

# MCMC with Gradient Information, Recent Advances in MCMC

CS698X: Topics in Probabilistic Modeling and Inference

Piyush Rai

# Plan

- Using gradient information in sampling
  - Langevin Dynamics and Stochastic Gradient Langevin Dynamics
  - Hamiltonian Monte Carlo (HMC)
- Parallel/Distributed MCMC



# Using Gradients in MCMC: Langevin Dynamics

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

And then accept/reject (MH)

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

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

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

Likelihood

Prior

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

And then accept/reject (MH)

$\eta$  set s.t. acceptance rate is around 0.6

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

Same as doing a gradient ascent step towards the posterior and injecting noise  $\epsilon_t \sim \mathcal{N}(0, \eta_t)$ . Noise ensures we aren't stuck at the MAP solution

Using gradient info in the proposal helps us move faster towards high-prob regions

- Note that the above is equivalent to

And then accept/reject (MH)

Known as **Metropolis-Adjusted Langevin Algorithm (MALA)**

Helps also incorporate the **curvature info** of the posterior

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

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

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  $p(\theta|\mathcal{D})$



# Langevin Dynamics: A Closer Look

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

And then accept/reject (MH)

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

Same as our target posterior

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

Above update is its discretization

Note that this is continuous time

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

where  $L(\theta_t) = -\log p(\mathcal{D}, \theta_t)$  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  $\mathcal{D}_t = \{x_{t1}, x_{t2}, \dots, x_{tN_t}\}$ , the (stochastic) Langevin dynamics update:

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

Almost as fast as doing SGD updates ☺

And then accept/reject (MH)

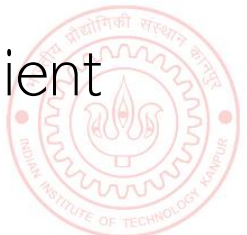
$$\theta^{(t)} \sim \mathcal{N}(\theta^*, \eta_t)$$

- 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

No need for accept/reject (MH)

- 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

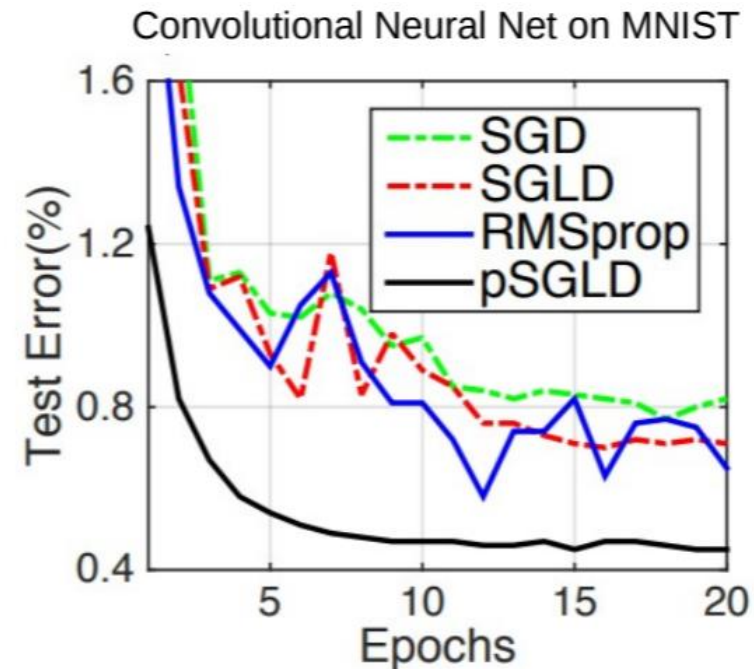
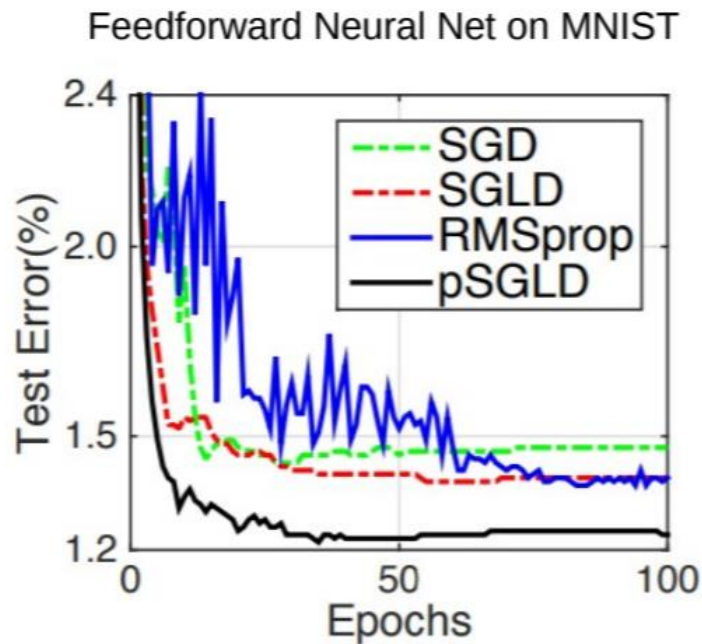
- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate  $\eta_t$  in all dimensions of  $\theta$
  - Doesn't apply to models where  $\theta$  is constrained (e.g., non-neg or prob. vector)
  - Needs the model to be differentiable (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
- SGLD in Riemannian space to handle constrained variables
  - Stoch. Grad. Riemannian Langevin Dynamics on the Probability Simplex (Patterson and Teh, 2013)

