LECTURE 16: MARKOV CHAIN MONTE CARLO (CONTD)

STAT 545: Intro. to Computational Statistics

Vinayak Rao

Purdue University

November 14, 2018

Markov chain Monte Carlo

We are interested in a distribution $\pi(x) = \frac{f(x)}{Z}$

(e.g. want the mean, quantiles etc.)

Monte Carlo: approximate with independent samples from π

MCMC: produce dependent samples via a Markov chain

$$x_0 \to x_1 \to x_2 \to x_3 \to \cdots \to x_{N-1} \to x_N$$

Use dependent samples to approximate integrals w.r.t. $\pi(x)$:

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i) \approx \mathbb{E}_\pi[g] \quad \text{as}$$

MCMC

Let $\mathcal{T}(x_i \to x_{i+1})$ be the Markov transition kernel. We require:

MCMC

Let $\mathcal{T}(x_i \to x_{i+1})$ be the Markov transition kernel. We require:

- π is a stationary distribution of $\mathcal{T}$:

$$\pi(x) = \int_{\mathcal{X}} \pi(y)\mathcal{T}(y \to x)\mathrm{d}y$$

MCMC

Let $\mathcal{T}(x_i \to x_{i+1})$ be the Markov transition kernel. We require:

- π is a stationary distribution of $\mathcal{T}$:

$$\pi(x) = \int_{\mathcal{X}} \pi(y)\mathcal{T}(y \to x)\mathrm{d}y$$

Also require that $\mathcal{T}$ be

- irreducible: not stuck to some part of $\mathcal{X}$ forever

MCMC

Let $\mathcal{T}(x_i \to x_{i+1})$ be the Markov transition kernel. We require:

- π is a stationary distribution of $\mathcal{T}$:
$$\pi(x) = \int_{\mathcal{X}} \pi(y)\mathcal{T}(y \to x)\mathrm{d}y$$

Also require that $\mathcal{T}$ be

- irreducible: not stuck to some part of $\mathcal{X}$ forever
- aperiodic: not stuck to some part of $\mathcal{X}$ for e.g. even iterations. Can fix this with a ‘lazy’ Markov chain that allows self-transitions.

MCMC

Let $\mathcal{T}(x_i \to x_{i+1})$ be the Markov transition kernel. We require:

- π is a stationary distribution of $\mathcal{T}$:

$$\pi(x) = \int_{\mathcal{X}} \pi(y)\mathcal{T}(y \to x)\mathrm{d}y$$

Also require that $\mathcal{T}$ be

- irreducible: not stuck to some part of $\mathcal{X}$ forever
- aperiodic: not stuck to some part of $\mathcal{X}$ for e.g. even iterations. Can fix this with a ‘lazy’ Markov chain that allows self-transitions.

Finally, for infinite state-spaces (e.g. the real line), need an additional condition:

- positive recurrent: revisits every neighborhood infinitely often

Ergodicity

With these conditions, our chain is *ergodic*

For any initialization:

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i) \to \mathbb{E}_{\pi}[g] \quad \text{as } N \to \infty \qquad \text{(Ergodicity)}$$

Ergodicity

With these conditions, our chain is *ergodic*

For any initialization:

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i) \to \mathbb{E}_\pi[g] \quad \text{as } N \to \infty \qquad \text{(Ergodicity)}$$

We eventually forget the arbitrary initialization.

Typically, we discard the first B burn-in samples.

Ergodicity

With these conditions, our chain is *ergodic*

For any initialization:

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i) \to \mathbb{E}_\pi[g] \quad \text{as } N \to \infty \qquad \text{(Ergodicity)}$$

We eventually forget the arbitrary initialization.

Typically, we discard the first B burn-in samples.

A good transition kernel has:

- A short burn-in period.
- Fast mixing (small dependence across samples).

Markov chain Monte Carlo

The Markov transition kernel $\mathcal{T}$ must satisfy

$$\pi(x_{n+1}) = \int_{\mathcal{X}} \pi(x_n)\mathcal{T}(x_{n+1}|x_n)\mathrm{d}x_n$$

Markov chain Monte Carlo

The Markov transition kernel $\mathcal{T}$ must satisfy

$$\pi(x_{n+1}) = \int_{\mathcal{X}} \pi(x_n)\mathcal{T}(x_{n+1}|x_n)\mathrm{d}x_n$$

Usually, we enforce the stronger condition of detailed balance:

$$\pi(x_{n+1})\mathcal{T}(x_n|x_{n+1}) = \pi(x_n)\mathcal{T}(x_{n+1}|x_n)$$

(Sufficient but not necessary)

The problem

Given some probability density $\pi(x) = f(x)/Z$:

- How do you construct a transition kernel $\mathcal{T}$ with π as it's stationary distribution?
- How do you construct a *good* transition kernel

Focus of a huge literature.

