Markov Chain Monte Carlo for Machine Learning

Sara Beery, Natalie Bernat, and Eric Zhan

Advanced Topics in Machine Learning
California Institute of Technology

April 20th, 2017

Table of Contents

History of Monte Carlo methods

- Enrico Fermi used to calculate incredibly accurate predictions using statistical sampling methods when he had insomnia, in order to impress his friends.

History of Monte Carlo methods

- Enrico Fermi used to calculate incredibly accurate predictions using statistical sampling methods when he had insomnia, in order to impress his friends.
- Sam Ulam used sampling to approximate the probability that a solitaire layout would be successful after the combinatorics proved too difficult.

History of Monte Carlo methods

- Enrico Fermi used to calculate incredibly accurate predictions using statistical sampling methods when he had insomnia, in order to impress his friends.
- Sam Ulam used sampling to approximate the probability that a solitaire layout would be successful after the combinatorics proved too difficult.
- Both of these examples get at the heart of Monte Carlo methods: selecting a statistical sample to approximate a hard combinatorial problem by a much simpler problem.

History of Monte Carlo methods

- Enrico Fermi used to calculate incredibly accurate predictions using statistical sampling methods when he had insomnia, in order to impress his friends.
- Sam Ulam used sampling to approximate the probability that a solitaire layout would be successful after the combinatorics proved too difficult.
- Both of these examples get at the heart of Monte Carlo methods: selecting a statistical sample to approximate a hard combinatorial problem by a much simpler problem.
- These ideas have been extended to fields across science, spurred on by the development of computers

Table of Contents

Bayesian Inference and Learning

We often need to solve the following intractable problems in Bayesian Statistics, given unknown variables $x \in X$ and data $y \in Y$:

- **Normalization**: we need to compute the normalizing factor $\int_X p(y|x')p(x')dx'$ in Bayes' theorem in order to compute the posterior $p(x|y)$ given $p(x)$ and $p(y|x)$

Bayesian Inference and Learning

We often need to solve the following intractable problems in Bayesian Statistics, given unknown variables $x \in X$ and data $y \in Y$:

- **Normalization**: we need to compute the normalizing factor $\int_X p(y|x')p(x')dx'$ in Bayes' theorem in order to compute the posterior $p(x|y)$ given $p(x)$ and $p(y|x)$
- **Marginalization**: Computing the marginal posterior $p(x|y) = \int_Z p(x, z|y)dz$ given the joint posterior of $(x, z) \in X \times Z$

Bayesian Inference and Learning

We often need to solve the following intractable problems in Bayesian Statistics, given unknown variables $x \in X$ and data $y \in Y$:

- **Normalization**: we need to compute the normalizing factor $\int_X p(y|x')p(x')dx'$ in Bayes' theorem in order to compute the posterior $p(x|y)$ given $p(x)$ and $p(y|x)$
- **Marginalization**: Computing the marginal posterior $p(x|y) = \int_Z p(x, z|y)dz$ given the joint posterior of $(x, z) \in X \times Z$
- **Expectation**: Obtain the expectation $\mathbb{E}_{p(x|y)}(f(x)) = \int_Z f(x)p(x|y)dx$ for some function $f : X \to \mathbb{R}^{n_f}$, such as the conditional mean or the conditional covariance

Other problems that can be approximated with MCMC

- **Statistical mechanics**: Computing the partition function of a system can involve a large sum over possible configurations
- **Optimization**: Finding the optimal solution over a large feasible set
- **Penalized likelihood model selection**: Avoiding wasting computing resources on computing maximum likelihood estimates over a large initial set of models

Caltech

Table of Contents

Introduction to Monte Carlo

We first draw N i.i.d. samples ${x^{(i)}}_{i=1}^{N}$ from a target density $p(x)$ on a high-dimensional space X. We can use these N samples to estimate the target density $p(x)$ using a point-mass function:

$$p_N(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(x) \tag{1}$$

where $\delta_{x^{(i)}}(x)$ is the Dirac-delta mass located at $x(i)$

Introduction to Monte Carlo

This allows us to approximate integrals or large sums $I(f)$ with tractable, smaller sums $I_N(f)$. These converge by the strong law of large numbers:

$$I_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \xrightarrow[N\to\infty]{a.s.} I(f) = \int_X f(x)p(x)dx \tag{2}$$

