CSci 8980: Advanced Topics in Graphical Models

MCMC, Gibbs Sampling

Instructor: Arindam Banerjee

September 27, 2007

Problems

- Primarily of two types: Integration and Optimization

Problems

- Primarily of two types: Integration and Optimization
- Bayesian inference and learning

Problems

- Primarily of two types: Integration and Optimization
- Bayesian inference and learning
  - Computing normalization in Bayesian methods

$$p(y|x) = \frac{p(y)p(x|y)}{\int_{y'} p(y')p(x|y')dy'}$$

Problems

- Primarily of two types: Integration and Optimization
- Bayesian inference and learning
  - Computing normalization in Bayesian methods

$$p(y|x) = \frac{p(y)p(x|y)}{\int_{y'} p(y')p(x|y')dy'}$$

  - Marginalization: $p(y|x) = \int_z p(y,z|x)dz$

Problems

- Primarily of two types: Integration and Optimization
- Bayesian inference and learning
  - Computing normalization in Bayesian methods

$$p(y|x) = \frac{p(y)p(x|y)}{\int_{y'} p(y')p(x|y')dy'}$$

  - Marginalization: $p(y|x) = \int_z p(y, z|x)dz$
  - Expectation:

$$E_{y|x}[f(y)] = \int_y f(y)p(y|x)dy$$

Problems

- Primarily of two types: Integration and Optimization
- Bayesian inference and learning
  - Computing normalization in Bayesian methods

$$p(y|x) = \frac{p(y)p(x|y)}{\int_{y'} p(y')p(x|y')dy'}$$

  - Marginalization: $p(y|x) = \int_z p(y, z|x)dz$
  - Expectation:

$$E_{y|x}[f(y)] = \int_y f(y)p(y|x)dy$$

- Statistical mechanics: Computing the partition function

$$Z = \sum_s \exp\left[-\frac{E(s)}{kT}\right]$$

Problems

- Primarily of two types: Integration and Optimization
- Bayesian inference and learning
  - Computing normalization in Bayesian methods

$$p(y|x) = \frac{p(y)p(x|y)}{\int_{y'} p(y')p(x|y')dy'}$$

  - Marginalization: $p(y|x) = \int_z p(y, z|x)dz$
  - Expectation:

$$E_{y|x}[f(y)] = \int_y f(y)p(y|x)dy$$

- Statistical mechanics: Computing the partition function

$$Z = \sum_s \exp\left[-\frac{E(s)}{kT}\right]$$

- Optimization, Model Selection, etc.

Monte Carlo Principle

- Target density $p(x)$ on a high-dimensional space

Monte Carlo Principle

- Target density $p(x)$ on a high-dimensional space
- Draw i.i.d. samples $\{x_i\}_{i=1}^n$ from $p(x)$

Monte Carlo Principle

- Target density $p(x)$ on a high-dimensional space
- Draw i.i.d. samples $\{x_i\}_{i=1}^{n}$ from $p(x)$
- Construct empirical point mass function

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}(x)$$

Monte Carlo Principle

- Target density $p(x)$ on a high-dimensional space
- Draw i.i.d. samples $\{x_i\}_{i=1}^n$ from $p(x)$
- Construct empirical point mass function

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i}(x)$$

- One can approximate integrals/sums by

$$I_n(f) = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \xrightarrow[n\to\infty]{a.s.} I(f) = \int_x f(x)p(x)dx$$

Monte Carlo Principle

- Target density $p(x)$ on a high-dimensional space
- Draw i.i.d. samples $\{x_i\}_{i=1}^n$ from $p(x)$
- Construct empirical point mass function

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}(x)$$

- One can approximate integrals/sums by

$$I_n(f) = \frac{1}{n} \sum_{i=1}^{n} f(x_i) \xrightarrow[n \to \infty]{a.s.} I(f) = \int_x f(x) p(x) dx$$

- Unbiased estimate $I_n(f)$ converges by strong law

Monte Carlo Principle

- Target density $p(x)$ on a high-dimensional space
- Draw i.i.d. samples $\{x_i\}_{i=1}^n$ from $p(x)$
- Construct empirical point mass function

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}(x)$$

- One can approximate integrals/sums by

$$I_n(f) = \frac{1}{n} \sum_{i=1}^{n} f(x_i) \xrightarrow[n \to \infty]{a.s.} I(f) = \int_x f(x) p(x) dx$$

- Unbiased estimate $I_n(f)$ converges by strong law
- For finite σ_f^2, central limit theorem implies

$$\sqrt{n}(I_n(f) - I(f)) \underset{n \to \infty}{\Longrightarrow} \mathcal{N}(0, \sigma_f^2)$$

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$
- $q(x)$ satisfies $p(x) \leq Mq(x), M < \infty$

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$
- $q(x)$ satisfies $p(x) \leq Mq(x), M < \infty$
- Algorithm: For $i = 1, \cdots, n$

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$
- $q(x)$ satisfies $p(x) \leq Mq(x), M < \infty$
- Algorithm: For $i = 1, \cdots, n$
  1. Sample $x_i \sim q(x)$ and $u \sim \mathcal{U}(0, 1)$

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$
- $q(x)$ satisfies $p(x) \leq Mq(x), M < \infty$
- Algorithm: For $i = 1, \cdots, n$
  1. Sample $x_i \sim q(x)$ and $u \sim \mathcal{U}(0, 1)$
  2. If $u < \frac{p(x_i)}{Mq(x_i)}$, accept x_i, else reject

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$
- $q(x)$ satisfies $p(x) \leq Mq(x), M < \infty$
- Algorithm: For $i = 1, \cdots, n$
    1. Sample $x_i \sim q(x)$ and $u \sim \mathcal{U}(0, 1)$
    2. If $u < \frac{p(x_i)}{Mq(x_i)}$, accept x_i, else reject
- Issues:

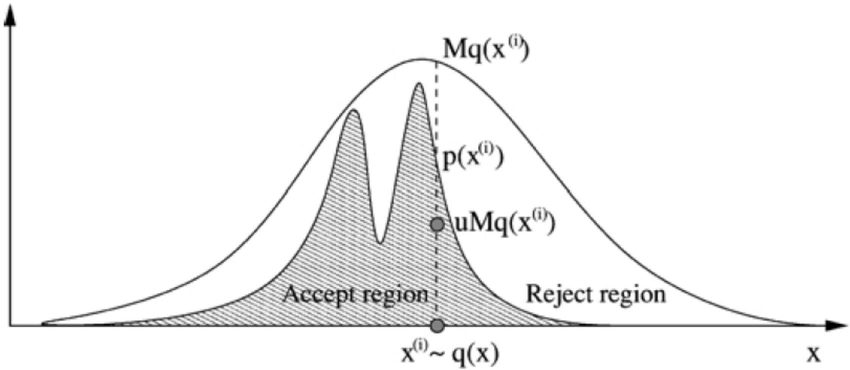
Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$
- $q(x)$ satisfies $p(x) \leq Mq(x), M < \infty$
- Algorithm: For $i = 1, \cdots, n$
  1. Sample $x_i \sim q(x)$ and $u \sim \mathcal{U}(0, 1)$
  2. If $u < \frac{p(x_i)}{Mq(x_i)}$, accept x_i, else reject
- Issues:
  - Tricky to bound $p(x)/q(x)$ with a reasonable constant M

