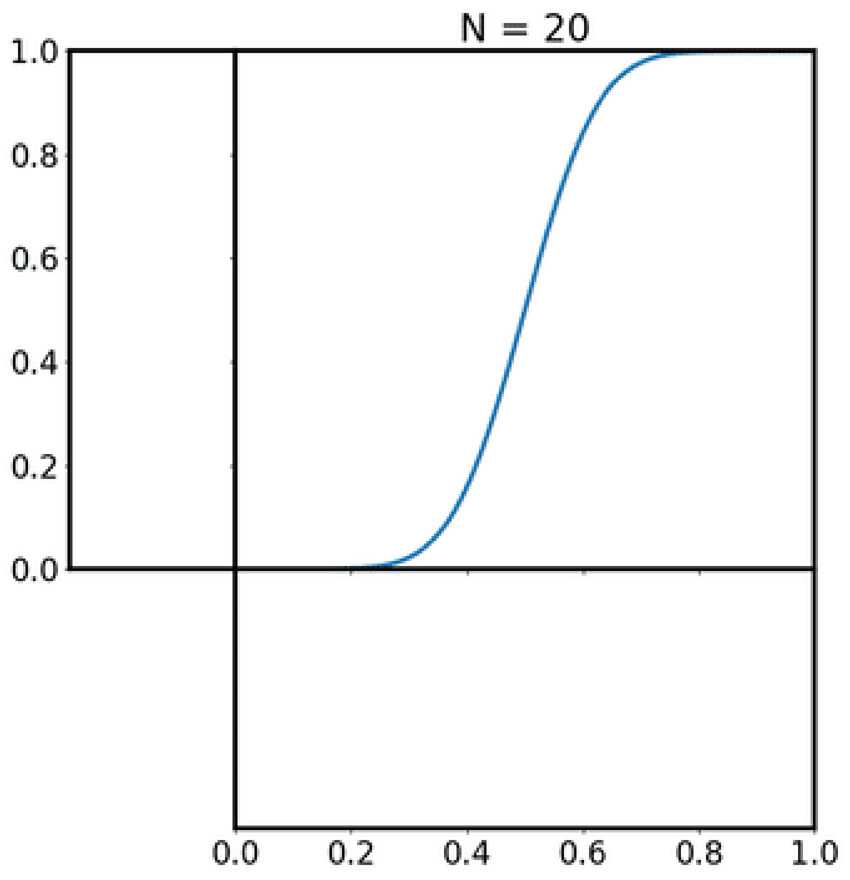
Answer The CDF of iid samples has a **standard uniform** distribution!

$$\{P(X \leq x_i)\}_{i=1}^{N} \sim \text{Uniform}(0, 1)$$

Question How can we use this fact to sample *any* RV?

Answer Apply this relationship in reverse:

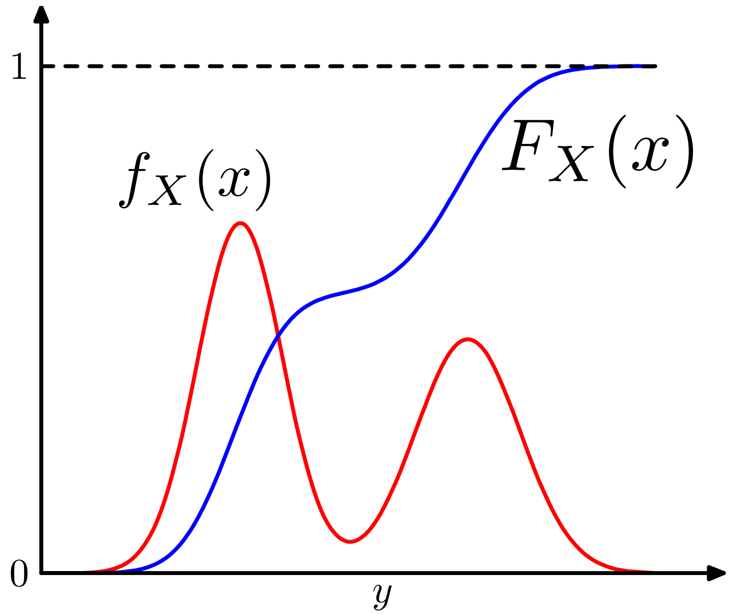
1. Sample iid standard uniform RVs
2. Compute *inverse* CDF
3. Result are samples from the target



Inverse Transform Sampling

- Input: Independent standard uniform variables $U_1, U_2, U_3, \ldots$
- We can use these to exactly sample from any continuous distribution using the cumulative distribution function:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(z)\, dz$$



- Assuming continuous CDF is invertible:

$$h(u) = F_X^{-1}(u)$$
$$X_i = h(U_i)$$

Requires us to have access to inverse CDF

$$P(X_i \leq x) = P(h(U_i) \leq x) = P(U_i \leq F_X(x)) = F_X(x)$$

This function transforms uniform variables to our target distribution!

Inverse Transform Sampling

- Very nice trick that applies to *all* continuous RVs (in theory)
- Yay, we know how to sample any RV right? Wrong…
- Don’t always have the *inverse* CDF (or cannot calculated it)
- Doesn’t extend easily to multivariate RVs (that’s why I only showed 1-dimensional)

Rejection Sampling

Assume

- Access to easy-to-sample distribution $q(z)$
- Constant *k* such that $\widetilde{p}(z) \leq k \cdot q(z)$

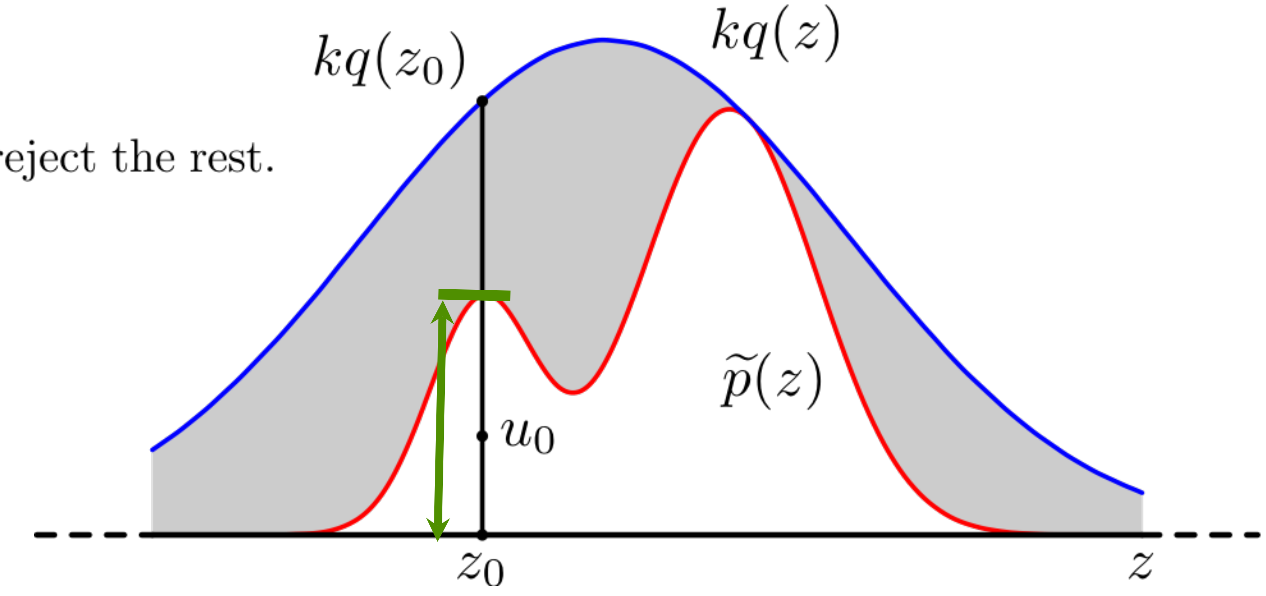
Proposal Distribution
Where we can use one of methods on previous slides to sample efficiently

Algorithm

1) Sample $q(z)$

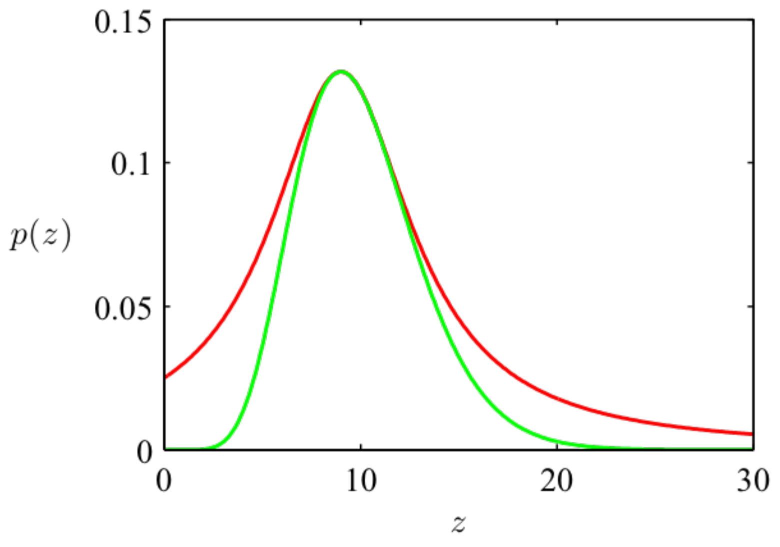
2) Keep samples in proportion to $\dfrac{\tilde{p}(z)}{k \bullet q(z)}$ and reject the rest.

Example Uses Gaussian proposal q to draw samples from multimodal distribution p



Rejection Sampling

- Rejection sampling is hopeless in high dimensions, but is useful for sampling low dimensional "building block" functions.
- For example, the Box-Muller method for generating samples from a Gaussian uses rejection sampling.



A second example where a gamma distribution is approximated by a Cauchy proposal distribution.

Inference (and related) Tasks

- Simulation: $x \sim p(x) = \frac{1}{Z} f(x)$
- Compute expectations: $\mathbb{E}[\phi(x)] = \int p(x)\phi(x)\, dx$
- Optimization: $x^* = \arg\max_x f(x)$
- Compute normalizer / marginal likelihood: $Z = \int f(x)\, dx$

Monte Carlo Integration

One reason to sample a distribution is to approximate expected values under that distribution…

Expected value of function $f(x)$ w.r.t. distribution $p(x)$ given by,

$$\mathbb{E}_p[f(x)] = \int p(x) f(x) \, dx \equiv \mu$$

- Doesn't always have a closed-form for arbitrary functions
- Suppose we have iid samples: $\{x_i\}_{i=1}^N \sim p(x)$
- *Monte Carlo* estimate of expected value,

$$\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^N f(x_i) \approx \mathbb{E}_p[f(x)]$$

Monte Carlo Integration

$$\hat{\mu}_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \approx \mathbb{E}_p[f(x)]$$

- Expectation estimated from *empirical distribution* of L samples:

$$\hat{p}_N(x) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}(x) \qquad \{x_i\}_{i=1}^N \sim p(x)$$

- The *Dirac delta* is *loosely* defined as a piecewise function:

$$\delta_{x_i}(x) = \begin{cases} +\infty & x = x_i \\ 0 & x \neq x_i \end{cases}$$

Caveat This is technically incorrect. Dirac is only well-defined within integrals, $\int \delta_{\bar{x}}(x) f(x)\, dx = f(\bar{x})$ but it gets the intuition across.

- For any *N* this estimator, a random variable, is *unbiased*:

$$\mathbb{E}[\hat{\mu}_N] = \frac{1}{N}\sum_{i=1}^{N} f(x_i) = \mathbb{E}_p[f(x)]$$

Monte Carlo Asymptotics

$$\hat{\mu}_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \approx \mathbb{E}_p[f(x)]$$

- Estimator variance reduces at rate 1/N:

$$\mathrm{Var}[\hat{\mu}_N] = \frac{1}{N}\mathrm{Var}[f] = \frac{1}{N}\mathbf{E}\left[(f(x)-\mu)^2\right]$$

Independent of dimensionality of random variable X

- If the true variance is **finite** have *central limit theorem*:

$$\sqrt{N}(\hat{\mu}_N - \mu) \underset{N\to\infty}{\Longrightarrow} \mathcal{N}(0, \mathrm{Var}[f])$$

- Even if true variance is **infinite** have *laws of large numbers*:

Weak Law

$$\lim_{N\to\infty} \Pr\left(|\hat{\mu}_N - \mu| < \epsilon\right) = 1, \qquad \text{for any} \epsilon > 0$$

Strong Law

$$\Pr\left(\lim_{N\to\infty} \hat{\mu}_N = \mu\right) = 1$$

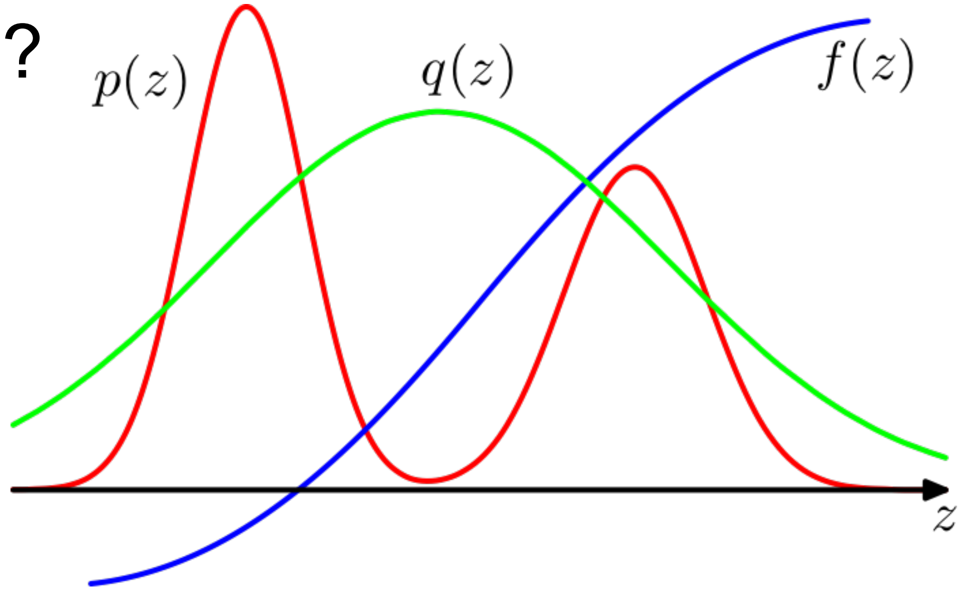
Importance Sampling

Can we estimate $\mathbb{E}_p[f]$ without sampling p(z)?

$$\mathbb{E}_p[f] = \int f(z)p(z)\,dz$$

q(z) is an easy-to-sample proposal distribution

$$= \int f(z)\frac{p(z)}{q(z)}q(z)\,dz$$



Monte Carlo estimate over samples $\{z_i\}_{i=1}^N \sim q$ from proposal q(z):

$$\mathbb{E}_p[f] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(z_i)}{q(z_i)}f(z_i)$$

Key: We can sample from an "easy" distribution q(z) instead!

