Markov Chain Monte Carlo

COS 424/524, SML 302: Fundamentals of Machine Learning

Professor Engelhardt

COS424/524, SML302

Lecture 22

Markov chain Monte Carlo

We have learned about many probabilistic models for analyzing data.

We talked about using maximum likelihood estimation and expectation maximization (for latent variable models) to fit models to data.

Sometimes this is hard to do. Why?

Today we will learn about Markov chain Monte Carlo (MCMC) methods that approximate the full posterior distribution of parameters.

EM and MLE approaches

Why are EM and MLE methods limited?

- sometimes hard to derive
- sometimes limited to models with simple conjugate priors
- sometimes limited to exponential family distributions
- sometimes point estimates of parameters are insufficient

Key point: when you want to 'test out' a model, often deriving MLE or EM approaches are burdensome.

MCMC approaches

When should you consider using MCMC?

- when the posterior $p(z)$ is computable *up to a normalizing constant*
- when it is difficult to sample from the posterior
- when you are willing to wait for the solution
- when you want to quantify the full posterior distribution of the samples, not just point estimates

What can Monte Carlo methods do?

Monte Carlo methods involve repeated sampling to find a result.

The main use of Monte Carlo methods is for integration and optimization problems.

Monte Carlo methods are useful for many situations we have encountered in this course.

What can Monte Carlo methods do? Integration.

Let $z \in \Theta$ be the set of unknown variables, including parameters and latent variables. Let $y \in \mathcal{D}$ be data observations.

Many of the questions we ask of our fitted models require (often) intractable integrals.

- *Normalization*: To compute posterior $p(z \mid y)$ using prior $p(z)$ and likelihood $p(y \mid z)$, compute normalizing constant in Bayes theorem:

$$p(z \mid y) = \frac{p(y \mid z)p(z)}{\int_\Theta p(y \mid z)p(z)dz}.$$

What can Monte Carlo methods do? Integration.

- *Marginalization:* When $z = (z_1, z_2)$, we may want the marginal posterior instead of full posterior:

$$p(z_1 \mid y) = \int_{\mathcal{Z}_2} p(z_1, z_2 \mid y) dz_2.$$

- *Expectation:* many data analyses involve expected value of some function of latent variables with respect to posterior:

$$\mathrm{E}_{p(z|y)}(f(z)) = \int_{\mathcal{Z}} f(z) p(z \mid y) dz.$$

What can Monte Carlo methods do? Optimization.

- *Optimization*:
  - Find a value that maximizes some objective function
  - Often, cannot explore all options in a computationally feasible way
  - Space may be non-convex and hard to search
- *Model selection:*
  - The space of models is often very large (e.g., ℓ_0 penalized regression)
  - We would like to search this space efficiently when simple relaxations of the space are not possible.

Monte Carlo: overview

The basic idea behind Monte Carlo methods is the following:

Monte Carlo methods

- Given data $\mathcal{D}$ and a model $\mathcal{M}$ with parameters, latent variables z.
- Generate $s = 1 : S$ unweighted samples from the posterior: $z^s \sim p(z \mid \mathcal{D}, \mathcal{M})$
- Use samples to compute expected values: $\mathrm{E}[f(z) \mid \mathcal{D}] \approx \frac{1}{S} \sum_{s=1}^{S} f(z^s)$, including marginal expectations

Monte Carlo: why this works for integrals

Draw an IID set of samples $\{z^s\}_{s=1}^S$ from a target density $p(z \mid x)$.

These S samples can be used to approximate this target density:

$$p(z \mid x) \approx p_S(z) = \frac{1}{S} \sum_{s=1}^{S} \delta_{z^s}(z).$$

In other words, we approximate the probability of z by computing the fraction of S samples from $p(x)$ that have value z.

As a consequence, we can approximate integrals in this framework:

$$\int_{\mathcal{Z}} f(z)p(z)dz \approx \frac{1}{S} \sum_{s=1}^{S} f(z^s).$$

Monte Carlo: why this works for optimization

By same reasoning, we can find maximum with respect to objective function $p(z)$:

$$\hat{z} \approx \arg \max_{z^s; s=1,\ldots,S} p(z^s)$$

It is generally inefficient to sample from the posterior for optimization

Simulated annealing algorithms sample from a distribution that approximates support on the objective function maxima, making our sampling for optimization efficient.

