Inference via Sampling (Contd), and Gradient-based and Online MCMC

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

March 6, 2019

Recap: Markov Chain Monte Carlo (MCMC)

- MCMC generates a sequence of "samples" $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(L)}$ based on a first-order Markov Chain

$$\boldsymbol{z}^{(\ell+1)} \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$$

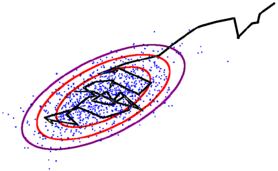
- The proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$ is also known as transition function (or transition kernel)

Recap: Markov Chain Monte Carlo (MCMC)

- MCMC generates a sequence of "samples" $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(L)}$ based on a first-order Markov Chain

$$\boldsymbol{z}^{(\ell+1)} \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$$

- The proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$ is also known as transition function (or transition kernel)
- MCMC basically does a random walk that (eventually) converges to the target distribution $p(\boldsymbol{z})$

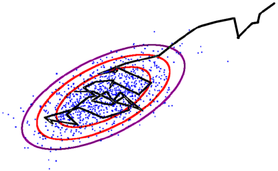


Recap: Markov Chain Monte Carlo (MCMC)

- MCMC generates a sequence of "samples" $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(L)}$ based on a first-order Markov Chain

$$\boldsymbol{z}^{(\ell+1)} \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$$

- The proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$ is also known as transition function (or transition kernel)
- MCMC basically does a random walk that (eventually) converges to the target distribution $p(\boldsymbol{z})$



- The generated samples give a sample based approximation of $p(\boldsymbol{z})$

Recap: The MH Sampling Algorithm

Goal: Generate samples from a probability distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$

The MH Sampling Algorithm

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

For $\ell = 0, \ldots, L-1$

- Sample $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$ and $u \sim \text{Unif}(0,1)$
- Compute the acceptance probability $A(\boldsymbol{z}^*, \boldsymbol{z}^{(\ell)}) = \min\left(1, \frac{\tilde{p}(\boldsymbol{z}^*)q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^*)}{\tilde{p}(\boldsymbol{z}^{(\ell)})q(\boldsymbol{z}^*|\boldsymbol{z}^{(\ell)})}\right)$
- If $u < A(\boldsymbol{z}^*, \boldsymbol{z}^{(\ell)})$ then set $\boldsymbol{z}^{(\ell+1)} = \boldsymbol{z}^*$ else $\boldsymbol{z}^{(\ell+1)} = \boldsymbol{z}^{(\ell)}$

Recap: The MH Sampling Algorithm

Goal: Generate samples from a probability distribution $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$

The MH Sampling Algorithm

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

For $\ell = 0, \ldots, L-1$

- Sample $\boldsymbol{z}^* \sim q(\boldsymbol{z}|\boldsymbol{z}^{(\ell)})$ and $u \sim \text{Unif}(0,1)$
- Compute the acceptance probability $A(\boldsymbol{z}^*, \boldsymbol{z}^{(\ell)}) = \min\left(1, \frac{\tilde{p}(\boldsymbol{z}^*)q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^*)}{\tilde{p}(\boldsymbol{z}^{(\ell)})q(\boldsymbol{z}^*|\boldsymbol{z}^{(\ell)})}\right)$
- If $u < A(\boldsymbol{z}^*, \boldsymbol{z}^{(\ell)})$ then set $\boldsymbol{z}^{(\ell+1)} = \boldsymbol{z}^*$ else $\boldsymbol{z}^{(\ell+1)} = \boldsymbol{z}^{(\ell)}$

Note: Computing acceptance prob. can be expensive in general, e.g., for posterior inference in which case $\tilde{p}(\boldsymbol{z})$ represents an unnormalized posterior $p(\mathbf{X}|\mathbf{Z})p(\mathbf{Z})$, which is product of likelihood and prior

Recap: Gibbs Sampling

- An instance of MH sampling where the acceptance probability $= 1$
- Based on sampling $\boldsymbol{z}$ one "component" at a time with proposal $=$ conditional distribution