The problem

Given some probability density $\pi(x) = f(x)/Z$:

- How do you construct a transition kernel $\mathcal{T}$ with π as it's stationary distribution?
- How do you construct a *good* transition kernel

Focus of a huge literature.

One approach: the Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm

The simplest and most widely applicable MCMC algorithm. Featured in Dongarra & Sullivan (2000)'s list of top 10 algoirithms.

1. Metropolis Algorithm for Monte Carlo
2. Simplex Method for Linear Programming
3. Krylov Subspace Iteration Methods
4. The Decompositional Approach to Matrix Computations
5. The Fortran Optimizing Compiler
6. QR Algorithm for Computing Eigenvalues
7. Quicksort Algorithm for Sorting
8. Fast Fourier Transform
9. Integer Relation Detection
10. Fast Multipole Method

The Metropolis-Hastings algorithm

A random walk algorithm

Choose a proposal distrib. $q(x_{new}|x_{old})$. E.g. $x_{new} \sim \mathcal{N}(x_{old}, \sigma^2 I)$

The Metropolis-Hastings algorithm

A random walk algorithm

Choose a proposal distrib. $q(x_{new}|x_{old})$. E.g. $x_{new} \sim \mathcal{N}(x_{old}, \sigma^2 I)$

Initialize chain at some starting point x_0.

Repeat:

- Propose a new point x^* according to $q(x^*|x_n)$.
- Define $\alpha = \min\left(1, \frac{\pi(x^*)q(x_n|x^*)}{\pi(x_n)q(x^*|x_n)}\right) = \min\left(1, \frac{f(x^*)q(x_n|x^*)}{f(x_n)q(x^*|x_n)}\right)$
- Set $x_{n+1} = x^*$ with probability α, else $x_{n+1} = x_n$.

The Metropolis-Hastings algorithm

A random walk algorithm

Choose a proposal distrib. $q(x_{new}|x_{old})$. E.g. $x_{new} \sim \mathcal{N}(x_{old}, \sigma^2 I)$

Initialize chain at some starting point x_0.

Repeat:

- Propose a new point x^* according to $q(x^*|x_n)$.
- Define $\alpha = \min\left(1, \frac{\pi(x^*)q(x_n|x^*)}{\pi(x_n)q(x^*|x_n)}\right) = \min\left(1, \frac{f(x^*)q(x_n|x^*)}{f(x_n)q(x^*|x_n)}\right)$
- Set $x_{n+1} = x^*$ with probability α, else $x_{n+1} = x_n$.

Comments:

- Do not need to calculate the normalization constant Z.
- Accept/reject steps ensure this has the correct distribution.

The Metropolis-Hastings algorithm

A random walk algorithm

Choose a proposal distrib. $q(x_{new}|x_{old})$. E.g. $x_{new} \sim \mathcal{N}(x_{old}, \sigma^2 I)$

Initialize chain at some starting point x_0.

Repeat:

- Propose a new point x^* according to $q(x^*|x_n)$.
- Define $\alpha = \min\left(1, \frac{\pi(x^*)q(x_n|x^*)}{\pi(x_n)q(x^*|x_n)}\right) = \min\left(1, \frac{f(x^*)q(x_n|x^*)}{f(x_n)q(x^*|x_n)}\right)$
- Set $x_{n+1} = x^*$ with probability α, else $x_{n+1} = x_n$.

Comments:

- Do not need to calculate the normalization constant Z.
- Accept/reject steps ensure this has the correct distribution.
- On rejection, keep old sample (i.e. there will be repetition)

The Metropolis-Hastings algorithm

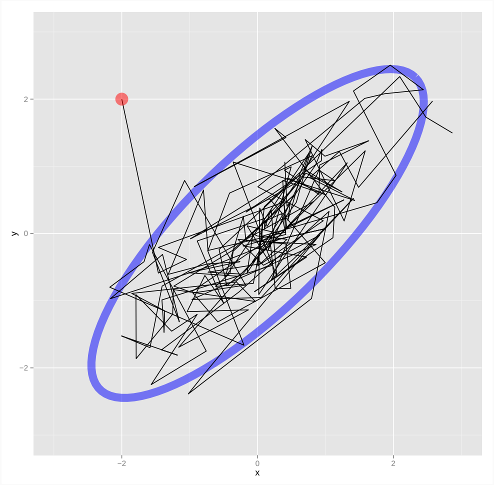
For a symmetric proposal ($q(x^*|x_n) = q(x_n|x^*)$):

$$\alpha = \min\left(1, \frac{f(x^*)}{f(x_n)}\right)$$

The Metropolis algorithm.