Rejection Sampling

- Target density $p(x)$ is known, but hard to sample
- Use an easy to sample *proposal distribution* $q(x)$
- $q(x)$ satisfies $p(x) \leq Mq(x), M < \infty$
- Algorithm: For $i = 1, \cdots, n$
  1. Sample $x_i \sim q(x)$ and $u \sim \mathcal{U}(0, 1)$
  2. If $u < \frac{p(x_i)}{Mq(x_i)}$, accept x_i, else reject
- Issues:
  - Tricky to bound $p(x)/q(x)$ with a reasonable constant M
  - If M is too large, acceptance probability is small

Rejection Sampling (Contd.)



Importance Sampling

- For a proposal distribution $q(x)$, with $w(x) = p(x)/q(x)$

$$I(f) = \int_x f(x)w(x)q(x)dx$$

Importance Sampling

- For a proposal distribution $q(x)$, with $w(x) = p(x)/q(x)$

$$I(f) = \int_x f(x)w(x)q(x)dx$$

- $w(x)$ is the importance weight

Importance Sampling

- For a proposal distribution $q(x)$, with $w(x) = p(x)/q(x)$

$$I(f) = \int_x f(x)w(x)q(x)dx$$

- $w(x)$ is the importance weight
- Monte Carlo estimate of $I(f)$ based on samples $x_i \sim q(x)$

$$\hat{I}_n(f) = \sum_{i=1}^{n} f(x_i)w(x_i)$$

Importance Sampling

- For a proposal distribution $q(x)$, with $w(x) = p(x)/q(x)$

$$I(f) = \int_x f(x)w(x)q(x)dx$$

- $w(x)$ is the importance weight
- Monte Carlo estimate of $I(f)$ based on samples $x_i \sim q(x)$

$$\hat{I}_n(f) = \sum_{i=1}^{n} f(x_i)w(x_i)$$

- The estimator is unbiased, and converges to $I(f)$ a.s.

Importance Sampling (Contd.)

- Choose $q(x)$ that minimizes variance of $\hat{I}_n(f)$

$$\text{var}_q(f(x)w(x)) = E_q[f^2(x)w^2(x)] - I^2(f)$$

Importance Sampling (Contd.)

- Choose $q(x)$ that minimizes variance of $\hat{I}_n(f)$

$$\mathrm{var}_q(f(x)w(x)) = E_q[f^2(x)w^2(x)] - I^2(f)$$

- Applying Jensen's and optimizing, we get

$$q^*(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

Importance Sampling (Contd.)

- Choose $q(x)$ that minimizes variance of $\hat{I}_n(f)$

$$\text{var}_q(f(x)w(x)) = E_q[f^2(x)w^2(x)] - I^2(f)$$

- Applying Jensen's and optimizing, we get

$$q^*(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

- Efficient sampling focuses on regions of high $|f(x)|p(x)$

Importance Sampling (Contd.)

- Choose $q(x)$ that minimizes variance of $\hat{I}_n(f)$

$$\text{var}_q(f(x)w(x)) = E_q[f^2(x)w^2(x)] - I^2(f)$$

- Applying Jensen's and optimizing, we get

$$q^*(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

- Efficient sampling focuses on regions of high $|f(x)|p(x)$
- Super efficient sampling, variance lower than even $q(x) = p(x)$

Importance Sampling (Contd.)

- Choose $q(x)$ that minimizes variance of $\hat{I}_n(f)$

$$\text{var}_q(f(x)w(x)) = E_q[f^2(x)w^2(x)] - I^2(f)$$

- Applying Jensen's and optimizing, we get

$$q^*(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

- Efficient sampling focuses on regions of high $|f(x)|p(x)$
- Super efficient sampling, variance lower than even $q(x) = p(x)$
- Exploited to evaluate probability of rare events, $q(x) \propto \mathbb{I}_E(x)p(x)$

Importance Sampling (Contd.)

- Choose $q(x)$ that minimizes variance of $\hat{I}_n(f)$

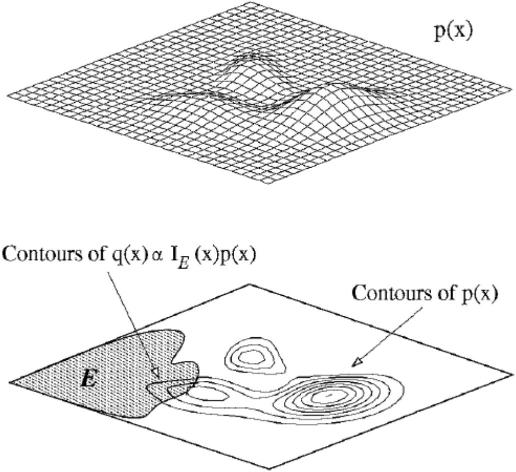
$$\mathrm{var}_q(f(x)w(x)) = E_q[f^2(x)w^2(x)] - I^2(f)$$

- Applying Jensen's and optimizing, we get

$$q^*(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

- Efficient sampling focuses on regions of high $|f(x)|p(x)$
- Super efficient sampling, variance lower than even $q(x) = p(x)$
- Exploited to evaluate probability of rare events, $q(x) \propto \mathbb{I}_E(x)p(x)$

Importance Sampling (Contd.)



Markov Chains

- Use a Markov chain to explore the state space

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1},\ldots,x_1) = T(x_i|x_{i-1})$$

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1},\ldots,x_1) = T(x_i|x_{i-1})$$

- A chain is homogenous if T is invariant for all i

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1}, \ldots, x_1) = T(x_i|x_{i-1})$$

- A chain is homogenous if T is invariant for all i
- MC will stabilize into an invariant distribution if

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1},\ldots,x_1) = T(x_i|x_{i-1})$$

- A chain is homogenous if T is invariant for all i
- MC will stabilize into an invariant distribution if
  1. Irreducible, transition graph is connected

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1}, \ldots, x_1) = T(x_i|x_{i-1})$$

- A chain is homogenous if T is invariant for all i
- MC will stabilize into an invariant distribution if
  1. Irreducible, transition graph is connected
  2. Aperiodic, does not get trapped in cycles

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1}, \dots, x_1) = T(x_i|x_{i-1})$$

- A chain is homogenous if T is invariant for all i
- MC will stabilize into an invariant distribution if
  1. Irreducible, transition graph is connected
  2. Aperiodic, does not get trapped in cycles
- Sufficient condition to ensure $p(x)$ is the invariant distribution

$$p(x_i)T(x_{i-1}|x_i) = p(x_{i-1})T(x_i|x_{i-1})$$

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1}, \ldots, x_1) = T(x_i|x_{i-1})$$