Gibbs Sampling

Initialize $\boldsymbol{z}^{(0)} = [z_1^{(0)}, z_2^{(0)}, \ldots, z_M^{(0)}]$ randomly

For $\ell = 1, \ldots, L$

- Sample $\boldsymbol{z}^{(\ell)}$ by sampling one component at a time (usually cyclic manner)

$$
\begin{aligned}
z_1^{(\ell)} &\sim p(z_1 | z_2^{(\ell-1)}, z_3^{(\ell-1)}, \ldots, z_M^{(\ell-1)}) \\
z_2^{(\ell)} &\sim p(z_2 | z_1^{(\ell)}, z_3^{(\ell-1)}, \ldots, z_M^{(\ell-1)}) \\
&\vdots \\
z_{M-1}^{(\ell)} &\sim p(z_{M-1} | z_1^{(\ell)}, \ldots, z_{M-2}^{(\ell)}, z_M^{(\ell-1)}) \\
z_M^{(\ell)} &\sim p(z_M | z_1^{(\ell)}, z_2^{(\ell)}, \ldots, z_{M-1}^{(\ell)})
\end{aligned}
$$

Recap: Gibbs Sampling

- An instance of MH sampling where the acceptance probability $= 1$
- Based on sampling $\boldsymbol{z}$ one "component" at a time with proposal $=$ conditional distribution

Gibbs Sampling

Initialize $\boldsymbol{z}^{(0)} = [z_1^{(0)}, z_2^{(0)}, \ldots, z_M^{(0)}]$ randomly

For $\ell = 1, \ldots, L$

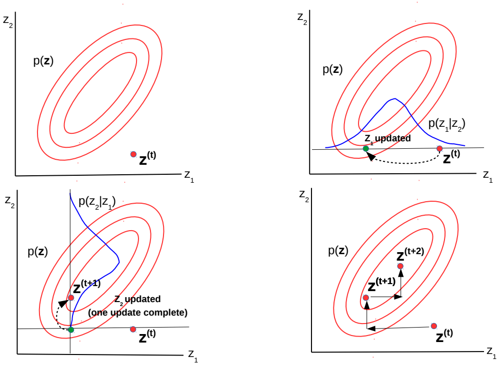
- Sample $\boldsymbol{z}^{(\ell)}$ by sampling one component at a time (usually cyclic manner)

$$
\begin{aligned}
z_1^{(\ell)} &\sim p(z_1 | z_2^{(\ell-1)}, z_3^{(\ell-1)}, \ldots, z_M^{(\ell-1)}) \\
z_2^{(\ell)} &\sim p(z_2 | z_1^{(\ell)}, z_3^{(\ell-1)}, \ldots, z_M^{(\ell-1)}) \\
&\vdots \\
z_{M-1}^{(\ell)} &\sim p(z_{M-1} | z_1^{(\ell)}, \ldots, z_{M-2}^{(\ell)}, z_M^{(\ell-1)}) \\
z_M^{(\ell)} &\sim p(z_M | z_1^{(\ell)}, z_2^{(\ell)}, \ldots, z_{M-1}^{(\ell)})
\end{aligned}
$$

- Very easy to derive if the conditional distributions are easy to obtain

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)



Note that Gibbs updates are like co-ordinate ascent

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"
- If $p(\boldsymbol{z}_m)$ and $p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ are conjugate then the CP is straightforward

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"
- If $p(\boldsymbol{z}_m)$ and $p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ are conjugate then the CP is straightforward
- Another way to get each CP $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ is by following this

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"
- If $p(\boldsymbol{z}_m)$ and $p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ are conjugate then the CP is straightforward
- Another way to get each CP $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ is by following this
  - Write down the expression of $p(\mathbf{X}, \mathbf{Z})$

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"
- If $p(\boldsymbol{z}_m)$ and $p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ are conjugate then the CP is straightforward
- Another way to get each CP $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ is by following this
  - Write down the expression of $p(\mathbf{X}, \mathbf{Z})$
  - Terms that contain $\boldsymbol{z}_m$ represent the CP of $\boldsymbol{z}_m$ (up to proportionality constant)

