Approximate Inference: Sampling Methods (3)

Piyush Rai

Probabilistic Machine Learning (CS772A)

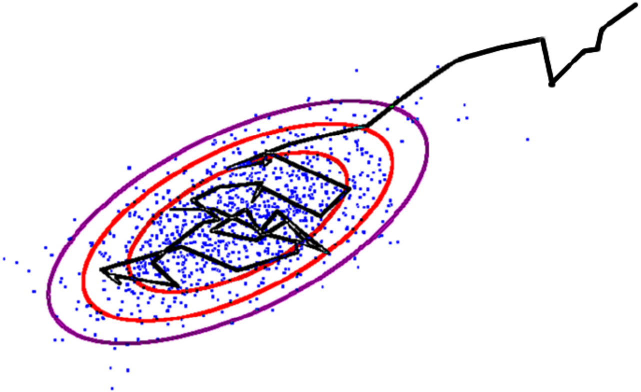
Oct 5, 2017

Recap

Markov Chain Monte Carlo (MCMC)

Generates samples from a target distribution by following a first-order Markov chain

$$z^{(1)} \rightarrow z^{(2)} \rightarrow \ldots \rightarrow z^{(L)}$$



Markov Chain

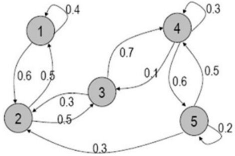
- Consider a sequence of random variables $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(L)}$
- A first-order Markov Chain assumes

$$p(\boldsymbol{z}^{(\ell+1)}|\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(\ell)}) = p(\boldsymbol{z}^{(\ell+1)}|\boldsymbol{z}^{(\ell)}) \qquad \forall \ell$$

- A first order Markov chain can be defined by the following
  - An initial state distribution $p(\boldsymbol{z}^{(0)})$
  - Transition probabilities $T_\ell(\boldsymbol{z}^{(\ell)}, \boldsymbol{z}^{(\ell+1)})$ define our proposal distribution $q(\boldsymbol{z}^{(\ell+1)}|\boldsymbol{z}^{(\ell)})$

Transition probabilities can be defined using a *K*x*K* table if **z** is a discrete r.v. with *K* possible values

$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0.4 & 0.6 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.0 & 0.7 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.3 & 0.6 \\ 0.0 & 0.3 & 0.0 & 0.5 & 0.2 \end{bmatrix} \end{array}$$



Markov Chain

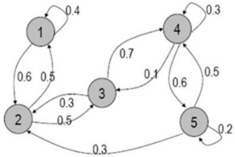
- Consider a sequence of random variables $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(L)}$
- A first-order Markov Chain assumes

$$p(\boldsymbol{z}^{(\ell+1)}|\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(\ell)}) = p(\boldsymbol{z}^{(\ell+1)}|\boldsymbol{z}^{(\ell)}) \qquad \forall \ell$$

- A first order Markov chain can be defined by the following
  - An initial state distribution $p(\boldsymbol{z}^{(0)})$
  - Transition probabilities $T_\ell(\boldsymbol{z}^{(\ell)}, \boldsymbol{z}^{(\ell+1)})$ define our proposal distribution $q(\boldsymbol{z}^{(\ell+1)}|\boldsymbol{z}^{(\ell)})$

Transition probabilities can be defined using a KxK table if $\mathbf{z}$ is a discrete r.v. with K possible values

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0.4 & 0.6 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.0 & 0.7 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.3 & 0.6 \\ 0.0 & 0.3 & 0.0 & 0.5 & 0.2 \end{bmatrix} \end{array}$$



- Homogeneous Markov Chain: Transition probabilities $T_\ell = T$ (same everywhere along the chain)

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(z)}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

- Initialize $\boldsymbol{z}^{(0)}$

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

- Initialize $\boldsymbol{z}^{(0)}$
- For $\tau = 0 : T - 1$

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

- Initialize $\boldsymbol{z}^{(0)}$
- For $\tau = 0 : T - 1$
    - Sample $u \sim U(0, 1)$

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

- Initialize $\boldsymbol{z}^{(0)}$
- For $\tau = 0 : T - 1$
    - Sample $u \sim U(0, 1)$
    - Sample $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

- Initialize $\boldsymbol{z}^{(0)}$
- For $\tau = 0 : T - 1$
  - Sample $u \sim U(0, 1)$
  - Sample $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$
  - If $u < A(\boldsymbol{z}^*, \boldsymbol{z}^{(\tau)})$ where A is defined as above

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

- Initialize $\boldsymbol{z}^{(0)}$
- For $\tau = 0 : T - 1$
  - Sample $u \sim U(0, 1)$
  - Sample $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$
  - If $u < A(\boldsymbol{z}^*, \boldsymbol{z}^{(\tau)})$ where A is defined as above
    - $\boldsymbol{z}^{(\tau+1)} = \boldsymbol{z}^*$

Metropolis-Hastings (MH) Sampling

- Goal: Generate samples from a distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$. Assume $\tilde{p}(\boldsymbol{z})$ can be evaluated for any $\boldsymbol{z}$
- Given the current sample $\boldsymbol{z}^{(\tau)}$, assume a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ for the next sample
- In each step, draw $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$ and accept the sample $\boldsymbol{z}^*$ with probability

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

The overall algorithm for MH sampling will be as follows

- Initialize $\boldsymbol{z}^{(0)}$
- For $\tau = 0 : T-1$
    - Sample $u \sim U(0,1)$
    - Sample $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\tau)})$
    - If $u < A(\boldsymbol{z}^*, \boldsymbol{z}^{(\tau)})$ where A is defined as above
        - $\boldsymbol{z}^{(\tau+1)} = \boldsymbol{z}^*$
    - Else
        - $\boldsymbol{z}^{(\tau+1)} = \boldsymbol{z}^{(\tau)}$

Gibbs Sampling (Geman & Geman, 1984)

- Suppose we wish to sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$

Gibbs Sampling (Geman & Geman, 1984)

- Suppose we wish to sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$
- Suppose we can't sample from $p(\boldsymbol{z})$ but can still sample from each local conditional $p(z_i|\boldsymbol{z}_{-i})$

Gibbs Sampling (Geman & Geman, 1984)

- Suppose we wish to sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$
- Suppose we can't sample from $p(\boldsymbol{z})$ but can still sample from each local conditional $p(z_i|\boldsymbol{z}_{-i})$
    - Can we done easily if we have a locally conjugate model (e.g., Gaussian matrix factorization)

Gibbs Sampling (Geman & Geman, 1984)

- Suppose we wish to sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$
- Suppose we can't sample from $p(\boldsymbol{z})$ but can still sample from each local conditional $p(z_i|\boldsymbol{z}_{-i})$
  - Can we done easily if we have a locally conjugate model (e.g., Gaussian matrix factorization)
- Gibbs sampling uses the conditionals $p(z_i|\boldsymbol{z}_{-i})$ as the proposal distribution for sampling z_i

Gibbs Sampling (Geman & Geman, 1984)