Introduction to Monte Carlo

- Note that $I_N(f)$ is unbiased.
- If the variance of $f(x)$ satisfies $\sigma_f^2(x) = \mathbb{E}_{p(x)}(f^2(x)) - I^2(f) < \infty$, then $var(I_N(f)) = \frac{\sigma_f^2(x)}{N}$
- Bounded variance of f also guarantees convergence in distribution of the error via the central limit theorem:

$$\sqrt{N}(I_N(f) - I(f)) \underset{N\to\infty}{\Longrightarrow} \mathcal{N}(0, \sigma_f^2(x)) \tag{3}$$

Inverse Transform Sampling

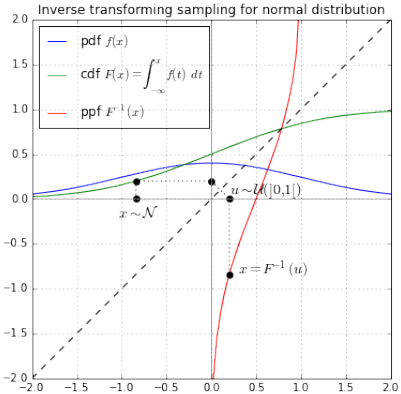
How do we sample from "easy-to-sample" distributions?

Suppose we want to sample from a distribution f with CDF F.

- Sample $u \sim Uniform[0, 1]$
- Compute $x = F^{-1}(u)$
  (This gives us the largest $x : P(-\infty < f < x) <= u)$
- Then x will be sampled according to f

Inverse Transform Sampling

Example: $\mathcal{N}(0,1)$



But for more complicated distributions, we may not have an explicit formula for $F^{-1}(u)$. In these cases we can consider other sampling methods.

Rejection Sampling

Main idea:

- It can be difficult to sample from complicated distributions
- Instead, we will sample from a simple distribution that contains our complicated distribution (an upper bound), and reject any sample that is not within the complicated distribution
- If we take enough samples, this will give us a good approximation under the Monte Carlo principle

Rejection Sampling: Sampling from a density

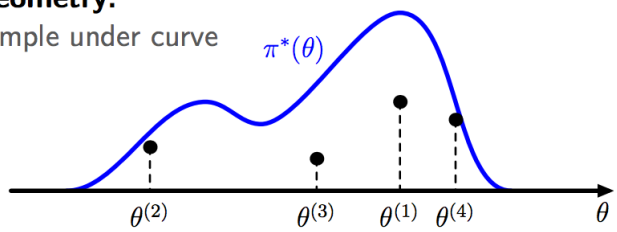
This is how we sample if we can compute the inverse CDF

Math: $A^{(s)} \sim \text{Uniform}[0,1], \quad \theta^{(s)} = \Phi^{-1}(A^{(s)})$

where cdf $\Phi(\theta) = \int_{-\infty}^{\theta} \pi(\theta')\, \mathrm{d}\theta'$

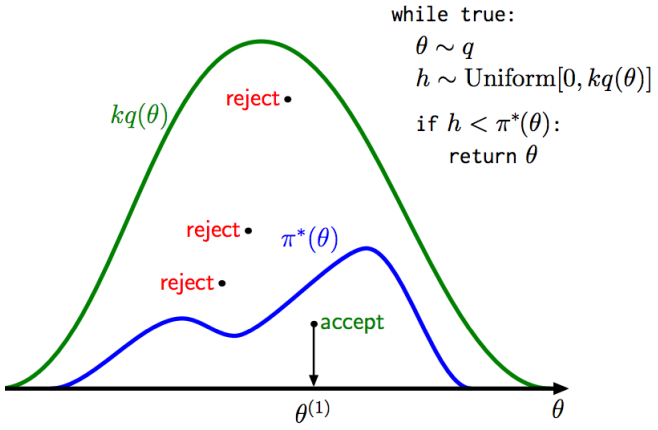
Geometry:

