Gradient-based and Online Sampling Methods, Recent Advances in Sampling Methods

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

March 11, 2019

Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample. For example,

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

  .. and then we accept/reject the generated sample

Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample. For example,

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

  .. and then we accept/reject the generated sample

- Langevin dynamics: Use posterior's gradient info in the proposal as follows

$$\text{do} \quad \theta^{(t)} \quad \sim \quad \mathcal{N}(\theta^{*}, \eta_t) \quad + \quad \text{MH accept/reject}$$

Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample. For example,

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

  .. and then we accept/reject the generated sample

- Langevin dynamics: Use posterior's gradient info in the proposal as follows

$$\begin{aligned} \text{do} \quad \theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta_t) \quad + \quad \text{MH accept/reject} \\ \text{where} \quad \theta^* &= \theta^{(t-1)} + \frac{\eta_t}{2} \nabla_\theta [\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} \end{aligned}$$

Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample. For example,

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

  .. and then we accept/reject the generated sample

- Langevin dynamics: Use posterior's gradient info in the proposal as follows

$$\begin{aligned} \text{do} \quad \theta^{(t)} \quad &\sim \quad \mathcal{N}(\theta^*, \eta_t) \quad + \quad \text{MH accept/reject} \\ \text{where} \quad \theta^* \quad &= \quad \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} \end{aligned}$$

- Note that the above is equivalent to

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample. For example,

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

.. and then we accept/reject the generated sample

- Langevin dynamics: Use posterior's gradient info in the proposal as follows

$$\begin{aligned} \text{do} \quad \theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta_t) \quad + \quad \text{MH accept/reject} \\ \text{where} \quad \theta^* &= \theta^{(t-1)} + \frac{\eta_t}{2} \nabla_\theta [\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} \end{aligned}$$

- Note that the above is equivalent to

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2} \nabla_\theta [\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

.. which is same as gradient based optimization for MAP + injected noise $\epsilon_t \sim \mathcal{N}(0, \eta_t)$

Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample. For example,

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

.. and then we accept/reject the generated sample

- Langevin dynamics: Use posterior's gradient info in the proposal as follows

$$\begin{aligned} \text{do} \quad \theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta_t) \quad + \quad \text{MH accept/reject} \\ \text{where} \quad \theta^* &= \theta^{(t-1)} + \frac{\eta_t}{2} \nabla_\theta [\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} \end{aligned}$$

- Note that the above is equivalent to

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2} \nabla_\theta [\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

.. which is same as gradient based optimization for MAP + injected noise $\epsilon_t \sim \mathcal{N}(0, \eta_t)$

- Incorporating gradients in proposals takes us to high-prob regions faster

Using Gradients in MCMC: Langevin Dynamics

- MCMC uses a random-walk based proposal to generate the next sample. For example,

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

  .. and then we accept/reject the generated sample

- Langevin dynamics: Use posterior's gradient info in the proposal as follows

$$\begin{aligned} \text{do} \quad \theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta_t) \quad + \quad \text{MH accept/reject} \\ \text{where} \quad \theta^* &= \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} \end{aligned}$$

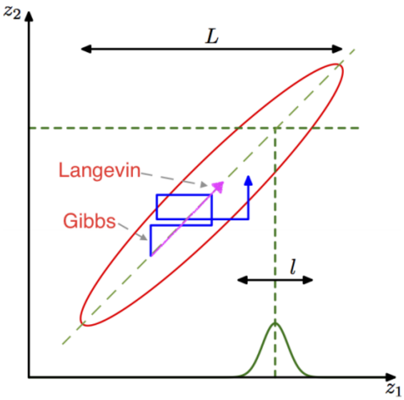
- Note that the above is equivalent to

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

  .. which is same as gradient based optimization for MAP + injected noise $\epsilon_t \sim \mathcal{N}(0, \eta_t)$

- Incorporating gradients in proposals takes us to high-prob regions faster
- After some waiting period T_0, the iterates $\{\theta^{(t)}\}_{T_0+1}^{T_0+S}$ are MCMC samples from the target $p(\theta|\mathcal{D})$

Using Gradients in MCMC: Langevin Dynamics



Langevin Dynamics: A Closer Look

- LD still seems like magic! Is generating MCMC samples really as easy as computing MAP?