Importance Sampling

IS weights are the ratio of target / proposal distributions:

$$\mathbb{E}_p[f] \approx \frac{1}{N}\sum_{i=1}^{N} {\color{blue}w_i} f(z_i) \qquad \text{where} \qquad w_i = \frac{p(z_i)}{q(z_i)}$$

But we often do not know the normalizer of the target distribution,

$$p(z) = \frac{1}{Z_p}\widetilde{p}(z) \qquad \text{where} \qquad Z_p = \int \widetilde{p}(z)\, dz$$

Can only evaluate unnormalized target

Can we evaluate IS estimate in terms of unnormalized weights?

$$\widetilde{w}_i = \frac{\widetilde{p}(z_i)}{q(z_i)}$$

Yes! Let's see how…

Importance Sampling (Normalized)

Recall, the importance sampling estimate is given by,

$$\mathbb{E}_p[f] \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(z_i)}{q(z_i)} f(z_i)$$

With normalized target and proposal distributions, respectively:

$$p(z) = \frac{1}{Z_p} \widetilde{p}(z) \qquad\qquad q(z) = \frac{1}{Z_p} \widetilde{q}(z)$$

Substitute and pull out ratio of normalizers,

$$\mathbb{E}_p[f] \approx \left(\frac{Z_q}{Z_p} \right) \frac{1}{N} \sum_{i=1}^{N} \frac{\widetilde{p}(z_i)}{\widetilde{q}(z_i)} f(z_i)$$

Need to compute this…

Easy to compute

Importance Sampling (Normalized)

Idea Compute importance sampling estimate of target normalizer:

$$Z_p = \int \widetilde{p}(z)\, dz = \int \frac{\widetilde{p}(z)}{q(z)} q(z)\, dz \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\widetilde{p}(z_i)}{q(z_i)}$$

Typically we have normalized proposal $q(z)$ so $Z_q = 1$ and,

$$\frac{Z_p}{Z_q} \approx \frac{1}{N} \sum_{i=1} \frac{\widetilde{p}(z_i)}{q(z_i)} = \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_i$$

Where $\widetilde{w}_i$ are our *unnormalized importance weights,*

$$\widetilde{w}_i = \frac{\widetilde{p}(z_i)}{q(z_i)}$$

We can compute this!

Importance Sampling (normalized)

Given samples $\{z_i\}_{i=1}^N \sim q$ we can write the IS estimate as,

$$\mathbb{E}_p[f] \approx \left(\frac{Z_q}{Z_p}\right) \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_i f(z_i)$$

The ratio of normalizers is approximated by normalized weights,

$$\frac{Z_p}{Z_q} \approx \frac{1}{N} \sum_{i=1} \widetilde{w}_i$$

Substituting the normalized weights yields,

$$\mathbb{E}_p[f] \approx \frac{\sum_{i=1}^N \widetilde{w}_i f(z_i)}{\sum_{j=1}^N \widetilde{w}_j} \qquad \text{where} \qquad \widetilde{w}_j = \frac{\widetilde{p}(z_j)}{\widetilde{q}(z_j)}$$

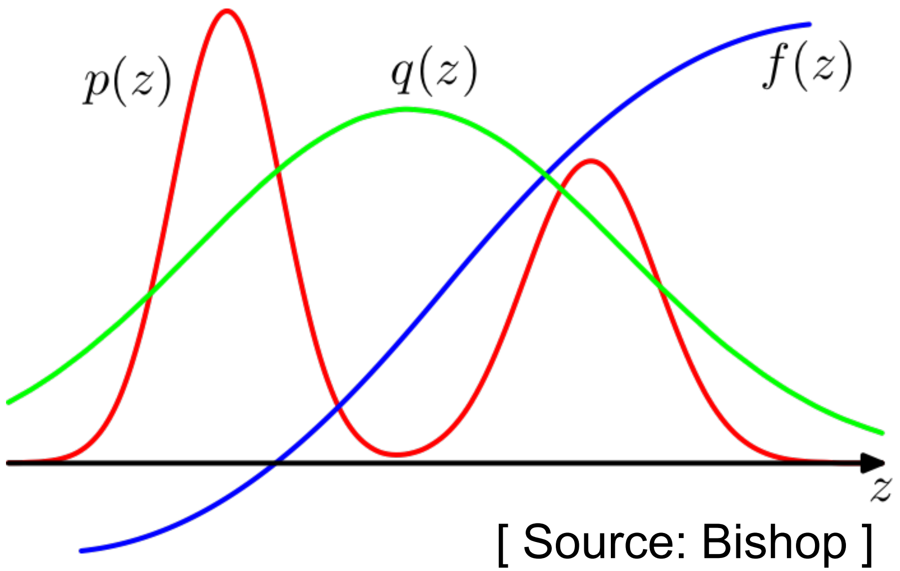
Importance Sampling On-A-Slide

1. Simulate from tractable distribution

$$\{z_i\}_{i=1}^N \sim q(z)$$

2. Compute importance weights & normalize

$$\widetilde{w}_i = \frac{\widetilde{p}(z_i)}{q(z_i)} \qquad w_i = \frac{\widetilde{w}_i}{\sum_{i=1}^N \widetilde{w}_i}$$



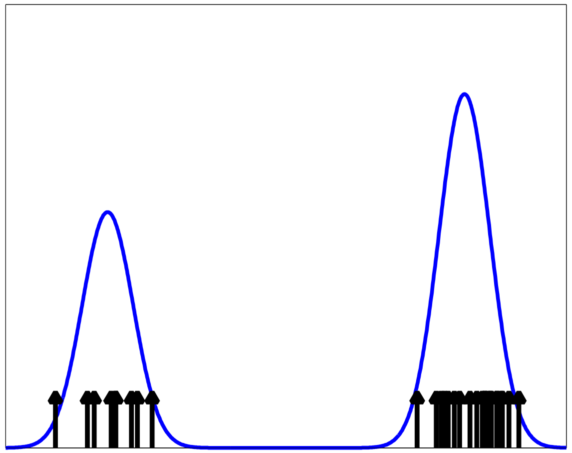
[Source: Bishop]

3. Compute importance-weighted expectation

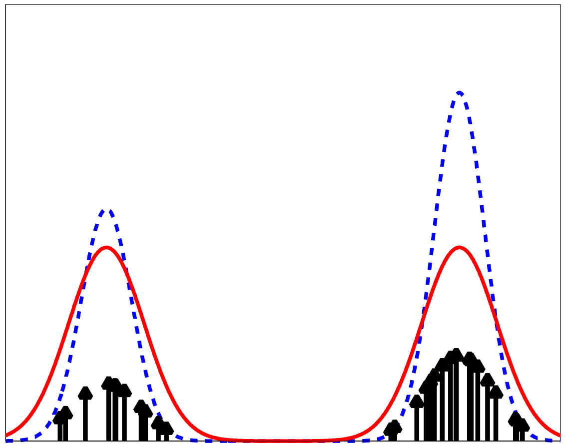
$$\mathbf{E}_p[f(z)] \approx \sum_{i=1}^N w_i f(z_i) \equiv \hat{f}$$

Note There is no 1/N term since it is part of the normalized IS weights

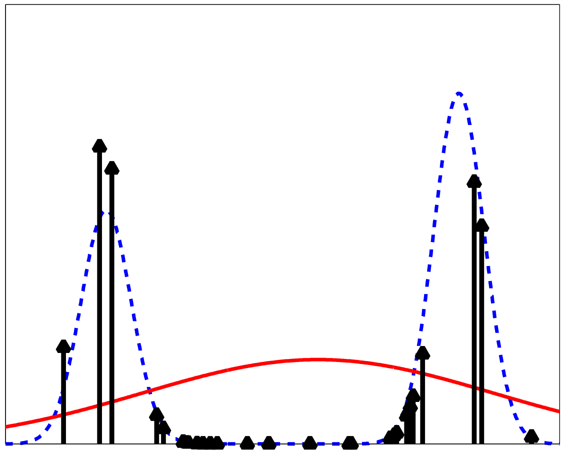
Selecting Proposal Distributions



Target Distribution



Good Proposal

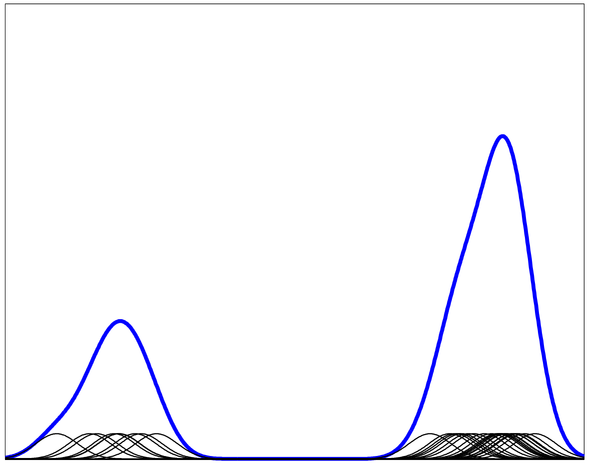


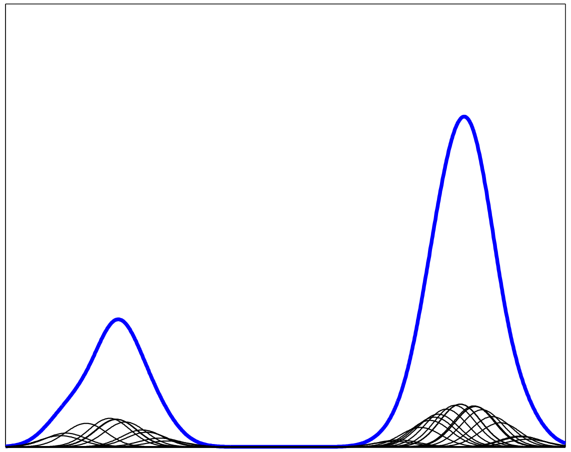
Poor Proposal

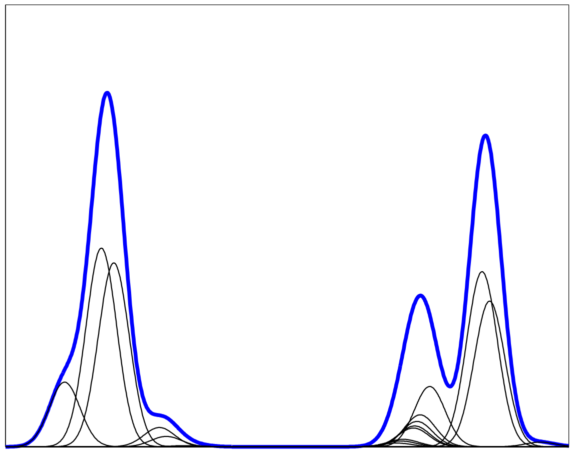
Kernel or Parzen window estimators interpolate to predict density:

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

$$w^{(\ell)} \propto \frac{p(x^{(\ell)})}{q(x^{(\ell)})}$$







Importance Sampling

Q: What is a good proposal distribution?

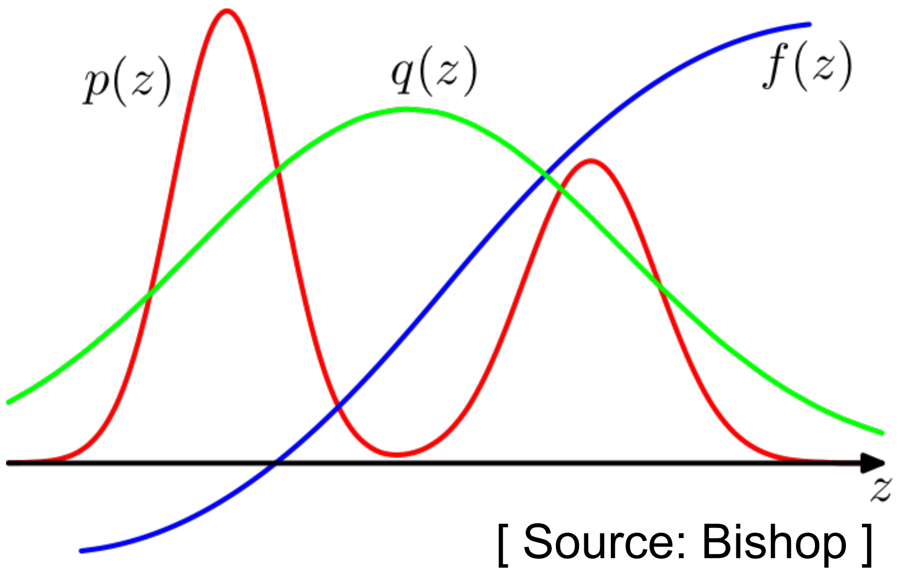
A: Minimize estimator variance

$$q^* = \arg\min_q \mathrm{Var}_q(\hat{f})$$

Minimum variance obtained when,

$$q^* \propto |f(z)|p(z)$$

E.g. can do better than q=p



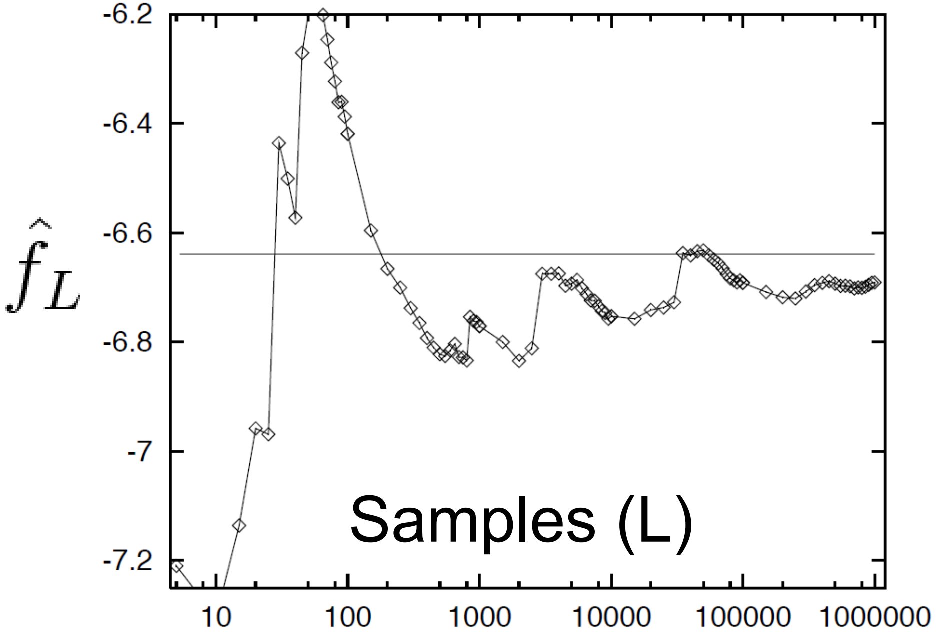
[Source: Bishop]

Estimator variance scales catastrophically with dimension:

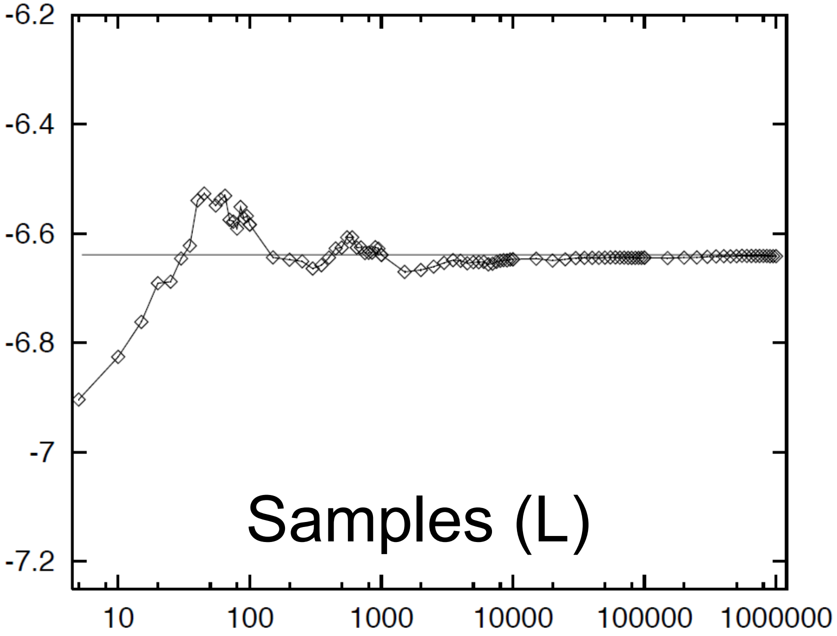
e.g. for N-dim. X and Gaussian q(x): $\mathrm{Var}_{q^*}(\hat{f}) = \exp(\sqrt{2N})$

Selecting Proposal Distributions

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



Gaussian Proposal



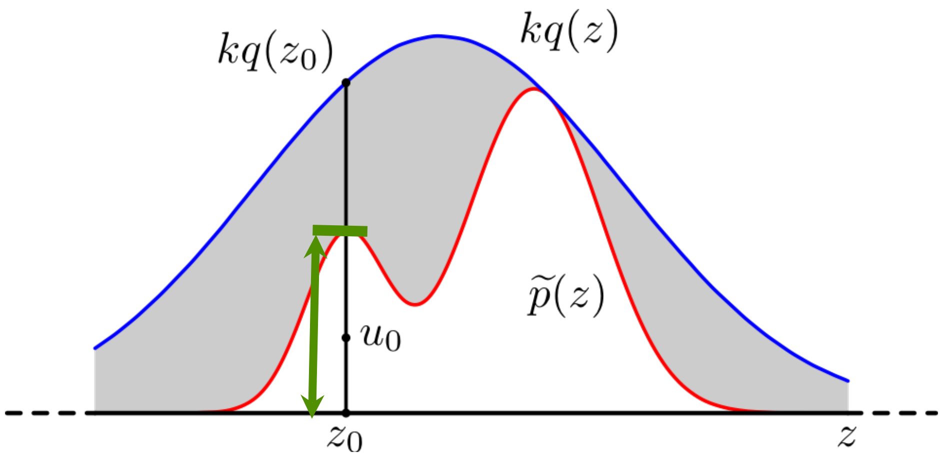
Cauchy (Student's-t) Proposal

Empirical variance of weights may not predict estimator variance!