The Metropolis-Hastings algorithm

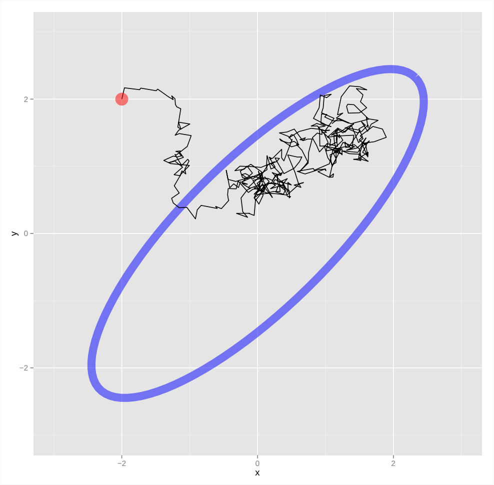
How do we chose the proposal variance?



$$\sigma^2 = 1$$

The Metropolis-Hastings algorithm

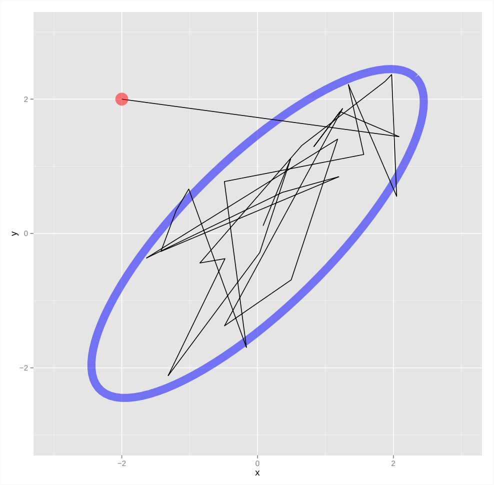
How do we chose the proposal variance?



$$\sigma^2 = .1$$

The Metropolis-Hastings algorithm

How do we chose the proposal variance?



$$\sigma^2 = 5$$

Does this satisfy detailed balance?

First, what is the transition kernel $\mathcal{T}(x_{n+1}|x_n)$?

Does this satisfy detailed balance?

First, what is the transition kernel $\mathcal{T}(x_{n+1}|x_n)$?

Prob. of moving from x_n to x_{n+1} is $\alpha(x_{n+1}, x_n)q(x_{n+1}|x_n)$.

Does this satisfy detailed balance?

First, what is the transition kernel $\mathcal{T}(x_{n+1}|x_n)$?

Prob. of moving from x_n to x_{n+1} is $\alpha(x_{n+1}, x_n)q(x_{n+1}|x_n)$.

Prob. of accepting move at x_n is $\alpha(x) = \int_{\mathcal{X}} \alpha(y, x_n)q(y|x_n)\mathrm{d}y$.

Does this satisfy detailed balance?

First, what is the transition kernel $\mathcal{T}(x_{n+1}|x_n)$?

Prob. of moving from x_n to x_{n+1} is $\alpha(x_{n+1}, x_n)q(x_{n+1}|x_n)$.

Prob. of accepting move at x_n is $\alpha(x) = \int_{\mathcal{X}} \alpha(y, x_n)q(y|x_n)\mathrm{d}y$.

Prob. of rejection at x_n is $r(x_n) = 1 - \alpha(x_n)$.

Does this satisfy detailed balance?

First, what is the transition kernel $\mathcal{T}(x_{n+1}|x_n)$?

Prob. of moving from x_n to x_{n+1} is $\alpha(x_{n+1}, x_n)q(x_{n+1}|x_n)$.

Prob. of accepting move at x_n is $\alpha(x) = \int_{\mathcal{X}} \alpha(y, x_n)q(y|x_n)\mathrm{d}y$.

