Approx. Inference via Sampling (Contd): MCMC with Gradients, Recent Advances

CS772A: Probabilistic Machine Learning
Piyush Rai
Plan for today

- Some other aspects of MCMC
- MCMC with gradient
- Some other recent advances
Sampling Methods: Label Switching Issue

- Suppose we are given samples $Z^{(1)}, Z^{(2)}, \ldots, Z^{(S)}$ from the posterior $p(Z|X)$

- We can’t always simply “average” them to get the “posterior mean” $\bar{Z}$

- Why: Non-identifiability of latent vars in models with multiple equivalent posterior modes

- Example: In clustering via 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
  - Say, in GMM, $z_n^{(1)} = [1,0]$ and $z_n^{(2)} = [0,1]$, both may imply the same
  - Averaging will give $\bar{z}_n = [0.5,0.5]$, which is incorrect

- Quantities not affected by permutations of dims of $Z$ can be safely averaged
  - E.g., probability that two points belong to the same cluster (e.g., in GMM)
  - Predicting the mean of an entry $r_{ij}$ in matrix factorization $\frac{1}{S} \sum_{s=1}^{S} u_i^{(s)} v_j^{(s)}$
MCMC: Some Practical Aspects

- Choice of proposal distribution is important
  - For MH sampling, Gaussian proposal is popular when \( z \) is continuous, e.g.,
    \[
    q(z|z^{(t-1)}) = \mathcal{N}(z|z^{(t-1)}, H)
    \]
  - Other options: Mixture of proposal distributions, data-driven or adaptive proposals

- Autocorrelation. Can show that when approximating \( f^* = \mathbb{E}[f] \) using \( \{Z^{(s)}\}_{s=1}^{S} \)
  \[
  \bar{f} = \frac{1}{S} \sum_{s=1}^{S} f_s
  \]
  \[
  \text{var}_{\text{MCMC}}[\bar{f}] = \text{var}_{\text{MC}}[\bar{f}] + \frac{1}{S^2} \sum_{s \neq t} \mathbb{E}[(f_s - f^*)(f_t - f^*)]
  \]
  \[
  \text{value of } f \text{ using } s^{th} \text{ MCMC sample}
  \]
  \[
  \text{Monte Carlo assumes uncorrelated samples}
  \]
  \[
  \text{Effective Sample Size (ESS)} = \frac{\text{var}_{\text{MC}}[\bar{f}]}{\text{var}_{\text{MCMC}}[\bar{f}]}
  \]
  \[
  \text{Lower is better}
  \]

- Autocorrelation function (ACF) at lag \( t \):
  \[
  \rho_t = \frac{1}{S-t} \sum_{s=1}^{S-t} (f_s - \bar{f})(f_{s+t} - \bar{f})
  \]
  \[
  \frac{1}{S-1} \sum_{s=1}^{S} (f_s - \bar{f})^2
  \]

- Multiple Chains: Run multiple chains, take union of generated samples
Coming Up Next

- Avoiding the random-walk behavior of MCMC
  - Using gradient information of the posterior
- Scalable MCMC methods
Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample, e.g.,

\[ \theta(t) \sim \mathcal{N}(\theta(t-1), \eta_t) \]

And then accept/reject (MH)

Will use \( \theta \) to denote all the unknowns

Can use automatic differentiation methods for this

- Langevin dynamics: Use (unnormalized) posterior’s gradient info in the proposal as

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

\[ \theta(t) \sim \mathcal{N}(\theta^*, \eta_t) \]

And then accept/reject (MH)

Likelihood

Prior

Move towards the mode of the posterior (like finding MAP est)

- Note that the above is equivalent to

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

And then accept/reject (MH)

\( \eta \) set s.t. acceptance rate is around 0.6

Helps also incorporate the curvature info of the posterior

If gradient is pre-multiplied by a preconditioner matrix \( M(\theta_t) \): Simplified Manifold MALA

One option to use for \( M(\theta_t) \) is the second derivative of the unnorm. posterior

- After some waiting period \( T_0 \), iterates \( \{\theta(t)\}_{t=T_0+1}^{T_0+S} \) are MCMC samples from \( p(\theta|D) \)