- A chain is homogenous if T is invariant for all i
- MC will stabilize into an invariant distribution if
  1. Irreducible, transition graph is connected
  2. Aperiodic, does not get trapped in cycles
- Sufficient condition to ensure $p(x)$ is the invariant distribution

$$p(x_i)T(x_{i-1}|x_i) = p(x_{i-1})T(x_i|x_{i-1})$$

- MCMC samplers, invariant distribution = target distribution

Markov Chains

- Use a Markov chain to explore the state space
- Markov chain in a discrete space is a process with

$$p(x_i|x_{i-1},\ldots,x_1) = T(x_i|x_{i-1})$$

- A chain is homogenous if T is invariant for all i
- MC will stabilize into an invariant distribution if
  1. Irreducible, transition graph is connected
  2. Aperiodic, does not get trapped in cycles
- Sufficient condition to ensure $p(x)$ is the invariant distribution

$$p(x_i)T(x_{i-1}|x_i) = p(x_{i-1})T(x_i|x_{i-1})$$

- MCMC samplers, invariant distribution = target distribution
- Design of samplers for fast convergence

Markov Chains (Contd.)

- Random walker on the web

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages
  - Aperiodicity, do not get stuck in a loop

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages
  - Aperiodicity, do not get stuck in a loop
- PageRank uses $T = L + E$

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages
  - Aperiodicity, do not get stuck in a loop
- PageRank uses $T = L + E$
  - $L =$ link matrix for the web graph

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages
  - Aperiodicity, do not get stuck in a loop
- PageRank uses $T = L + E$
  - $L =$ link matrix for the web graph
  - $E =$ uniform random matrix, to ensure irreducibility, aperiodicity

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages
  - Aperiodicity, do not get stuck in a loop
- PageRank uses $T = L + E$
  - $L =$ link matrix for the web graph
  - $E =$ uniform random matrix, to ensure irreducibility, aperiodicity
- Invariant distribution $p(x)$ represents rank of webpage x

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages
  - Aperiodicity, do not get stuck in a loop
- PageRank uses $T = L + E$
  - $L =$ link matrix for the web graph
  - $E =$ uniform random matrix, to ensure irreducibility, aperiodicity
- Invariant distribution $p(x)$ represents rank of webpage x
- Continuous spaces, T becomes an integral kernel K

$$\int_{x_i} p(x_i) K(x_{i+1}|x_i) dx_i = p(x_{i+1})$$

Markov Chains (Contd.)

- Random walker on the web
  - Irreducibility, should be able to reach all pages
  - Aperiodicity, do not get stuck in a loop
- PageRank uses $T = L + E$
  - $L =$ link matrix for the web graph
  - $E =$ uniform random matrix, to ensure irreducibility, aperiodicity
- Invariant distribution $p(x)$ represents rank of webpage x
- Continuous spaces, T becomes an integral kernel K

$$\int_{x_i} p(x_i)K(x_{i+1}|x_i)dx_i = p(x_{i+1})$$

- $p(x)$ is the corresponding eigenfunction

The Metropolis-Hastings Algorithm

- Most popular MCMC method

The Metropolis-Hastings Algorithm

- Most popular MCMC method
- Based on a proposal distribution $q(x^*|x)$

The Metropolis-Hastings Algorithm

- Most popular MCMC method
- Based on a proposal distribution $q(x^*|x)$
- Algorithm: For $i = 0, \dots, (n-1)$

The Metropolis-Hastings Algorithm

- Most popular MCMC method
- Based on a proposal distribution $q(x^*|x)$
- Algorithm: For $i = 0, \ldots, (n-1)$
  - Sample $u \sim \mathcal{U}(0,1)$

The Metropolis-Hastings Algorithm

- Most popular MCMC method
- Based on a proposal distribution $q(x^*|x)$
- Algorithm: For $i = 0, \ldots, (n-1)$
  - Sample $u \sim \mathcal{U}(0, 1)$
  - Sample $x^* \sim q(x^*|x_i)$

The Metropolis-Hastings Algorithm

- Most popular MCMC method
- Based on a proposal distribution $q(x^*|x)$
- Algorithm: For $i = 0, \ldots, (n-1)$
  - Sample $u \sim \mathcal{U}(0, 1)$
  - Sample $x^* \sim q(x^*|x_i)$
  - Then

$$x_{i+1} = \begin{cases} x^* & \text{if} \quad u < A(x_i, x^*) = \min\left\{1, \frac{p(x^*)q(x_i|x^*)}{p(x_i)q(x^*|x_i)}\right\} \\ x_i & \text{otherwise} \end{cases}$$

The Metropolis-Hastings Algorithm

- Most popular MCMC method
- Based on a proposal distribution $q(x^*|x)$
- Algorithm: For $i = 0, \ldots, (n-1)$
  - Sample $u \sim \mathcal{U}(0,1)$
  - Sample $x^* \sim q(x^*|x_i)$
  - Then

$$x_{i+1} = \begin{cases} x^* & \text{if} \quad u < A(x_i, x^*) = \min\left\{1, \frac{p(x^*)q(x_i|x^*)}{p(x_i)q(x^*|x_i)}\right\} \\ x_i & \text{otherwise} \end{cases}$$

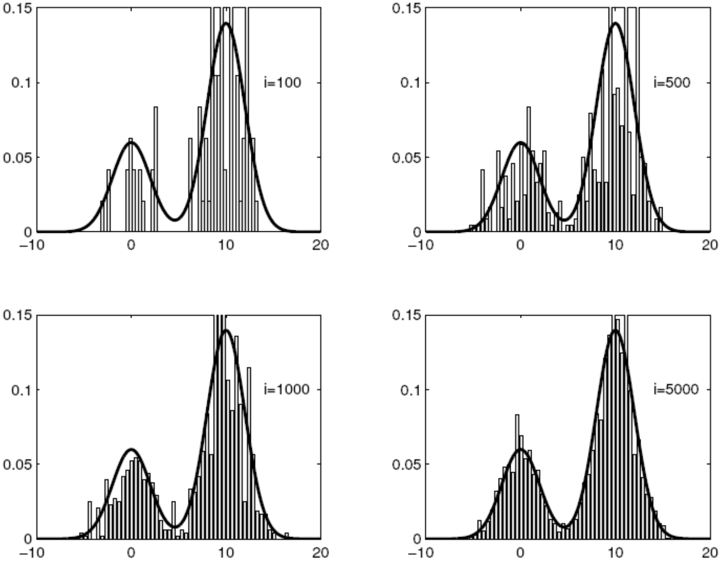
- The transition kernel is

$$K_{MH}(x_{i+1}|x_i) = q(x_{i+1}|x_i)A(x_i, x_{i+1}) + \delta_{x_i}(x_{i+1})r(x_i)$$

where $r(x_i)$ is the term associated with rejection

$$r(x_i) = \int_x q(x|x_i)(1 - A(x_i, x))dx$$

The Metropolis-Hastings Algorithm (Contd.)