sample under curve

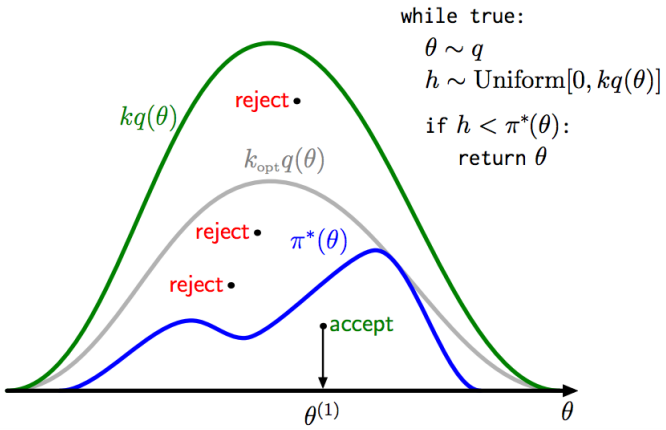


Rejection Sampling: Sampling from a density

And this is a good way to approximate if you can't

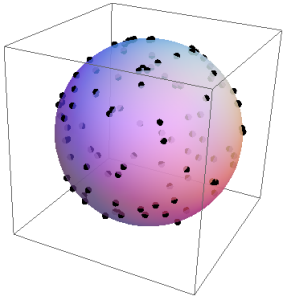


Rejection Sampling: Sampling from a density



Rejection Sampling: Drawbacks

Consider approximating the area of the unit sphere by sampling uniformly from the simplex and rejecting any sample not inside the sphere. In low dimensions, this is fine, but as the number of dimensions gets large, the volume of the unit sphere goes to zero whereas the volume of the simplex stays one, resulting in an intractable number of rejected samples.



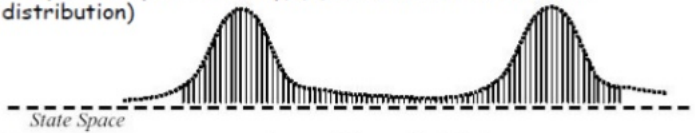
Rejection sampling is impractical in high dimensions!

Importance Sampling

- Importance sampling is used to estimate properties of a particular distribution of interest. Essentially we are transforming a difficult integral into an expectation over a simpler proposal distribution.
- Relies on drawing samples from a proposal distribution in order to develop the approximation
- Convergence time depends heavily on an appropriate choice of proposal distribution.

Importance Sampling: visual example

Sample from prior belief q(x) (for instance, the uniform distribution)



Compute importance weights, w(x) = p(x) /q(x)

Resample particles according to importance weights to get p(x)
Samples with high weights chosen many times; density reflects pdf

Importance Sampling

If we choose proposal distribution $q(x)$ such that $supp(p(x)) \subset supp(q(x))$ then we can write

$$I(f) = \int f(x)w(x)dx \tag{4}$$

where $w(x) = \frac{p(x)}{q(x)}$ is known as the **importance weight**

Importance Sampling

If we can simulate N i.i.d. samples ${x^{(i)}}_{i+1}^{N}$ from $q(x)$ and evaluate $w(x^{(i)})$ then we can approximate $I(f)$ as

$$I_N(f) = \sum_{i=1}^{N} f(x^{(i)}) w(x^{(i)}) \tag{5}$$

which converges to $I(f)$ by the strong law of large numbers

Importance Sampling

- Choosing an appropriate $q(x)$ is very important. In importance sampling, we can choose an optimal q by finding one that will minimize the variance of $I_N(f)$
- By sampling from areas of high "importance" we can achieve very high sampling efficiency
- It can be hard to find a candidate q in high dimensions. One strategy is to use **adaptive importance sampling** which parametrizes q.

Table of Contents

Markov Chain Monte Carlo

Main Idea:

- Given a target distribution p, construct a Markov chain over the sample space χ with invariant distribution equal to p.
- Perform a random walk and a sample at each iteration.
- Use the sample histogram to approximate the target distribution p.
- This is called the **Metropolis-Hastings** algorithm.

Markov Chain Monte Carlo

Main Idea:

- Given a target distribution p, construct a Markov chain over the sample space χ with invariant distribution equal to p.
- Perform a random walk and a sample at each iteration.
- Use the sample histogram to approximate the target distribution p.
- This is called the **Metropolis-Hastings** algorithm.

Two important things to note for MCMC:

- We assume we can evaluate $p(x)$ for any $x \in \chi$.
- We only need to know $p(x)$ up to a normalizing constant.

Review: Markov chains

Suppose a stochastic process $x^{(i)}$ can only take n values in $\chi = \{x_1, x_2, ..., x_n\}$.

Then $x^{(i)}$ is a **Markov chain** if $Pr(x^{(i)} \mid x^{(i-1)}, ..., x^{(1)}) = T(x^{(i)} \mid x^{(i-1)})$.

Transition matrix T is $n \times n$ with row-sum 1 (aka stochastic matrix).

- $T_{ij} = T(x_j \mid x_i)$ "transition probability from state x_i to state x_j"

A row vector π is a **stationary distribution** if $\pi T = \pi$.

A row vector π is a (unique) **invariant/limiting distribution** if $\lim_{n\to\infty} \mu T^n = \pi$ for all initial distributions μ.

An invariant distribution is also a stationary distribution.