"Bayesian Learning via Stochastic Gradient Langevin Dynamics" by Welling and Teh (2011)
Langevin Dynamics: A Closer Look

- Is generating MCMC samples really as easy as computing MAP?
- Recall the form of Langevin Dynamics updates

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

- Equivalent to discretization of an SDE with equilibrium distribution \( \propto \exp(\log p(D, \theta)) \)

\[ d\theta_t = -\nabla L(\theta_t) dt + \sqrt{2} dB_t \]

where \( L(\theta_t) = -\log p(D, \theta_t) \) and \((B_t)_{t \geq 0}\) is Brownian motion s.t. \( \Delta B_t \) are i.i.d. Gaussian r.v.s

- Discretization introduces some error which is corrected by MH accept/reject step
- Note: As learning rate \( \eta_t \) decreases, discretization error also decreases and rejection rate tends to zero
- Note: Gradient computations require all the data (thus slow)
  - Solution: Use stochastic gradients - Stochastic Gradient Langevin Dynamics (SGLD)
Stochastic Gradient Langevin Dynamics (SGLD)

- An “online” MCMC method: Langevin Dynamics with minibatches to compute gradients

- Given minibatch $\mathcal{D}_t = \{x_{t1}, x_{t2}, \ldots, x_{tN_t}\}$, the (stochastic) Langevin dynamics update:

$$
\theta^* = \theta^{(t-1)} + \frac{\eta_t}{2} \nabla_{\theta} \left[ \frac{N}{|\mathcal{D}_t|} \sum_{n=1}^{N_t} \log p(x_{tn}|\theta) + \log p(\theta) \right]
$$

- Choice of the learning rate is important. For convergence, $\eta_t = a(b + t)^{-\kappa}$
  
  - Switching to constant learning rates (after a few iterations) often helps convergence

- As $\eta_t$ becomes very very small, acceptance prob. becomes close to 1

- Recent flurry of work on this topic (see “Bayesian Learning via Stochastic Gradient Langevin Dynamics” by Welling and Teh (2011) and follow-up works)
Improvements to SGLD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate $\eta_t$ in all dimensions of $\theta$
  - Doesn’t apply to models where $\theta$ is constrained (e.g., non-neg or prob. vector)
  - Needs the model to be differentiable (since it needs $\nabla_\theta \log p(\mathcal{D}, \theta)$)

- A lot of recent work on improving the basic SGLD to handle such limitations

- Introducing the curvature information in the gradients, e.g.,
  - Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring (Ahn et al, 2012), and Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks (Li et al, 2016)
  - These methods use a preconditioner matrix in the learning rate to improve convergence
  - This also allows different amounts of updates in different dimensions

- SLGD in Riemannian space to handle constrained variables
  - Stoch. Grad. Riemannian Langevin Dynamics on the Probability Simplex (Patterson and Teh, 2013)
Applications of SGLD

- Popular for Bayesian neural networks and other complex Bayesian models
- Reason: SGLD = backprop based updates + Gaussian noise

(Figure: Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks (Li et al, 2016))
Other Recent “SGD-inspired” Sampling Algorithms

- Run SGD and use SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea using stochastic weight avg (SWA)

\[
\theta_{SWA} = \frac{1}{T} \sum_{t=1}^{T} \theta_t
\]

\[
\bar{\theta}^2 = \frac{1}{T} \sum_{t=1}^{T} \theta_t^2, \quad \Sigma_{\text{diag}} = \text{diag}(\bar{\theta}^2 - \theta_{SWA}^2)
\]

- If we want full cov., we can use a low-rank approx. of $\Sigma$ (see Maddox et al for details)
- Reason it works: SGD is asymptotically Normal under certain conditions
- For a more detailed theory of SGD and MCMC, may also refer to this very nice paper: Stochastic Gradient Descent as Approximate Bayesian Inference (Mandt et al, 2017)
- Such algos can give not too accurate but very fast posterior approx for complex models

Approach known as SWA-Gaussian (SWAG)