The Metropolis-Hastings Algorithm (Contd.)

- By construction

$$p(x_i)K_{MH}(x_{i+1}|x_i) = p(x_{i+1})K_{MH}(x_i|x_{i+1})$$

The Metropolis-Hastings Algorithm (Contd.)

- By construction

$$p(x_i)K_{MH}(x_{i+1}|x_i) = p(x_{i+1})K_{MH}(x_i|x_{i+1})$$

- Implies $p(x)$ is the invariant distribution

The Metropolis-Hastings Algorithm (Contd.)

- By construction

$$p(x_i)K_{MH}(x_{i+1}|x_i) = p(x_{i+1})K_{MH}(x_i|x_{i+1})$$

- Implies $p(x)$ is the invariant distribution
- Basic properties

The Metropolis-Hastings Algorithm (Contd.)

- By construction

$$p(x_i)K_{MH}(x_{i+1}|x_i) = p(x_{i+1})K_{MH}(x_i|x_{i+1})$$

- Implies $p(x)$ is the invariant distribution
- Basic properties
  - Irreducibility, ensure support of q contains support of p

The Metropolis-Hastings Algorithm (Contd.)

- By construction

$$p(x_i)K_{MH}(x_{i+1}|x_i) = p(x_{i+1})K_{MH}(x_i|x_{i+1})$$

- Implies $p(x)$ is the invariant distribution
- Basic properties
  - Irreducibility, ensure support of q contains support of p
  - Aperiodicity, ensured since rejection is always a possibility

The Metropolis-Hastings Algorithm (Contd.)

- By construction

$$p(x_i)K_{MH}(x_{i+1}|x_i) = p(x_{i+1})K_{MH}(x_i|x_{i+1})$$

- Implies $p(x)$ is the invariant distribution
- Basic properties
  - Irreducibility, ensure support of q contains support of p
  - Aperiodicity, ensured since rejection is always a possibility
- Independent sampler: $q(x^*|x_i) = q(x^*)$ so that

$$A(x_i, x^*) = \min\left\{1, \frac{p(x^*)q(x_i)}{q(x^*)p(x_i)}\right\}$$

The Metropolis-Hastings Algorithm (Contd.)

- By construction

$$p(x_i)K_{MH}(x_{i+1}|x_i) = p(x_{i+1})K_{MH}(x_i|x_{i+1})$$

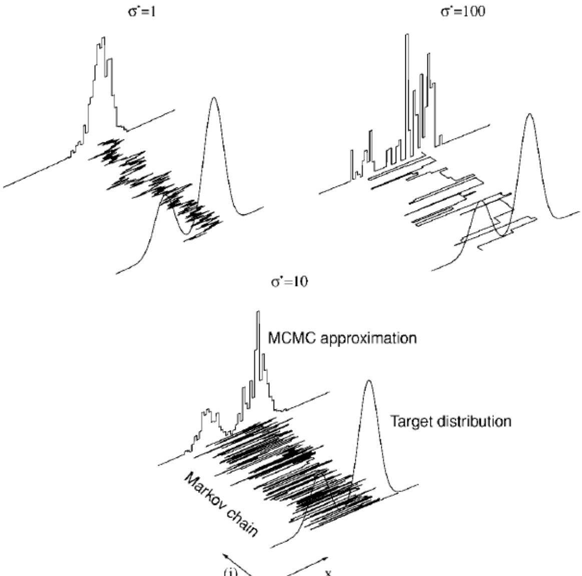
- Implies $p(x)$ is the invariant distribution
- Basic properties
  - Irreducibility, ensure support of q contains support of p
  - Aperiodicity, ensured since rejection is always a possibility
- Independent sampler: $q(x^*|x_i) = q(x^*)$ so that

$$A(x_i, x^*) = \min\left\{1, \frac{p(x^*)q(x_i)}{q(x^*)p(x_i)}\right\}$$

- Metropolis sampler: symmetric $q(x^*|x_i) = q(x_i|x^*)$

$$A(x_i, x^*) = \min\left\{1, \frac{p(x^*)}{p(x_i)}\right\}$$

The Metropolis-Hastings Algorithm (Contd.)



Simulated Annealing

- Problem: To find global maximum of $p(x)$

Simulated Annealing

- Problem: To find global maximum of $p(x)$
- Initial idea: Run MCMC, estimate $\hat{p}(x)$, compute max

Simulated Annealing

- Problem: To find global maximum of $p(x)$
- Initial idea: Run MCMC, estimate $\hat{p}(x)$, compute max
- Issue: MC may not come close to the mode(s)

Simulated Annealing

- Problem: To find global maximum of $p(x)$
- Initial idea: Run MCMC, estimate $\hat{p}(x)$, compute max
- Issue: MC may not come close to the mode(s)
- Simulate a *non-homogenous* Markov chain

Simulated Annealing

- Problem: To find global maximum of $p(x)$
- Initial idea: Run MCMC, estimate $\hat{p}(x)$, compute max
- Issue: MC may not come close to the mode(s)
- Simulate a *non-homogenous* Markov chain
- Invariant distribution at iteration i is $p_i(x) \propto p^{1/T_i}(x)$

Simulated Annealing

- Problem: To find global maximum of $p(x)$
- Initial idea: Run MCMC, estimate $\hat{p}(x)$, compute max
- Issue: MC may not come close to the mode(s)
- Simulate a *non-homogenous* Markov chain
- Invariant distribution at iteration i is $p_i(x) \propto p^{1/T_i}(x)$
- Sample update follows

$$x_{i+1} = \begin{cases} x^* & \text{if} \quad u < A(x_i, x^*) = \min\left\{1, \dfrac{p^{\frac{1}{T_i}}(x^*)q(x_i|x^*)}{p^{\frac{1}{T_i}}(x_i)q(x^*|x_i)}\right\} \\ x_i & \text{otherwise} \end{cases}$$

Simulated Annealing

- Problem: To find global maximum of $p(x)$
- Initial idea: Run MCMC, estimate $\hat{p}(x)$, compute max
- Issue: MC may not come close to the mode(s)
- Simulate a *non-homogenous* Markov chain
- Invariant distribution at iteration i is $p_i(x) \propto p^{1/T_i}(x)$
- Sample update follows

$$x_{i+1} = \begin{cases} x^* & \text{if} \quad u < A(x_i, x^*) = \min\left\{1, \dfrac{p^{\frac{1}{T_i}}(x^*)q(x_i|x^*)}{p^{\frac{1}{T_i}}(x_i)q(x^*|x_i)}\right\} \\ x_i & \text{otherwise} \end{cases}$$

