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” $z^{(1)}, z^{(2)}, \ldots, z^{(L)}$ based on a first-order Markov Chain

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

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

- MCMC basically does a random walk that (eventually) converges to the target distribution $p(z)$

- The generated samples give a sample based approximation of $p(z)$
Recap: The MH Sampling Algorithm

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

The MH Sampling Algorithm

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

For \( \ell = 0, \ldots, L - 1 \)

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

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

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

**Gibbs Sampling**

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

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

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

  $z_1^{(\ell)} \sim p(z_1|z_2^{(\ell-1)}, z_3^{(\ell-1)}, \ldots, z_M^{(\ell-1)})$

  $z_2^{(\ell)} \sim p(z_2|z_1^{(\ell)}, z_3^{(\ell-1)}, \ldots, z_M^{(\ell-1)})$

  \[\vdots\]

  $z_{M-1}^{(\ell)} \sim p(z_{M-1}|z_1^{(\ell)}, \ldots, z_{M-2}^{(\ell)}, z_M^{(\ell-1)})$

  $z_M^{(\ell)} \sim p(z_M|z_1^{(\ell)}, z_2^{(\ell)}, \ldots, z_{M-1}^{(\ell)})$

- 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 \( p(Z|X) \) where \( Z = [z_1, z_2, \ldots, z_M] \)
- Gibbs sampling requires the conditional posteriors \( p(z_m|Z_{-m}, X) \) for \( m = 1, \ldots, M \)
- In general, \( p(z_m|Z_{-m}, X) \propto p(z_m)p(X|z_m, Z_{-m}) \) where \( Z_{-m} \) is “known”
- If \( p(z_m) \) and \( p(X|z_m, Z_{-m}) \) are conjugate then the CP is straightforward
- Another way to get each CP \( p(z_m|Z_{-m}, X) \) is by following this
  - Write down the expression of \( p(X, Z) \)
  - Terms that contain \( z_m \) represent the CP of \( z_m \) (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 \( p(z_m|Z_{-m}, X) \), we only need to condition on terms in Markov Blanket of \( z_m \)
- Markov Blanket of a variable: Its parents, children, and other parents of its children
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

\[ p(x, z, \mu, \Sigma, \pi) = p(x|z, \mu, \Sigma)p(z|\pi)p(\pi) \prod_{k=1}^{K} p(\mu_k)p(\Sigma_k) \]

\[ = \left( \prod_{i=1}^{N} \prod_{k=1}^{K} (\pi_kN(x_i|\mu_k, \Sigma_k))^{I(z_i=k)} \right) \times \]

\[ \text{Dir}(\pi|\alpha) \prod_{k=1}^{K} N(\mu_k|m_0, V_0)IW(\Sigma_k|S_0, v_0) \]

\[ p(z_i = k|x_i, \mu, \Sigma, \pi) \propto \pi_kN(x_i|\mu_k, \Sigma_k) \]

\[ p(\pi|z) = \text{Dir}(\{\alpha_k + \sum_{i=1}^{N} I(z_i = k)\})^{K}_{k=1} \]

\[ p(\mu_k|\Sigma_k, z, x) = N(\mu_k|m_k, V_k) \]

\[ V_k^{-1} = V_0^{-1} + N_k\Sigma_k^{-1} \]

\[ m_k = V_k(\Sigma_k^{-1}N_k\bar{x}_k + V_0^{-1}m_0) \]

\[ N_k \triangleq \sum_{i=1}^{N} I(z_i = k) \]

\[ \bar{x}_k \triangleq \frac{\sum_{i=1}^{N} I(z_i = k)x_i}{N_k} \]

\[ p(\Sigma_k|\mu_k, z, x) = IW(\Sigma_k|S_k, v_k) \]

\[ S_k = S_0 + \sum_{i=1}^{N} I(z_i = k)(x_i - \mu_k)(x_i - \mu_k)^T \]

\[ \nu_k = \nu_0 + N_k \]
Gibbs Sampling: Another Not-So-Simple Example

\[ y_{i,j} = x_{i,j}^T w_j + \epsilon_{i,j} \]
\[ w_j \sim N(\mu_w, \Sigma_w) \]
\[ \mu_w \sim N(\mu_0, V_0) \]
\[ \Sigma_w \sim IW(\eta_0, S_0^{-1}) \]
\[ \sigma^2 \sim IG(\nu_0/2, \nu_0\sigma_0^2/2) \]