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"
- If $p(\boldsymbol{z}_m)$ and $p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ are conjugate then the CP is straightforward
- Another way to get each CP $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ is by following this
  - Write down the expression of $p(\mathbf{X}, \mathbf{Z})$
  - Terms that contain $\boldsymbol{z}_m$ represent the CP of $\boldsymbol{z}_m$ (up to proportionality constant)
  - Note: Sometimes it's easier to look at the log of everything (like we did while deriving mean-field VI)

Deriving A Gibbs Sampler: The General Recipe

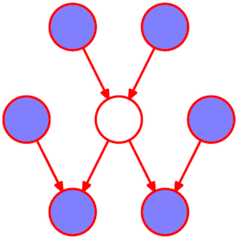
- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"
- If $p(\boldsymbol{z}_m)$ and $p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ are conjugate then the CP is straightforward
- Another way to get each CP $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ is by following this
  - Write down the expression of $p(\mathbf{X}, \mathbf{Z})$
  - Terms that contain $\boldsymbol{z}_m$ represent the CP of $\boldsymbol{z}_m$ (up to proportionality constant)
  - Note: Sometimes it's easier to look at the log of everything (like we did while deriving mean-field VI)
- Also remember: In $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$, we only need to condition on terms in Markov Blanket of $\boldsymbol{z}_m$

Deriving A Gibbs Sampler: The General Recipe

- Suppose our target distribution is a posterior distribution $p(\mathbf{Z}|\mathbf{X})$ where $\mathbf{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M]$
- Gibbs sampling requires the conditional posteriors $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ for $m = 1, \ldots, M$
- In general, $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X}) \propto p(\boldsymbol{z}_m)p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ where $\mathbf{Z}_{-m}$ is "known"
- If $p(\boldsymbol{z}_m)$ and $p(\mathbf{X}|\boldsymbol{z}_m, \mathbf{Z}_{-m})$ are conjugate then the CP is straightforward
- Another way to get each CP $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$ is by following this
  - Write down the expression of $p(\mathbf{X}, \mathbf{Z})$
  - Terms that contain $\boldsymbol{z}_m$ represent the CP of $\boldsymbol{z}_m$ (up to proportionality constant)
  - Note: Sometimes it's easier to look at the log of everything (like we did while deriving mean-field VI)
- Also remember: In $p(\boldsymbol{z}_m|\mathbf{Z}_{-m}, \mathbf{X})$, we only need to condition on terms in Markov Blanket of $\boldsymbol{z}_m$
- Markov Blanket of a variable: Its parents, children, and other parents of its children

An Aside: Markov Blanket

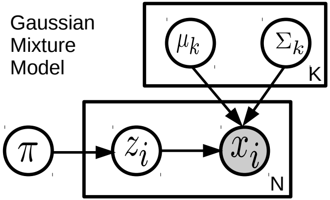
- Markov Blanket of a variable: Its parents, children, and other parents of its children



- Very helpful in quickly seeing what to condition on when deriving CPs in complex models

Gibbs Sampling: A Not-So-Simple Example

Gaussian Mixture Model



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

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

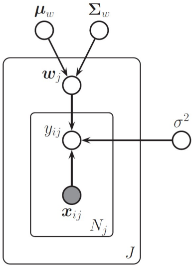
$$
p(\boldsymbol{\pi}|\mathbf{z}) \quad = \quad \mathrm{Dir}(\{\alpha_k + \sum_{i=1}^{N} \mathbb{I}(z_i = k)\}_{k=1}^{K})
$$