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

$$\mathbb{E}_q[\hat{f}_L] = \mathbb{E}_p[f] \qquad \mathrm{Var}_q[\hat{f}_L] = \frac{1}{L}\mathrm{Var}_q[f(x)w(x)] \qquad w(x) = \frac{p(x)}{q(x)}$$

Monte Carlo Methods Summary

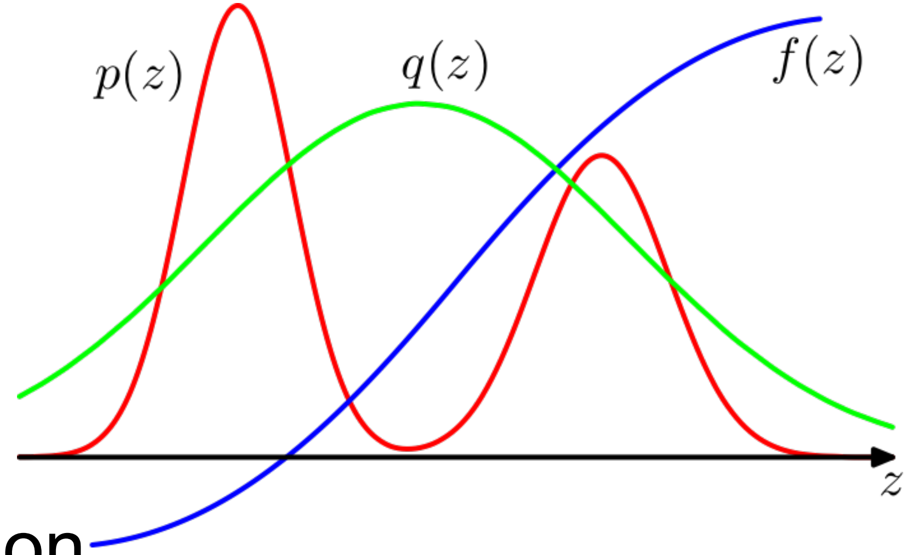


Rejection sampling

- Choose q such that: $\widetilde{p}(z) \leq k \cdot q(z)$
- Sample q(z) and keep with probability: $\frac{\widetilde{p}(z)}{k \cdot q(z)}$

Pro: Efficient, easy to implement

Con: Acceptance rate evaporates as dimension increases



Importance Sampling

$$\mathbf{E}_p[f(z)] \approx \sum_{l=1}^{L} \frac{\widetilde{r}^{(l)}}{\sum_{i=1}^{L} \widetilde{r}^{(i)}} f(z^{(l)}) \qquad \widetilde{r}^{(l)} = \frac{\widetilde{p}(z^{(l)})}{q(z^{(l)})}$$

Pro: Efficient, easy to implement

Con: Variance grows exponentially in dimension

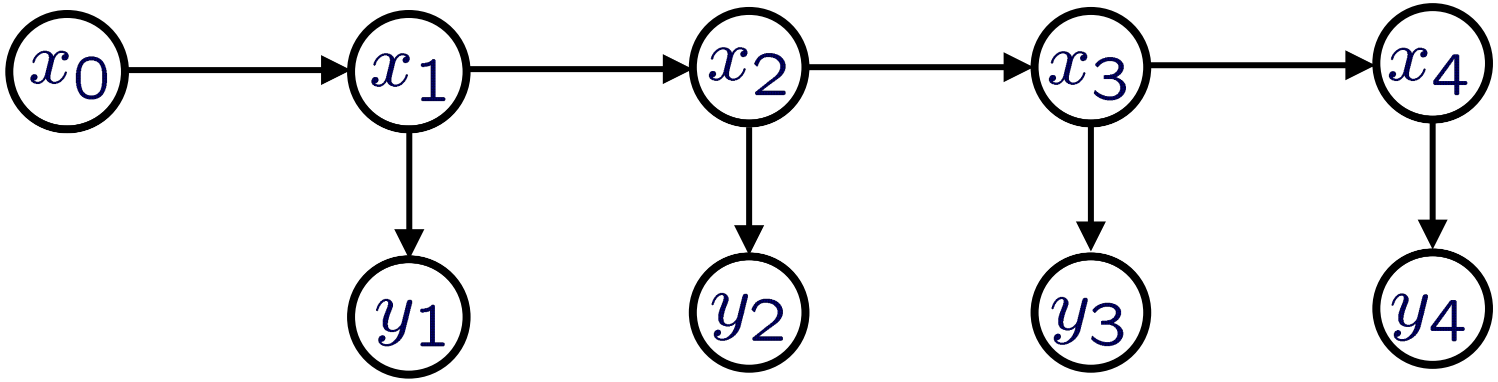
Outline

- Monte Carlo Estimation
- Sequential Monte Carlo
- Markov Chain Monte Carlo

Outline

- Monte Carlo Estimation
- Sequential Monte Carlo
- Markov Chain Monte Carlo

Non-linear State Space Models



$x_t \in \mathbb{R}^d$

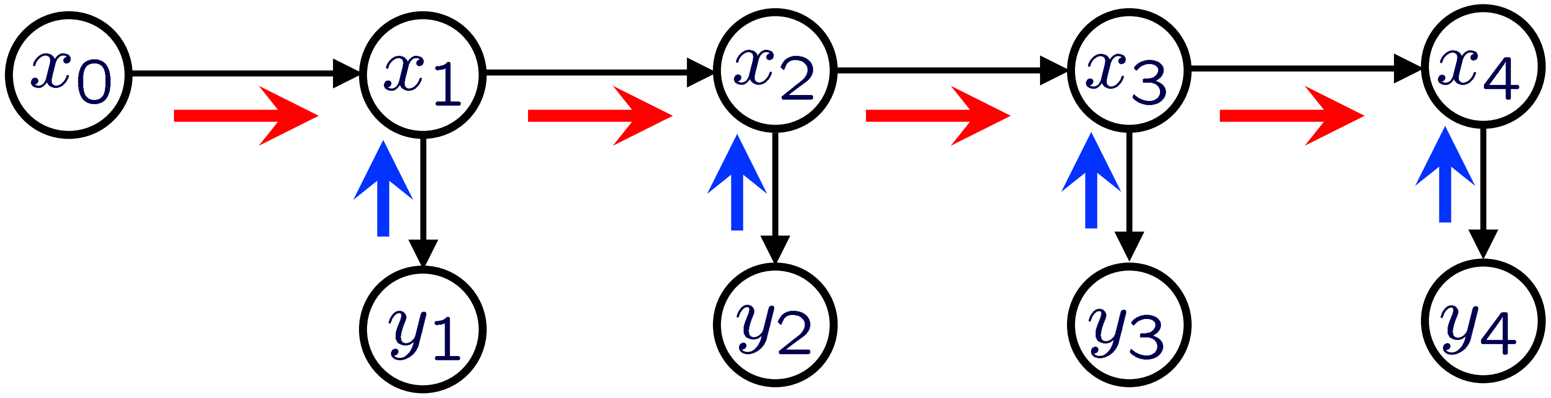
$y_t \in \mathbb{R}^k$

$$x_{t+1} = f(x_t, w_t) \qquad w_t \sim \mathcal{F}$$

$$y_t = g(x_t, v_t) \qquad v_t \sim \mathcal{G}$$

- 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 (SIS)



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

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

- Construct efficient proposal using Markov structure

$$q(x \mid y) = q(x_0) \prod_{t=1}^{T} q(x_t \mid x_{t-1}, y_t) \qquad q(x_t \mid x_{t-1}, y_t) \approx p(x_t \mid x_{t-1}, y)$$

Computing the weights is easy with this type of proposal!

Recursive Weight Updating

Recall the importance weights are given by,

$$w^{(\ell)} \propto \frac{p(x^{(\ell)} \mid y)}{q(x^{(\ell)} \mid y)} \propto \frac{p(x^{(\ell)}, y)}{q(x^{(\ell)} \mid y)}$$

Plugging in the factorization of p and q weights at time t are:

$$w_t^{(\ell)} \propto \frac{p(x_0^{(\ell)})}{q(x_0^{(\ell)})} \frac{p(x_1^{(\ell)} \mid x_0^{(\ell)})p(y_1 \mid x_1^{(\ell)})}{q(x_1^{(\ell)} \mid x_0^{(\ell)}, y_1)} \cdots \frac{p(x_t^{(\ell)} \mid x_{t-1}^{(\ell)})p(y_t \mid x_t^{(\ell)})}{q(x_t^{(\ell)} \mid x_{t-1}^{(\ell)}, y_t)}$$

Therefore, by recursion we have that weights at time $t+1$ are:

$$w_{t+1}^{(\ell)} \propto w_t^{(\ell)} \frac{p(x_{t+1}^{(\ell)} \mid x_t^{(\ell)})p(y_{t+1} \mid x_{t+1}^{(\ell)})}{q(x_{t+1}^{(\ell)} \mid x_t^{(\ell)}, y_t)}$$

Sequential Importance Sampling (SIS)

For ℓ = 1,…,N

Sample initial N particles from proposal prior: $x_0^{(\ell)} \sim q_0$

Compute initial importance weights: $w_0^{(\ell)} \propto p(x_0^{(\ell)}) \div q(x_0^{(\ell)})$

For t=1,…T

For ℓ =1,…N

Propagate particles: $x_t^{(\ell)} \sim q(x_t \mid x_{t-1}^{(\ell)}, y_t)$

Compute *unnormalized* weights,

$$\widetilde{w}_t^{(\ell)} = w_{t-1}^{(\ell)} \frac{p(x_t^{(\ell)} | x_{t-1}^{(\ell)}) p(y_t | x_t^{(\ell)})}{q(x_t^{(\ell)} | x_{t-1}^{(\ell)}, y_t)}$$

Normalize weights: $w_t^{(\ell)} = \widetilde{w}_t^{(\ell)} \div \sum_i \widetilde{w}_t^{(i)}$

Filter mean estimate: $\hat{x}_t = \sum_\ell w_t^{(\ell)} x_t^{(\ell)}$

Particle Filters: The Movie

(M. Isard, 1996)

Weight Degeneration

Sequential importance sampling does not work!

- Sample trajectories $x^{(\ell)}$ are high-dimensional and become unlikely

- In time, unnormalized weights approach zero with high probability,

$$\lim_{t \to \infty} \widetilde{w}_t^{(\ell)} = 0$$

- Normalized weights approach *one-hot vector,*

$$w_t^{(\ell)} = \widetilde{w}_t^{(\ell)} \div \sum_i \widetilde{w}_t^{(i)} = \begin{cases} 1 & \text{if } \widetilde{w}_t^{(\ell)} = \max \widetilde{w}_t \\ 0 & \text{otherwise} \end{cases}$$

Particle Resampling

$$p(x_t \mid y_{\bar{t}}) \approx \sum_{\ell=1}^{L} \omega_t^{(\ell)} \delta_{x_t^{(\ell)}}(x_t) \quad \Rightarrow \quad p(x_t \mid y_{\bar{t}}) \approx \sum_{\ell=1}^{L} \frac{1}{L} \delta_{\bar{x}_t^{(\ell)}}(x_t)$$

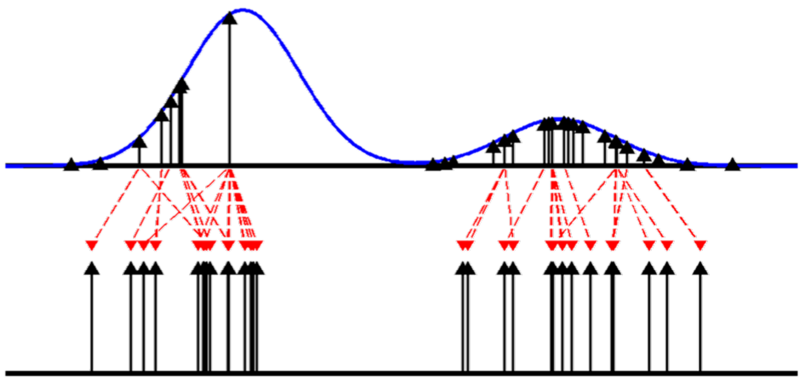
where $y_{\bar{t}} = \{y_1, \ldots, y_t\}$

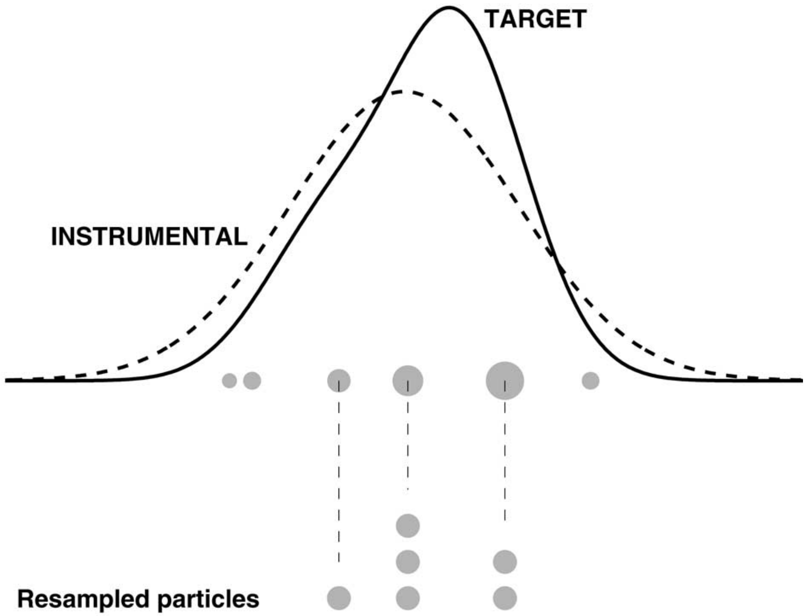
$$\bar{x}_t^{(\ell)} = x_t^{(j_\ell)}$$

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

$$j_\ell \sim \mathrm{Cat}(\omega_t)$$

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





Sequential IS with Resampling : Particle Filter

Initialize: N samples $\widetilde{x}_0^{(\ell)} \sim q_0$ and weights $w_0^{(\ell)} \propto p(x_0^{(\ell)}) \div q(x_0^{(\ell)})$