- T_i decreases following a cooling schedule, $\lim_{i\to\infty} T_i = 0$

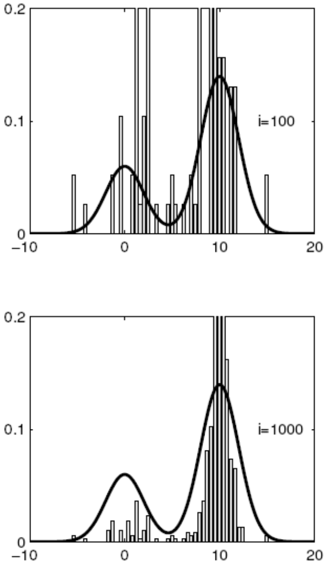
Simulated Annealing

- Problem: To find global maximum of $p(x)$
- Initial idea: Run MCMC, estimate $\hat{p}(x)$, compute max
- Issue: MC may not come close to the mode(s)
- Simulate a *non-homogenous* Markov chain
- Invariant distribution at iteration i is $p_i(x) \propto p^{1/T_i}(x)$
- Sample update follows

$$x_{i+1} = \begin{cases} x^* & \text{if} \quad u < A(x_i, x^*) = \min\left\{1, \dfrac{p^{\frac{1}{T_i}}(x^*)q(x_i|x^*)}{p^{\frac{1}{T_i}}(x_i)q(x^*|x_i)}\right\} \\ x_i & \text{otherwise} \end{cases}$$

- T_i decreases following a cooling schedule, $\lim_{i\to\infty} T_i = 0$
- Cooling schedule needs proper choice, e.g., $T_i = \frac{1}{C\log(i+T_0)}$

Simulated Annealing (Contd.)



Monte Carlo EM

- E-step involves computing an expectation

$$Q(\theta, \theta_n) = \int_x \log p(x, z|\theta) p(z|x, \theta_n) dx$$

Monte Carlo EM

- E-step involves computing an expectation

$$Q(\theta, \theta_n) = \int_x \log p(x, z|\theta) p(z|x, \theta_n) dx$$

- Estimate the expectation using MCMC

Monte Carlo EM

- E-step involves computing an expectation

$$Q(\theta, \theta_n) = \int_x \log p(x, z|\theta) p(z|x, \theta_n) dx$$

- Estimate the expectation using MCMC
- Draw samples using MH with acceptance probability

$$A(z, z^*) = \min \left\{ 1, \frac{p(x|z^*, \theta_n) p(z^*|\theta_n) q(z|z^*)}{p(x|z, \theta_n) p(z|\theta_n) q(z^*|z)} \right\}$$

Monte Carlo EM

- E-step involves computing an expectation

$$Q(\theta, \theta_n) = \int_x \log p(x, z|\theta) p(z|x, \theta_n) dx$$

- Estimate the expectation using MCMC
- Draw samples using MH with acceptance probability

$$A(z, z^*) = \min \left\{ 1, \frac{p(x|z^*, \theta_n) p(z^*|\theta_n) q(z|z^*)}{p(x|z, \theta_n) p(z|\theta_n) q(z^*|z)} \right\}$$

- Several variants:

Monte Carlo EM

- E-step involves computing an expectation

$$Q(\theta, \theta_n) = \int_x \log p(x, z|\theta) p(z|x, \theta_n) dx$$

- Estimate the expectation using MCMC
- Draw samples using MH with acceptance probability

$$A(z, z^*) = \min \left\{ 1, \frac{p(x|z^*, \theta_n) p(z^*|\theta_n) q(z|z^*)}{p(x|z, \theta_n) p(z|\theta_n) q(z^*|z)} \right\}$$

- Several variants:
  - Stochastic EM: Draw one sample

Monte Carlo EM

- E-step involves computing an expectation

$$Q(\theta, \theta_n) = \int_x \log p(x, z|\theta) p(z|x, \theta_n) dx$$

- Estimate the expectation using MCMC
- Draw samples using MH with acceptance probability

$$A(z, z^*) = \min \left\{ 1, \frac{p(x|z^*, \theta_n) p(z^*|\theta_n) q(z|z^*)}{p(x|z, \theta_n) p(z|\theta_n) q(z^*|z)} \right\}$$

- Several variants:
  - Stochastic EM: Draw one sample
  - Monte Carlo EM: Draw multiple samples

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p
  - Cycle kernel $K_1 K_2$ converges to p

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p
  - Cycle kernel $K_1 K_2$ converges to p
- Mixtures can use global and local proposals

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p
  - Cycle kernel $K_1 K_2$ converges to p
- Mixtures can use global and local proposals
  - Global proposals explore the entire space (with probability α)

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p
  - Cycle kernel $K_1 K_2$ converges to p
- Mixtures can use global and local proposals
  - Global proposals explore the entire space (with probability α)
  - Local proposals discover finer details (with probability $(1-\alpha)$)

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p
  - Cycle kernel $K_1 K_2$ converges to p
- Mixtures can use global and local proposals
  - Global proposals explore the entire space (with probability α)
  - Local proposals discover finer details (with probability $(1-\alpha)$)
- Example: Target has many narrow peaks

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p
  - Cycle kernel $K_1 K_2$ converges to p
- Mixtures can use global and local proposals
  - Global proposals explore the entire space (with probability α)
  - Local proposals discover finer details (with probability $(1-\alpha)$)
- Example: Target has many narrow peaks
  - Global proposal gets the peaks

Mixtures of MCMC Kernels

- Powerful property of MCMC: Combination of Samplers
- Let K_1, K_2 be kernels with invariant distribution p
  - Mixture kernel $\alpha K_1 + (1-\alpha)K_2, \alpha \in [0,1]$ converges to p
  - Cycle kernel $K_1 K_2$ converges to p
- Mixtures can use global and local proposals
  - Global proposals explore the entire space (with probability α)
  - Local proposals discover finer details (with probability $(1-\alpha)$)
- Example: Target has many narrow peaks
  - Global proposal gets the peaks
  - Local proposals get the neighborhood of peaks (random walk)

Cycles of MCMC Kernels

- Split a multi-variate state into blocks

Cycles of MCMC Kernels

- Split a multi-variate state into blocks
- Each block can be updated separately

Cycles of MCMC Kernels

- Split a multi-variate state into blocks
- Each block can be updated separately
- Convergence is faster if correlated variables are blocked

Cycles of MCMC Kernels

- Split a multi-variate state into blocks
- Each block can be updated separately
- Convergence is faster if correlated variables are blocked
- Transition kernel is given by

$$K_{MHCycle}(x^{(i+1)}|x^{(i)}) = \prod_{j=1}^{n_b} K_{MH(j)}(x_{b_j}^{(i+1)}|x_{b_j}^{(i)}, x_{-[b_j]}^{(i+1)})$$