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

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

Gibbs Sampling: Another Not-So-Simple Example

J schools Regression Problem



$$y_{ij} = \mathbf{x}_{ij}^T \mathbf{w}_j + \epsilon_{ij}$$

$$\mathbf{w}_j \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$$

$$\begin{aligned} \boldsymbol{\mu}_w &\sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{V}_0) \\ \boldsymbol{\Sigma}_w &\sim \mathrm{IW}(\eta_0, \mathbf{S}_0^{-1}) \\ \sigma^2 &\sim \mathrm{IG}(\nu_0/2, \nu_0\sigma_0^2/2) \end{aligned}$$

$$\begin{aligned} p(\mathbf{w}_j|\mathcal{D}_j, \boldsymbol{\theta}) &= \mathcal{N}(\mathbf{w}_j|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ \boldsymbol{\Sigma}_j^{-1} &= \boldsymbol{\Sigma}^{-1} + \mathbf{X}_j^T\mathbf{X}_j/\sigma^2 \\ \boldsymbol{\mu}_j &= \boldsymbol{\Sigma}_j(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \mathbf{X}_j^T\mathbf{y}_j/\sigma^2) \end{aligned}$$

$$\begin{aligned} p(\boldsymbol{\mu}_w|\mathbf{w}_{1:J}, \boldsymbol{\Sigma}_w) &= \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N) \\ \boldsymbol{\Sigma}_N^{-1} &= \mathbf{V}_0^{-1} + J\boldsymbol{\Sigma}^{-1} \\ \boldsymbol{\mu}_N &= \boldsymbol{\Sigma}_N(\mathbf{V}_0^{-1}\boldsymbol{\mu}_0 + J\boldsymbol{\Sigma}^{-1}\overline{\mathbf{w}}) \\ \overline{\mathbf{w}} &= \frac{1}{J}\sum_j \mathbf{w}_j \end{aligned}$$

$$\begin{aligned} p(\boldsymbol{\Sigma}_w|\boldsymbol{\mu}_w, \mathbf{w}_{1:J}) &= \mathrm{IW}((\mathbf{S}_0 + \mathbf{S}_\mu)^{-1}, \eta_0 + J) \\ \mathbf{S}_\mu &= \sum_j (\mathbf{w}_j - \boldsymbol{\mu}_w)(\mathbf{w}_j - \boldsymbol{\mu}_w)^T \end{aligned}$$

$$\begin{aligned} p(\sigma^2|\mathcal{D}, \mathbf{w}_{1:J}) &= \mathrm{IG}([\nu_0 + N]/2, [\nu_0\sigma_0^2 + \mathrm{SSR}(\mathbf{w}_{1:J})]/2) \\ \mathrm{SSR}(\mathbf{w}_{1:J}) &= \sum_{j=1}^{J}\sum_{i=1}^{N_j}(y_{ij} - \mathbf{w}_j^T\mathbf{x}_{ij})^2 \end{aligned}$$

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
- Many variations exist, e.g.,

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
- Many variations exist, e.g.,
  - Blocked Gibbs: sample multiple variables jointly (sometimes possible)

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
- 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

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
- 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 (note: collapsing is a more general idea, can also be used in other inference algorithms such as VI)

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
- 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 (note: collapsing is a more general idea, can also be used in other inference algorithms such as VI)
  - MH within Gibbs

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
- 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 (note: collapsing is a more general idea, can also be used in other inference algorithms such as VI)
  - MH within Gibbs
- Instead of sampling from the conditionals, an alternative is to use the mode of the conditional.

Gibbs Sampling: Some Comments

- One of the most popular MCMC algorithm
- Very easy to derive and implement for locally conjugate models
- 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 (note: collapsing is a more general idea, can also be used in other inference algorithms such as VI)
  - MH within Gibbs
- Instead of sampling from the conditionals, an alternative is to use the mode of the conditional.
  - Called the "Iterative Conditional Mode" (ICM) algorithm (doesn't give the posterior though)

Sampling Methods: Label Switching Issue

- A subtle but important issue

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes
- Example: In a clustering model (e.g., GMM), the likelihood is invariant to how we label clusters

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes
- Example: In a clustering model (e.g., GMM), the likelihood is invariant to how we label clusters
  - What we call cluster 1 in one sample may be cluster 2 in the next sample

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes
- Example: In a clustering model (e.g., GMM), the likelihood is invariant to how we label clusters
  - What we call cluster 1 in one sample may be cluster 2 in the next sample
- Therefore averaging latent variables across samples can be meaningless