For t=1,…T

If Resampling:

Resample $x_{t-1}^{(\ell)}$ from $\widetilde{x}_{t-1}$ according to normalized weights w_{t-1} (with replacement)

Set uniform weights $w_{t-1} = 1/N$

Else: Set $x_{t-1} = \widetilde{x}_{t-1}$

For ℓ=1,…N

Propagate particles: $x_t^{(\ell)} \sim q(x_t \mid x_{t-1}^{(\ell)}, y_t)$

Compute *unnormalized* weights, $\widetilde{w}_t^{(\ell)} = w_{t-1}^{(\ell)} \frac{p(x_t^{(\ell)}|x_{t-1}^{(\ell)})p(y_t|x_t^{(\ell)})}{q(x_t^{(\ell)}|x_{t-1}^{(\ell)},y_t)}$

Normalize weights: $w_t^{(\ell)} = \widetilde{w}_t^{(\ell)} \div \sum_i \widetilde{w}_t^{(i)}$

Filter mean estimate: $\hat{x}_t = \sum_\ell w_t^{(\ell)} x_t^{(\ell)}$

“Bootstrap” Proposal

Recall that the full proposal distribution factorizes as,

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

A convenient choice is to sample from the prior distribution,

$$q(x) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1})$$

This is easy to sample, and weight updates simplify,

$$w_{t+1}^{(\ell)} \propto w_t^{(\ell)} \frac{\cancel{p(x_{t+1}^{(\ell)} \mid x_t^{(\ell)})} p(y_{t+1} \mid x_{t+1}^{(\ell)})}{\cancel{p(x_{t+1}^{(\ell)} \mid x_t^{(\ell)})}} = w_t^{(\ell)} p(y_{t+1} \mid x_{t+1}^{(\ell)})$$

“Correct” weights with data likelihood

Bootstrap Particle Filter

Initialize: N samples $\tilde{x}_0^{(\ell)} \sim q_0$ and weights $w_0^{(\ell)} \propto p(x_0^{(\ell)}) \div q(x_0^{(\ell)})$

For t=1,…T

If Resampling:

Resample $x_{t-1}^{(\ell)}$ from $\tilde{x}_{t-1}$ according to normalized weights w_{t-1} (with replacement)

Set uniform weights $w_{t-1} = 1/N$

Else: Set $x_{t-1} = \tilde{x}_{t-1}$

For ℓ=1,…N

Propagate particles: $\tilde{x}_t^{(\ell)} \sim p(x_t \mid x_{t-1}^{(\ell)})$

Compute *unnormalized* weights, $\tilde{w}_t^{(\ell)} = w_{t-1}^{(\ell)} p(y_t \mid \tilde{x}_t^{(\ell)})$

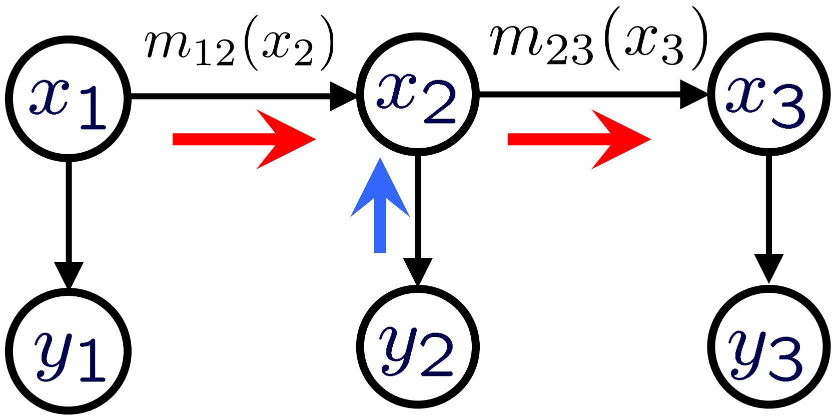
Changes for Bootstrap

Normalize weights: $w_t^{(\ell)} = \tilde{w}_t^{(\ell)} \div \sum_i \tilde{w}_t^{(i)}$

Filter mean estimate: $\hat{x}_t = \sum_\ell w_t^{(\ell)} x_t^{(\ell)}$

Particle Filtering Algorithms

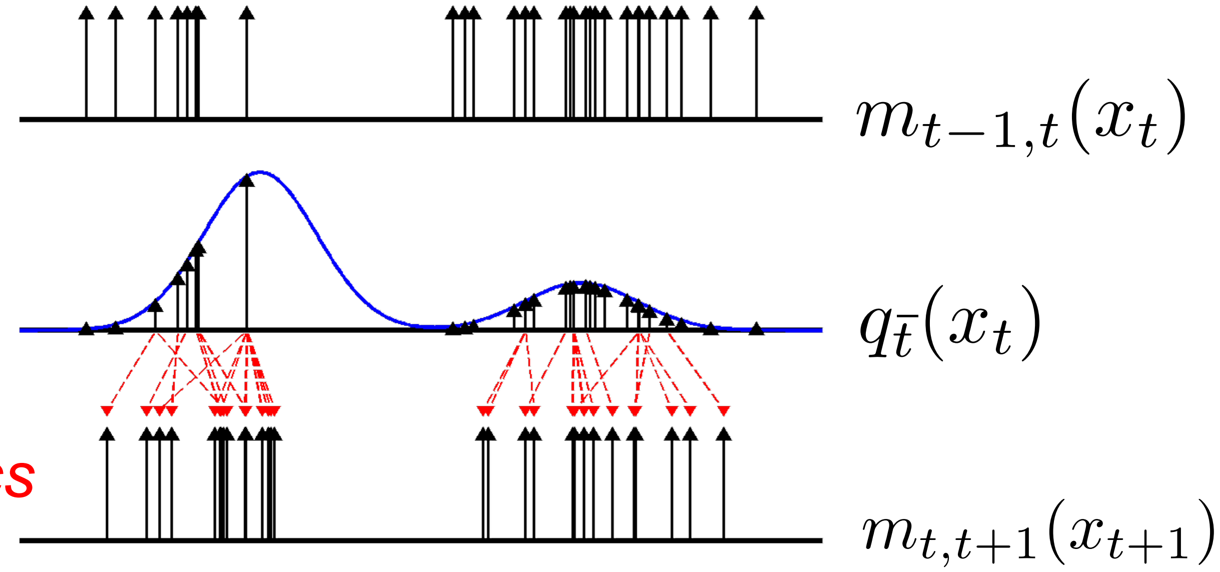
- Represent state estimates using a set of samples
- Propagate over time using *sequential* *importance sampling* *with* *resampling*



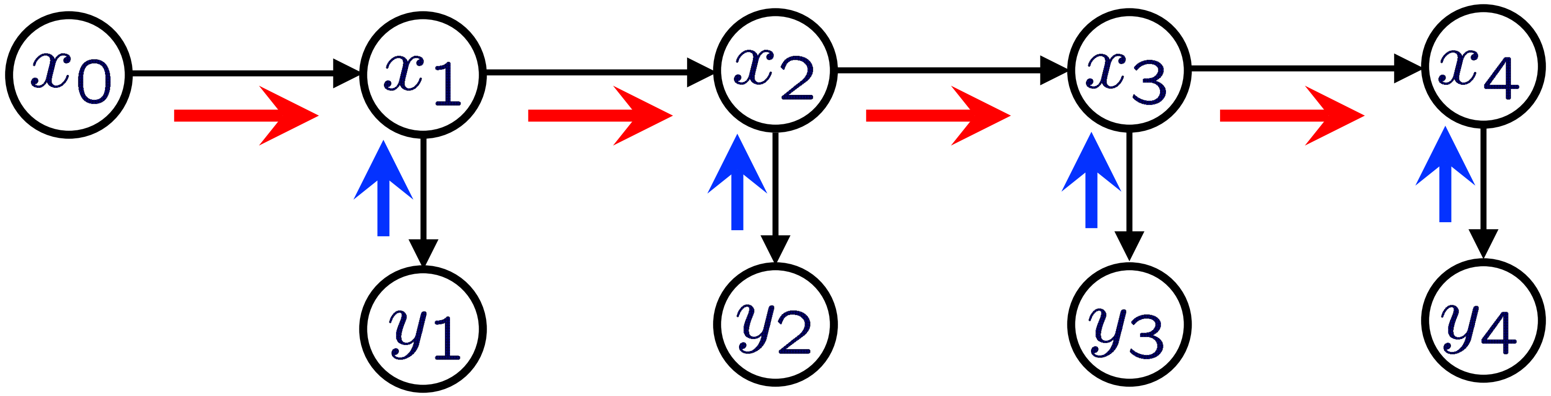
Sample-based density estimate

Weight by observation likelihood

Resample & propagate by dynamics



BP for State-Space Models



$$m_{t-1,t}(x_t) \propto p\left(x_t \mid y_{\overline{t-1}}\right) \qquad \text{where} \qquad y_{\bar{t}} = \{y_1, \ldots, y_t\}$$

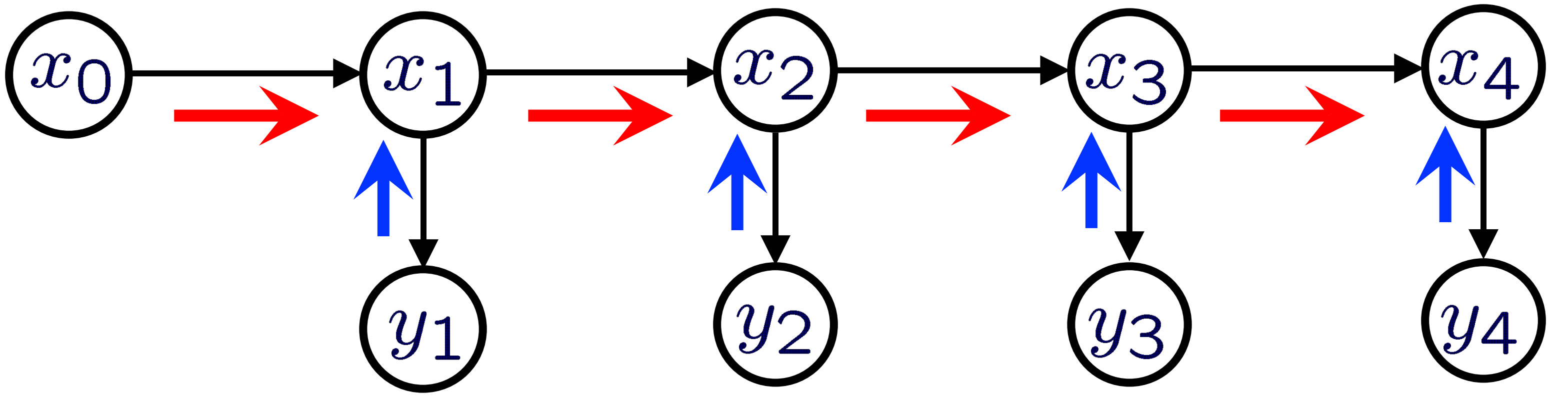
$$m_{t-1,t}(x_t)\, p(y_t \mid x_t) \propto p(x_t \mid y_{\bar{t}}) = q_{\bar{t}}(x_t)$$

Prediction (Integral/Sum step of BP):

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

Inference (Product step of BP): $q_{\bar{t}}(x_t) = \frac{1}{Z_t} m_{t-1,t}(x_t) p(y_t \mid x_t)$

Particle Filter: Measurement Update



Incoming message: A set of *L* weighted particles

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

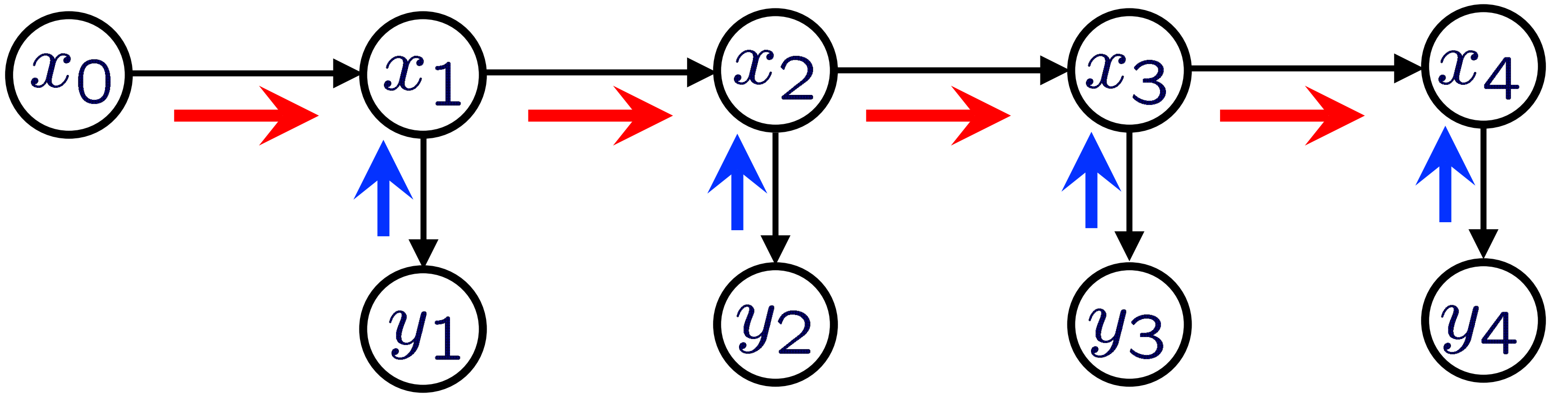
Bayes' Rule: Posterior at particles proportional to prior times likelihood

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

$$q_{\bar{t}}(x_t) = \sum_{\ell=1}^{L} w_t^{(\ell)} \delta(x_t, x_t^{(\ell)}) \qquad w_t^{(\ell)} \triangleq \frac{w_{t-1,t}^{(\ell)}\, p(y_t \mid x_t^{(\ell)})}{\sum_{m=1}^{L} w_{t-1,t}^{(m)}\, p(y_t \mid x_t^{(m)})}$$