Review: Markov chains

Theorem: A Markov chain has an invariant distribution if it is:

- **Irreducible**: There is a positive probability to get from any state to all other states.
- **Aperiodic**: The *gcd* of the lengths of all path from a state to itself is 1, for all states x_i: $gcd\{n > 0 : Pr(x^{(n)} = x_i \mid x^{(0)} = x_i) > 0\}$

Review: Markov chains

Theorem: A Markov chain has an invariant distribution if it is:

- **Irreducible**: There is a positive probability to get from any state to all other states.
- **Aperiodic**: The *gcd* of the lengths of all path from a state to itself is 1, for all states x_i: $gcd\{n > 0 : Pr(x^{(n)} = x_i \mid x^{(0)} = x_i) > 0\}$

If we construct an irreducible and aperiodic Markov chain T such that $pT = p$ (i.e. p is a stationary distribution), then p is necessarily the invariant distribution.

This is the key behind the Metropolis-Hastings algorithm.

Reversibility / Detailed Balance Condition

How can we ensure that p is a stationary distribution of T?

Check if T satisfies the **reversibility / detailed balance** condition:

$$p(x^{(i+1)})\,T(x^{(i)} \mid x^{(i+1)}) = p(x^{(i)})\,T(x^{(i+1)} \mid x^{(i)}) \tag{6}$$

Reversibility / Detailed Balance Condition

How can we ensure that p is a stationary distribution of T?

Check if T satisfies the **reversibility / detailed balance** condition:

$$p(x^{(i+1)})\,T(x^{(i)} \mid x^{(i+1)}) = p(x^{(i)})\,T(x^{(i+1)} \mid x^{(i)}) \tag{6}$$

Summing over $x^{(i)}$ on both sides gives us:

$$p(x^{(i+1)}) = \sum_{x^{(i)}} p(x^{(i)})\,T(x^{(i+1)} \mid x^{(i)}) \tag{7}$$

This is equivalent to $pT = p$.

Discrete vs. Continuous Markov chains

Transition matrix T becomes an **integral/transition kernel** K.

Detailed balance condition is the same:

$$p(x^{(i+1)})K(x^{(i)} \mid x^{(i+1)}) = p(x^{(i)})K(x^{(i+1)} \mid x^{(i)}) \tag{8}$$

Integral instead of summation:

$$p(x^{(i+1)}) = \int_{\chi} p(x^{(i)})K(x^{(i+1)} \mid x^{(i)})dx^{(i)} \tag{9}$$

Metropolis-Hastings Algorithm

Initialize $x^{(0)}$

For i from 0 to $N-1$:

- Sample $u \sim \mathit{Uniform}[0,1]$
- Sample $x^* \sim q(x^* \mid x^{(i)})$
- If $u < A(x^{(i)}, x^*) = \min\left\{1, \frac{p(x^*)q(x^{(i)}|x^*)}{p(x^{(i)})q(x^*|x^{(i)})}\right\}$:

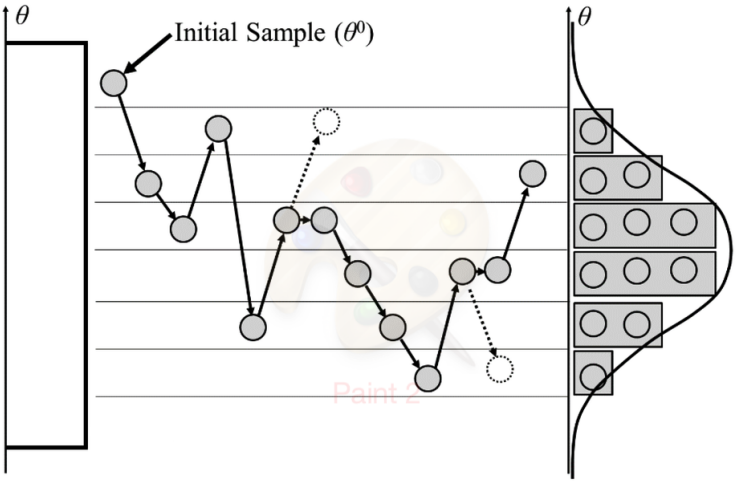
  $x^{(i+1)} = x^*$

  Else:

  $x^{(i+1)} = x^{(i)}$

q is called the **proposal distribution** and should be easy-to-sample.
$A(x^{(i)}, x^*)$ is called the **acceptance probability** of moving from $x^{(i)}$ to x^*.