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes
- Example: In a clustering model (e.g., GMM), the likelihood is invariant to how we label clusters
  - What we call cluster 1 in one sample may be cluster 2 in the next sample
- Therefore averaging latent variables across samples can be meaningless
- Quantities not affected by permutations of latent variables can be safely averaged

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes
- Example: In a clustering model (e.g., GMM), the likelihood is invariant to how we label clusters
  - What we call cluster 1 in one sample may be cluster 2 in the next sample
- Therefore averaging latent variables across samples can be meaningless
- Quantities not affected by permutations of latent variables can be safely averaged
  - E.g., probability that two points belong to the same cluster (e.g., in GMM)

Sampling Methods: Label Switching Issue

- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes
- Example: In a clustering model (e.g., GMM), the likelihood is invariant to how we label clusters
  - What we call cluster 1 in one sample may be cluster 2 in the next sample
- Therefore averaging latent variables across samples can be meaningless
- Quantities not affected by permutations of latent variables can be safely averaged
  - E.g., 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: Some Other Aspects

- Choice of proposal distribution is important

MCMC: Some Other Aspects

- Choice of proposal distribution is important
  - For MH sampling, Gaussian proposal is popular when $\boldsymbol{z}$ is continuous, e.g.,

$$q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^{(\ell-1)}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}^{(\ell-1)}, \mathbf{H})$$

    where $\mathbf{H}$ is the Hessian at the MAP of the target distribution

MCMC: Some Other Aspects

- Choice of proposal distribution is important
  - For MH sampling, Gaussian proposal is popular when $\boldsymbol{z}$ is continuous, e.g.,

$$q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^{(\ell-1)}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}^{(\ell-1)}, \mathbf{H})$$

    where $\mathbf{H}$ is the Hessian at the MAP of the target distribution
  - More sophisticated proposals: Mixture of proposal distributions, data-driven or adaptive proposals

MCMC: Some Other Aspects

- Choice of proposal distribution is important
  - For MH sampling, Gaussian proposal is popular when $\boldsymbol{z}$ is continuous, e.g.,

$$q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^{(\ell-1)}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}^{(\ell-1)}, \mathbf{H})$$

    where $\mathbf{H}$ is the Hessian at the MAP of the target distribution
  - More sophisticated proposals: Mixture of proposal distributions, data-driven or adaptive proposals
- Autocorrelation. Can show that when approximating $f^* = \mathbb{E}[f]$ using S samples $\{\boldsymbol{z}^{(s)}\}_{s=1}^{S}$

$$\text{var}_{MCMC}[\bar{f}] = \text{var}_{MC}[\bar{f}] + \frac{1}{S^2}\sum_{s \neq t}\mathbb{E}[(f_s - f^*)(f_t - f^*)]$$

MCMC: Some Other Aspects

- Choice of proposal distribution is important
  - For MH sampling, Gaussian proposal is popular when $\boldsymbol{z}$ is continuous, e.g.,

$$q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^{(\ell-1)}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}^{(\ell-1)}, \mathbf{H})$$

    where $\mathbf{H}$ is the Hessian at the MAP of the target distribution
  - More sophisticated proposals: Mixture of proposal distributions, data-driven or adaptive proposals
- Autocorrelation. Can show that when approximating $f^* = \mathbb{E}[f]$ using S samples $\{\boldsymbol{z}^{(s)}\}_{s=1}^S$

$$\text{var}_{MCMC}[\bar{f}] = \text{var}_{MC}[\bar{f}] + \frac{1}{S^2}\sum_{s\neq t}\mathbb{E}[(f_s - f^*)(f_t - f^*)], \quad \text{Effective Sample Size (ESS)} = \frac{\text{var}_{MC}[f]}{\text{var}_{MCMC}[f]}$$