Prob. of rejection at x_n is $r(x_n) = 1 - \alpha(x_n)$.

We then have:

$$\mathcal{T}(x_{n+1}|x_n) = \alpha(x_{n+1}, x_n)q(x_{n+1}|x_n) + r(x_n)\delta(x_n = x_{n+1})$$

Does this satisfy detailed balance?

First, what is the transition kernel $\mathcal{T}(x_{n+1}|x_n)$?

Prob. of moving from x_n to x_{n+1} is $\alpha(x_{n+1}, x_n)q(x_{n+1}|x_n)$.

Prob. of accepting move at x_n is $\alpha(x) = \int_{\mathcal{X}} \alpha(y, x_n)q(y|x_n)\mathrm{d}y$.

Prob. of rejection at x_n is $r(x_n) = 1 - \alpha(x_n)$.

We then have:

$$\mathcal{T}(x_{n+1}|x_n) = \alpha(x_{n+1}, x_n)q(x_{n+1}|x_n) + r(x_n)\delta(x_n = x_{n+1})$$

We want to show detailed balance:

$$\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$$

The Metropolis-Hastings algorithm

Detailed balance: $\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$

Consider the LHS:

$$\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_n)\left(\alpha(x_{n+1}, x_n)q(x_{n+1}, x_n) + r(x_n)\delta(x_n = x_{n+1})\right)$$

The Metropolis-Hastings algorithm

Detailed balance: $\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$

Consider the LHS:

$$\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_n)\left(\alpha(x_{n+1}, x_n)q(x_{n+1}, x_n) + r(x_n)\delta(x_n = x_{n+1})\right)$$

The first term is: $$\frac{f(x_n)}{Z}\min\left(1, \frac{f(x_{n+1})q(x_n|x_{n+1})}{f(x_n)q(x_{n+1}|x_n)}\right)q(x_{n+1}|x_n)$$

The Metropolis-Hastings algorithm

Detailed balance: $\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$

Consider the LHS:

$$\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_n)\left(\alpha(x_{n+1},x_n)q(x_{n+1},x_n) + r(x_n)\delta(x_n = x_{n+1})\right)$$

The first term is:

$$\begin{aligned} &\frac{f(x_n)}{Z}\min\left(1, \frac{f(x_{n+1})q(x_n|x_{n+1})}{f(x_n)q(x_{n+1}|x_n)}\right)q(x_{n+1}|x_n) \\ &= \frac{1}{Z}\min\left(f(x_n)q(x_{n+1}|x_n), f(x_{n+1})q(x_n|x_{n+1})\right) \end{aligned}$$

The Metropolis-Hastings algorithm

Detailed balance: $\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$

Consider the LHS:

$$\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_n)\left(\alpha(x_{n+1}, x_n)q(x_{n+1}, x_n) + r(x_n)\delta(x_n = x_{n+1})\right)$$

The first term is:

$$\begin{aligned}\frac{f(x_n)}{Z}\min\left(1, \frac{f(x_{n+1})q(x_n|x_{n+1})}{f(x_n)q(x_{n+1}|x_n)}\right)q(x_{n+1}|x_n) \\ = \frac{1}{Z}\min\left(f(x_n)q(x_{n+1}|x_n), f(x_{n+1})q(x_n|x_{n+1})\right)\end{aligned}$$

First term of RHS has this form too.

The Metropolis-Hastings algorithm

Detailed balance: $\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$

Consider the LHS:

$$\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_n)\left(\alpha(x_{n+1}, x_n)q(x_{n+1}, x_n) + r(x_n)\delta(x_n = x_{n+1})\right)$$

The first term is:

$$\begin{aligned} &\frac{f(x_n)}{Z}\min\left(1, \frac{f(x_{n+1})q(x_n|x_{n+1})}{f(x_n)q(x_{n+1}|x_n)}\right)q(x_{n+1}|x_n) \\ &= \frac{1}{Z}\min\left(f(x_n)q(x_{n+1}|x_n), f(x_{n+1})q(x_n|x_{n+1})\right) \end{aligned}$$

First term of RHS has this form too.