Metropolis-Hastings Algorithm



Example of MH algorithm

$p(x) \propto 0.1\exp(-0.2x^2) + 0.7\exp(-0.2(x-10)^2)$ bimodal distribution

$q(x^* \mid x^{(i)}) = \mathcal{N}(x^{(i)}, 100)$

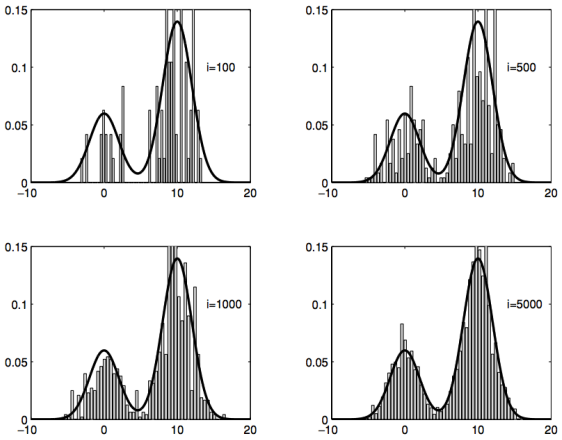


Figure 6. Target distribution and histogram of the MCMC samples at different iteration points.

Verify: detailed balance condition

The transition kernel for the MH algorithm is:

$$K_{MH}(x^{(i+1)} \mid x^{(i)}) = q(x^{(i+1)} \mid x^{(i)}) \cdot A(x^{(i)}, x^{(i+1)}) + \delta_{x^{(i)}}(x^{(i+1)}) \cdot r(x^{(i)}) \quad (10)$$

where $r(x^{(i)}) = \int_{\chi} q(x^* \mid x^{(i)})(1 - A(x^{(i)}, x^*))dx^*$ is the "rejection" term

Verify: detailed balance condition

The transition kernel for the MH algorithm is:

$$K_{MH}(x^{(i+1)} \mid x^{(i)}) = q(x^{(i+1)} \mid x^{(i)}) \cdot A(x^{(i)}, x^{(i+1)}) + \delta_{x^{(i)}}(x^{(i+1)}) \cdot r(x^{(i)}) \quad (10)$$

where $r(x^{(i)}) = \int_{\chi} q(x^* \mid x^{(i)})(1 - A(x^{(i)}, x^*))dx^*$ is the "rejection" term

We can check that K_{MH} satisfies the detailed balance condition:

$$p(x^{(i+1)})K_{MH}(x^{(i)} \mid x^{(i+1)}) = p(x^{(i)})K_{MH}(x^{(i+1)} \mid x^{(i)}) \quad (11)$$

Verify: irreducibility and aperiodicity

Irreducibility holds as long as the support of q includes the support of p.

Aperiodicity holds by construction. By allowing for rejection, we are creating self-loops in our Markov chain.

Disadvantages of MH algorithm

Samples are not independent

- Nearby samples are correlated.
- Can use "**thinning**" by sampling every d steps, but this is often unnecessary.
- Just take more samples.
- Theoretically, the sample histogram is an unbiased estimator for p.

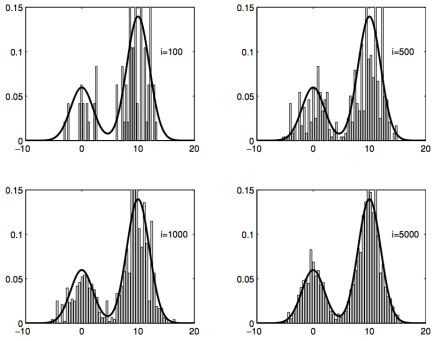
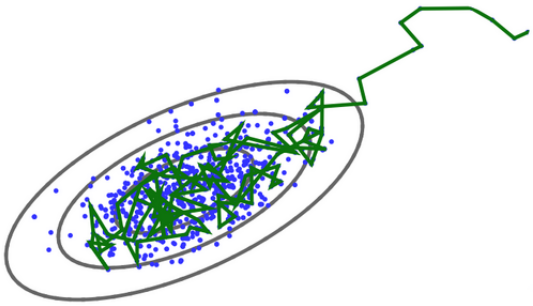


Figure 6. Target distribution and histogram of the MCMC samples at different iteration points.

Disadvantages of MH algorithm

Initial samples are biased

- Starting state not chosen according to p, meaning the random walk might start in a low density region of p.
- We can employ a **burn-in** period, where the first few samples are discarded.
- How many samples to discard depends on the **mixing time** of the Markov chain.



Convergence of Random Walk

How many time-steps do we need to converge to p?

- Theoretically, infinite.
- But being "close" is often a good enough approximation.

Convergence of Random Walk

How many time-steps do we need to converge to p?

- Theoretically, infinite.
- But being "close" is often a good enough approximation.