A Simple Baseline for Bayesian Uncertainty in Deep Learning, Maddox et al (2019)
Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an “auxiliary variable sampler” and incorporates gradient information.
- Uses the idea of simulating a Hamiltonian Dynamics of a physical system.
- Consider the target posterior \( p(\theta | D) \propto \exp(-U(\theta)) \).
- Think of \( \theta \) as “position” then \( U(\theta) = -\log p(D|\theta)p(\theta) \) is like “potential energy”.
- Let’s introduce an auxiliary variable - the momentum \( r \) of the system.
- Can now define a joint distribution over the position and momentum as
  \[
p(\theta, r) \propto \exp \left(-U(\theta) - \frac{1}{2} r^\top M^{-1} r \right) = p(\theta|D)p(r)
  \]
- The total energy (potential + kinetic) or the Hamiltonian of the system
  \[
  H(\theta, r) = U(\theta) + \frac{1}{2} r^\top M^{-1} r = U(\theta) + K(r)
  \]
- Given a sample \((\theta, r)\) from \( p(\theta, r) \), ignoring \( r \), \( \theta \) will be a sample from \( p(\theta|D) \).
Generating Samples in HMC

- Given an initial $(\theta, r)$, Hamiltonian Dynamics defines how $(\theta, r)$ changes w.r.t. time $t$
  \[
  \frac{\partial \theta}{\partial t} = \frac{\partial H}{\partial r} = \frac{\partial K}{\partial r} \\
  \frac{\partial r}{\partial t} = -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}
  \]
  \[
  (H(\theta, r) = U(\theta) + \frac{1}{2} r^T M^{-1} r = U(\theta) + K(r))
  \]

- We can use these equations to update $(\theta, r) \rightarrow (\theta^*, r^*)$ by discretizing time

- For $s = 1: S$, sample as follows
  - Initialize $\theta_0 = \theta^{(s-1)}$, $r_* \sim \mathcal{N}(0, I)$ and $r_0 = r_* - \rho \frac{\partial U}{\partial \theta} |_{\theta_0}$
  - Do $L$ “leapfrog” steps with learning rates $\rho_\ell = \rho$ for $\ell < L$ and $\rho_L = \rho/2$
    - For $\ell = 1: L$
      \[
      \theta_\ell = \theta_{\ell-1} + \rho \frac{\partial K}{\partial r} |_{r_{\ell-1}} \\
      r_\ell = r_{\ell-1} - \rho_\ell \frac{\partial U}{\partial \theta} |_{\theta_\ell}
      \]
  - Perform MH accept/reject test on $(\theta_L, r_L)$. If accepted $\theta^{(s)} = \theta_L$

- The momentum forces exploring different regions instead of getting driven to regions where the MAP solution is

Reason: Getting analytical solutions for the above requires integrals which is in general intractable

$L$ usually set to 5 and learning rate tuned to make acceptance rate around 90%

A single sample generated by taking $L$ steps
HMC in Practice

- HMC typically has very low rejection rate (that too, primarily due to discretization error)
- Performance can be sensitive to $L$ (no. of leapfrog steps) and step-sizes, tuning hard
- A lot of renewed interest in HMC (you may check out NUTS - No U-turn Sampler – doesn’t require setting $L$)
  - Prob. Prog. packages e.g., Tensorflow Prob., Stan, etc, contain implementations of HMC
- Can also do HMC on minibatches (Stochastic Gradient HMC - Chen et al, 2014)
- An illustration: SGHMC vs other methods on MNIST classification (Bayesian neural net)
Suppose our goal is to compute the posterior of \( \theta \in \mathbb{R}^D \) (assuming \( N \) is very large):

\[
p(\theta|X) \propto p(\theta)p(X|\theta) = p(\theta) \prod_{n=1}^{N} p(x_n|\theta)
\]

Suppose we have \( J \) machines with data partitioned as \( X = \{X(j)\}_{j=1}^{J} \).

Let’s assume that the posterior \( p(\theta|X) \) factorizes as

\[
p(\theta|X) = \prod_{i=1}^{J} p^{(i)}(\theta|X^{(i)})
\]

Here \( p^{(j)}(\theta|X^{(j)}) \propto p(\theta)^{1/J} \prod_{x_n \in X^{(j)}} p(x_n|\theta) \) is known as the “subset posterior.”

Assume the \( j^{th} \) machine generates \( T \) MCMC samples \( \{\theta_{j,t}\}_{t=1}^{T} \).