Monte Carlo: why this works for integration

As an example with respect to (marginal) expectations:

$$\int_{\mathcal{Z}} f(z)p(z)dz = \mathrm{E}[f(z) \mid \mathcal{D}] \approx \frac{1}{S}\sum_{s=1}^{S} f(z^s)$$

By generating a large number of samples from the posterior, we can achieve arbitrary levels of accuracy in these approximations.

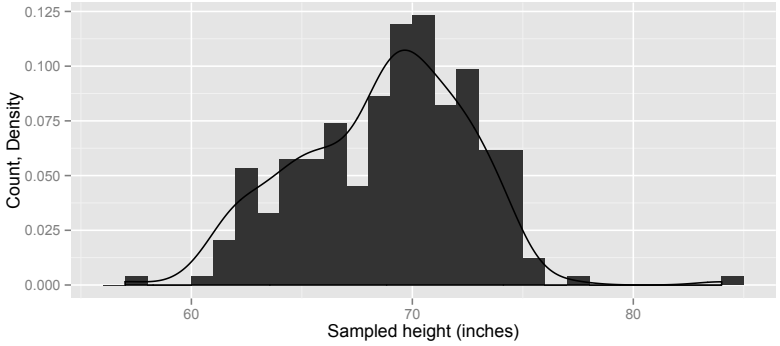
Much of the practical work in sampling is concerned with how to sample IID from the posterior. This can be difficult!

Running example: Gaussian mixture model

Let's use the following example to motivate this discussion:

Modeling height as a Gaussian mixture model

- Here, the data $\mathcal{D}$ are the set of hight data collected from this class
- Here, let the model $\mathcal{M}$ be a Gaussian mixture model with $K = 3$.



Running example: Gaussian mixture model

In this example, how would we think about sampling?

Modeling height as a Gaussian mixture model

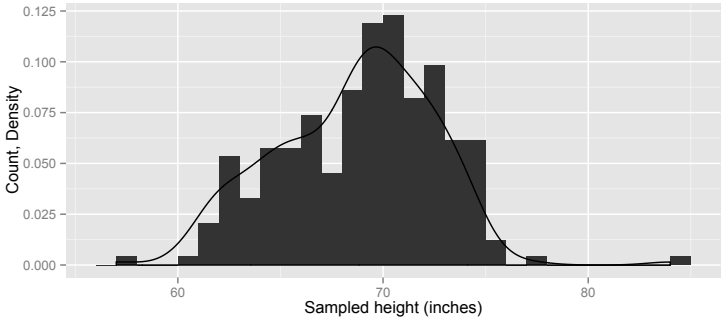
- Latent variables and parameters are:
  - μ, σ^2: K cluster means, variances
  - π: K cluster proportions
  - (c: n cluster assignment variables for each sample)
- The posterior probability is: $p(\mu, \sigma^2, \pi \mid c, \mathcal{D}) \propto \prod_{i=1}^{n} \pi_{c_i} \mathcal{N}(x_i | \mu_{c_i}, \sigma^2_{c_i})$.
- We can put priors on the parameters without distributions: π, μ, σ^2.

Running example: Gaussian mixture model

In this example, what is the question we are asking?

Modeling height as a Gaussian mixture model

- Given data $\mathcal{D}$, find parameters that maximize posterior distribution.
- We might estimate uncertainty in these parameter estimates.



We will discuss four sampling algorithms

Monte Carlo methods

- Rejection sampling
- Importance sampling

Markov chain Monte Carlo methods

- Metropolis Hastings
- Gibbs sampling

Key point: Sampling from the posterior of most useful models is difficult; but computing the (unnormalized) posterior density of a point is possible.

How would you sample from the posterior of a Gaussian mixture model?

Rejection sampling

Since we cannot sample directly from $p(z \mid \mathcal{D})$:

- choose a *proposal distribution* $q(z)$ that is easy to sample from such that, for some M, $Mq(z) \geq p(z)$
- sample $z^s \sim q(z)$ from the proposal distribution
- if z^s is likely with respect to $p(z \mid x)$, keep it; otherwise, reject it.
- By rejecting samples according to their probability under $p(z)$, we are left with samples from $p(z)$.