For the second term, $r(x_n)\delta(x_n = x_{n+1}) = r(x_{n+1})\delta(x_{n+1} = x_n)$

The Metropolis-Hastings algorithm

Detailed balance: $\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$

Consider the LHS:

$$\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_n)\left(\alpha(x_{n+1}, x_n)q(x_{n+1}, x_n) + r(x_n)\delta(x_n = x_{n+1})\right)$$

The first term is:
$$\begin{aligned} &\frac{f(x_n)}{Z}\min\left(1, \frac{f(x_{n+1})q(x_n|x_{n+1})}{f(x_n)q(x_{n+1}|x_n)}\right)q(x_{n+1}|x_n) \\ &= \frac{1}{Z}\min\left(f(x_n)q(x_{n+1}|x_n), f(x_{n+1})q(x_n|x_{n+1})\right) \end{aligned}$$

First term of RHS has this form too.

For the second term, $r(x_n)\delta(x_n = x_{n+1}) = r(x_{n+1})\delta(x_{n+1} = x_n)$

Thus, $\pi(x_n)\mathcal{T}(x_{n+1}|x_n) = \pi(x_{n+1})\mathcal{T}(x_n|x_{n+1})$

Gibbs sampling

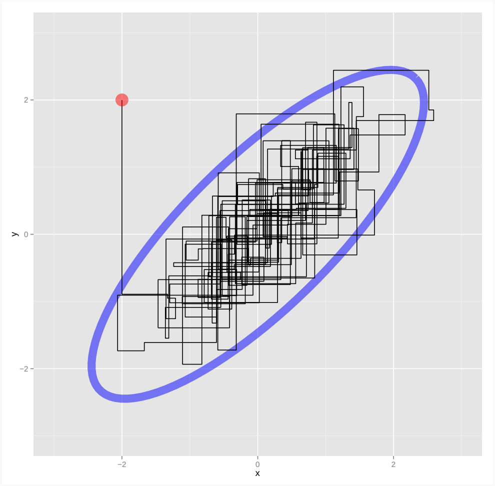
Consider a Markov chain over a set of variables $(x_1, \cdots, x_d)$.

Gibbs sampling cycles though these sequentially (or randomly). At the *i*th step, it updates x_i conditioned on the the rest:

$$x_i \sim \pi(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \pi(x_i | \mathbf{x}_{\setminus i})$$

Often these conditionals have a much simpler form than the joint.

Gibbs sampling



Detailed balance for the sequential Gibbs sampler

Does it satisfy stationarity?

Does it satisfy irreducibility?

Is it aperiodic?

Detailed balance for the randomized Gibbs sampler

Suppose we update component i with prob. ρ_i. Let x and x' differ only in component i. Then:

$$\mathcal{T}(\mathsf{x}'|\mathsf{x}) = \rho_i \pi(x_i'|\mathsf{x}_{\setminus i})$$

Also

$$\begin{aligned} \pi(\mathsf{x})\mathcal{T}(\mathsf{x}'|\mathsf{x}) &= \pi(\mathsf{x})\rho_i \pi(x_i'|\mathsf{x}_{\setminus i}) \\ &= \pi(\mathsf{x}_{\setminus i})\pi(x_i|\mathsf{x}_{\setminus i})\rho_i \pi(x_i'|\mathsf{x}_{\setminus i}) \end{aligned}$$

From symmetry (or by calculating RHS), we have detailed balance.

Detailed balance for Gibbs sampler

Under mild conditions, Gibbs sampling is irreducible.

Detailed balance for Gibbs sampler

Under mild conditions, Gibbs sampling is irreducible.

Can break down under constraints. E.g. two perfectly couple variables.

Performance deteriorates with strong coupling between variables.

Poor mixing due to coupled variables is always a concern.

Detailed balance for Gibbs sampler

Under mild conditions, Gibbs sampling is irreducible.
Can break down under constraints. E.g. two perfectly couple variables.
Performance deteriorates with strong coupling between variables.
Poor mixing due to coupled variables is always a concern.

Advantages: Simple, with no free parameters.

Detailed balance for Gibbs sampler

Under mild conditions, Gibbs sampling is irreducible.
Can break down under constraints. E.g. two perfectly couple variables.
Performance deteriorates with strong coupling between variables.
Poor mixing due to coupled variables is always a concern.

Advantages: Simple, with no free parameters.
Often, conditional independencies in a model along with suitable conjugate priors allow efficient ‘blocked-Gibbs samplers’.

Gibbs sampling

More generally, Gibbs sampling can update more than one component at each step.

E.g. consider a Markov chain over 5 variables $(x_1, x_2, x_3, x_4, x_5)$.