We need a way to combine these subset posteriors using a “consensus”:

\[
\hat{\theta}_1, \ldots, \hat{\theta}_T = \text{CONSENSUSSAMPLES} (\{\theta_{j,1}, \ldots, \theta_{j,T}\}_{j=1}^{J})
\]
Parallel/Distributed MCMC

- Many ways to compute the consensus samples. Let’s look at two of them

- Approach 1: **Weighted Average**: \( \hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t} \) where \( W_j \) can be learned as follows
  - Assuming Gaussian likelihood and Gaussian prior
    \[
    \tilde{\Sigma}_j = \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
    \Sigma = (\Sigma_0^{-1} + \sum_{j=1}^{J} \tilde{\Sigma}_j^{-1})^{-1} \quad (\Sigma_0 \text{ is the prior's covariance}) \\
    W_j = \Sigma(\Sigma_0^{-1}/J + \tilde{\Sigma}_j^{-1})
    \]

- Approach 2: Fit \( J \) Gaussians, one for each \( \{\theta_{j,t}\}_{t=1}^{T} \) and take their product
  \[
  \bar{\mu}_j = \text{sample mean of } \{\theta_{j,1}, \ldots, \theta_{j,T}\}, \quad \tilde{\Sigma}_j = \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
  \hat{\Sigma}_j = (\sum_{j=1}^{J} \tilde{\Sigma}_j^{-1})^{-1}, \quad \bar{\mu}_j = \hat{\Sigma}_j(\sum_{j=1}^{J} \tilde{\Sigma}_j^{-1} \bar{\mu}_j) \quad (\text{cov and mean of prod. of Gaussians}) \\
  \hat{\theta}_t \sim \mathcal{N}(\bar{\mu}_j, \hat{\Sigma}_j), t = 1, \ldots, T \quad (\text{the final consensus samples})
  \]

- For detailed proofs and other approaches, may refer to the reference below

Patterns of Scalable Bayesian Inference (Angelino et al, 2016)
Approximate Inference: VI vs Sampling

- VI approximates a posterior distribution $p(Z|X)$ by another distribution $q(Z|\phi)$
- Sampling uses $S$ samples $Z^{(1)}, Z^{(2)}, \ldots, Z^{(S)}$ to approximate $p(Z|X)$
- Sampling can be used within VI (ELBO approx using Monte-Carlo)
- In terms of “comparison” between VI and sampling, a few things to be noted
  - **Convergence**: VI only has local convergence, sampling (in theory) can give exact posterior
  - **Storage**: Sampling based approx needs to storage all samples, VI only needs var. params $\phi$
  - **Prediction Cost**: Sampling always requires Monte-Carlo avging for posterior predictive; with VI, sometimes we can get closed form posterior predictive

PPD if using sampling: $$p(x_*|X) = \int p(x_*|Z)p(Z|X)dZ \approx \frac{1}{S} \sum_{s=1}^{S} p(x_*|Z^{(s)})$$

PPD if using VI: $$p(x_*|X) = \int p(x_*|Z)p(Z|X)dZ \approx \int p(x_*|Z)q(Z|\phi)dZ$$

- There is some work on “compressing” sampling-based approximations*

---

*“Compact approximations to Bayesian predictive distributions” by Snelson and Ghaharamani, 2005; and “Bayesian Dark Knowledge” by Korattikara et al, 2015
Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diff.)
- Conjugate models with one “main” parameter: Straightforward posterior updates
- MLE-II/MAP-II: Often useful for estimating the hyperparameters
- EM: If we want to do MLE/MAP for models with latent variables
  - Very general algorithm, can also be made online
  - Used when we want point estimates for some unknowns and posterior over others
  - Can use it for hyperparameter estimation as well
  - Often better than using direct gradient methods
- VI and sampling methods can be used to get full posterior for complex models
  - Quite easy if we have local conjugacy (VI has closed form updates, Gibbs sampler is easy to derive)
  - In other cases, we have general VI with Monte-Carlo gradients, MH sampling
  - MCMC can also make use of gradient info (LD/SGLD)
- For large-scale problems, online/distributed VI/MCMC, or SGD based posterior approx