- Suppose we wish to sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$
- Suppose we can't sample from $p(\boldsymbol{z})$ but can still sample from each local conditional $p(z_i|\boldsymbol{z}_{-i})$
  - Can we done easily if we have a locally conjugate model (e.g., Gaussian matrix factorization)
- Gibbs sampling uses the conditionals $p(z_i|\boldsymbol{z}_{-i})$ as the proposal distribution for sampling z_i
- Gibbs sampling samples from these conditionals in a cyclic order

Gibbs Sampling (Geman & Geman, 1984)

- Suppose we wish to sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$
- Suppose we can't sample from $p(\boldsymbol{z})$ but can still sample from each local conditional $p(z_i|\boldsymbol{z}_{-i})$
  - Can we done easily if we have a locally conjugate model (e.g., Gaussian matrix factorization)
- Gibbs sampling uses the conditionals $p(z_i|\boldsymbol{z}_{-i})$ as the proposal distribution for sampling z_i
- Gibbs sampling samples from these conditionals in a cyclic order
- Gibbs sampling is equivalent to Metropolis Hastings sampling with **acceptance prob. $= 1$**

$$A(\boldsymbol{z}^*, \boldsymbol{z}) = \frac{p(\boldsymbol{z}^*)q(\boldsymbol{z}|\boldsymbol{z}^*)}{p(\boldsymbol{z})q(\boldsymbol{z}^*|\boldsymbol{z})} = \frac{p(z_i^*|\boldsymbol{z}_{-i}^*)p(\boldsymbol{z}_{-i}^*)p(z_i|\boldsymbol{z}_{-i}^*)}{p(z_i|\boldsymbol{z}_{-i})p(\boldsymbol{z}_{-i})p(z_i^*|\boldsymbol{z}_{-i})} = 1$$

where we use the fact that $\boldsymbol{z}_{-i}^* = \boldsymbol{z}_{-i}$

Gibbs Sampling (Geman & Geman, 1984)

- Suppose we wish to sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$
- Suppose we can't sample from $p(\boldsymbol{z})$ but can still sample from each local conditional $p(z_i|\boldsymbol{z}_{-i})$
  - Can we done easily if we have a locally conjugate model (e.g., Gaussian matrix factorization)
- Gibbs sampling uses the conditionals $p(z_i|\boldsymbol{z}_{-i})$ as the proposal distribution for sampling z_i
- Gibbs sampling samples from these conditionals in a cyclic order
- Gibbs sampling is equivalent to Metropolis Hastings sampling with **acceptance prob. = 1**

$$A(\boldsymbol{z}^*, \boldsymbol{z}) = \frac{p(\boldsymbol{z}^*)q(\boldsymbol{z}|\boldsymbol{z}^*)}{p(\boldsymbol{z})q(\boldsymbol{z}^*|\boldsymbol{z})} = \frac{p(z_i^*|\boldsymbol{z}_{-i}^*)p(\boldsymbol{z}_{-i}^*)p(z_i|\boldsymbol{z}_{-i}^*)}{p(z_i|\boldsymbol{z}_{-i})p(\boldsymbol{z}_{-i})p(z_i^*|\boldsymbol{z}_{-i})} = 1$$

  where we use the fact that $\boldsymbol{z}_{-i}^* = \boldsymbol{z}_{-i}$

- **Note:** Even <u>w/o local conjugacy</u>, if we can sample from local conditionals, Gibbs sampling applies!

Gibbs Sampling: Sketch of the Algorithm

M: Total number of unknowns to be sampled ($\boldsymbol{z} = [z_1, \ldots, z_M]$), T: number of Gibbs iterations

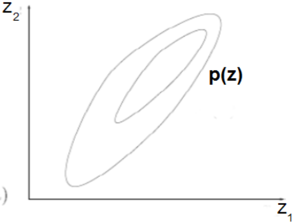
1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.

     $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.

     $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.

Gibbs Sampling: A Simple Example

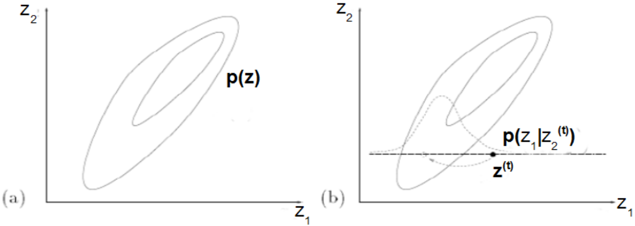
Can sample from a 2-D Gaussian using 1-D Gaussians (recall that if the joint distribution is a 2-D Gaussian, conditionals will simply be 1-D Gaussians)



(a)

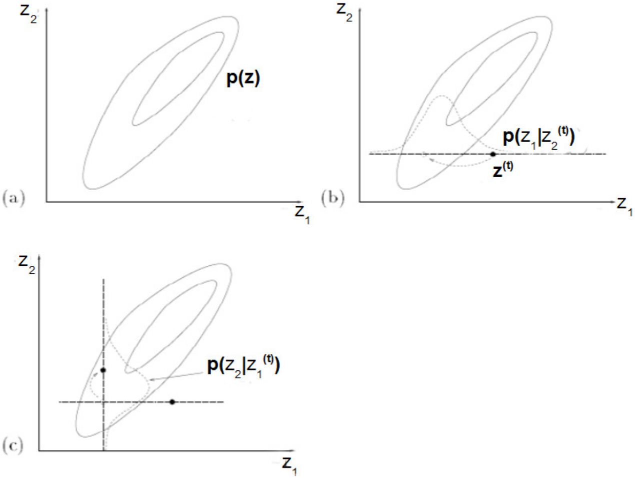
Gibbs Sampling: A Simple Example

Can sample from a 2-D Gaussian using 1-D Gaussians (recall that if the joint distribution is a 2-D Gaussian, conditionals will simply be 1-D Gaussians)



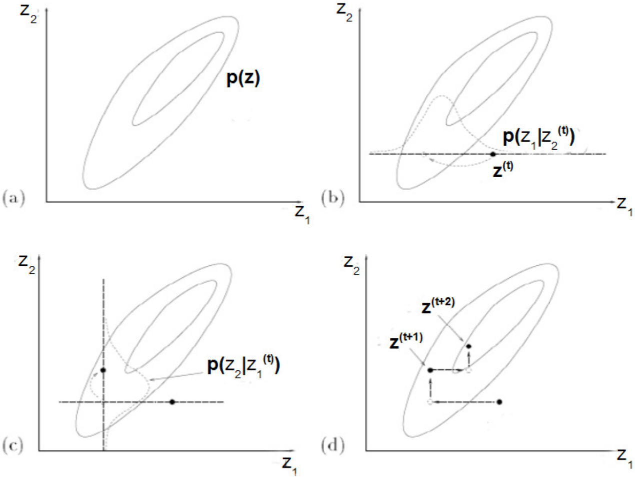
Gibbs Sampling: A Simple Example

Can sample from a 2-D Gaussian using 1-D Gaussians (recall that if the joint distribution is a 2-D Gaussian, conditionals will simply be 1-D Gaussians)