MCMC: Some Other Aspects

- Choice of proposal distribution is important
  - For MH sampling, Gaussian proposal is popular when $\boldsymbol{z}$ is continuous, e.g.,

$$q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^{(\ell-1)}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}^{(\ell-1)}, \mathbf{H})$$

    where $\mathbf{H}$ is the Hessian at the MAP of the target distribution
  - More sophisticated proposals: Mixture of proposal distributions, data-driven or adaptive proposals
- Autocorrelation. Can show that when approximating $f^* = \mathbb{E}[f]$ using S samples $\{\boldsymbol{z}^{(s)}\}_{s=1}^{S}$

$$\text{var}_{MCMC}[\bar{f}] = \text{var}_{MC}[\bar{f}] + \frac{1}{S^2}\sum_{s \neq t}\mathbb{E}[(f_s - f^*)(f_t - f^*)], \quad \text{Effective Sample Size (ESS)} = \frac{\text{var}_{MC}[f]}{\text{var}_{MCMC}[f]}$$

- In above, f_s is value of f computed using the s^t MCMC sample $\boldsymbol{z}^{(s)}$. Assume $\bar{f} = \frac{1}{S}\sum_{s=1}^{S} f_s$
- Autocorrelation function (ACF) at lag t is define as $\rho_t = \frac{\frac{1}{S-t}\sum_{s=1}^{S-t}(f_s - \bar{f})(f_{s+t} - \bar{f})}{\frac{1}{S-1}\sum_{s=1}^{S}(f_s - \bar{f})^2}$. Lower is better!

MCMC: Some Other Aspects

- Choice of proposal distribution is important
  - For MH sampling, Gaussian proposal is popular when $\boldsymbol{z}$ is continuous, e.g.,

$$q(\boldsymbol{z}^{(\ell)}|\boldsymbol{z}^{(\ell-1)}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}^{(\ell-1)}, \mathbf{H})$$

    where $\mathbf{H}$ is the Hessian at the MAP of the target distribution
  - More sophisticated proposals: Mixture of proposal distributions, data-driven or adaptive proposals
- Autocorrelation. Can show that when approximating $f^* = \mathbb{E}[f]$ using S samples $\{\boldsymbol{z}^{(s)}\}_{s=1}^{S}$

$$\text{var}_{MCMC}[\bar{f}] = \text{var}_{MC}[\bar{f}] + \frac{1}{S^2}\sum_{s \neq t} \mathbb{E}[(f_s - f^*)(f_t - f^*)], \quad \text{Effective Sample Size (ESS)} = \frac{\text{var}_{MC}[f]}{\text{var}_{MCMC}[f]}$$

- In above, f_s is value of f computed using the s^t MCMC sample $\boldsymbol{z}^{(s)}$. Assume $\bar{f} = \frac{1}{S}\sum_{s=1}^{S} f_s$
- Autocorrelation function (ACF) at lag t is define as $\rho_t = \frac{\frac{1}{S-t}\sum_{s=1}^{S-t}(f_s - \bar{f})(f_{s+t} - \bar{f})}{\frac{1}{S-1}\sum_{s=1}^{S}(f_s - \bar{f})^2}$. Lower is better!
- Multiple Chains: Run multiple chains, take union of generated samples (ignoring burn-in samples)

MCMC and Random Walk

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

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

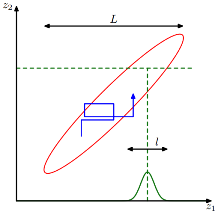
- .. and then we accept/reject (if doing MH) or always accept (if doing Gibbs sampling)

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: $\theta = [z_1, z_2]$)

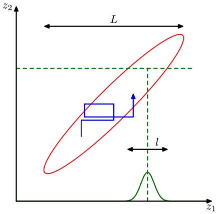


MCMC and Random Walk

- 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: $\theta = [z_1, z_2]$)



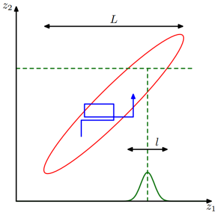
- Can be especially critical when the components of θ are highly correlated

MCMC 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
- Using gradient info of the posterior can be helpful in avoiding the random walk (more in next class)

Using Gradient Info via Langevin Dynamics

- Constructs proposal distribution using gradient of the log-posterior

Using Gradient Info via Langevin Dynamics

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

Using Gradient Info via Langevin Dynamics

- Constructs proposal distribution using gradient of the log-posterior
- 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