Alternately updating $x_1, x_2 | x_3, x_4, x_5$, and then $x_3, x_4, x_5 | x_1, x_2$ form a correct Gibbs sampler

- If simulating these conditionals is easy, this can be more efficient than naively sampling one component at a time.

Gibbs sampling

More generally, Gibbs sampling can update more than one component at each step.

E.g. consider a Markov chain over 5 variables $(x_1, x_2, x_3, x_4, x_5)$.

Alternately updating $x_1, x_2 | x_3, x_4, x_5$, and then $x_3, x_4, x_5 | x_1, x_2$ form a correct Gibbs sampler

- If simulating these conditionals is easy, this can be more efficient than naively sampling one component at a time.

Can also have overlaps: e.g. update $x_1, x_2, x_3 | x_5$, and then $x_3, x_4, x_5 | x_1, x_2$ form a correct Gibbs sampler

Convince yourself this is correct

Back to the Mixture of Gaussians (MoG)

Observations come from one of K components

- each component is a Gaussian
- Component c has parameters $\theta_c = (\mu_c, \Sigma_c)$, its mean and covariance

To generate the ith observation:

$$c_i \sim \pi \qquad \text{Sample it's cluster assignment}$$

$$x_i \sim \mathcal{N}(x_i | \mu_{c_i}, \Sigma_{c_i}) \qquad \text{Sample it's value}$$

Back to the Mixture of Gaussians (MoG)

Observations come from one of *K* components

- each component is a Gaussian
- Component *c* has parameters $\theta_c = (\mu_c, \Sigma_c)$, its mean and covariance

To generate the *i*th observation:

$$\begin{aligned} c_i &\sim \pi && \text{Sample it's cluster assignment} \\ x_i &\sim \mathcal{N}(x_i \mid \mu_{c_i}, \Sigma_{c_i}) && \text{Sample it's value} \end{aligned}$$

Given data *X*, how did we estimate the parameters $\pi, \{\mu_c, \Sigma_c\}$?

Back to the Mixture of Gaussians (MoG)

Observations come from one of K components

- each component is a Gaussian
- Component c has parameters $\theta_c = (\mu_c, \Sigma_c)$, its mean and covariance

To generate the ith observation:

$$\begin{aligned} c_i &\sim \pi && \text{Sample it's cluster assignment} \\ x_i &\sim \mathcal{N}(x_i | \mu_{c_i}, \Sigma_{c_i}) && \text{Sample it's value} \end{aligned}$$

Given data X, how did we estimate the parameters $\pi, \{\mu_c, \Sigma_c\}$?

What does a Bayesian approach involve?

Bayesian inference for MoG

Place a prior over π: conjugate is a Dirichlet prior

Place a prior over the cluster parameters $\theta_c = (\mu_c, \Sigma_c)$: conjugate is the normal inverse-Wishart distribution.

Bayesian inference for MoG

Place a prior over π: conjugate is a Dirichlet prior

Place a prior over the cluster parameters $\theta_c = (\mu_c, \Sigma_c)$: conjugate is the normal inverse-Wishart distribution.

Given data $X = \{x_1, \cdots, x_N\}$, we want to sample from the distribution $p(\pi, \{\mu_c, \Sigma_c\}_{c=1}^{K} | X)$

Bayesian inference for MoG (contd.)

We will sample from the distribution $p(C, \pi, \{\mu_c, \Sigma_c\}_{c=1}^K | X)$, including the cluster assignments $C = \{c_1, \cdots, c_N\}$.

We do this by Gibbs sampling

Bayesian inference for MoG (contd.)

We will sample from the distribution $p(C, \pi, \{\mu_c, \Sigma_c\}_{c=1}^{K} | X)$, including the cluster assignments $C = \{c_1, \cdots, c_N\}$.

We do this by Gibbs sampling

How do we do this?

Gibbs sampling for MoG

Initialize MCMC chain, by randomly assigning observations to one of the K clusters.

Then, repeat:

1) Given cluster assignments C, sample π and $\{\mu_c, \Sigma_c\}$

- These are simple conjugate distributions (think about this and HW 5)

Gibbs sampling for MoG

Initialize MCMC chain, by randomly assigning observations to one of the K clusters.

Then, repeat:

1) Given cluster assignments C, sample π and $\{\mu_c, \Sigma_c\}$

- These are simple conjugate distributions (think about this and HW 5)

2) Given parameters, sample cluster assignments C.

- We did this for the case of the EM algorithm: just simulate from the cluster ”responsibilities”