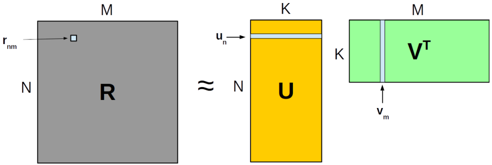
Gibbs Sampling: A Simple Example

Can sample from a 2-D Gaussian using 1-D Gaussians (recall that if the joint distribution is a 2-D Gaussian, conditionals will simply be 1-D Gaussians)



Some Other Examples of Gibbs Sampling

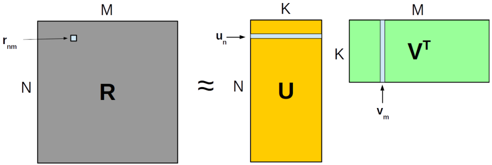
Gibbs Sampling for Probabilistic Matrix Factorization



- Recall the low-rank probabilistic matrix factorization model

$$r_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}$$

Gibbs Sampling for Probabilistic Matrix Factorization

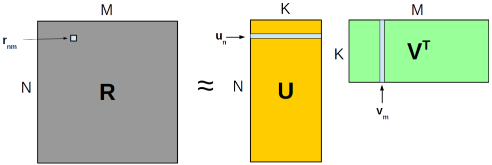


- Recall the low-rank probabilistic matrix factorization model

$$r_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}$$

- Assuming $\epsilon_{ij} \sim \mathcal{N}(\epsilon_{ij}|0, \beta^{-1})$, we have the following Gaussian likelihood for each observation

Gibbs Sampling for Probabilistic Matrix Factorization



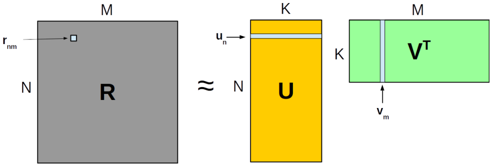
- Recall the low-rank probabilistic matrix factorization model

$$r_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}$$

- Assuming $\epsilon_{ij} \sim \mathcal{N}(\epsilon_{ij}|0, \beta^{-1})$, we have the following Gaussian likelihood for each observation

$$p(r_{ij}|\boldsymbol{u}_i, \boldsymbol{v}_j) = \mathcal{N}(r_{ij}|\boldsymbol{u}_i^\top \boldsymbol{v}_j, \beta^{-1})$$

Gibbs Sampling for Probabilistic Matrix Factorization



- Recall the low-rank probabilistic matrix factorization model

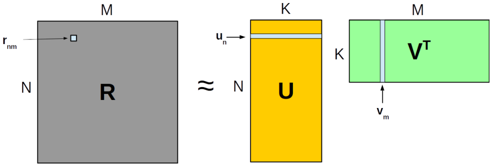
$$r_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}$$

- Assuming $\epsilon_{ij} \sim \mathcal{N}(\epsilon_{ij}|0, \beta^{-1})$, we have the following Gaussian likelihood for each observation

$$p(r_{ij}|\boldsymbol{u}_i, \boldsymbol{v}_j) = \mathcal{N}(r_{ij}|\boldsymbol{u}_i^\top \boldsymbol{v}_j, \beta^{-1})$$

- With Gaussian priors $p(\boldsymbol{u}_i) = \mathcal{N}(\boldsymbol{u}_i|\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K)$, $p(\boldsymbol{v}_j) = \mathcal{N}(\boldsymbol{v}_j|\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$, we have local conjugacy

Gibbs Sampling for Probabilistic Matrix Factorization



- Recall the low-rank probabilistic matrix factorization model

$$r_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}$$

- Assuming $\epsilon_{ij} \sim \mathcal{N}(\epsilon_{ij}|0, \beta^{-1})$, we have the following Gaussian likelihood for each observation

$$p(r_{ij}|\boldsymbol{u}_i, \boldsymbol{v}_j) = \mathcal{N}(r_{ij}|\boldsymbol{u}_i^\top \boldsymbol{v}_j, \beta^{-1})$$

- With Gaussian priors $p(\boldsymbol{u}_i) = \mathcal{N}(\boldsymbol{u}_i|\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K)$, $p(\boldsymbol{v}_j) = \mathcal{N}(\boldsymbol{v}_j|\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$, we have local conjugacy
- Local conditional posteriors for $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ have closed form (lecture 13). Simple Gibbs sampling!

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$
2. For step $s = 1, \ldots, S$

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$
2. For step $s = 1, \ldots, S$
   - Sample $\mathbf{U}^{(s)}$: For $i = 1, \ldots, N$, sample a new $\boldsymbol{u}_i$ from its conditional posterior

$$\boldsymbol{u}_i^{(s)} \sim \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{\mu}_{u_i}^{(s)}, \boldsymbol{\Sigma}_{u_i}^{(s)})$$

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$
2. For step $s = 1, \ldots, S$
   - Sample $\mathbf{U}^{(s)}$: For $i = 1, \ldots, N$, sample a new $\boldsymbol{u}_i$ from its conditional posterior

$$\boldsymbol{u}_i^{(s)} \sim \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{\mu}_{u_i}^{(s)}, \boldsymbol{\Sigma}_{u_i}^{(s)})$$

where $\boldsymbol{\Sigma}_{u_i}^{(s)} = (\lambda_u \mathbf{I} + \beta \sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j^{(s-1)} \boldsymbol{v}_j^{(s-1)\top})^{-1}$ and $\boldsymbol{\mu}_{u_i}^{(s)} = \boldsymbol{\Sigma}_{u_i}^{(s)}(\beta \sum_{j:(i,j)\in\Omega} r_{ij} \boldsymbol{v}_j^{(s-1)})$

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$
2. For step $s = 1, \ldots, S$
   - Sample $\mathbf{U}^{(s)}$: For $i = 1, \ldots, N$, sample a new $\boldsymbol{u}_i$ from its conditional posterior

$$\boldsymbol{u}_i^{(s)} \sim \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{\mu}_{u_i}^{(s)}, \boldsymbol{\Sigma}_{u_i}^{(s)})$$

   where $\boldsymbol{\Sigma}_{u_i}^{(s)} = (\lambda_u \mathbf{I} + \beta \sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j^{(s-1)} \boldsymbol{v}_j^{(s-1)\top})^{-1}$ and $\boldsymbol{\mu}_{u_i}^{(s)} = \boldsymbol{\Sigma}_{u_i}^{(s)}(\beta \sum_{j:(i,j)\in\Omega} r_{ij} \boldsymbol{v}_j^{(s-1)})$
   - Sample $\mathbf{V}^{(s)}$: For $j = 1, \ldots, M$, sample a new $\boldsymbol{v}_j$ from its conditional posterior

$$\boldsymbol{v}_j^{(s)} \sim \mathcal{N}(\boldsymbol{v}_j | \boldsymbol{\mu}_{v_j}^{(s)}, \boldsymbol{\Sigma}_{v_j}^{(s)})$$