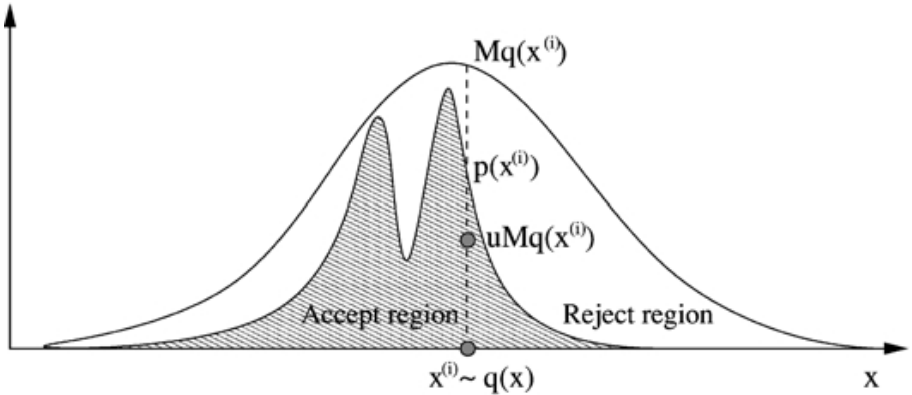
Understand efficiency: the number of times we have to reject sample z^s on average for each accept.

Rejection sampling: algorithm

Rejection sampling

- Set $s = 1$;
- Repeat until $s = S$:
  - sample $z^s \sim q(z)$; sample $u \sim Unif(0, 1)$.
  - if $u < \frac{p(z^s)}{Mq(z^s)}$, accept z^s and increment s;
  - Otherwise, reject z^s

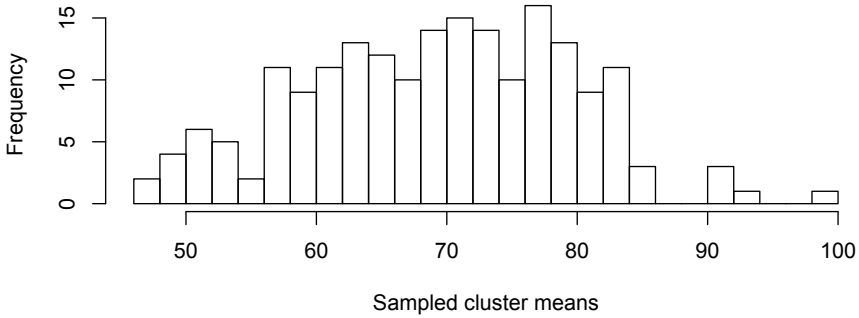
Rejection sampling: intuition



The "art" of rejection sampling is to select a proposal distribution that is easy to sample from and as close to $p(z)$ as possible.

From [Andrieu et al. 2003]

Rejection sampling in practice

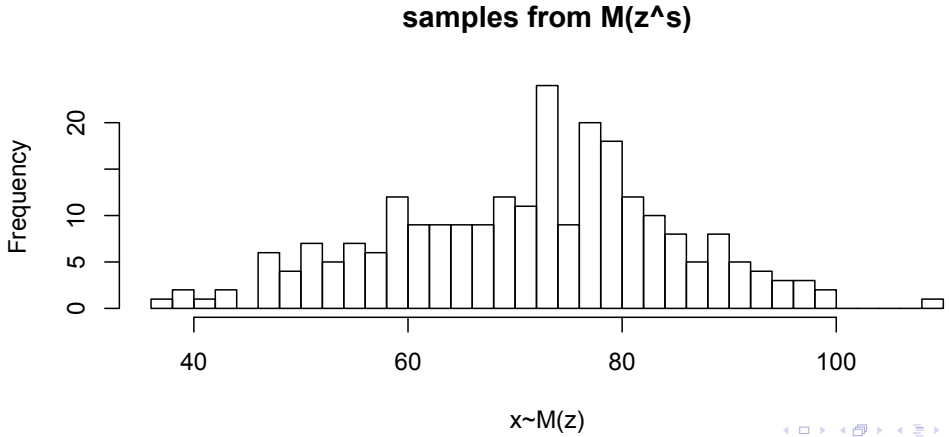


Proposal distribution: a Gaussian prior with empirical means and variances and Gamma with mean equal to the empirical variance

I tried a number of M until one gave me a reasonable acceptance rate. Here, I accepted $195/500,000$ iterations.