Variance of importance weights ***increases*** *with each update*

Particle Filter: Sample Propagation



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

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

Prediction: Sample next state conditioned on current particles

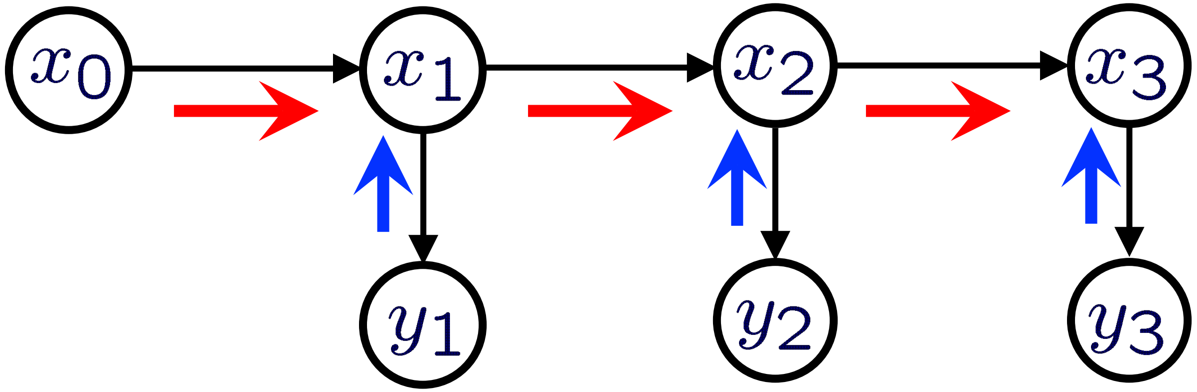
$$m_{t,t+1}(x_{t+1}) = \sum_{\ell=1}^{L} w_{t,t+1}^{(\ell)} \delta(x_{t+1}, x_{t+1}^{(\ell)})$$

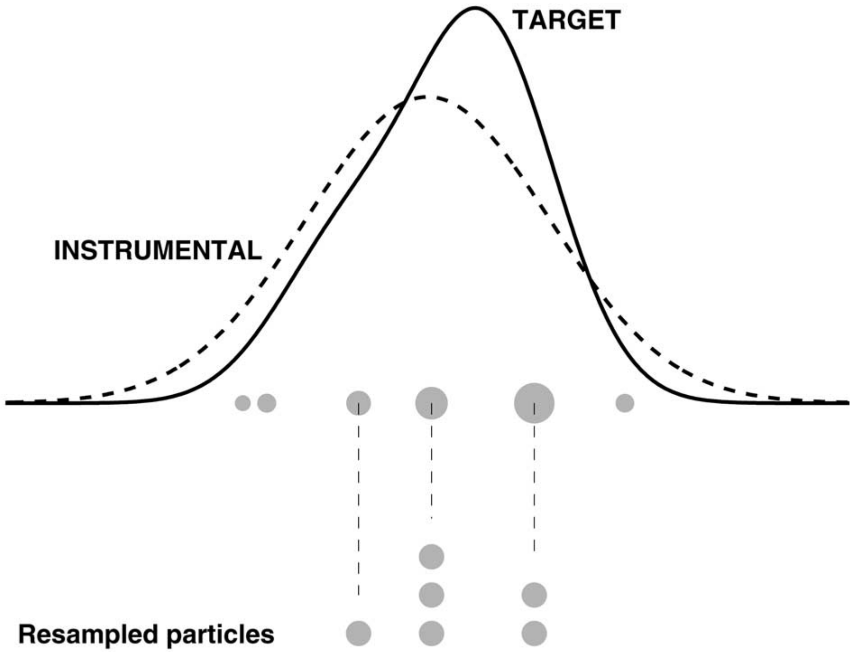
$$x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid x_t^{(\ell)})$$

$$w_{t,t+1}^{(\ell)} = w_t^{(\ell)}$$

Assumption for now: Can exactly simulate temporal dynamics

Particle Filter: Resampling





State Posterior Estimate:

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

Prediction: Sample next state conditioned on randomly chosen particles

$$m_{t,t+1}(x_{t+1}) = \sum_{\ell=1}^{L} w_{t,t+1}^{(\ell)} \delta(x_{t+1}, x_{t+1}^{(\ell)})$$

$$\tilde{x}_t^{(\ell)} \sim q_{\bar{t}}(x_t)$$

$$x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid \tilde{x}_t^{(\ell)})$$

$$w_{t,t+1}^{(\ell)} = 1/L$$

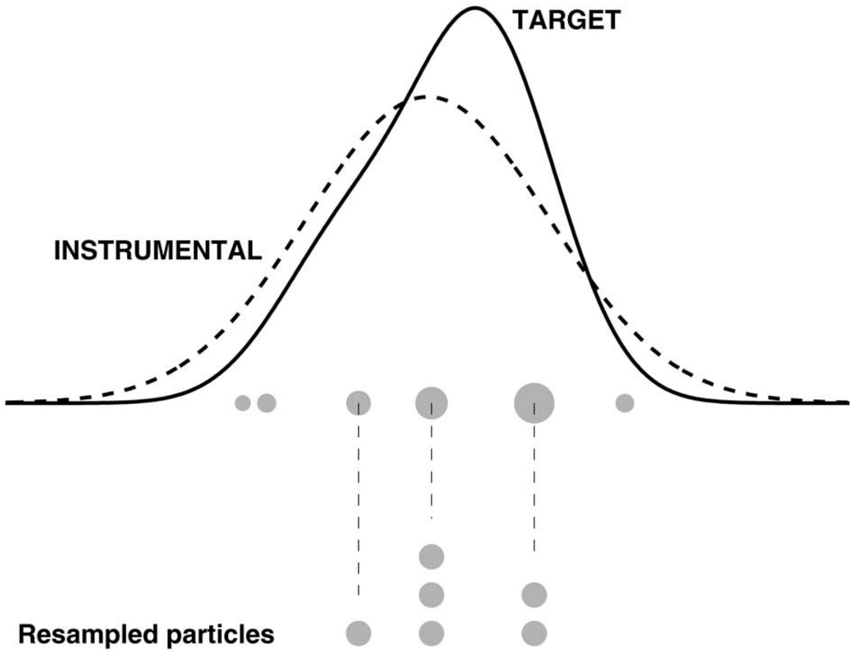
Resampling with replacement preserves expectations, but increases the variance of subsequent estimators

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:

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

Prediction: Sample next state conditioned on randomly chosen particles

$$m_{t,t+1}(x_{t+1}) = \sum_{\ell=1}^{L} w_{t,t+1}^{(\ell)} \delta(x_{t+1}, x_{t+1}^{(\ell)})$$

$$\tilde{x}_t^{(\ell)} \sim q_{\bar{t}}(x_t)$$

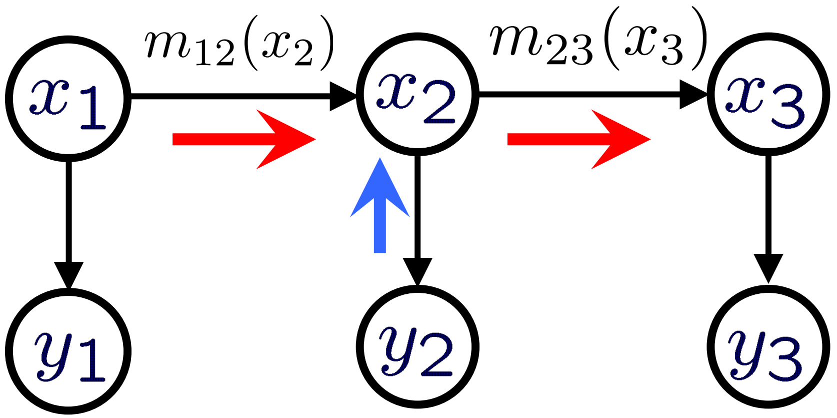
$$x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid \tilde{x}_t^{(\ell)})$$

$$w_{t,t+1}^{(\ell)} = 1/L$$

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

Particle Filtering Algorithms

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



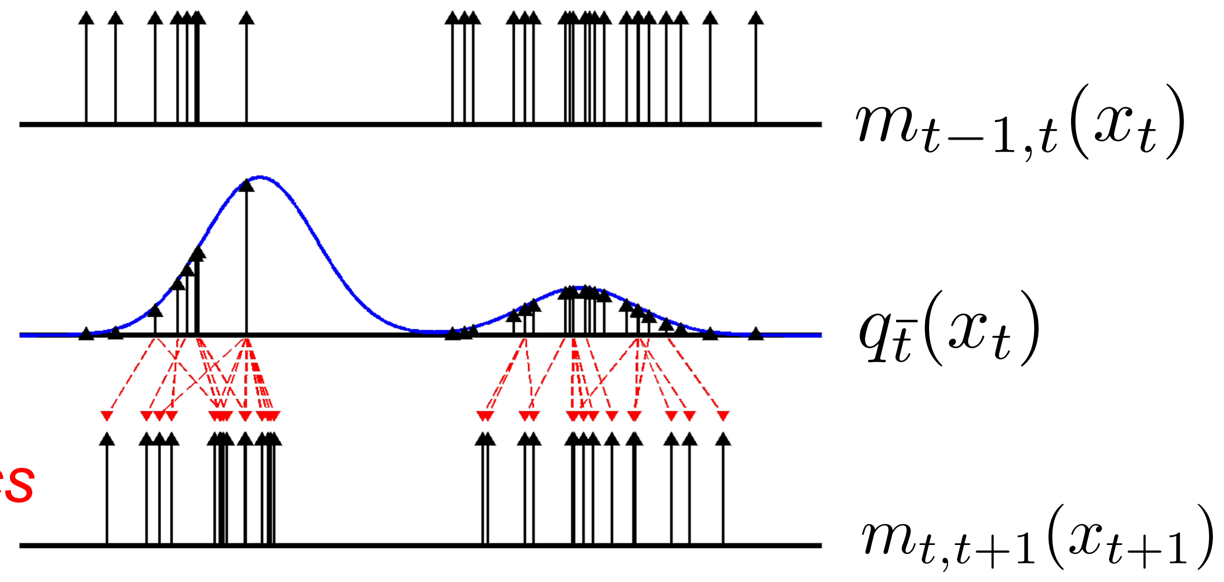
Sample-based density estimate

↓

Weight by observation likelihood

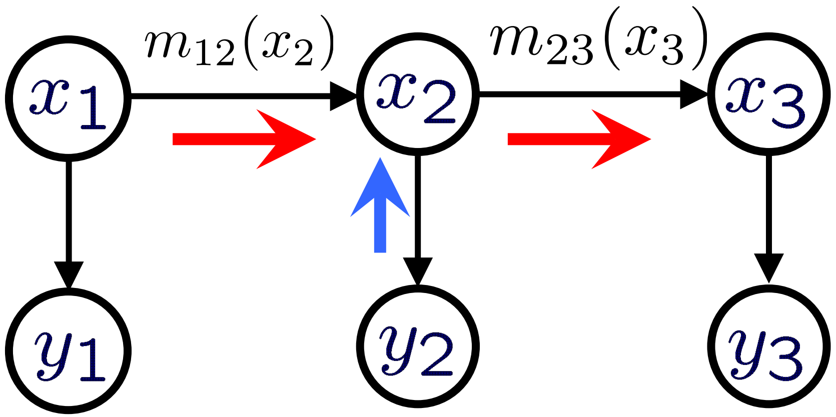
↓

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:

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

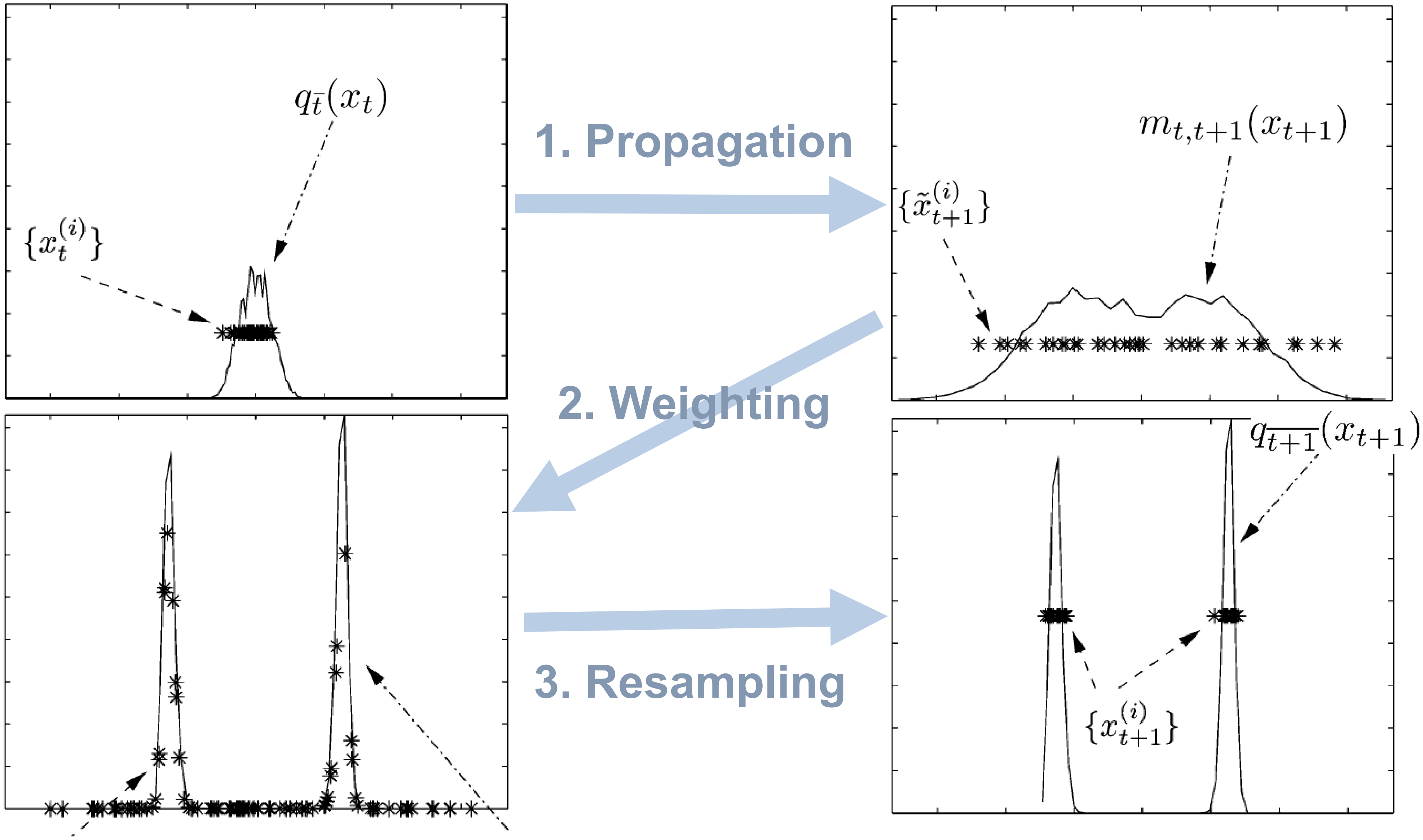
Account for observation via importance weights:

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

Sample from forward dynamics distribution of next state:

$$m_{t,t+1}(x_{t+1}) \approx \sum_{m=1}^{L} \frac{1}{L} \delta_{x_{t+1}^{(m)}}(x_{t+1}) \qquad x_{t+1}^{(m)} \sim \sum_{\ell=1}^{L} w_t^{(\ell)} p(x_{t+1} \mid x_t^{(\ell)})$$