Cycles of MCMC Kernels

- Split a multi-variate state into blocks
- Each block can be updated separately
- Convergence is faster if correlated variables are blocked
- Transition kernel is given by

$$K_{MHCycle}(x^{(i+1)}|x^{(i)}) = \prod_{j=1}^{n_b} K_{MH(j)}(x_{b_j}^{(i+1)}|x_{b_j}^{(i)}, x_{-[b_j]}^{(i+1)})$$

- Trade-off on block size

Cycles of MCMC Kernels

- Split a multi-variate state into blocks
- Each block can be updated separately
- Convergence is faster if correlated variables are blocked
- Transition kernel is given by

$$K_{MHCycle}(x^{(i+1)}|x^{(i)}) = \prod_{j=1}^{n_b} K_{MH(j)}(x_{b_j}^{(i+1)}|x_{b_j}^{(i)}, x_{-[b_j]}^{(i+1)})$$

- Trade-off on block size
  - If block size is small, chain takes long time to explore the space

Cycles of MCMC Kernels

- Split a multi-variate state into blocks
- Each block can be updated separately
- Convergence is faster if correlated variables are blocked
- Transition kernel is given by

$$K_{MHCycle}(x^{(i+1)}|x^{(i)}) = \prod_{j=1}^{n_b} K_{MH(j)}(x_{b_j}^{(i+1)}|x_{b_j}^{(i)}, x_{-[b_j]}^{(i+1)})$$

- Trade-off on block size
  - If block size is small, chain takes long time to explore the space
  - If block size is large, acceptance probability is low

Cycles of MCMC Kernels

- Split a multi-variate state into blocks
- Each block can be updated separately
- Convergence is faster if correlated variables are blocked
- Transition kernel is given by

$$K_{MHCycle}(x^{(i+1)}|x^{(i)}) = \prod_{j=1}^{n_b} K_{MH(j)}(x_{b_j}^{(i+1)}|x_{b_j}^{(i)}, x_{-[b_j]}^{(i+1)})$$

- Trade-off on block size
  - If block size is small, chain takes long time to explore the space
  - If block size is large, acceptance probability is low
- Gibbs sampling effectively uses block size of 1

The Gibbs Sampler

- For a d-dimensional vector x, assume we know

$$p(x_j|x_{-j}) = p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \cdots, x_d)$$

The Gibbs Sampler

- For a d-dimensional vector x, assume we know

$$p(x_j|x_{-j}) = p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \cdots, x_d)$$

- Gibbs sampler uses the following proposal distribution

$$q(x^*|x^{(i)}) = \begin{cases} p(x_j^*|x_{-j}^{(i)}) & \text{if } x_{-j}^* = x_{-j}^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

The Gibbs Sampler

- For a d-dimensional vector x, assume we know

$$p(x_j|x_{-j}) = p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \cdots, x_d)$$

- Gibbs sampler uses the following proposal distribution

$$q(x^*|x^{(i)}) = \begin{cases} p(x_j^*|x_{-j}^{(i)}) & \text{if } \; x_{-j}^* = x_{-j}^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

- The acceptance probability

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

The Gibbs Sampler

- For a d-dimensional vector x, assume we know

$$p(x_j|x_{-j}) = p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \cdots, x_d)$$

- Gibbs sampler uses the following proposal distribution

$$q(x^*|x^{(i)}) = \begin{cases} p(x_j^*|x_{-j}^{(i)}) & \text{if} \quad x_{-j}^* = x_{-j}^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

- The acceptance probability

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

- Deterministic scan: All samples are accepted

The Gibbs Sampler (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$

The Gibbs Sampler (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)} \ldots, x_d^{(i)})$

The Gibbs Sampler (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - Sample $x_2^{(i+1)} \sim p(x_1 | x_1^{(i+1)}, x_3^{(i)} \ldots, x_d^{(i)})$

The Gibbs Sampler (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - Sample $x_2^{(i+1)} \sim p(x_1 | x_1^{(i+1)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - $\cdots$

The Gibbs Sampler (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - Sample $x_2^{(i+1)} \sim p(x_1 | x_1^{(i+1)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - $\ldots$
  - Sample $x_d^{(i+1)} \sim p(x_d | x_1^{(i+1)}, \ldots, x_{d-1}^{(i+1)})$

The Gibbs Sampler (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - Sample $x_2^{(i+1)} \sim p(x_1 | x_1^{(i+1)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - $\ldots$
  - Sample $x_d^{(i+1)} \sim p(x_d | x_1^{(i+1)}, \ldots, x_{d-1}^{(i+1)})$
- Possible to have MH steps inside a Gibbs sampler

The Gibbs Sampler (Contd.)

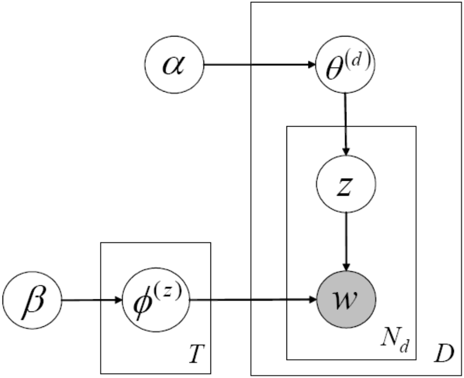
- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - Sample $x_2^{(i+1)} \sim p(x_1 | x_1^{(i+1)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - $\cdots$
  - Sample $x_d^{(i+1)} \sim p(x_d | x_1^{(i+1)}, \ldots, x_{d-1}^{(i+1)})$
- Possible to have MH steps inside a Gibbs sampler
- For $d = 2$, Gibbs sampler is the data augmentation algorithm

The Gibbs Sampler (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (N-1)$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - Sample $x_2^{(i+1)} \sim p(x_1 | x_1^{(i+1)}, x_3^{(i)} \ldots, x_d^{(i)})$
  - $\cdots$
  - Sample $x_d^{(i+1)} \sim p(x_d | x_1^{(i+1)}, \ldots, x_{d-1}^{(i+1)})$
- Possible to have MH steps inside a Gibbs sampler
- For $d = 2$, Gibbs sampler is the data augmentation algorithm
- For Bayes nets, the conditioning is on the Markov blanket

$$p(x_j | x_{-j}) = p(x_j | x_{pa(j)}) \prod_{k \in ch(j)} p(x_k | pa(k))$$

Bayesian LDA



Gibbs Sampler for Bayesian LDA

- The conditional distribution

$$p(z_\ell = h|\mathbf{z}_{-\ell}, \mathbf{w}) \propto p(z_\ell = h|z_{-\ell})p(w_\ell|z_\ell = h, \mathbf{z}_{-\ell}, \mathbf{w}_{-\ell})$$

Gibbs Sampler for Bayesian LDA

- The conditional distribution

$$p(z_\ell = h|\mathbf{z}_{-\ell}, \mathbf{w}) \propto p(z_\ell = h|z_{-\ell})p(w_\ell|z_\ell = h, \mathbf{z}_{-\ell}, \mathbf{w}_{-\ell})$$