Rejection sampling in practice

Plotting the means is only part of the picture; Here, 244 samples from the generative model with parameters z^s.



Importance sampling

Instead of probabilistically rejecting samples, accept them all, with weights

- choose a *proposal distribution* $q(z)$ that is easy to sample from
- sample $z^s \sim q(z)$ from the proposal distribution
- compute $w(z^s) = \frac{p(z^s)}{q(z^s)}$
- Then $\int f(z)w(z)q(z)dz \approx \sum_{s=1}^{S} f(z^s)w(z^s)$
- Unbiased estimator, but hard to choose q sometimes (efficiency)

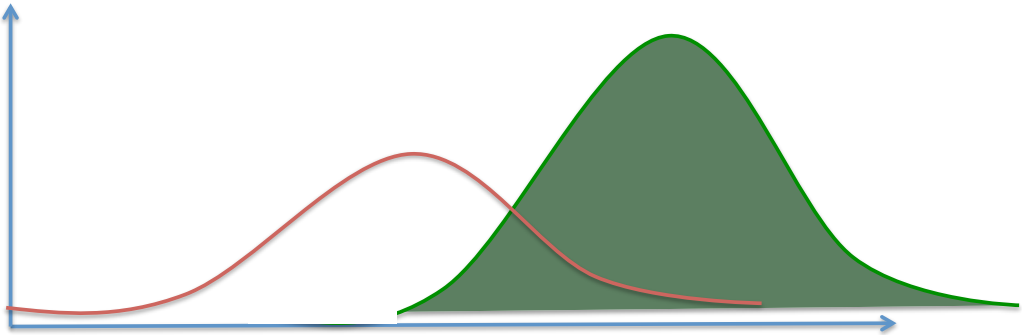
Importance sampling: algorithm

Importance sampling

- For $s = 1 : S$
  - sample $z^s \sim q(z)$.
  - compute $w(z) = \frac{p(z^s)}{q(z^s)}$

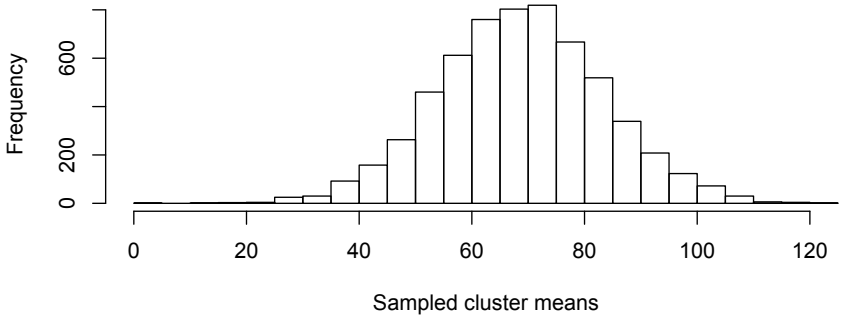
Importance sampling: intuition

The "art" of importance sampling is to select a proposal distribution that is easy to sample from and as close to $p(z)$ as possible.



How can you evaluate the efficiency of your sampler?