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$
2. For step $s = 1, \ldots, S$
   - Sample $\mathbf{U}^{(s)}$: For $i = 1, \ldots, N$, sample a new $\boldsymbol{u}_i$ from its conditional posterior

$$\boldsymbol{u}_i^{(s)} \sim \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{\mu}_{u_i}^{(s)}, \boldsymbol{\Sigma}_{u_i}^{(s)})$$

   where $\boldsymbol{\Sigma}_{u_i}^{(s)} = (\lambda_u \mathbf{I} + \beta \sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j^{(s-1)} \boldsymbol{v}_j^{(s-1)\top})^{-1}$ and $\boldsymbol{\mu}_{u_i}^{(s)} = \boldsymbol{\Sigma}_{u_i}^{(s)}(\beta \sum_{j:(i,j)\in\Omega} r_{ij} \boldsymbol{v}_j^{(s-1)})$

   - Sample $\mathbf{V}^{(s)}$: For $j = 1, \ldots, M$, sample a new $\boldsymbol{v}_j$ from its conditional posterior

$$\boldsymbol{v}_j^{(s)} \sim \mathcal{N}(\boldsymbol{v}_j | \boldsymbol{\mu}_{v_j}^{(s)}, \boldsymbol{\Sigma}_{v_j}^{(s)})$$

   where $\boldsymbol{\Sigma}_{v_j}^{(s)} = (\lambda_v \mathbf{I} + \beta \sum_{i:(i,j)\in\Omega} \boldsymbol{u}_i^{(s)} \boldsymbol{u}_i^{(s)\top})^{-1}$ and $\boldsymbol{\mu}_{v_j}^{(s)} = \boldsymbol{\Sigma}_{v_j}^{(s)}(\beta \sum_{i:(i,j)\in\Omega} r_{ij} \boldsymbol{u}_i^{(s)})$

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$
2. For step $s = 1, \ldots, S$
   - Sample $\mathbf{U}^{(s)}$: For $i = 1, \ldots, N$, sample a new $\boldsymbol{u}_i$ from its conditional posterior

$$\boldsymbol{u}_i^{(s)} \sim \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{\mu}_{u_i}^{(s)}, \boldsymbol{\Sigma}_{u_i}^{(s)})$$

   where $\boldsymbol{\Sigma}_{u_i}^{(s)} = (\lambda_u \mathbf{I} + \beta \sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j^{(s-1)} \boldsymbol{v}_j^{(s-1)\top})^{-1}$ and $\boldsymbol{\mu}_{u_i}^{(s)} = \boldsymbol{\Sigma}_{u_i}^{(s)}(\beta \sum_{j:(i,j)\in\Omega} r_{ij} \boldsymbol{v}_j^{(s-1)})$
   - Sample $\mathbf{V}^{(s)}$: For $j = 1, \ldots, M$, sample a new $\boldsymbol{v}_j$ from its conditional posterior

$$\boldsymbol{v}_j^{(s)} \sim \mathcal{N}(\boldsymbol{v}_j | \boldsymbol{\mu}_{v_j}^{(s)}, \boldsymbol{\Sigma}_{v_j}^{(s)})$$

   where $\boldsymbol{\Sigma}_{v_j}^{(s)} = (\lambda_v \mathbf{I} + \beta \sum_{i:(i,j)\in\Omega} \boldsymbol{u}_i^{(s)} \boldsymbol{u}_i^{(s)\top})^{-1}$ and $\boldsymbol{\mu}_{v_j}^{(s)} = \boldsymbol{\Sigma}_{v_j}^{(s)}(\beta \sum_{i:(i,j)\in\Omega} r_{ij} \boldsymbol{u}_i^{(s)})$

In the end, we will have S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^S$ approximating $p(\mathbf{U}, \mathbf{V}|\mathbf{R})$.

Gibbs Sampling for Probabilistic Matrix Factorization

1. Randomly initialize the latent factors $\mathbf{U}^{(0)} = \{\boldsymbol{u}_i^{(0)}\}_{i=1}^N$ and $\mathbf{V}^{(0)} = \{\boldsymbol{v}_j^{(0)}\}_{j=1}^M$
2. For step $s = 1, \ldots, S$
   - Sample $\mathbf{U}^{(s)}$: For $i = 1, \ldots, N$, sample a new $\boldsymbol{u}_i$ from its conditional posterior

$$\boldsymbol{u}_i^{(s)} \sim \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{\mu}_{u_i}^{(s)}, \boldsymbol{\Sigma}_{u_i}^{(s)})$$

   where $\boldsymbol{\Sigma}_{u_i}^{(s)} = (\lambda_u \mathbf{I} + \beta \sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j^{(s-1)} \boldsymbol{v}_j^{(s-1)\top})^{-1}$ and $\boldsymbol{\mu}_{u_i}^{(s)} = \boldsymbol{\Sigma}_{u_i}^{(s)}(\beta \sum_{j:(i,j)\in\Omega} r_{ij} \boldsymbol{v}_j^{(s-1)})$

   - Sample $\mathbf{V}^{(s)}$: For $j = 1, \ldots, M$, sample a new $\boldsymbol{v}_j$ from its conditional posterior

$$\boldsymbol{v}_j^{(s)} \sim \mathcal{N}(\boldsymbol{v}_j | \boldsymbol{\mu}_{v_j}^{(s)}, \boldsymbol{\Sigma}_{v_j}^{(s)})$$

   where $\boldsymbol{\Sigma}_{v_j}^{(s)} = (\lambda_v \mathbf{I} + \beta \sum_{i:(i,j)\in\Omega} \boldsymbol{u}_i^{(s)} \boldsymbol{u}_i^{(s)\top})^{-1}$ and $\boldsymbol{\mu}_{v_j}^{(s)} = \boldsymbol{\Sigma}_{v_j}^{(s)}(\beta \sum_{i:(i,j)\in\Omega} r_{ij} \boldsymbol{u}_i^{(s)})$

In the end, we will have S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^S$ approximating $p(\mathbf{U}, \mathbf{V} | \mathbf{R})$.In practice, we discard the first few samples and thereafter collect one sample after every few steps until we get a total of S samples

Gibbs Sampling for Gaussian Mixture Model

- Recall the GMM, K clusters with parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ and mixing prop. $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$

Gibbs Sampling for Gaussian Mixture Model

- Recall the GMM, K clusters with parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ and mixing prop. $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$
- Joint distribution of data $\boldsymbol{x} = \{x_1, \ldots, x_N\}$, latent cluster ids $\boldsymbol{z} = \{z_1, \ldots, z_N\}$, and other unknowns

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) &= p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma})p(\mathbf{z}|\boldsymbol{\pi})p(\boldsymbol{\pi}) \prod_{k=1}^K p(\boldsymbol{\mu}_k)p(\boldsymbol{\Sigma}_k) \\
&= \left(\prod_{i=1}^N \prod_{k=1}^K (\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))^{\mathbb{I}(z_i=k)}\right) \times \\
&\quad \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_{k=1}^K \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_0, \mathbf{V}_0)\mathrm{IW}(\boldsymbol{\Sigma}_k|\mathbf{S}_0, \nu_0)
\end{aligned}
$$