Based on reparametrizing the constrained variables to make them unconstrained



# 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<sup>8</sup>

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

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

Approach known as  
SWA-Gaussian (SWAG)

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

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_{SWA}, \Sigma_{\text{diag}})$$

- 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



# Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an “auxiliary variable sampler” and incorporates gradient info
- Uses the idea of simulating a Hamiltonian Dynamics of a physical system
- Consider the target posterior  $p(\theta|\mathcal{D}) \propto \exp(-U(\theta))$
- Think of  $\theta$  as “position” then  $U(\theta) = -\log p(\mathcal{D}|\theta)p(\theta)$  is like “potential energy”
- Let’s introduce an auxiliary variable - the momentum  $\mathbf{r}$  of the system
- Can now define a joint distribution over the position and momentum as

$$p(\theta, \mathbf{r}) \propto \exp\left(-U(\theta) - \frac{1}{2}\mathbf{r}^\top M^{-1}\mathbf{r}\right) = p(\theta|\mathcal{D})p(\mathbf{r})$$

- The total energy (potential + kinetic) or the Hamiltonian of the system

Constant w.r.t. time

$$H(\theta, \mathbf{r}) = U(\theta) + \frac{1}{2}\mathbf{r}^\top M^{-1}\mathbf{r} = U(\theta) + K(\mathbf{r})$$

- Given a sample  $(\theta, \mathbf{r})$  from  $p(\theta, \mathbf{r})$ , ignoring  $\mathbf{r}$ ,  $\theta$  will be a sample from  $p(\theta|\mathcal{D})$



# Generating Samples in HMC

- Given an initial  $(\theta, \mathbf{r})$ , Hamiltonian Dynamics defines how  $(\theta, \mathbf{r})$  changes w.r.t. time  $t$

$$\begin{aligned} \frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \mathbf{r}} = \frac{\partial K}{\partial \mathbf{r}} \\ \frac{\partial \mathbf{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta} \end{aligned} \quad ( H(\theta, \mathbf{r}) = U(\theta) + \frac{1}{2} \mathbf{r}^\top M^{-1} \mathbf{r} = U(\theta) + K(\mathbf{r}) )$$

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

- For  $s = 1:S$ , sample as follows

- Initialize  $\theta_0 = \theta^{(s-1)}$ ,  $\mathbf{r}_* \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $\mathbf{r}_0 = \mathbf{r}_* - \frac{\rho}{2} \frac{\partial U}{\partial \theta} |_{\theta_0}$
- Do  $L$  “leapfrog” steps with learning rates  $\rho_\ell = \rho$  for  $\ell < L$  and  $\rho_L = \rho/2$ 
  - For  $\ell = 1:L$

$$\theta_\ell = \theta_{\ell-1} + \rho \frac{\partial K}{\partial \mathbf{r}} |_{\mathbf{r}_{\ell-1}}$$

$$\mathbf{r}_\ell = \mathbf{r}_{\ell-1} - \rho_\ell \frac{\partial U}{\partial \theta} |_{\theta_\ell}$$