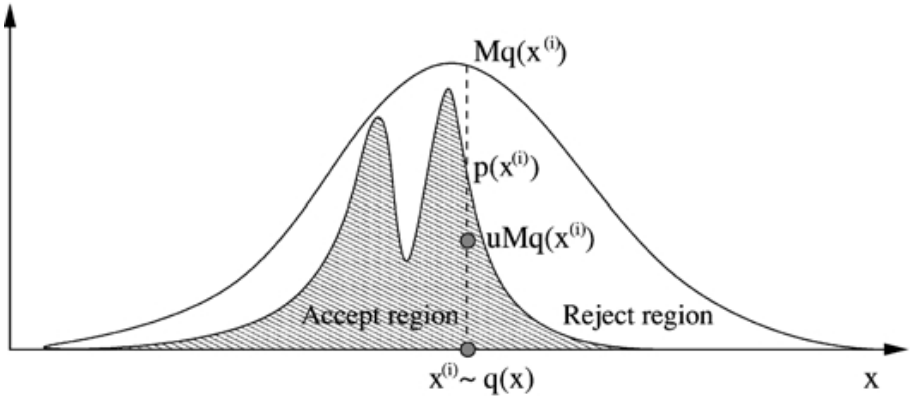
Importance sampling in practice



Proposal distribution was a Gaussian prior with empirical means and variances and Gamma with mean equal to the empirical variance

Importance sampling gives (near) zero probability to all 2,000 samples.

What have we learned about Monte Carlo methods?



Difficult to select proposal distributions for models in use that are both

- easy to sample from
- good approximations to the true posterior distribution

From [Andrieu et al. 2003]

Monte Carlo versus Markov chain Monte Carlo

Monte Carlo versus MCMC

- Monte Carlo methods are *not-iterative*: all samples are generated from the same posterior
- Markov chain Monte Carlo methods are *iterative*: posterior is updated after each sample

Markov chain Monte Carlo methods (MCMC)

Instead of randomly sampling from a proposal distribution, we can use a Markov chain to explore the latent space in a systematic way.

Intuitively, we would like our walk through parameter space to spend *as much time as possible in the highest probability regions*.

Metropolis-Hastings

MH sampling, as in MC sampling, uses the idea of a proposal distribution.

In MH, the proposal distribution is conditional on the current state: $q(z^* \mid z)$. This induces a *Markov chain*.

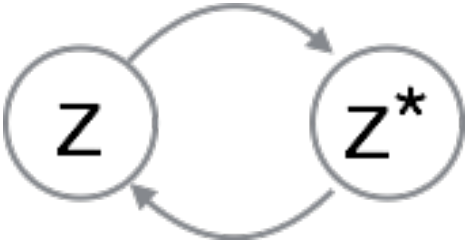
The novel element of this Markov chain is that it is on a continuous state space, not a discrete state space.

Metropolis-Hastings: what does this look like?

Each proposal z^* in sample space is selected within some (probabilistic) radius of the current state z.

When $p(z^*)$ scaled by probability of returning to current state from proposed state $q(z \mid z^*)$ is greater than $p(z)$ scaled by probability of moving from current to proposed state $q(z^* \mid z)$, MH accepts proposal.

Otherwise, MH flips a (biased) coin to decide to leave better current state.



Metropolis-Hastings: algorithm

Metropolis-Hastings

- Initialize z^1
- For $s = 1 : S$
  - sample $z^* \sim q(z^* \mid z^s)$
  - sample $u \sim Unif(0, 1)$
  - If $u < \min\left\{1, \frac{p(z^*)q(z^s|z^*)}{p(z^s)q(z^*|z^s)}\right\}$, then $z^{s+1} = z^*$
  - else $z^{s+1} = z^s$

Metropolis: algorithm

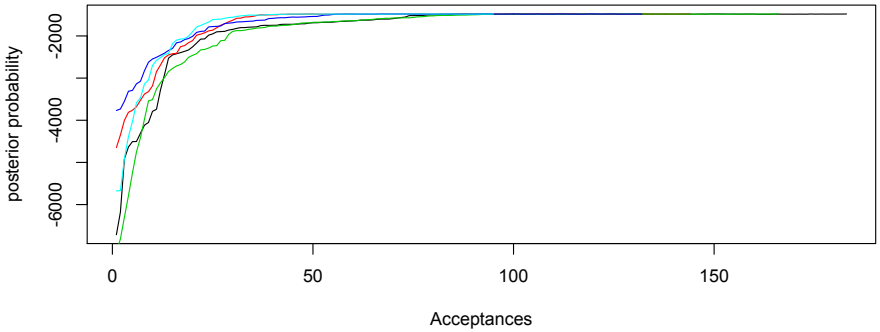
For Metropolis, the proposal is symmetric, $q(z \mid z^*) = q(z^* \mid z)$.