Bootstrap Particle Filter Summary

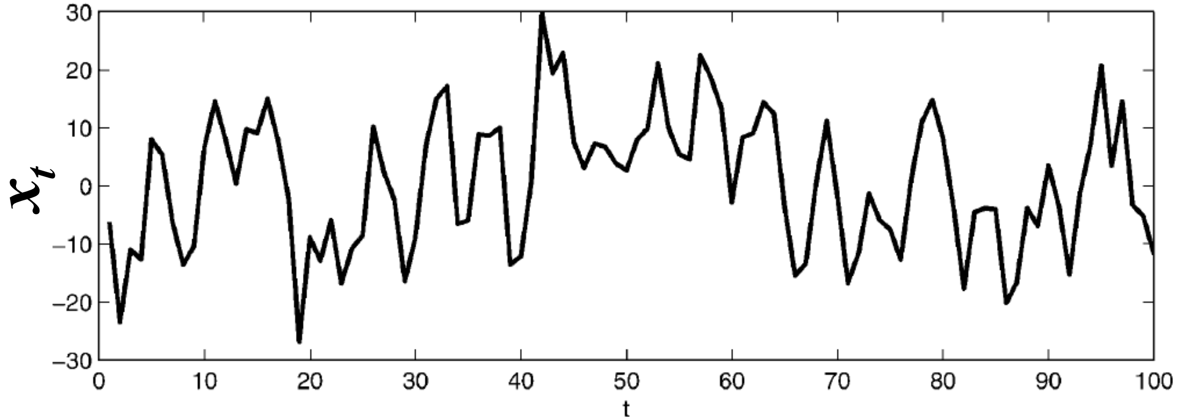


[Source: Cappe]

Toy Nonlinear Model

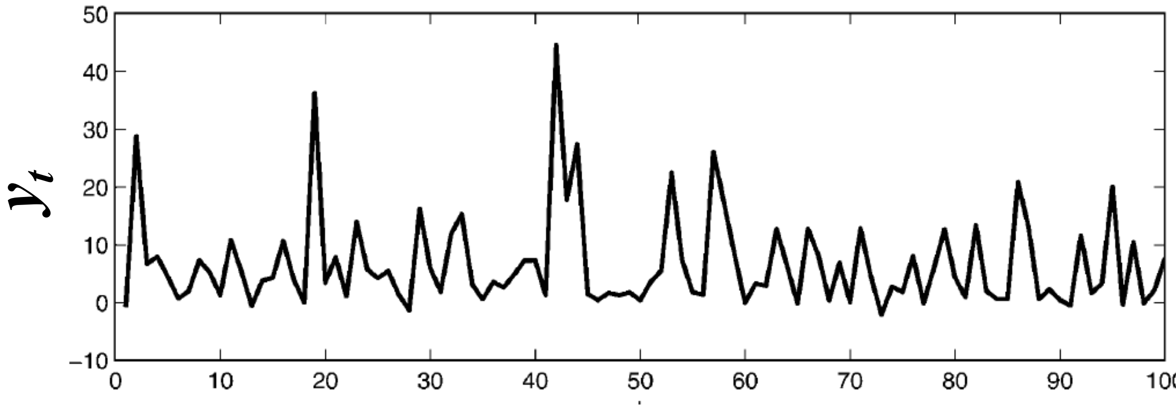
Nonlinear dynamics and observation model…

Dynamics



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

Measurement

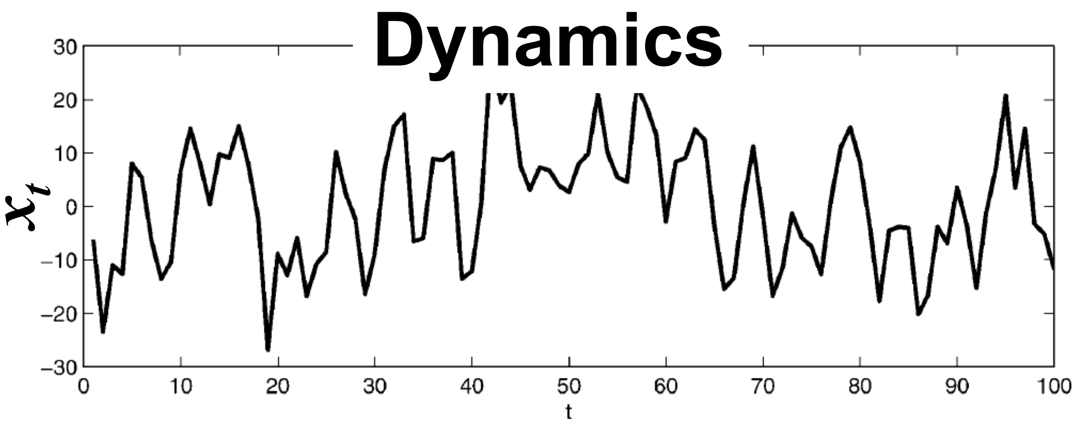


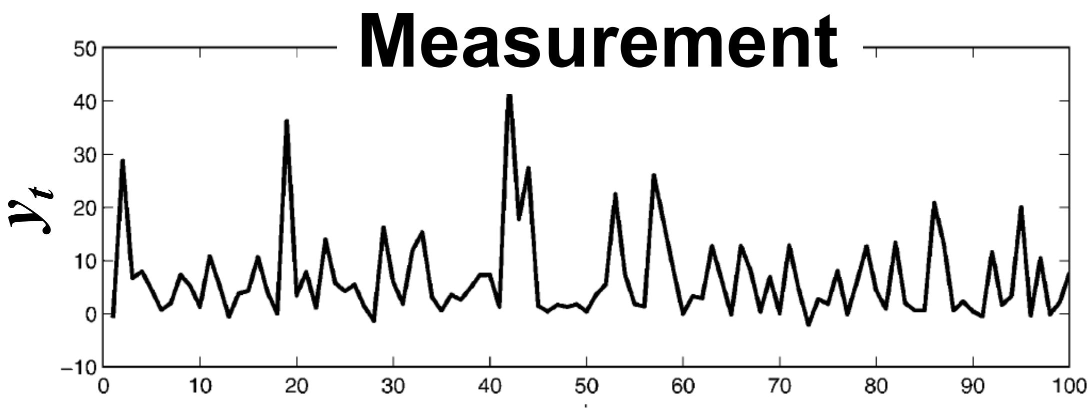
$$y_t = \frac{x_t^2}{20} + v_t$$

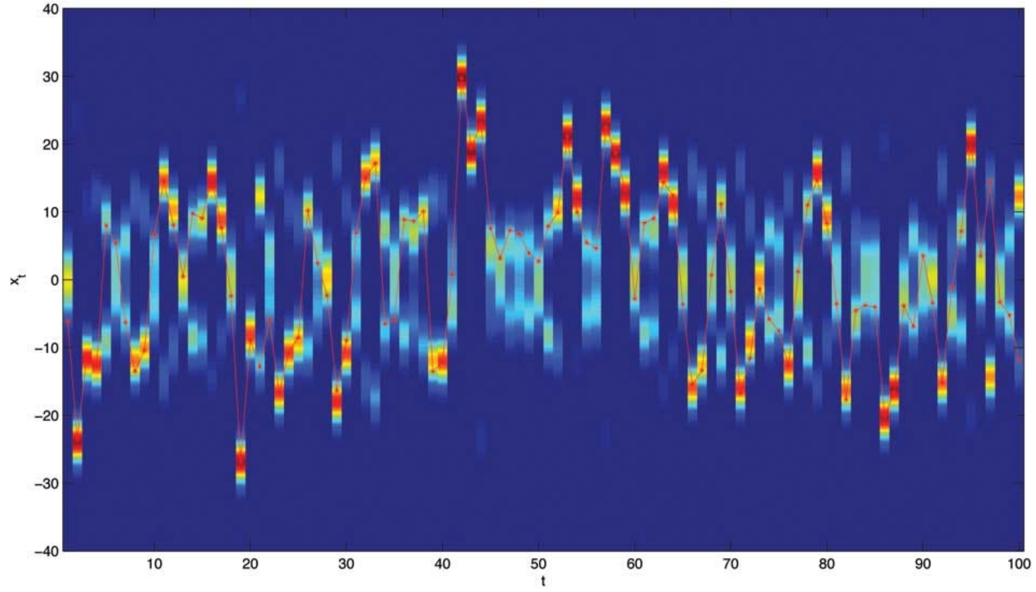
Gaussian noise model, $u_t \sim \mathcal{N}(0, \sigma_x^2)$ and $v_t \sim \mathcal{N}(0, \sigma_y^2)$

…filter equations lack closed form.

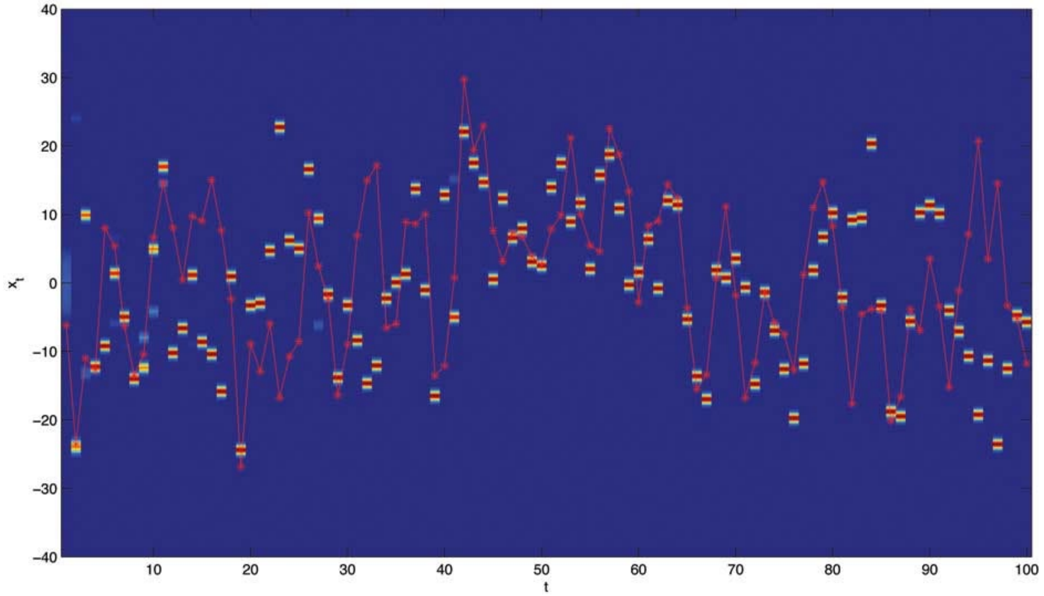
Toy Nonlinear Model







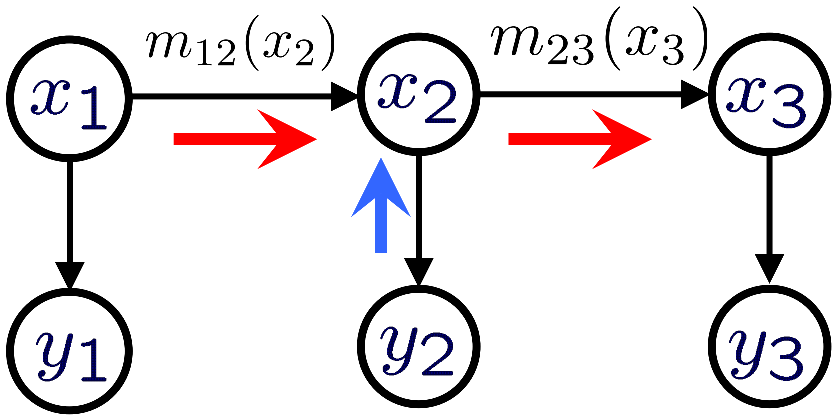
Particle Filter Marginal KDEs



Full Sequence Importance Sampling

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:

$$p(x_{t-1} \mid y_{t-1}, \ldots, y_1) \approx \sum_{\ell=1}^{L} \frac{1}{L} \delta_{x_{t-1}^{(\ell)}}(x_{t-1})$$

- Sample from a *proposal distribution q*:

$$x_t^{(\ell)} \sim q(x_t \mid x_{t-1}^{(\ell)}, y_t) \approx p(x_t \mid x_{t-1}^{(\ell)}, y_t)$$

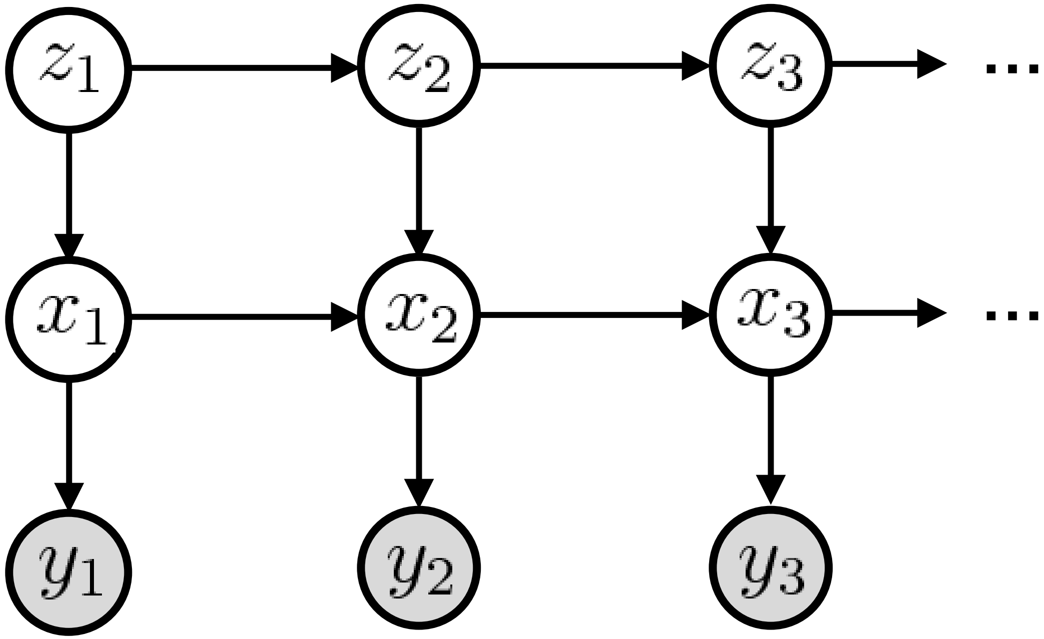
- Account for *observation and proposal* via importance weights:

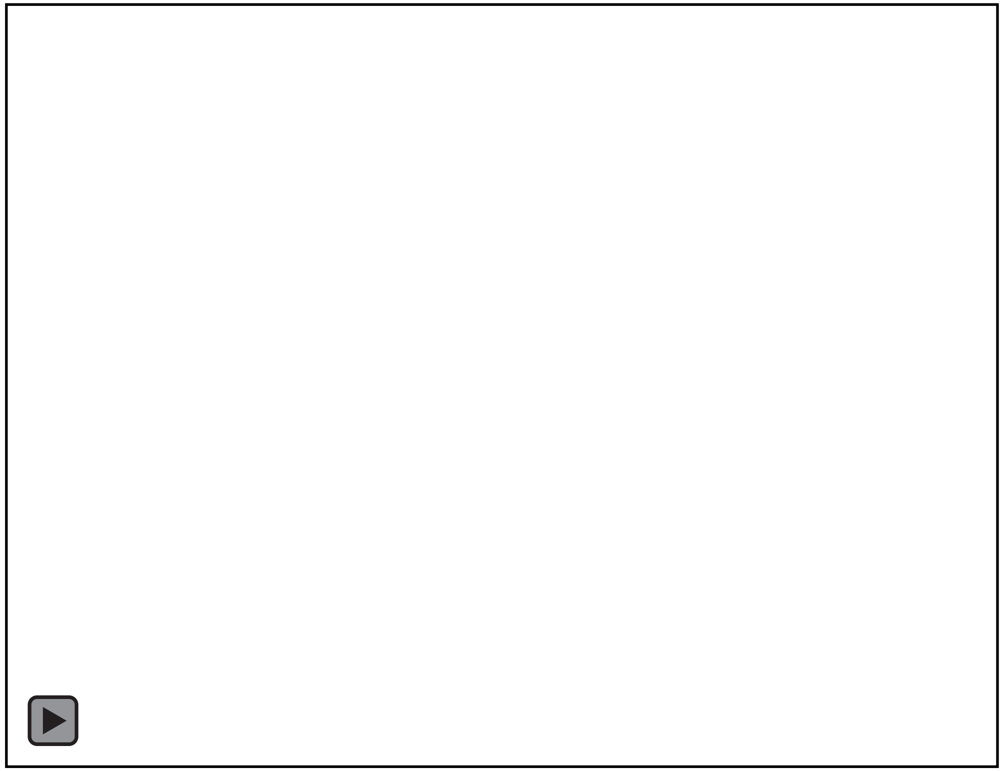
$$w_t^{(\ell)} \propto \frac{p(x_t^{(\ell)} \mid x_{t-1}^{(\ell)}) p(y_t \mid x_t^{(\ell)})}{q(x_t^{(\ell)} \mid x_{t-1}^{(\ell)}, y_t)}$$

- Resample to avoid particle degeneracy:

$$p(x_t \mid y_t, \ldots, y_1) \approx \sum_{\ell=1}^{L} \frac{1}{L} \delta_{x_t^{(\ell)}}(x_t) \qquad x_t^{(\ell)} \sim \sum_{m=1}^{L} w_t^{(m)} \delta_{x_t^{(m)}}(x_t)$$

Switching State-Space Model





Colors indicate 3 writing modes

[Video: Isard & Blake, ICCV 1998.]