Total Variation norm:

- Finite case: $\delta(q,p) = \frac{1}{2}\sum_{x\in\chi} |q(x) - p(x)|$
- General case: $\delta(q,p) = \sup_{A\subseteq\chi} |q(A) - p(A)|$

ϵ-**mixing time**: $\tau_x(\epsilon) = \min\{t : \delta(K_x^{(t')}, p) \leq \epsilon, \quad \forall t' > t\}$

Convergence of Random Walk

How many time-steps do we need to converge to p?

- Theoretically, infinite.
- But being "close" is often a good enough approximation.

Total Variation norm:

- Finite case: $\delta(q,p) = \frac{1}{2}\sum_{x\in\chi}|q(x)-p(x)|$
- General case: $\delta(q,p) = \sup_{A\subseteq\chi}|q(A)-p(A)|$

ϵ-**mixing time**: $\tau_x(\epsilon) = \min\{t : \delta(K_x^{(t')}, p) \leq \epsilon, \quad \forall t' > t\}$

Mixing times of a Markov chain can be bounded by its second eigenvalue and its conductance (measure of connectivity).

CS139 Advanced Algorithms covers this in great detail.

Other MCMC algorithms

Many MCMC algorithms are just special cases of the MH algorithm.

Independent sampler

- Assume $q(x^* \mid x^{(i)}) = q(x^*)$, independent of current state
- $A(x^{(i)}, x^*) = \min\left\{1, \frac{p(x^*)q(x^{(i)})}{p(x^{(i)})q(x^*)}\right\} = \min\left\{1, \frac{w(x^*)}{w(x^{(i)})}\right\}$

Metropolis algorithm

- Assume $q(x^* \mid x^{(i)}) = q(x^{(i)} \mid x^*)$, symmetric random walk
- $A(x^{(i)}, x^*) = \min\left\{1, \frac{p(x^*)}{p(x^{(i)})}\right\}$

Other MCMC algorithms

Simulated Annealing

Monte Carlo EM

Hybrid Monte Carlo

Slice Sampler

Reversible jump MCMC

Gibbs Sampler

Gibbs Sampler

Assume each x is an n-dimensional vector and we know the full conditional probabilities: $p(x_j \mid x_1, ..., x_{j-1}, x_{j+1}, ..., x_n) = p(x_j \mid x_{-j})$. Furthermore, assume the full conditional probabilities are easy-to-sample.

Then we can choose proposal distribution for each j:

$$q(x^* \mid x^{(i)}) = \begin{cases} p(x_j^* \mid x_{-j}^{(i)}) & \text{if } x_{-j}^* = x_{-j}^{(i)}, \text{ meaning all other coordinates match} \\ 0 & \text{otherwise} \end{cases}$$

With acceptance probability:

$$A(x^{(i)}, x^*) = \min\left\{1, \frac{p(x^*)p(x_j^{(i)} | x_{-j}^{(i)})}{p(x^{(i)})p(x_j^* | x_{-j}^{(i)})}\right\} = \min\left\{1, \frac{p(x_{-j}^*)}{p(x_{-j}^{(i)})}\right\} = 1$$

Gibbs Sampler

At each iteration, we are changing one coordinate of our current state. We can choose which coordinate to change randomly, or we can go in order.

Gibbs Sampler

At each iteration, we are changing one coordinate of our current state. We can choose which coordinate to change randomly, or we can go in order.

For directed graphical models (aka Bayesian networks), we can write the full conditional probability of x_j in terms of its **Markov blanket**:

$$p(x_j \mid x_{-j}) = p(x_j \mid x_{parent(j)}) \prod_{k \in children(j)} p(x_k \mid x_{parent(k)}) \tag{12}$$

Recall: the Markov blanket $MB(x_j)$ of x_j is the minimal set of nodes such that x_j is independent from the rest of the graph if $MB(x_j)$ is observed. For directed graphical models, $MB(x_j) = \{parents(x_j), children(x_j), parents(children(x_j))\}$.