Then the acceptance ratio simplifies:

Metropolis

- Initialize z^1
- For $s = 1 : S$
  - sample $z^* \sim q(z^* \mid z^s)$
  - sample $u \sim Unif(0, 1)$
  - If $u < \min\left\{1, \frac{p(z^*)}{p(z^s)}\right\}$, then $z^{s+1} = z^*$
  - else $z^{s+1} = z^s$

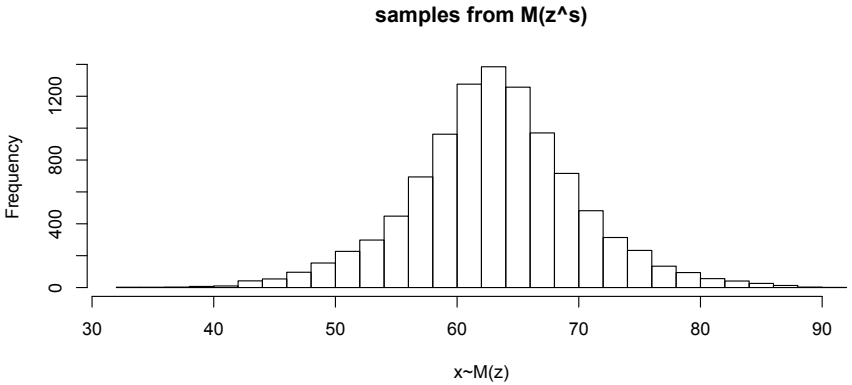
MH in practice



Ran 100,000 iterations of MH on Gaussian mixture model.

- proposal distribution of cluster means: Gaussian (mean μ_k)
- proposal distribution of cluster variances: gamma (expected value σ_k^2)
- proposal distribution of proportions: Dirichlet (parameters $C \cdot \pi$)

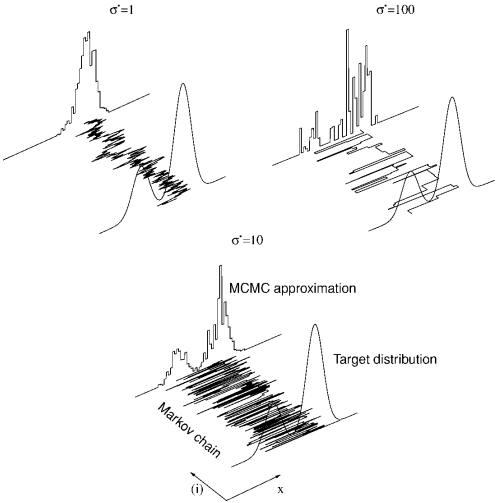
MH in practice



What if we, as before, simulate data from the final sample?

It is a blob; no clear separation of clusters (as opposed to real data)

Proposal distribution: large vs small steps?



Selecting the right conditional proposal distribution is important

Small step sizes:

- exploits the current position
- stays in one posterior mode
- may have difficulty finding high probability region

Large step sizes:

- explores the whole space
- moves between posterior modes easily
- may spend time in low probability regions

Convergence and mixing rates

- *Mixing rate* refers to the rate at which a Markov chain approaches the *stationary distribution*.
- Once a Markov chain is sampling according to the stationary distribution, it (by definition) will never leave.
- *Only when a Markov chain is in the stationary distribution* are we sampling from the *posterior distribution*.
- A number of modeling decisions can improve the mixing rate, including ample exploration.

MCMC: Gibbs sampling

Gibbs sampling, like importance sampling in MC approaches, accepts all proposal distributions.

Gibbs sampling selects a proposal distribution that, by definition, produces an acceptance probability equal to one.

Gibbs sampling is useful for sampling in a high dimensional space (as we are in with Gaussian mixture models)

MCMC: Gibbs sampling

Let $z = \{z_1, z_2, \ldots, z_p\}$ for p-dimensional space we are sampling (for GMMs, $p = 8$).

Define $p(z_j \mid z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_p)$, or, more succinctly, $p(z_j \mid z_{-j})$.

Let

$$q(z^* \mid z^s) = \begin{cases} p(z_j^* \mid z_{-j}^s) & \text{if } z_{-j}^* = z_{-j}^s \\ 0 & \text{otherwise} \end{cases}$$

Then we can see that the acceptance probability is always one.