Gibbs Sampling for Gaussian Mixture Model

- Recall the GMM, K clusters with parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ and mixing prop. $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$
- Joint distribution of data $\boldsymbol{x} = \{x_1, \ldots, x_N\}$, latent cluster ids $\boldsymbol{z} = \{z_1, \ldots, z_N\}$, and other unknowns

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) &= p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(\mathbf{z}|\boldsymbol{\pi}) p(\boldsymbol{\pi}) \prod_{k=1}^{K} p(\boldsymbol{\mu}_k) p(\boldsymbol{\Sigma}_k) \\
&= \left(\prod_{i=1}^{N} \prod_{k=1}^{K} \left(\pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)^{\mathbb{I}(z_i = k)} \right) \times \\
&\quad \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k | \mathbf{m}_0, \mathbf{V}_0) \mathrm{IW}(\boldsymbol{\Sigma}_k | \mathbf{S}_0, \nu_0)
\end{aligned}
$$

- We want to infer the posterior distribution $p(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}|\boldsymbol{x})$ over the unknowns

Gibbs Sampling for Gaussian Mixture Model

The model has local conjugacy (except for the cluster id z_i, but that isn't a problem)

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) &= p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma})p(\mathbf{z}|\boldsymbol{\pi})p(\boldsymbol{\pi}) \prod_{k=1}^{K} p(\boldsymbol{\mu}_k)p(\boldsymbol{\Sigma}_k) \\
&= \left(\prod_{i=1}^{N} \prod_{k=1}^{K} (\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))^{\mathbb{I}(z_i=k)} \right) \times \\
&\quad \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_0, \mathbf{V}_0)\mathrm{IW}(\boldsymbol{\Sigma}_k|\mathbf{S}_0, \nu_0)
\end{aligned}
$$

$$
\boxed{p(z_i = k|\mathbf{x}_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \propto \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}
$$

$$
\boxed{p(\boldsymbol{\pi}|\mathbf{z}) = \mathrm{Dir}(\{\alpha_k + \sum_{i=1}^{N} \mathbb{I}(z_i = k)\}_{k=1}^{K})}
$$

$$
\begin{aligned}
p(\boldsymbol{\mu}_k|\boldsymbol{\Sigma}_k, \mathbf{z}, \mathbf{x}) &= \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, \mathbf{V}_k) \\
\mathbf{V}_k^{-1} &= \mathbf{V}_0^{-1} + N_k \boldsymbol{\Sigma}_k^{-1} \\
\mathbf{m}_k &= \mathbf{V}_k(\boldsymbol{\Sigma}_k^{-1} N_k \overline{\mathbf{x}}_k + \mathbf{V}_0^{-1}\mathbf{m}_0) \\
N_k &\triangleq \sum_{i=1}^{N} \mathbb{I}(z_i = k) \\
\overline{\mathbf{x}}_k &\triangleq \frac{\sum_{i=1}^{N} \mathbb{I}(z_i = k)\mathbf{x}_i}{N_k}
\end{aligned}
$$

$$
\begin{aligned}
p(\boldsymbol{\Sigma}_k|\boldsymbol{\mu}_k, \mathbf{z}, \mathbf{x}) &= \mathrm{IW}(\boldsymbol{\Sigma}_k|\mathbf{S}_k, \nu_k) \\
\mathbf{S}_k &= \mathbf{S}_0 + \sum_{i=1}^{N} \mathbb{I}(z_i = k)(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \\
\nu_k &= \nu_0 + N_k
\end{aligned}
$$

Gibbs Sampling for Gaussian Mixture Model

The model has local conjugacy (except for the cluster id z_i, but that isn't a problem)

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) &= p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(\mathbf{z}|\boldsymbol{\pi}) p(\boldsymbol{\pi}) \prod_{k=1}^{K} p(\boldsymbol{\mu}_k) p(\boldsymbol{\Sigma}_k) \\
&= \left(\prod_{i=1}^{N} \prod_{k=1}^{K} \left(\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)^{\mathbb{I}(z_i=k)} \right) \times \\
&\quad \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_0, \mathbf{V}_0) \mathrm{IW}(\boldsymbol{\Sigma}_k|\mathbf{S}_0, \nu_0)
\end{aligned}
$$

$$
\boxed{p(z_i = k|\mathbf{x}_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \propto \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}
$$

$$
\boxed{p(\boldsymbol{\pi}|\mathbf{z}) = \mathrm{Dir}(\{\alpha_k + \sum_{i=1}^{N} \mathbb{I}(z_i = k)\}_{k=1}^{K})}
$$

$$
\boxed{
\begin{aligned}
p(\boldsymbol{\mu}_k|\boldsymbol{\Sigma}_k, \mathbf{z}, \mathbf{x}) &= \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, \mathbf{V}_k) \\
\mathbf{V}_k^{-1} &= \mathbf{V}_0^{-1} + N_k \boldsymbol{\Sigma}_k^{-1} \\
\mathbf{m}_k &= \mathbf{V}_k(\boldsymbol{\Sigma}_k^{-1} N_k \overline{\mathbf{x}}_k + \mathbf{V}_0^{-1} \mathbf{m}_0) \\
N_k &\triangleq \sum_{i=1}^{N} \mathbb{I}(z_i = k) \\
\overline{\mathbf{x}}_k &\triangleq \frac{\sum_{i=1}^{N} \mathbb{I}(z_i = k) \mathbf{x}_i}{N_k}
\end{aligned}
}
$$

$$
\boxed{
\begin{aligned}
p(\boldsymbol{\Sigma}_k|\boldsymbol{\mu}_k, \mathbf{z}, \mathbf{x}) &= \mathrm{IW}(\boldsymbol{\Sigma}_k|\mathbf{S}_k, \nu_k) \\
\mathbf{S}_k &= \mathbf{S}_0 + \sum_{i=1}^{N} \mathbb{I}(z_i = k)(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \\
\nu_k &= \nu_0 + N_k
\end{aligned}
}
$$

Each Gibbs iteration cycles through sampling these variables one-at-a-time (conditioned on the rest)

Gibbs Sampling: Some Comments

- Perhaps the most popular MCMC algorithm

Gibbs Sampling: Some Comments

- Perhaps the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
    - But even without local conjugacy (if local conditionals can be obtained)

Gibbs Sampling: Some Comments

- Perhaps the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
  - But even without local conjugacy (if local conditionals can be obtained)
- Many variations exist, e.g.,

Gibbs Sampling: Some Comments

- Perhaps the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
  - But even without local conjugacy (if local conditionals can be obtained)
- Many variations exist, e.g.,
  - Blocked Gibbs: sample multiple variables jointly (sometimes possible)

Gibbs Sampling: Some Comments

- Perhaps the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
  - But even without local conjugacy (if local conditionals can be obtained)
- Many variations exist, e.g.,
  - Blocked Gibbs: sample multiple variables jointly (sometimes possible)
  - Rao-Blackwellized Gibbs: Can collapse (i.e., integrate out) the unneeded variables while sampling. Also called "collapsed" Gibbs sampling

Gibbs Sampling: Some Comments

- Perhaps the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
  - But even without local conjugacy (if local conditionals can be obtained)
- Many variations exist, e.g.,
  - Blocked Gibbs: sample multiple variables jointly (sometimes possible)
  - Rao-Blackwellized Gibbs: Can collapse (i.e., integrate out) the unneeded variables while sampling. Also called "collapsed" Gibbs sampling
- Instead of sampling from the conditionals, an alternative is to use the mode of the conditional.