Discrete switching state:

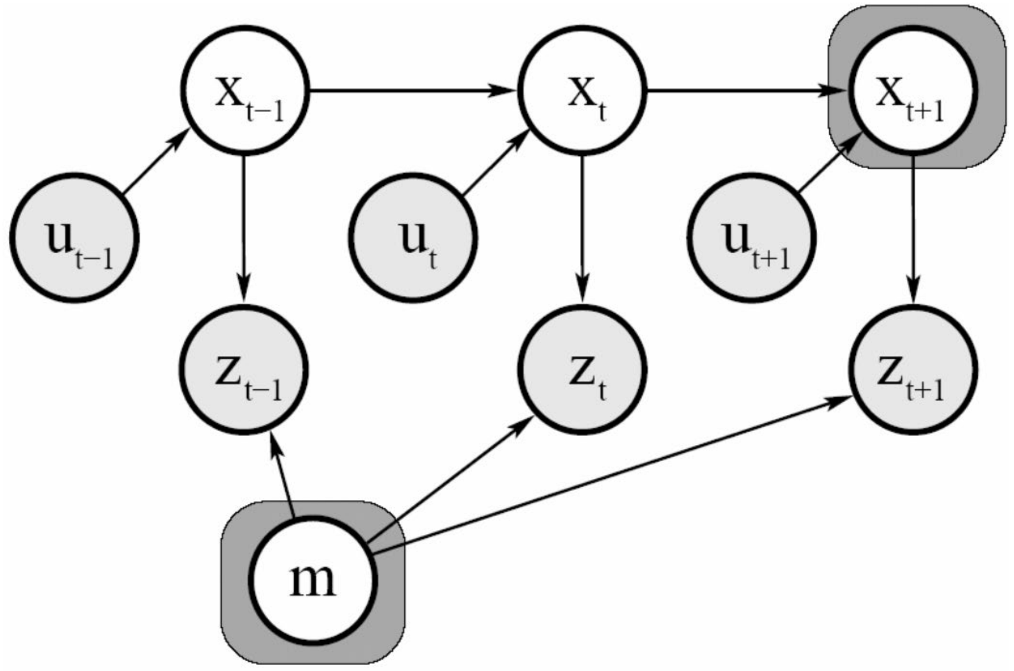
$z_t \mid z_{t-1} \sim \mathrm{Cat}(\pi(z_{t-1}))$ With stochastic transition matrix π

Switching state selects dynamics:

$x_t \mid x_{t-1} \sim \mathcal{N}(A_{z_t} x_{t-1}, \Sigma_{z_t})$ (e.g. Nonlinear Gaussian)

Example: Particle Filters for SLAM

Simultaneous Localization & Mapping (FastSLAM, Montemerlo 2003)



$$p(x_t, m \mid z_{1:t}, u_{1:t})$$

x_t = State of the robot at time t

m = Map of the environment

$z_{1:t}$ = Sensor inputs from time 1 to t

$u_{1:t}$ = Control inputs from time 1 to t

Raw odometry (controls)

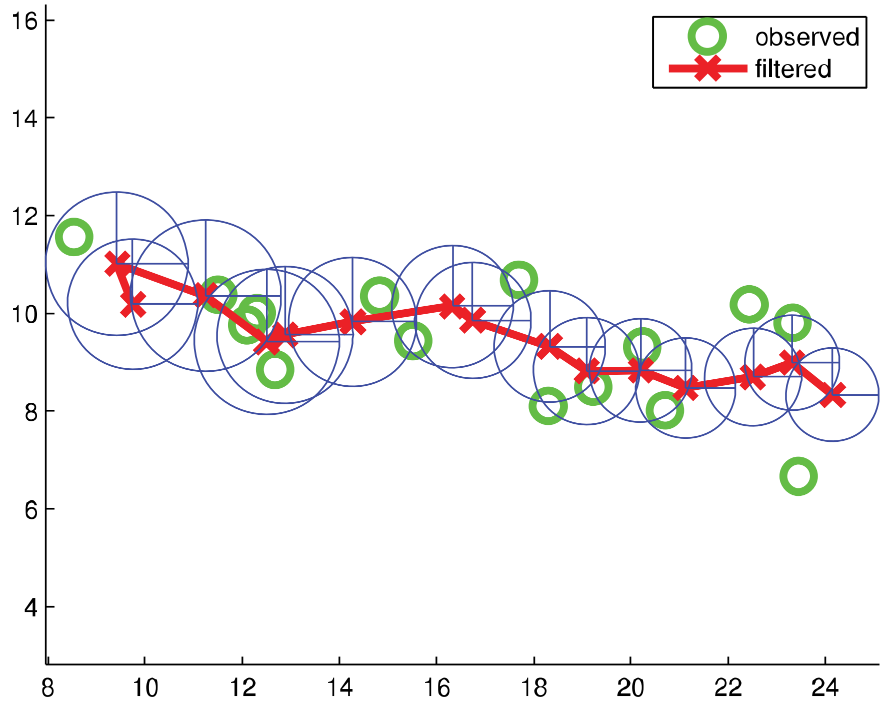
True trajectory (GPS)

Inferred trajectory & landmarks

Dynamical System Inference

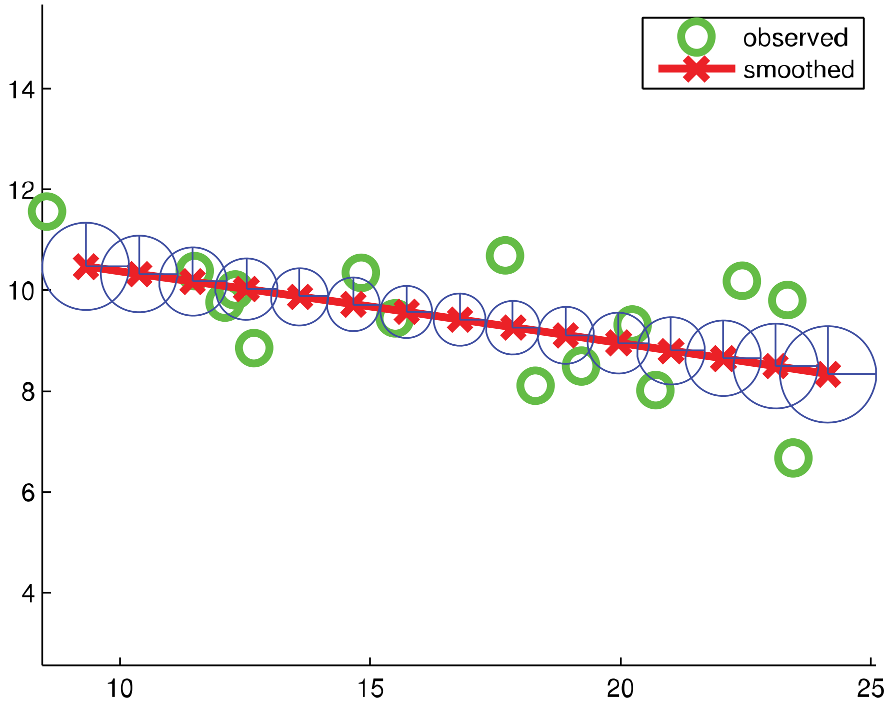
Define shorthand notation: $y_1^{t-1} \triangleq \{y_1, \ldots, y_{t-1}\}$

Filtering



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

Smoothing

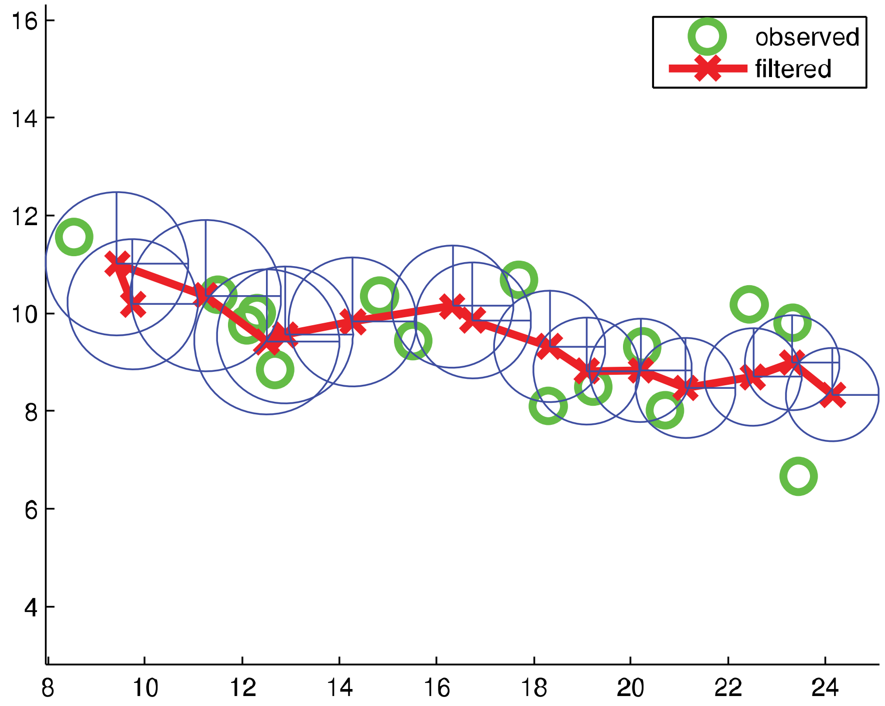


Compute full posterior marginal $p(x_t \mid y_1^T)$ at each time t

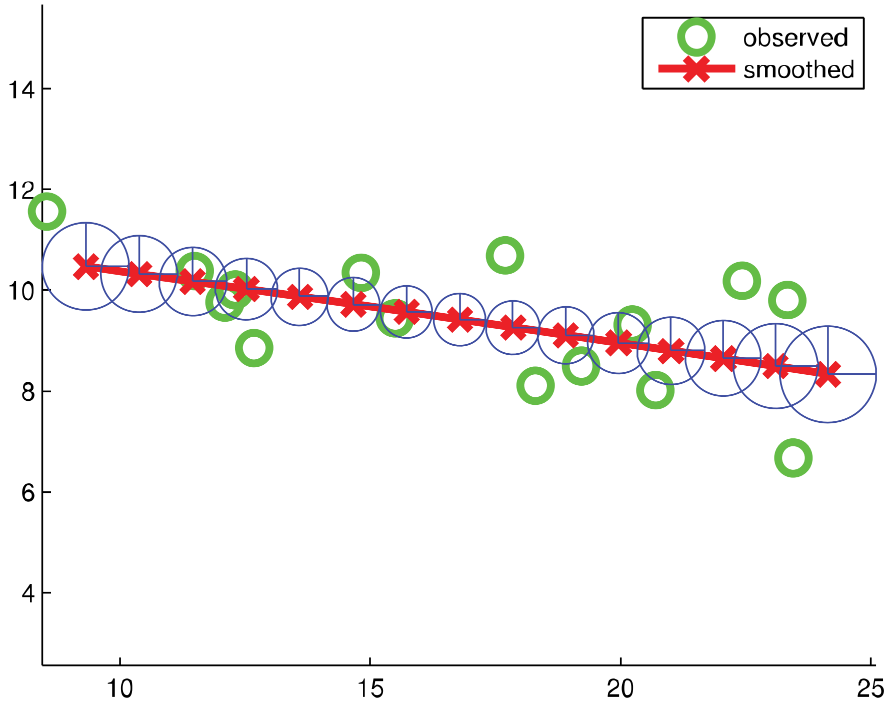
Dynamical System Inference

Define shorthand notation: $y_1^{t-1} \triangleq \{y_1, \ldots, y_{t-1}\}$

Filtering

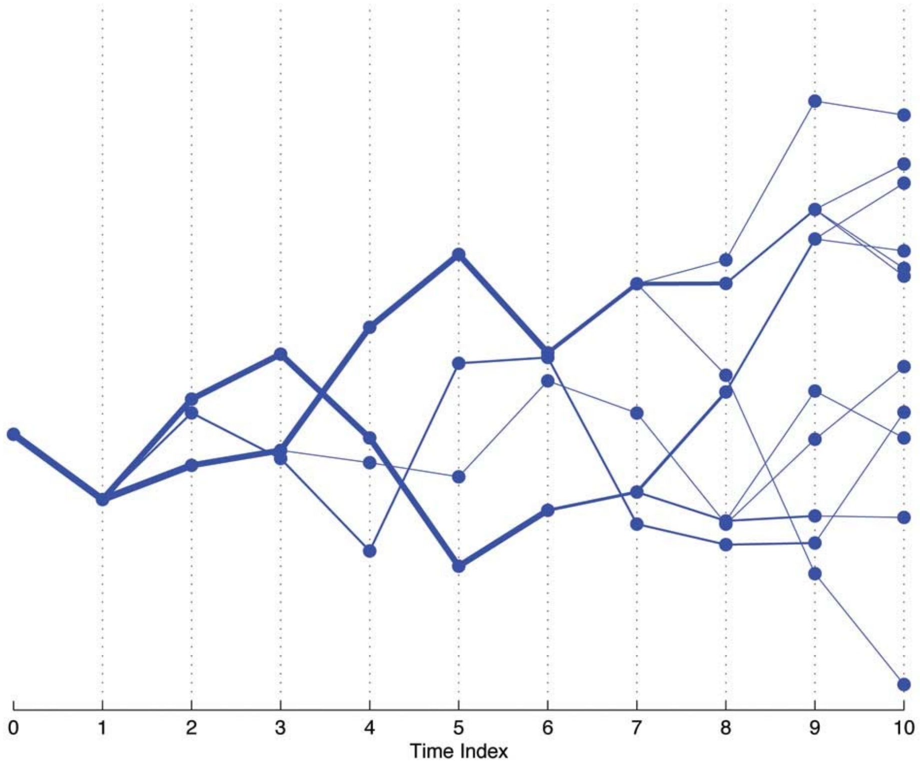


Smoothing



If estimates at time t are not needed *immediately*, then better *smoothed* estimates are possible by incorporating future observations