Using Gradient Info via Langevin Dynamics

- Constructs proposal distribution using gradient of the log-posterior
- 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

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

Using Gradient Info via Langevin Dynamics

- Constructs proposal distribution using gradient of the log-posterior
- 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

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

Using Gradient Info via Langevin Dynamics

- Constructs proposal distribution using gradient of the log-posterior
- 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

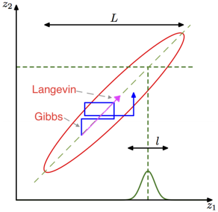
$$\begin{aligned}
\theta^* &= \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} \\
\theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta) \qquad \text{(and then accept/reject using an MH step)}
\end{aligned}$$

Using Gradient Info via Langevin Dynamics

- Constructs proposal distribution using gradient of the log-posterior
- 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

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

- This method is called Langevin dynamics (Neal, 2010). Has its origins in statistical Physics. (Move proposal's mean towards posterior's mode)



Langevin Dynamics (Contd)

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

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

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test
- 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

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test
- 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
- Almost as efficient computationally as standard gradient ascent/descent based MAP estimation

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test
- 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
- Almost as efficient computationally as standard gradient ascent/descent based MAP estimation
- A few technical conditions (Welling and Teh, 2011)

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test
- 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
- Almost as efficient computationally as standard gradient ascent/descent based MAP estimation
- 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)

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test
- 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
- Almost as efficient computationally as standard gradient ascent/descent based MAP estimation
- 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

Langevin Dynamics (Contd)

- 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 $\theta^{(t)}$ using MH test
- 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
- Almost as efficient computationally as standard gradient ascent/descent based MAP estimation
- 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!

Stochastic Gradient (Online) Langevin Dynamics

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

Stochastic Gradient (Online) Langevin Dynamics

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

Stochastic Gradient (Online) Langevin Dynamics

- 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

Stochastic Gradient (Online) Langevin Dynamics

- 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

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

Stochastic Gradient (Online) Langevin Dynamics

- 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

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

Stochastic Gradient (Online) Langevin Dynamics

- 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

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

Stochastic Gradient (Online) Langevin Dynamics

- 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

Stochastic Gradient (Online) Langevin Dynamics

- 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

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

$$\theta^{(t)} \sim \mathcal{N}(\theta^*, \sigma^2) \quad \text{then} \quad \text{accept/reject}$$

- Basically, instead of doing gradient descent, SGLD does stochastic gradient descent + MH
  - Valid under some technical conditions on learning rate η_t, noise variance σ^2, etc.

Stochastic Gradient (Online) Langevin Dynamics

- 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 η_t, noise variance σ^2, 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)

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large
  - Almost as efficient as doing MAP estimation using stochastic gradient methods

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large
  - Almost as efficient as doing MAP estimation using stochastic gradient methods
- Applies to non-conjugate models easily (so long as we can take derivatives)

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large
  - Almost as efficient as doing MAP estimation using stochastic gradient methods
- Applies to non-conjugate models easily (so long as we can take derivatives)
- Several improvements on SGLD in the past couple of years

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large
  - Almost as efficient as doing MAP estimation using stochastic gradient methods
- Applies to non-conjugate models easily (so long as we can take derivatives)
- Several improvements on SGLD in the past couple of years
  - Better choice of learning rate and pre-conditioners for improving convergence

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large
  - Almost as efficient as doing MAP estimation using stochastic gradient methods
- Applies to non-conjugate models easily (so long as we can take derivatives)
- Several improvements on SGLD in the past couple of years
  - Better choice of learning rate and pre-conditioners for improving convergence
  - Extending to the case when θ has some constraints (e.g., a point on simplex)

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large
  - Almost as efficient as doing MAP estimation using stochastic gradient methods
- Applies to non-conjugate models easily (so long as we can take derivatives)
- Several improvements on SGLD in the past couple of years
  - Better choice of learning rate and pre-conditioners for improving convergence
  - Extending to the case when θ has some constraints (e.g., a point on simplex)
  - Theoretical analysis and justification for the "correctness" of the procedure