Table of Contents

Sequential MC and Particle Filters

Sequential MC methods provide **online** approximations of probability distributions for inherently **sequential data**

Sequential MC and Particle Filters

Sequential MC methods provide **online** approximations of probability distributions for inherently **sequential data**

- **Assumptions**:
  - Initial distribution: $p(x_0)$
  - Dynamic model: $p(x_t|x_{0:t-1}, y_{1:t-1})$ (for t$\geq$1)
  - Measurement model: $p(y_t|x_{0:t}, y_{1:t-1})$ (for t$\geq$1)

Sequential MC and Particle Filters

Sequential MC methods provide **online** approximations of probability distributions for inherently **sequential data**

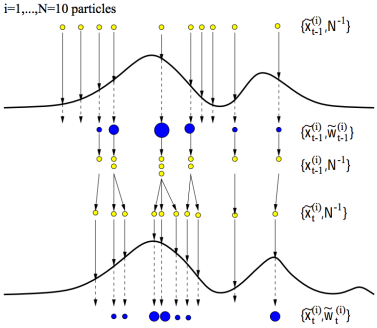
- **Assumptions**:
  - Initial distribution: $p(x_0)$
  - Dynamic model: $p(x_t|x_{0:t-1}, y_{1:t-1})$ (for t$\geq$1)
  - Measurement model: $p(y_t|x_{0:t}, y_{1:t-1})$ (for t$\geq$1)
- **Aim**: estimate the posterior $p(x_{0:t}|y_{1:t})$, including the *filtering distribution*, $p(x_t|y_{1:t})$, and sample N particles $\{x_{0:t}^{(i)}\}_{i=1}^{N}$ from that distribution

Sequential MC and Particle Filters

Sequential MC methods provide **online** approximations of probability distributions for inherently **sequential data**

- **Assumptions**:
  - Initial distribution: $p(x_0)$
  - Dynamic model: $p(x_t|x_{0:t-1}, y_{1:t-1})$ (for t$\geq$1)
  - Measurement model: $p(y_t|x_{0:t}, y_{1:t-1})$ (for t$\geq$1)
- **Aim**: estimate the posterior $p(x_{0:t}|y_{1:t})$, including the *filtering distribution*, $p(x_t|y_{1:t})$, and sample N particles $\{x_{0:t}^{(i)}\}_{i=1}^N$ from that distribution
- **Method**: Generate a weighted set of paths $\{\tilde{x}_{0:t}^{(i)}\}_{i=1}^N$, using an importance sampling framework with appropriate proposal distribution $q(\tilde{x}_{0:t}|y_{1:t})$

Sequential MC and Particle Filters



Sequential importance sampling step

– For $i = 1, ..., N$, sample from the transition priors

$$\tilde{x}_t^{(i)} \sim q_t\left(\tilde{x}_t | x_{0:t-1}^{(i)}, y_{1:t}\right)$$

and set

$$\tilde{x}_{0:t}^{(i)} \triangleq \left(\tilde{x}_t^{(i)}, x_{0:t-1}^{(i)}\right)$$

– For $i = 1, ..., N$, evaluate and normalize the importance weights

$$w_t^{(i)} \propto \frac{p\left(y_t | \tilde{x}_t^{(i)}\right) p\left(\tilde{x}_t^{(i)} | x_{0:t-1}^{(i)}, y_{1:t-1}\right)}{q_t\left(\tilde{x}_t^{(i)} | x_{0:t-1}^{(i)}, y_{1:t}\right)}.$$

Selection step

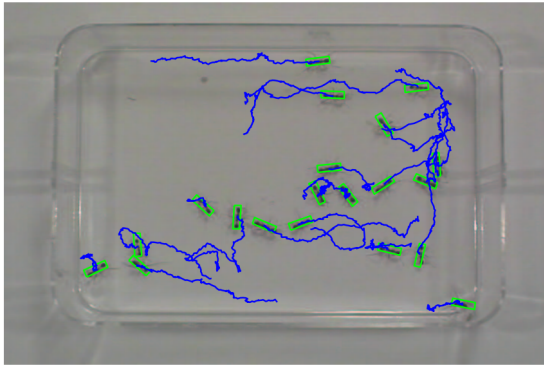
– Multiply/Discard particles $\left\{\tilde{x}_{0:t}^{(i)}\right\}_{i=1}^{N}$ with high/low importance weights $w_t^{(i)}$ to obtain N particles $\left\{x_{0:t}^{(i)}\right\}_{i=1}^{N}$.

Here, $q(\tilde{x}_{0:t}|y_{1:t}) = p(x_{0:t-1}|y_{1:t-1})q(\tilde{x}|x_{0:t-1}, y_{1:t})$

Vision Application

Tracking Multiple Interacting Targets with an MCMC-based Particle Filter

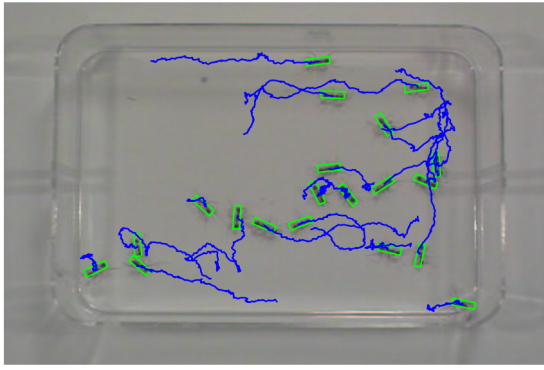
- **Aim**: Generate sequence of states X_{it}, for the i'th ant at frame t
- **Idea**: Use an independent particle filter for each ant