Gibbs Sampling: Some Comments

- Perhaps the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
  - But even without local conjugacy (if local conditionals can be obtained)
- Many variations exist, e.g.,
  - Blocked Gibbs: sample multiple variables jointly (sometimes possible)
  - Rao-Blackwellized Gibbs: Can collapse (i.e., integrate out) the unneeded variables while sampling. Also called "collapsed" Gibbs sampling
- Instead of sampling from the conditionals, an alternative is to use the mode of the conditional.
  - Called the "Itearative Conditional Mode" (ICM) algorithm (doesn't give the posterior though)

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}]$$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j]$$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}$$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}$$

- The variance can be likewise approximated as

$$\text{Var}[r_{ij}|\mathbf{R}] = \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}]$$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}$$

- The variance can be likewise approximated as

$$\text{Var}[r_{ij}|\mathbf{R}] = \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] \quad = \quad \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] + \text{Var}[\epsilon_{ij}]$$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^S$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^S$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] \approx \frac{1}{S}\sum_{s=1}^S \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}$$

- The variance can be likewise approximated as

$$\begin{aligned}
\text{Var}[r_{ij}|\mathbf{R}] = \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] &= \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] + \text{Var}[\epsilon_{ij}] \\
&= \mathbb{E}[(\boldsymbol{u}_i^\top \boldsymbol{v}_j)^2] - \left[\mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j]\right]^2 + \beta^{-1}
\end{aligned}$$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}$$

- The variance can be likewise approximated as

$$\begin{aligned}
\text{Var}[r_{ij}|\mathbf{R}] = \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] \quad &= \quad \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] + \text{Var}[\epsilon_{ij}] \\
&= \quad \mathbb{E}[(\boldsymbol{u}_i^\top \boldsymbol{v}_j)^2] - \left[\mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j]\right]^2 + \beta^{-1} \\
&\approx \quad \frac{1}{S}\sum_{s=1}^{S}\left(\boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}\right)^2 - \left(\frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}\right)^2 + \beta^{-1}
\end{aligned}$$

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}$$

- The variance can be likewise approximated as

$$\begin{aligned}
\text{Var}[r_{ij}|\mathbf{R}] = \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] &= \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] + \text{Var}[\epsilon_{ij}] \\
&= \mathbb{E}[(\boldsymbol{u}_i^\top \boldsymbol{v}_j)^2] - \left[\mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j]\right]^2 + \beta^{-1} \\
&\approx \frac{1}{S}\sum_{s=1}^{S} \left(\boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}\right)^2 - \left(\frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}\right)^2 + \beta^{-1}
\end{aligned}$$

- Question: Can't we average all samples to get a single $\mathbf{U}$ and a single $\mathbf{V}$ and use those?

Using MCMC Samples: An Example

- Consider making predictions in the PMF model. Gibbs sampling gave us S samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$
- How to make predictions for a missing r_{ij} (its predictive mean, its predictive variance)?
- Given the samples $\{\mathbf{U}^{(s)}, \mathbf{V}^{(s)}\}_{s=1}^{S}$ from the posterior, we can approximate the mean of r_{ij} as

$$\mathbb{E}[r_{ij}|\mathbf{R}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] = \mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}$$

- The variance can be likewise approximated as

$$\begin{aligned}
\text{Var}[r_{ij}|\mathbf{R}] = \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j + \epsilon_{ij}] \quad &= \quad \text{Var}[\boldsymbol{u}_i^\top \boldsymbol{v}_j] + \text{Var}[\epsilon_{ij}] \\
&= \quad \mathbb{E}[(\boldsymbol{u}_i^\top \boldsymbol{v}_j)^2] - \left[\mathbb{E}[\boldsymbol{u}_i^\top \boldsymbol{v}_j]\right]^2 + \beta^{-1} \\
&\approx \quad \frac{1}{S}\sum_{s=1}^{S}\left(\boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}\right)^2 - \left(\frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top} \boldsymbol{v}_j^{(s)}\right)^2 + \beta^{-1}
\end{aligned}$$

- Question: Can't we average all samples to get a single $\mathbf{U}$ and a single $\mathbf{V}$ and use those?
  - No. Reason: Mode or label switching

Mode/Label Switching

- The posterior of most latent variable models has multiple modes

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are <u>known</u>

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are known
- Each sampling iteration can give samples of latent variables from one of the modes

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are known
- Each sampling iteration can give samples of latent variables from one of the modes
- Example: Consider a clustering model like GMM. Likelihood is invariant to label permutations

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are known
- Each sampling iteration can give samples of latent variables from one of the modes
- Example: Consider a clustering model like GMM. Likelihood is invariant to label permutations
  - What one sample considers as cluster 1 may become cluster 2 for next sample

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are <u>known</u>
- Each sampling iteration can give samples of latent variables from one of the modes
- Example: Consider a clustering model like GMM. Likelihood is invariant to label permutations
  - What one sample considers as cluster 1 may become cluster 2 for next sample
- Therefore averaging latent variables or parameters across samples can be meaningless

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are <u>known</u>
- Each sampling iteration can give samples of latent variables from one of the modes
- Example: Consider a clustering model like GMM. Likelihood is invariant to label permutations
  - What one sample considers as cluster 1 may become cluster 2 for next sample
- Therefore averaging latent variables or parameters across samples can be meaningless
- Quantities not affected by permutations of latent variables can be safely averaged

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are known
- Each sampling iteration can give samples of latent variables from one of the modes
- Example: Consider a clustering model like GMM. Likelihood is invariant to label permutations
  - What one sample considers as cluster 1 may become cluster 2 for next sample
- Therefore averaging latent variables or parameters across samples can be meaningless
- Quantities not affected by permutations of latent variables can be safely averaged
  - Probability that two points belong to the same cluster (e.g., in GMM)

Mode/Label Switching

- The posterior of most latent variable models has multiple modes
  - .. even if it is unimodal when the latent variables are <u>known</u>
- Each sampling iteration can give samples of latent variables from one of the modes
- Example: Consider a clustering model like GMM. Likelihood is invariant to label permutations
  - What one sample considers as cluster 1 may become cluster 2 for next sample
- Therefore averaging latent variables or parameters across samples can be meaningless
- Quantities not affected by permutations of latent variables can be safely averaged
  - Probability that two points belong to the same cluster (e.g., in GMM)
  - Predicting the mean/variance of a missing entry r_{ij} in matrix factorization