- Notation:

Gibbs Sampler for Bayesian LDA

- The conditional distribution

$$p(z_\ell = h|\mathbf{z}_{-\ell}, \mathbf{w}) \propto p(z_\ell = h|z_{-\ell})p(w_\ell|z_\ell = h, \mathbf{z}_{-\ell}, \mathbf{w}_{-\ell})$$

- Notation:
  - $C^{DT}_{(d_{-\ell},h)}$ = words from d assigned to h, excluding current word

Gibbs Sampler for Bayesian LDA

- The conditional distribution

$$p(z_\ell = h|\mathbf{z}_{-\ell}, \mathbf{w}) \propto p(z_\ell = h|z_{-\ell})p(w_\ell|z_\ell = h, \mathbf{z}_{-\ell}, \mathbf{w}_{-\ell})$$

- Notation:
  - $C^{DT}_{(d_{-\ell},h)}$ = words from d assigned to h, excluding current word
  - $C^{WT}_{(w_{-\ell},h)}$ = w_ℓ assigned to h, excluding current word

Gibbs Sampler for Bayesian LDA

- The conditional distribution

$$p(z_\ell = h | \mathbf{z}_{-\ell}, \mathbf{w}) \propto p(z_\ell = h | z_{-\ell}) p(w_\ell | z_\ell = h, \mathbf{z}_{-\ell}, \mathbf{w}_{-\ell})$$

- Notation:
  - $C^{DT}_{(d_{-\ell}, h)}$ = words from d assigned to h, excluding current word
  - $C^{WT}_{(w_{-\ell}, h)}$ = w_ℓ assigned to h, excluding current word
- Then, the first term

$$p(z_\ell = h | z_{-\ell}) = \frac{C^{DT}_{(d_{-\ell}, h)} + \alpha}{\sum_{t=1}^{T} C^{DT}_{(d_{-\ell}, t)} + T\alpha}$$

Gibbs Sampler for Bayesian LDA

- The conditional distribution

$$p(z_\ell = h|\mathbf{z}_{-\ell}, \mathbf{w}) \propto p(z_\ell = h|z_{-\ell})p(w_\ell|z_\ell = h, \mathbf{z}_{-\ell}, \mathbf{w}_{-\ell})$$

- Notation:
  - $C^{DT}_{(d_{-\ell},h)}$ = words from d assigned to h, excluding current word
  - $C^{WT}_{(w_{-\ell},h)}$ = w_ℓ assigned to h, excluding current word
- Then, the first term

$$p(z_\ell = h|z_{-\ell}) = \frac{C^{DT}_{(d_{-\ell},h)} + \alpha}{\sum_{t=1}^{T} C^{DT}_{(d_{-\ell},t)} + T\alpha}$$

- The second term

$$p(w_\ell|z_\ell = h, \mathbf{z}_{-\ell}, \mathbf{w}_{-ell}) = \frac{C^{WT}_{(w_{-\ell},h)} + \beta}{\sum_{w=1}^{W} C^{WT}_{(w_{-\ell},h)} + W\beta}$$

Basic Idea

- Sometimes easier to sample from $p(x, u)$ rather than $p(x)$

Basic Idea

- Sometimes easier to sample from $p(x, u)$ rather than $p(x)$
- Sample (x_i, u_i), and then ignore u_i

Basic Idea

- Sometimes easier to sample from $p(x, u)$ rather than $p(x)$
- Sample (x_i, u_i), and then ignore u_i
- Consider two well-known examples:

Basic Idea

- Sometimes easier to sample from $p(x, u)$ rather than $p(x)$
- Sample (x_i, u_i), and then ignore u_i
- Consider two well-known examples:
  - Hybrid Monte Carlo

Basic Idea

- Sometimes easier to sample from $p(x, u)$ rather than $p(x)$
- Sample (x_i, u_i), and then ignore u_i
- Consider two well-known examples:
    - Hybrid Monte Carlo
    - Slice sampling

Hybrid Monte Carlo

- Uses gradient of the target distribution

Hybrid Monte Carlo

- Uses gradient of the target distribution
- Improves "mixing" in high dimensions

Hybrid Monte Carlo

- Uses gradient of the target distribution
- Improves "mixing" in high dimensions
- Effectively, take steps based on gradient of $p(x)$

Hybrid Monte Carlo

- Uses gradient of the target distribution
- Improves “mixing” in high dimensions
- Effectively, take steps based on gradient of $p(x)$
- Introduce auxiliary momentum variables $u \in \mathbb{R}^d$ with

$$p(x, u) = p(x) N(u; 0, \mathbb{I}_d)$$

Hybrid Monte Carlo

- Uses gradient of the target distribution
- Improves “mixing” in high dimensions
- Effectively, take steps based on gradient of $p(x)$
- Introduce auxiliary momentum variables $u \in \mathbb{R}^d$ with

$$p(x, u) = p(x) N(u; 0, \mathbb{I}_d)$$

- Gradient vector $\Delta(x) = \partial \log p(x) / \partial x$, step-size ρ

Hybrid Monte Carlo

- Uses gradient of the target distribution
- Improves "mixing" in high dimensions
- Effectively, take steps based on gradient of $p(x)$
- Introduce auxiliary momentum variables $u \in \mathbb{R}^d$ with

$$p(x, u) = p(x)N(u; 0, \mathbb{I}_d)$$

- Gradient vector $\Delta(x) = \partial \log p(x)/\partial x$, step-size ρ
- Gradient descent for L steps to get proposal candidate

Hybrid Monte Carlo

- Uses gradient of the target distribution
- Improves "mixing" in high dimensions
- Effectively, take steps based on gradient of $p(x)$
- Introduce auxiliary momentum variables $u \in \mathbb{R}^d$ with

$$p(x, u) = p(x) N(u; 0, \mathbb{I}_d)$$

- Gradient vector $\Delta(x) = \partial \log p(x) / \partial x$, step-size ρ
- Gradient descent for L steps to get proposal candidate
- When $L = 1$, one obtains the Langevin algorithm

$$x^* = x_0 + \rho u_0 = x^{(i-1)} + \rho(u^* + \rho \Delta(x^{(i-1)})/2)$$

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (n-1)$

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (n-1)$
    - Sample $v \sim \mathcal{U}[0,1]$, $u^* \sim \mathcal{N}(0, \mathbb{I}_d)$

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (n-1)$
  - Sample $v \sim \mathcal{U}[0,1]$, $u^* \sim \mathcal{N}(0, \mathbb{I}_d)$
  - Let $x_0 = x^{(i-1)}$, $u_0 = u^* + \rho\Delta(x_0)/2$

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (n-1)$
  - Sample $v \sim \mathcal{U}[0,1]$, $u^* \sim \mathcal{N}(0, \mathbb{I}_d)$
  - Let $x_0 = x^{(i-1)}$, $u_0 = u^* + \rho\Delta(x_0)/2$
  - For $\ell = 1, \ldots, L$, with $\rho_\ell = \rho, \ell < L$, $\rho_L = \rho/2$

$$x_\ell = x_{\ell-1} + \rho u_{\ell-1} \qquad u_\ell = u_{\ell-1} + \rho_\ell \Delta(x_\ell)$$