\[
p(w_j | D_j, \theta) = \mathcal{N}(w_j | \mu_j, \Sigma_j) \\
\Sigma_j^{-1} = \Sigma^{-1} + X_j^T X_j / \sigma^2 \\
\mu_j = \Sigma_j(\Sigma^{-1} \mu + X_j^T y_j / \sigma^2)
\]

\[
p(\mu_w | w_{1:J}, \Sigma_w) = \mathcal{N}(\mu | \mu_N, \Sigma_N) \\
\Sigma_N^{-1} = V_0^{-1} + J \Sigma^{-1} \\
\mu_N = \Sigma_N(\Sigma_0^{-1} \mu_0 + J \Sigma^{-1} \overline{w}) \\
\overline{w} = \frac{1}{J} \sum_j w_j
\]

\[
p(\Sigma_w | \mu_w, w_{1:J}) = \text{IW}((S_0 + S_\mu)^{-1}, \eta_0 + J) \\
S_\mu = \sum_j (w_j - \mu_w)(w_j - \mu_w)^T
\]

\[
p(\sigma^2 | D, w_{1:J}) = \text{IG}([\nu_0 + N]/2, [\nu_0\sigma_0^2 + \text{SSR}(w_{1:J})]/2) \\
\text{SSR}(w_{1:J}) = \sum_{j=1}^{J} \sum_{i=1}^{N_j} (y_{ij} - w_j^T x_{ij})^2
\]
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
- Suppose we are given samples $Z^{(1)}, \ldots, Z^{(S)}$ from the posterior $p(Z|X)$
- We can’t always simply “average” them to get the “posterior mean” $\bar{Z}$
- 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
  - For MH sampling, Gaussian proposal is popular when \( z \) is continuous, e.g.,
    \[
    q(z^{(\ell)} | z^{(\ell-1)}) = \mathcal{N}(z | z^{(\ell-1)}, H)
    \]
    where \( H \) 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 \( \{z^{(s)}\}_{s=1}^{S} \)
  \[
  \text{var}_{MCMC}[\bar{f}] = \text{var}_{MC}[\bar{f}] + \frac{1}{S^2} \sum_{s \neq t} \mathbb{E}[(f_s - f^*)(f_t - f^*)], \quad \text{Effective Sample Size (ESS)} = \frac{\text{var}_{MC}[f]}{\text{var}_{MCMC}[f]}
  \]
  - In above, \( f_s \) is value of \( f \) computed using the \( s^{th} \) MCMC sample \( z^{(s)} \). 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)

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

  ![Random Walk Diagram]

- Can be especially critical when the components of \( \theta \) 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
- 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

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

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

- This method is called **Langevin dynamics** (Neal, 2010). Has its origins in statistical Physics. (Move proposal’s mean towards posterior’s mode)
Note that the updates of $\theta$ can also be written in the form

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

where $\epsilon_t \sim \mathcal{N}(0, \eta)$

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 Langevin Dynamics** (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch \( D_t = \{x_{t1}, \ldots, x_{tN_t}\} \). Then the (stochastic) Langevin dynamics update is

\[
\theta^* = \theta^{(t-1)} + \eta_t \nabla_{\theta} \left[ \frac{N}{|D_t|} \sum_{n=1}^{N_t} \log p(x_{tn} | \theta) + \log p(\theta) \right],
\]

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

- Basically, instead of doing gradient descent, SGLD does stochastic gradient descent + MH
  - Valid under some technical conditions on learning rate \( \eta_t \), noise variance \( \sigma^2 \), 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)
- If not doing accept/reject, we just need to do the following for each minibatch of data

\[
\theta^{(t)} = \theta^{(t-1)} + \eta_t \nabla_{\theta} \left[ \frac{N}{|D_t|} \sum_{n=1}^{N_t} \log p(x_{tn}|\theta) + \log p(\theta) \right] + \epsilon_t
\]

- 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 \( \theta \) has some constraints (e.g., a point on simplex)
  - Theoretical analysis and justification for the “correctness” of the procedure