A Note On Smoothing



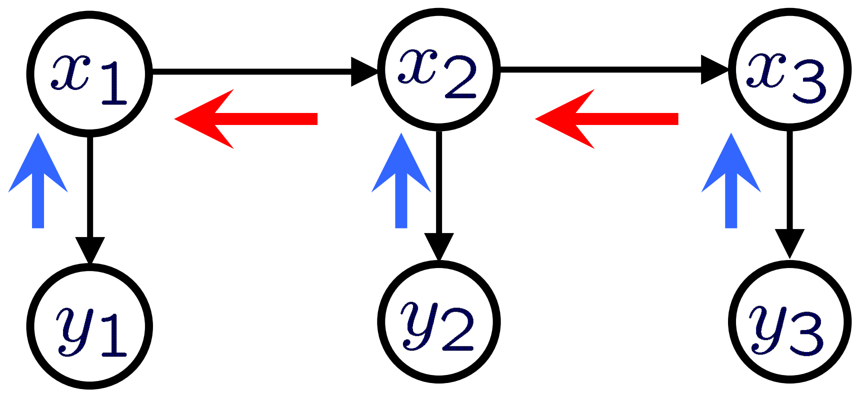
- Each resampling step discards states and they cannot subsequently restored
- Resampling introduces dependence across trajectories (common ancestors)
- Smoothed marginal estimates are generally poor
- Backwards simulation improves estimates of smoothed trajectories

Particle Filter Smoothing

Smoothing distribution factorizes as,

$$p(x_1^T \mid y_1^T) = p(x_T \mid y_1^T) \prod_{t=1}^{T-1} p(x_t \mid x_{t+1}, y_1^T)$$

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



Markov property removes dependence on $y_{t+1} \ldots y_T$

Filter distribution at time T

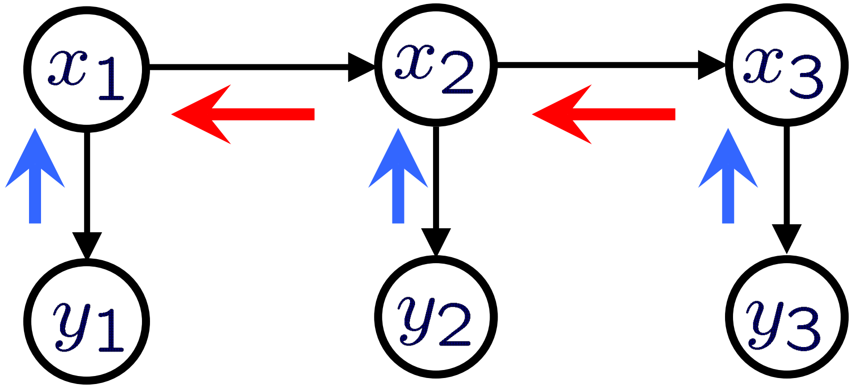
Suggests an algorithm to sample from $p(x_1^T \mid y_1^T)$:

1. Compute and store filter marginals, $p(x_t \mid y_1^t)$ for t=1,…,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),…,2,1 from, $x_t \sim p(x_t \mid x_{t+1}, y_1^t)$

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:

$$p(x_t \mid x_{t+1}, y_1^T) \approx \sum_{\ell=1}^{L} \rho_t^{(\ell)}(x_{t+1}) \delta(x_t - x_t^{(i)}) \quad \text{where} \quad \rho_t^{(i)}(x_{t+1}) = \frac{w_t^{(i)} p(x_{t+1} \mid x_t^{(i)})}{\sum_{l=1}^{L} w_t^{(l)} p(x_{t+1} \mid x_t^{(l)})}$$

Particle Filter Smoothing

Algorithm 5 Particle Smoother

for $t = 0$ to T **do** ▷ 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 $\widetilde{x}_T = x_T^{(i)}$ with probability $\omega_t^{(i)}$.

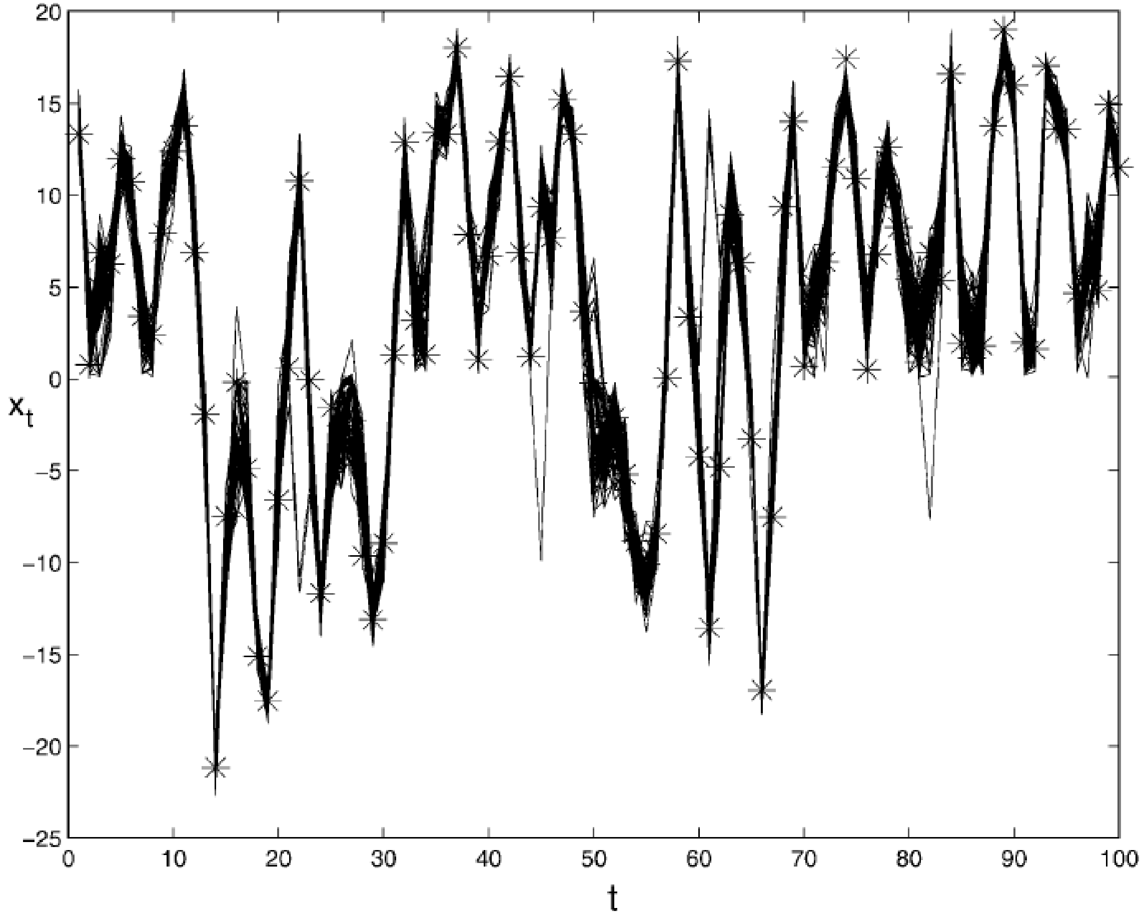
for $t = T - 1$ to 1 **do** ▷ Backward Pass Smoother

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

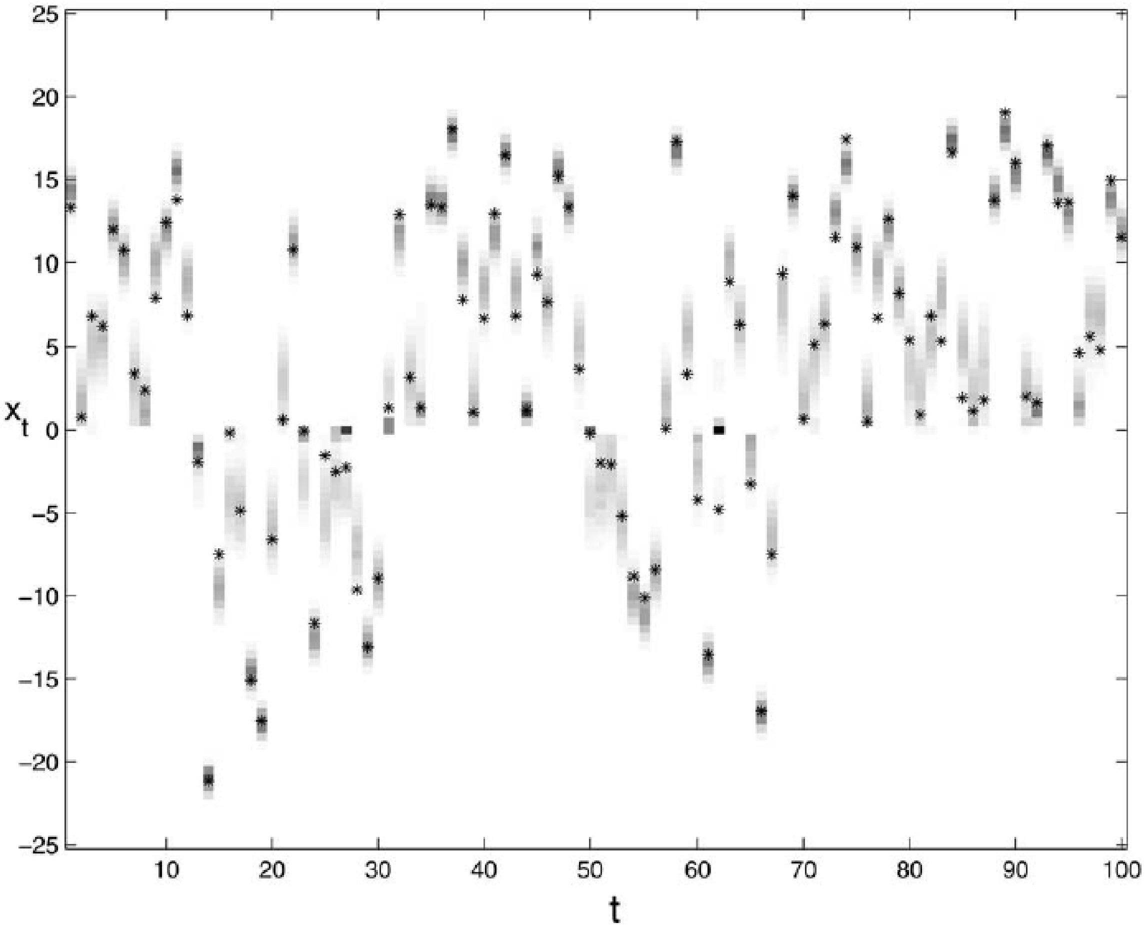
Choose $\widetilde{x}_t = x_t^{(i)}$ with probability $\rho_t^{(i)}$.

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 $p(x_1^t \mid y_1^t)$ to sample trajectory prior to the resampling step
- Other smoothing topics:
  - Generalized two-filter smoothing
  - MC approximation of posterior marginals $p(x_t \mid y_1^T)$
- *Maximum a posteriori* (MAP) particle filter
- Maximum likelihood parameter estimation using PF

Sequential Monte Carlo Summary

- Importance sampling for inference in nonlinear dynamical systems
- Using model dynamics as proposal allows recursive weight updates

$$q(x \mid y) = q(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}) \qquad\qquad w_t^{(\ell)} \propto w_{t-1}^{(\ell)} p(y_t \mid x_t^{(\ell)})$$

- All but one weight go to zero as prior/posterior diverge (degeneracy)
- Periodic resampling (with replacement) avoids weight degeneracy
- Each resampling step increases estimator variance (use sparingly)
- In practice, resample when effective sample size (ESS) below thresh

Outline

- Monte Carlo Estimation
- Sequential Monte Carlo
- Markov Chain Monte Carlo

See separate MCMC slides…

Monte Carlo Methods Summary

- Simulation: $x \sim p(x) = \frac{1}{Z} f(x)$ **Rejection sampling, MCMC**

- Compute expectations: $\mathbb{E}[\phi(x)] = \int p(x)\phi(x)\, dx$

  Importance sampling or any simulation method

- Optimization: $x^* = \arg\max_x f(x)$ **Simulated annealing**

- Compute normalizer / marginal likelihood: $Z = \int f(x)\, dx$

  Reverse importance sampling (Did not cover)

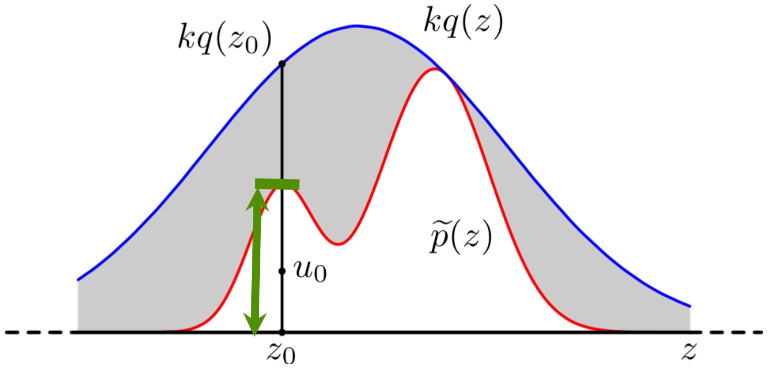
Monte Carlo Methods Summary

- In complex models we often have no other choice than to simulate realizations

- Rejection sampler choose proposal/constant s.t. $\widetilde{p}(z) \leq kq(z)$

1) Sample $q(z)$

2) Keep samples in proportion to $\frac{\tilde{p}(z)}{k \bullet q(z)}$ and reject the rest.



- Monte carlo estimate via independent samples $\{z^{(i)}\}_{i=1}^{L} \sim p$,

$$\mathbf{E}_p[f] \approx \frac{1}{L} \sum_{i=1}^{L} f(z^{(i)})$$

- Unbiased
- Consistent
- Law of large numbers
- Central limit theorem (if f is finite variance)

Monte Carlo Methods Summary

- Importance sampling estimate over samples $\{z^{(i)}\}_{i=1}^{L} \sim q$,

$$\mathbf{E}_p[f] \approx \sum_{i=1}^{L} w^{(i)} f(z^{(i)})$$

$$w^{(i)} \propto \frac{\widetilde{p}(z^{(i)})}{q(z^{(i)})}$$

Proposal

Importance Weights

- Avoids simulation of *p(z)* but variance scales exponentially with dim.
- Sequential importance sampling extends IS for sequence models, with proposal given by dynamics,

$$q(z) = q(z_0) \prod_{t=1}^{T} p(z_t \mid z_{t-1})$$

"Bootstrap" Particle Filter

$$w_t(z^{(i)}) \propto w_{t-1}(z^{(i-1)}) p(y_t \mid z_t^{(i)})$$

Recursively update weights

- **Resampling** step necessary to avoid weight degeneracy

Monte Carlo Methods Summary

- Lots of other methods to explore…
  - Hamiltonian Monte Carlo
  - Slice Sampling
  - Reversible Jump MCMC (and other *transdimensional samplers*)
  - Parallel Tempering

- Some good resources if you are interested…

Neal, R. “Probabilistic Inference Using Markov Chain Monte Carlo Methods”, U. Toronto, 1993

MacKay, D. J. “Introduction to Monte Carlo Methods”, Cambridge U., 1998

Andrieu, C., et al., “Introduction to MCMC for Machine Learning”, 2001