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (n-1)$
  - Sample $v \sim \mathcal{U}[0,1]$, $u^* \sim \mathcal{N}(0, \mathbb{I}_d)$
  - Let $x_0 = x^{(i-1)}$, $u_0 = u^* + \rho\Delta(x_0)/2$
  - For $\ell = 1, \ldots, L$, with $\rho_\ell = \rho, \ell < L$, $\rho_L = \rho/2$

$$x_\ell = x_{\ell-1} + \rho u_{\ell-1} \qquad u_\ell = u_{\ell-1} + \rho_\ell \Delta(x_\ell)$$

  - Set

$$(x^{(i+1)}, u^{(i+1)}) = \begin{cases} (x_L, u_L) & \text{if } A = \min\left\{1, \frac{p(x_L)}{p(x_i)} \exp\left(-\frac{1}{2}(\|u_L\|^2 - \|u^*\|^2)\right)\right\} \\ (x^{(i)}, u^{(i)}) & \text{otherwise} \end{cases}$$

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (n-1)$
  - Sample $v \sim \mathcal{U}[0,1]$, $u^* \sim \mathcal{N}(0, \mathbb{I}_d)$
  - Let $x_0 = x^{(i-1)}$, $u_0 = u^* + \rho\Delta(x_0)/2$
  - For $\ell = 1, \ldots, L$, with $\rho_\ell = \rho, \ell < L$, $\rho_L = \rho/2$

$$x_\ell = x_{\ell-1} + \rho u_{\ell-1} \qquad u_\ell = u_{\ell-1} + \rho_\ell \Delta(x_\ell)$$

  - Set

$$(x^{(i+1)}, u^{(i+1)}) = \begin{cases} (x_L, u_L) & \text{if } A = \min\left\{1, \frac{p(x_L)}{p(x_i)} \exp\left(-\frac{1}{2}(\|u_L\|^2 - \|u^*\|^2)\right)\right\} \\ (x^{(i)}, u^{(i)}) & \text{otherwise} \end{cases}$$

- Tradeoffs for ρ, L

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \ldots, (n-1)$
  - Sample $v \sim \mathcal{U}[0,1]$, $u^* \sim \mathcal{N}(0, \mathbb{I}_d)$
  - Let $x_0 = x^{(i-1)}$, $u_0 = u^* + \rho\Delta(x_0)/2$
  - For $\ell = 1, \ldots, L$, with $\rho_\ell = \rho, \ell < L$, $\rho_L = \rho/2$

$$x_\ell = x_{\ell-1} + \rho u_{\ell-1} \qquad u_\ell = u_{\ell-1} + \rho_\ell \Delta(x_\ell)$$

  - Set

$$(x^{(i+1)}, u^{(i+1)}) = \begin{cases} (x_L, u_L) & \text{if } A = \min\left\{1, \frac{p(x_L)}{p(x_i)} \exp\left(-\frac{1}{2}(\|u_L\|^2 - \|u^*\|^2)\right)\right\} \\ (x^{(i)}, u^{(i)}) & \text{otherwise} \end{cases}$$

- Tradeoffs for ρ, L
  - Large ρ gives low acceptance, small ρ needs many steps

Hybrid Monte Carlo (Contd.)

- Initialize $x^{(0)}$. For $i = 0, \dots, (n-1)$
  - Sample $v \sim \mathcal{U}[0,1]$, $u^* \sim \mathcal{N}(0, \mathbb{I}_d)$
  - Let $x_0 = x^{(i-1)}$, $u_0 = u^* + \rho\Delta(x_0)/2$
  - For $\ell = 1, \dots, L$, with $\rho_\ell = \rho, \ell < L$, $\rho_L = \rho/2$

$$x_\ell = x_{\ell-1} + \rho u_{\ell-1} \qquad u_\ell = u_{\ell-1} + \rho_\ell \Delta(x_\ell)$$

  - Set

$$(x^{(i+1)}, u^{(i+1)}) = \begin{cases} (x_L, u_L) & \text{if } A = \min\left\{1, \frac{p(x_L)}{p(x_i)} \exp\left(-\frac{1}{2}(\|u_L\|^2 - \|u^*\|^2)\right)\right\} \\ (x^{(i)}, u^{(i)}) & \text{otherwise} \end{cases}$$

- Tradeoffs for ρ, L
  - Large ρ gives low acceptance, small ρ needs many steps
  - Large L gives candidates far from x_0, but expensive

The Slice Sampler

- Construct extended target distribution

$$p^*(x, u) = \begin{cases} 1 & \text{if } 0 \leq u \leq p(x) \\ 0 & \text{otherwise} \end{cases}$$

The Slice Sampler

- Construct extended target distribution

$$p^*(x,u) = \begin{cases} 1 & \text{if } 0 \leq u \leq p(x) \\ 0 & \text{otherwise} \end{cases}$$

- It follows that: $\int p^*(x,u) = \int_0^{p(x)} du = p(x)$

The Slice Sampler

- Construct extended target distribution

$$p^*(x,u) = \begin{cases} 1 & \text{if } 0 \leq u \leq p(x) \\ 0 & \text{otherwise} \end{cases}$$

- It follows that: $\int p^*(x,u) = \int_0^{p(x)} du = p(x)$
- From the Gibbs sampling perspective

$$p(u|x) = \mathcal{U}[0, p(x)] \qquad p(x|u) = \mathcal{U}_A, A = \{x : p(x) \geq u\}$$

The Slice Sampler

- Construct extended target distribution

$$p^*(x, u) = \begin{cases} 1 & \text{if } 0 \leq u \leq p(x) \\ 0 & \text{otherwise} \end{cases}$$

- It follows that: $\int p^*(x, u) = \int_0^{p(x)} du = p(x)$
- From the Gibbs sampling perspective

$$p(u|x) = \mathcal{U}[0, p(x)] \qquad p(x|u) = \mathcal{U}_A, A = \{x : p(x) \geq u\}$$

- Algorithm is easy is A is easy to figure out

The Slice Sampler

- Construct extended target distribution

$$p^*(x,u) = \begin{cases} 1 & \text{if } 0 \le u \le p(x) \\ 0 & \text{otherwise} \end{cases}$$

- It follows that: $\int p^*(x,u) = \int_0^{p(x)} du = p(x)$
- From the Gibbs sampling perspective

$$p(u|x) = \mathcal{U}[0, p(x)] \qquad p(x|u) = \mathcal{U}_A, A = \{x : p(x) \ge u\}$$

- Algorithm is easy is A is easy to figure out
- Otherwise, several auxiliary variables need to be introduced

The Slice Sampler (Contd.)