Vision Application

Tracking Multiple Interacting Targets with an MCMC-based Particle Filter

- **Particle Filter Assumptions**:
  - **Motion Model**: Normally distributed, centered at location in previous frame X_{t-1}
  - **Measurement Model**: Image-comparison function ({how ant-like} - {how background-like})



Vision Application

Independent Particle Filters

- $P(X_{it}|Z^{t-1}) \approx \{X_{i,t-1}^{(r)}, \pi_{i,t-1}^{(r)}\}_{r=1}^{N}$

Vision Application

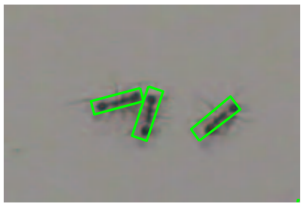
Independent Particle Filters

- $P(X_{it}|Z^{t-1}) \approx \{X_{i,t-1}^{(r)}, \pi_{i,t-1}^{(r)}\}_{r=1}^{N}$
- **Proposal distribution**: $q(X_{it}) = \sum_r \pi_{i,t-1}^{(r)} P(X_{i,t}|X_{i,t-1}^{(r)})$
- where $\pi_{i,t-1}^{(r)} = P(Z_{it}|X_{it}^{(r)})$

Vision Application

Independent Particle Filters

- $P(X_{it}|Z^{t-1}) \approx \{X_{i,t-1}^{(r)}, \pi_{i,t-1}^{(r)}\}_{r=1}^{N}$
- **Proposal distribution**: $q(X_{it}) = \sum_r \pi_{i,t-1}^{(r)} P(X_{i,t}|X_{i,t-1}^{(r)})$
- where $\pi_{i,t-1}^{(r)} = P(Z_{it}|X_{it}^{(r)})$



(a) frame 9043

(b) frame 9080

(c) 9083

Vision Application

Markov Random Field (MRF) Model & the Joint MRF Particle Filter

- Modify motion model to include pairwise interactions:
  - $P(X_t|X_{t-1}) \sim \prod_i P(X_{it}|X_{i,t-1}) \prod_{ij \in E} \psi(X_{it}, X_{jt})$
- **Proposal distribution (same as before!)**:
  - $q(X_t) = \sum_r \pi_{t-1}^{(r)} \prod_i P(X_{i,t}|X_{i,t-1}^{(r)})$
  - but now $\pi_{t-1}^{(r)} = \prod_{i=1}^n P(Z_{it}|X_{it}^{(r)}) \prod_{ij \in E} \psi(X_{i,t}^{(r)}, X_{j,t}^r)$

Vision Application

MCMC-based Particle Filter

- Because of pairwise interactions, Joint MRF Particle Filter scales badly with number of targets (ants)
- Intractable instance of importance sampling... smells like a good time to use an MCMC algorithm!

Vision Application

Tracking Multiple Interacting Targets with an MCMC-based Particle Filter

- Use Metropolis-Hastings instead of importance sampling
- Proposal Density: Simply the Motion Model

Table 1: Tracker failures observed in the 10,400 frame test sequence

Tracker	Proposal Density	Number of Samples	Number of Failu
MCMC	1	50	123
MCMC	1	100	49
MCMC	1	200	28
MCMC	1	1000	16
MCMC	2	50	124
MCMC	2	100	42
MCMC	2	200	27
MCMC	2	1000	16
single particle filter	n/a	10 per target	148
single particle filter	n/a	50 per target	125
single particle filter	n/a	100 per target	119
joint particle filter	n/a	50	544
joint particle filter	n/a	100	519
joint particle filter	n/a	200	479
joint particle filter	n/a	1000	392

References

- http://www.cs.ubc.ca/~arnaud/andrieu_defreitas_doucet_jordan_intromontecarlomachinelearning.pdf
- https://smartech.gatech.edu/bitstream/handle/1853/3246/03-35.pdf?sequence=1&isAllowed=y
- https://media.nips.cc/Conferences/2015/tutorialslides/nips-2015-monte-carlo-tutorial.pdf
- https://en.wikipedia.org/wiki/Inverse_transform_sampling
- http://www.mdpi.com/1996-1073/8/6/5538/htm