MCMC: Gibbs sampling

Let $q(z^* \mid z^s) = p(z_j^* \mid z_{-j}^s)$ if $z_{-j}^* = z_{-j}^s$ (i.e., all other parameters stay the same), and 0 otherwise. Then,

$$\begin{aligned} A(z^*) &= \min\left\{1, \frac{p(z^*)q(z^s \mid z^*)}{p(z^s)q(z^* \mid z^s)}\right\} \\ &= \min\left\{1, \frac{p(z^*)p(z_j^s \mid z_{-j}^s)}{p(z^s)p(z_j^* \mid z_{-j}^*)}\right\} \\ &= \min\left\{1, \frac{p(z_{-j}^*)}{p(z_{-j}^s)}\right\} \\ &= 1. \end{aligned}$$

Practically, we do not have to reject samples.

Gibbs sampling: adding auxiliary variables

Gibbs sampling using conditional probabilities allows us to sample in much higher dimensions easily.

Often, as in the mixture model case, adding additional *auxiliary* variables *faster mixing*.

E.g., in a mixture model, auxiliary variables might be cluster assignments for each sample c.

For a mixture model, Gibbs sampling has the flavor of EM, except that both the E-step and M-step are samples from a conditional distribution.

Gibbs sampling: algorithm

Gibbs sampling

- Initialize z^1
- For $s = 1 : S$
  - sample $z_1^{s+1} \sim p(z_1 \mid x, z_2^s, z_3^s, \ldots, z_p^s)$
  - sample $z_2^{s+1} \sim p(z_2 \mid x, z_1^{s+1}, z_3^s, \ldots, z_p^s)$
  - sample $z_3^{s+1} \sim p(z_3 \mid x, z_1^{s+1}, z_2^{s+1}, \ldots, z_p^s)$

    $\vdots$
  - sample $z_p^{s+1} \sim p(z_3 \mid x, z_1^{s+1}, z_2^{s+1}, \ldots, z_{p-1}^{s+1})$

Gibbs sampling in practice

Gibbs sampling for a Gaussian mixture model

- assign each sample to one of K mixture components in c
- iterate S times:
  - for each parameter z_j, draw a sample from $p(z_j \mid z_{-j}, c^s, \mathbf{y})$.
  - given new parameter values, draw a sample of c_i:

$$c_i \sim Mult\left(\frac{\pi_k^{s+1}\mathcal{N}\left(\mu_k^{s+1}, \sigma_k^{2(s+1)}\right)}{\sum_{\ell=1}^{K}\pi_\ell^{s+1}\mathcal{N}\left(\mu_\ell^{s+1}, \sigma_\ell^{2(s+1)}\right)}\right).$$

Gibbs sampling in practice

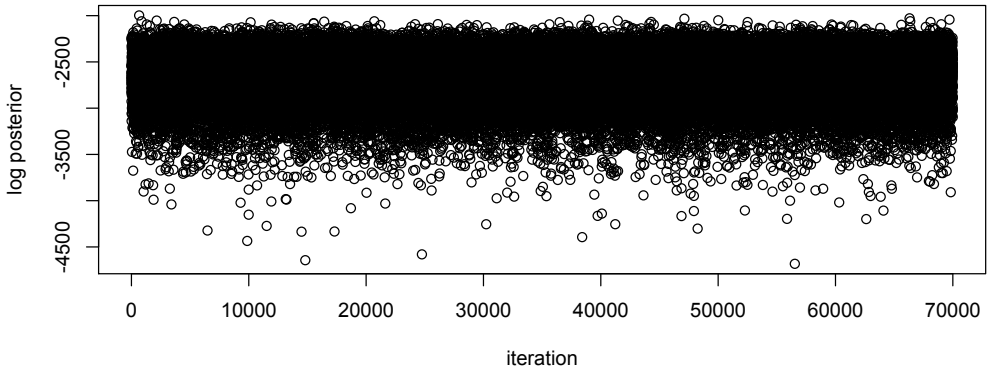
How do we compute these conditional probabilities?