- Perform MH accept/reject test on  $(\theta_L, \mathbf{r}_L)$ . 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

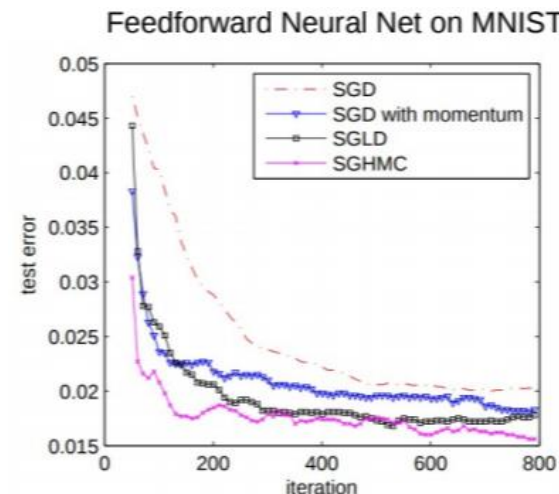
$L$  usually set to 5 and learning rate tuned to make acceptance rate around 90%

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)



(Figure: Stochastic Gradient Hamiltonian Monte Carlo (Chen et al, 2014))



# Parallel/Distributed MCMC

- Suppose our goal is to compute the posterior of  $\theta \in \mathbb{R}^D$  (assuming  $N$  is very large)

$$p(\theta|\mathbf{X}) \propto p(\theta)p(\mathbf{X}|\theta) = p(\theta) \prod_{n=1}^N p(\mathbf{x}_n|\theta)$$

- Suppose we have  $J$  machines with data partitioned as  $\mathbf{X} = \{\mathbf{X}^{(j)}\}_{j=1}^J$
- Let's assume that the posterior  $p(\theta|\mathbf{X})$  factorizes as

$$p(\theta|\mathbf{X}) = \prod_{j=1}^J p^{(j)}(\theta|\mathbf{X}^{(j)})$$

- Here  $p^{(j)}(\theta|\mathbf{X}^{(j)}) \propto p(\theta)^{1/J} \prod_{\mathbf{x}_n \in \mathbf{X}^{(j)}} p(\mathbf{x}_n|\theta)$  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”

$$\hat{\theta}_1, \dots, \hat{\theta}_T = \text{CONSENSUSSAMPLES}(\{\theta_{j,1}, \dots, \theta_{j,T}\}_{j=1}^J)$$



# 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

$$\bar{\Sigma}_j = \text{sample covariance of } \{\theta_{j,1}, \dots, \theta_{j,T}\}$$

$$\Sigma = (\Sigma_0^{-1} + \sum_{j=1}^J \bar{\Sigma}_j^{-1})^{-1} \quad (\Sigma_0 \text{ is the prior's covariance})$$

$$W_j = \Sigma(\Sigma_0^{-1}/J + \bar{\Sigma}_j^{-1})$$

These approaches can also be used to make VI parallel/distributed



- Approach 2: Fit  $J$  Gaussians, one for each  $\{\theta_{j,t}\}_{t=1}^T$  and take their product

$$\bar{\mu}_j = \text{sample mean of } \{\theta_{j,1}, \dots, \theta_{j,T}\}, \quad \bar{\Sigma}_j = \text{sample covariance of } \{\theta_{j,1}, \dots, \theta_{j,T}\}$$

$$\hat{\Sigma}_J = (\sum_{j=1}^J \bar{\Sigma}_j^{-1})^{-1}, \quad \hat{\mu}_J = \hat{\Sigma}_J (\sum_{j=1}^J \bar{\Sigma}_j^{-1} \bar{\mu}_j) \quad (\text{cov and mean of prod. of Gaussians})$$

$$\hat{\theta}_t \sim \mathcal{N}(\hat{\mu}_J, \hat{\Sigma}_J), t = 1, \dots, T \quad (\text{the final consensus samples})$$

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

# 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