MCMC and Random Walk

- MCMC methods use a proposal distribution to draw the next sample given the previous sample

$$\theta^{(t)} \sim \mathcal{N}(\theta^{(t-1)}, \sigma^2)$$

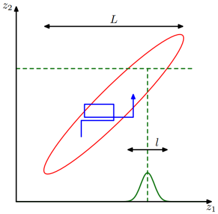
- .. and then we accept/reject (if doing MH) or always accept (if doing Gibbs sampling)

MCMC and Random Walk

- MCMC methods use a proposal distribution to draw the next sample given the previous sample

$$\theta^{(t)} \sim \mathcal{N}(\theta^{(t-1)}, \sigma^2)$$

- .. and then we accept/reject (if doing MH) or always accept (if doing Gibbs sampling)
- Such proposal distributions typically lead to a random-walk behavior (e.g., a zig-zag trajectory in Gibbs sampling) and may lead to very slow convergence (pic below: $\theta = [z_1, z_2]$)

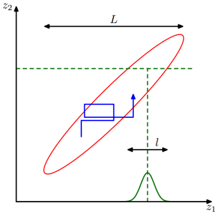


MCMC and Random Walk

- MCMC methods use a proposal distribution to draw the next sample given the previous sample

$$\theta^{(t)} \sim \mathcal{N}(\theta^{(t-1)}, \sigma^2)$$

- .. and then we accept/reject (if doing MH) or always accept (if doing Gibbs sampling)
- Such proposal distributions typically lead to a random-walk behavior (e.g., a zig-zag trajectory in Gibbs sampling) and may lead to very slow convergence (pic below: $\theta = [z_1, z_2]$)



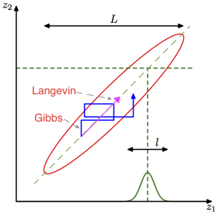
- Can be especially critical when the components of θ are highly correlated

MCMC using Posterior's Gradient Information

- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?

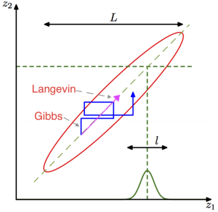
MCMC using Posterior's Gradient Information

- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?
  - Maybe. It can help us move faster towards high probability regions of the posterior $p(\theta|\mathcal{D})$



MCMC using Posterior's Gradient Information

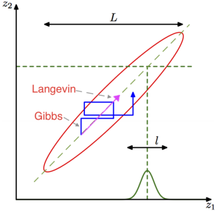
- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?
    - Maybe. It can help us move faster towards high probability regions of the posterior $p(\theta|\mathcal{D})$



- Gradient of the log-posterior: $\nabla_\theta \log \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \nabla_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]$

MCMC using Posterior's Gradient Information

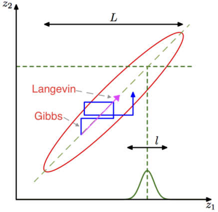
- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?
  - Maybe. It can help us move faster towards high probability regions of the posterior $p(\theta|\mathcal{D})$



- Gradient of the log-posterior: $\nabla_\theta \log \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \nabla_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]$
- Now let's construct a proposal and generate a random sample as follows

MCMC using Posterior's Gradient Information

- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?
  - Maybe. It can help us move faster towards high probability regions of the posterior $p(\theta|\mathcal{D})$

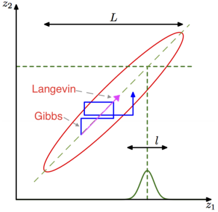


- Gradient of the log-posterior: $\nabla_\theta \log \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \nabla_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]$
- Now let's construct a proposal and generate a random sample as follows

$$\theta^* = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}}$$

MCMC using Posterior's Gradient Information

- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?
  - Maybe. It can help us move faster towards high probability regions of the posterior $p(\theta|\mathcal{D})$

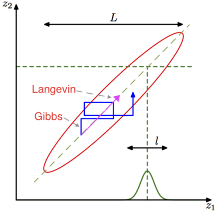


- Gradient of the log-posterior: $\nabla_\theta \log \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \nabla_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]$
- Now let's construct a proposal and generate a random sample as follows

$$\begin{aligned}\theta^* &= \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} \\ \theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta)\end{aligned}$$

MCMC using Posterior's Gradient Information

- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?
  - Maybe. It can help us move faster towards high probability regions of the posterior $p(\theta|\mathcal{D})$

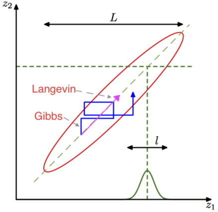


- Gradient of the log-posterior: $\nabla_\theta \log \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \nabla_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]$
- Now let's construct a proposal and generate a random sample as follows

$$\begin{aligned}
\theta^* &= \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} \\
\theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta) \qquad \text{(and then accept/reject using an MH step)}
\end{aligned}$$

MCMC using Posterior's Gradient Information

- Would somehow also incorporating gradient information of the posterior $p(\theta|\mathcal{D})$ be useful?
  - Maybe. It can help us move faster towards high probability regions of the posterior $p(\theta|\mathcal{D})$



- Gradient of the log-posterior: $\nabla_\theta \log \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \nabla_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]$
- Now let's construct a proposal and generate a random sample as follows

$$\theta^* = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}}$$

$$\theta^{(t)} \sim \mathcal{N}(\theta^*, \eta) \qquad \text{(and then accept/reject using an MH step)}$$

- This method is called Langevin dynamics (Neal, 2010). Has its origins in statistical Physics.

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)
- The random noise ensures that we aren't stuck just on the MAP estimate but explore the posterior

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)
- The random noise ensures that we aren't stuck just on the MAP estimate but explore the posterior
- A few technical conditions (Welling and Teh, 2011)

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)
- The random noise ensures that we aren't stuck just on the MAP estimate but explore the posterior
- A few technical conditions (Welling and Teh, 2011)
  - The noise variance needs to be controlled (here, we are setting it to twice the learning rate)

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)
- The random noise ensures that we aren't stuck just on the MAP estimate but explore the posterior
- A few technical conditions (Welling and Teh, 2011)
    - The noise variance needs to be controlled (here, we are setting it to twice the learning rate)
    - As $\eta \to 0$, the acceptance probability approaches 1 and we can always accept

MCMC using Posterior's Gradient Information

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)
- The random noise ensures that we aren't stuck just on the MAP estimate but explore the posterior
- A few technical conditions (Welling and Teh, 2011)
    - The noise variance needs to be controlled (here, we are setting it to twice the learning rate)
    - As $\eta \to 0$, the acceptance probability approaches 1 and we can always accept
- Note that the procedure is almost as fast as MAP estimation!