- for each parameter z_j, draw a sample from $p(z_j \mid z_{-j}, c^s, \mathbf{y})$.
- for example, draw a sample of μ_k:

$$\begin{aligned} \mu_k^{s+1} &\sim p(\mu_k \mid z_{-\mu_k}, c^s, \mathbf{y}) \\ &\propto p(\mathbf{y} \mid \mu_{\mathbf{k}}, \mathbf{z_k}, \mathbf{c^s} = \mathbf{k}, \mathbf{y_k}) \mathbf{p}(\mu_{\mathbf{k}} \mid \mu_{\mathbf{0}}, \sigma_{\mathbf{0}}^{\mathbf{2}}) \\ &= \mathcal{N}(\mu_k \mid \mu_0, \sigma_0^2) \prod_{i=1}^{n} \mathcal{N}(y_i \mid \mu_k, \sigma_k^2, c_i^s = k) \end{aligned}$$

Apply Bayes rule and use the conjugacy of the model distributions and available conditional distributions.

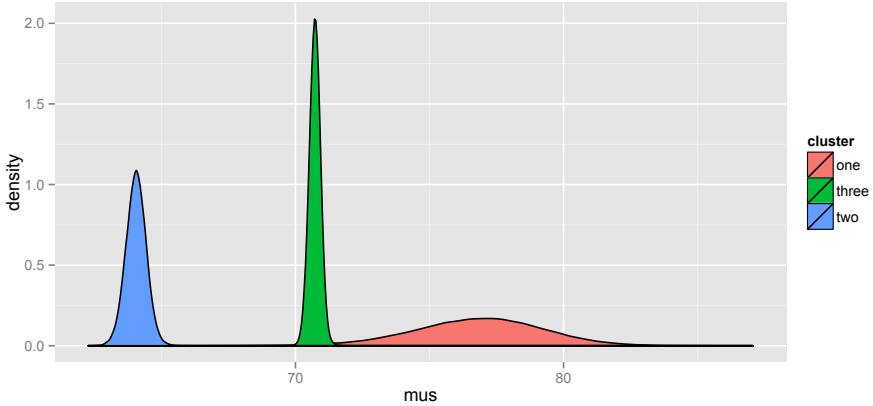
Gibbs sampling in practice



Ran 70,000 iterations of Gibbs sampling on a Gaussian mixture model.

- added auxiliary variables to assign each component to one cluster

Gibbs sampling in practice



- marginal posterior distributions of each cluster mean
- beautiful cluster separation
- terrible mixing – is this an advantage or a disadvantage?

Recap: discussed four sampling algorithms

Monte Carlo methods

- Rejection sampling: reject samples from proposal distribution
- Importance sampling: weight samples from proposal distribution

Markov chain Monte Carlo methods

- Metropolis-Hastings: add a Markov chain to the proposal distribution
- Gibbs sampling: sample from the conditional distribution for each parameter

Monte Carlo methods: background

Active area of research!

- theoretically: Markov chain questions, mixing rates, etc.
- practically: speeding up MCMC, parallelizing to large data
- MCMC ([Gelfand & Smith 1990]) "opened the MCMC candy shop," enabling practical Bayesian statistics
- Gibbs sampling is often (assuming your model has a proper posterior) the first thing you try to evaluate a model

Monte Carlo methods: history

"Being secret, the work of von Neumann and Ulam required a code name. A colleague of von Neumann and Ulam, Nicholas Metropolis, suggested using the name Monte Carlo, which refers to the Monte Carlo Casino in Monaco where Ulam's uncle would borrow money from relatives to gamble."

Monte Carlo Methods important in

- Manhattan project (von Neumann)
- Hydrogen bomb
- Physics, physical chemistry, and operations research

Wikipedia: Monte Carlo Methods

Additional Resources

- MLAPA: Chapter 24
- Andrieu et al. 2003 *An Introduction to MCMC for Machine Learning*
- MH Papers: [Hastings 1970] and [Metropolis et al. 1953]
- Gelfand & Smith 1990 *Sampling-Based Approaches to Calculating Marginal Densities*
- (video) Nando de Freitas *Monte Carlo simulation for statistical inference, model selection and decision making*
- Metacademy – Metropolis-Hastings
- Metacademy – Gibbs Sampling
- Metacademy – Slice Sampling
- Metacademy – Hamiltonian Monte Carlo