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

$$\theta^* = \theta^{(t-1)} + \eta_t \nabla_\theta \left[\frac{N}{|\mathcal{D}_t|} \sum_{n=1}^{N_t} \log p(\boldsymbol{x}_{tn}|\theta) + \log p(\theta) \right],$$

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

$$
\begin{aligned}
\theta^* &= \theta^{(t-1)} + \eta_t \nabla_\theta \left[\frac{N}{|\mathcal{D}_t|} \sum_{n=1}^{N_t} \log p(\boldsymbol{x}_{tn}|\theta) + \log p(\theta) \right], \\
\theta^{(t)} &\sim \mathcal{N}(\theta^*, \sigma^2)
\end{aligned}
$$

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

$$\begin{aligned} \theta^* &= \theta^{(t-1)} + \eta_t \nabla_\theta \left[\frac{N}{|\mathcal{D}_t|} \sum_{n=1}^{N_t} \log p(\boldsymbol{x}_{tn}|\theta) + \log p(\theta) \right], \\ \theta^{(t)} &\sim \mathcal{N}(\theta^*, \sigma^2) \quad \text{then} \quad \text{accept/reject} \end{aligned}$$

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

$$\begin{aligned} \theta^* &= \theta^{(t-1)} + \eta_t \nabla_\theta \left[\frac{N}{|\mathcal{D}_t|} \sum_{n=1}^{N_t} \log p(\boldsymbol{x}_{tn}|\theta) + \log p(\theta) \right], \\ \theta^{(t)} &\sim \mathcal{N}(\theta^*, \sigma^2) \quad \text{then} \quad \text{accept/reject} \end{aligned}$$

- Basically, instead of doing gradient descent, SGLD does stochastic gradient descent + MH

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

$$\begin{aligned}
\theta^* &= \theta^{(t-1)} + \eta_t \nabla_\theta \left[\frac{N}{|\mathcal{D}_t|} \sum_{n=1}^{N_t} \log p(\boldsymbol{x}_{tn}|\theta) + \log p(\theta) \right], \\
\theta^{(t)} &\sim \mathcal{N}(\theta^*, \sigma^2) \quad \text{then} \quad \text{accept/reject}
\end{aligned}$$

- Basically, instead of doing gradient descent, SGLD does stochastic gradient descent + MH
  - Valid under some technical conditions on learning rate, variance of proposal distribution, etc.

Scaling Up: Online MCMC Methods

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

$$
\begin{aligned}
\theta^* &= \theta^{(t-1)} + \eta_t \nabla_\theta \left[\frac{N}{|\mathcal{D}_t|} \sum_{n=1}^{N_t} \log p(\boldsymbol{x}_{tn}|\theta) + \log p(\theta) \right], \\
\theta^{(t)} &\sim \mathcal{N}(\theta^*, \sigma^2) \quad \text{then} \quad \text{accept/reject}
\end{aligned}
$$

- Basically, instead of doing gradient descent, SGLD does stochastic gradient descent + MH
  - Valid under some technical conditions on learning rate, variance of proposal distribution, etc.
- Recent flurry of work on this topic (see "Bayesian Learning via Stochastic Gradient Langevin Dynamics" by Welling and Teh (2011) and follow-up works)

Scaling Up: Distributed/Parallel MCMC Methods

- Allows scaling up MCMC algorithms by distributing data to multiple machines

Scaling Up: Distributed/Parallel MCMC Methods

- Allows scaling up MCMC algorithms by distributing data to multiple machines
- Consensus Monte Carlo algorithm is one such example

Scaling Up: Distributed/Parallel MCMC Methods

- Allows scaling up MCMC algorithms by distributing data to multiple machines
- Consensus Monte Carlo algorithm is one such example
- Consensus Monte Carlo partitions the total data $\mathcal{D}$ across M machines as $D_1, \ldots, D_M$

Scaling Up: Distributed/Parallel MCMC Methods

- Allows scaling up MCMC algorithms by distributing data to multiple machines
- Consensus Monte Carlo algorithm is one such example
- Consensus Monte Carlo partitions the total data $\mathcal{D}$ across M machines as $D_1, \ldots, D_M$
- The posterior is then approximated as

$$p(\theta|\mathcal{D}) \propto \prod_{m=1}^{M} p(\mathcal{D}_m|\theta)p(\theta)^{1/M}$$

Scaling Up: Distributed/Parallel MCMC Methods

- Allows scaling up MCMC algorithms by distributing data to multiple machines
- Consensus Monte Carlo algorithm is one such example
- Consensus Monte Carlo partitions the total data $\mathcal{D}$ across M machines as $D_1, \ldots, D_M$
- The posterior is then approximated as

$$p(\theta|\mathcal{D}) \propto \prod_{m=1}^{M} p(\mathcal{D}_m|\theta)p(\theta)^{1/M}$$

- Each "worker" m runs MCMC using its own data and generates samples $\theta_{m1}, \ldots, \theta_{mS}$

Scaling Up: Distributed/Parallel MCMC Methods

- Allows scaling up MCMC algorithms by distributing data to multiple machines
- Consensus Monte Carlo algorithm is one such example
- Consensus Monte Carlo partitions the total data $\mathcal{D}$ across M machines as $D_1, \ldots, D_M$
- The posterior is then approximated as

$$p(\theta|\mathcal{D}) \propto \prod_{m=1}^{M} p(\mathcal{D}_m|\theta)p(\theta)^{1/M}$$

- Each "worker" m runs MCMC using its own data and generates samples $\theta_{m1}, \ldots, \theta_{mS}$
- All workers' samples are combined as

$$\theta_s = (\sum_{m=1}^{M} W_m)^{-1} \sum_{m=1}^{M} W_m \theta_{ms} \qquad s = 1, \ldots, S$$

.. where W_m is the weight assigned to worker m (for more details, see "Bayes and Big Data: The Consensus Monte Carlo Algorithm" by Scott et al (2016)).

Some Comments

- MCMC is a very general technique for generating samples from a distribution

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
    - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods)

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods)
  - Several other alternatives too1 (e.g., Hamiltonian Monte Carlo) which we didn't discuss here

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods)
  - Several other alternatives too1 (e.g., Hamiltonian Monte Carlo) which we didn't discuss here
- MCMC can be scaled up to large datasets (online and distributed MCMC methods)

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods)
  - Several other alternatives too1 (e.g., Hamiltonian Monte Carlo) which we didn't discuss here
- MCMC can be scaled up to large datasets (online and distributed MCMC methods)
- Some other issues of MCMC

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods)
  - Several other alternatives too1 (e.g., Hamiltonian Monte Carlo) which we didn't discuss here
- MCMC can be scaled up to large datasets (online and distributed MCMC methods)
- Some other issues of MCMC
  - Assessing convergence can be hard (there exist methods but beyond the scope of this class)

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods)
  - Several other alternatives too1 (e.g., Hamiltonian Monte Carlo) which we didn't discuss here
- MCMC can be scaled up to large datasets (online and distributed MCMC methods)
- Some other issues of MCMC
  - Assessing convergence can be hard (there exist methods but beyond the scope of this class)
  - Representation of the distribution requires storing all the samples

Some Comments

- MCMC is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so converge can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods)
  - Several other alternatives too1 (e.g., Hamiltonian Monte Carlo) which we didn't discuss here
- MCMC can be scaled up to large datasets (online and distributed MCMC methods)
- Some other issues of MCMC
  - Assessing convergence can be hard (there exist methods but beyond the scope of this class)
  - Representation of the distribution requires storing all the samples (also, making predictions can be slow since it requires empirically averaging the predictions computed using each MCMC sample)