Langevin Dynamics: A Closer Look

- LD still seems like magic! Is generating MCMC samples really as easy as computing MAP?
- Recall the form of LD updates

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

Langevin Dynamics: A Closer Look

- LD still seems like magic! Is generating MCMC samples really as easy as computing MAP?
- Recall the form of LD updates

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

- Equivalent to a discretization of a stochastic diff. eqn. with equilib. distr $\propto \exp(\log p(\mathcal{D}, \theta))$

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

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

Langevin Dynamics: A Closer Look

- LD still seems like magic! Is generating MCMC samples really as easy as computing MAP?
- Recall the form of LD updates

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

- Equivalent to a discretization of a stochastic diff. eqn. with equilib. distr $\propto \exp(\log p(\mathcal{D}, \theta))$

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

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

- Discretization introduces some error which is corrected by MH accept/reject step

Langevin Dynamics: A Closer Look

- LD still seems like magic! Is generating MCMC samples really as easy as computing MAP?
- Recall the form of LD updates

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

- Equivalent to a discretization of a stochastic diff. eqn. with equilib. distr $\propto \exp(\log p(\mathcal{D}, \theta))$

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

.. where $L(\theta_t) = -\log p(\mathcal{D}, \theta_t)$ and $(B_t)_{t\geq 0}$ is Brownian motion s.t. ΔB_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 η_t decreases, discretization error also decreases (and rejection rate $\to 0$)

Langevin Dynamics: A Closer Look

- LD still seems like magic! Is generating MCMC samples really as easy as computing MAP?
- Recall the form of LD updates

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

- Equivalent to a discretization of a stochastic diff. eqn. with equilib. distr $\propto \exp(\log p(\mathcal{D},\theta))$

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

.. where $L(\theta_t) = -\log p(\mathcal{D},\theta_t)$ and $(B_t)_{t\geq 0}$ is Brownian motion s.t. ΔB_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 η_t decreases, discretization error also decreases (and rejection rate $\to 0$)
- Note: Gradient computations require all the data (thus slow)

Langevin Dynamics: A Closer Look

- LD still seems like magic! Is generating MCMC samples really as easy as computing MAP?
- Recall the form of LD updates

$$\boxed{\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\Big|_{\theta^{(t-1)}} + \epsilon_t} \quad + \quad \text{MH accept/reject}$$

- Equivalent to a discretization of a stochastic diff. eqn. with equilib. distr $\propto \exp(\log p(\mathcal{D}, \theta))$

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

.. where $L(\theta_t) = -\log p(\mathcal{D}, \theta_t)$ and $(B_t)_{t\geq 0}$ is Brownian motion s.t. ΔB_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 η_t decreases, discretization error also decreases (and rejection rate $\to 0$)
- 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

Stochastic Gradient Langevin Dynamics (SGLD)

- An "online" MCMC method: Langevin Dynamics with minibatches to compute gradients
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

Stochastic Gradient Langevin Dynamics (SGLD)

- An "online" MCMC method: Langevin Dynamics with minibatches to compute gradients
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

Stochastic Gradient Langevin Dynamics (SGLD)

- An "online" MCMC method: Langevin Dynamics with minibatches to compute gradients
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

Stochastic Gradient Langevin Dynamics (SGLD)

- An "online" MCMC method: Langevin Dynamics with minibatches to compute gradients
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

- Choice of the learning rate is important. For convergence, $\eta_t = a(b+t)^{-\kappa}$

Stochastic Gradient Langevin Dynamics (SGLD)

- An "online" MCMC method: Langevin Dynamics with minibatches to compute gradients
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

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

Stochastic Gradient Langevin Dynamics (SGLD)

- An "online" MCMC method: Langevin Dynamics with minibatches to compute gradients
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

- Choice of the learning rate is important. For convergence, $\eta_t = a(b+t)^{-\kappa}$
  - In practice however, switching to constant learning rates (after a few iterations) also helps convergence
- When the learning rate becomes very very small, acceptance prob. becomes close to 1 (so no more need to do MH accept/reject test; can accept every

Stochastic Gradient Langevin Dynamics (SGLD)

- An "online" MCMC method: Langevin Dynamics with minibatches to compute gradients
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

- Choice of the learning rate is important. For convergence, $\eta_t = a(b+t)^{-\kappa}$
  - In practice however, switching to constant learning rates (after a few iterations) also helps convergence
- When the learning rate becomes very very small, acceptance prob. becomes close to 1 (so no more need to do MH accept/reject test; can accept every sample)
- 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 SLGD

- The basic SGLD, although fairly simple, has many limitations

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)
  - Assumes that the model is differentiable

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)
  - Assumes that the model is differentiable
- A lot of recent work on improving the basic SGLD to handle such limitations

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)
  - Assumes that the model is differentiable
- A lot of recent work on improving the basic SGLD to handle such limitations. Some examples
  - 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)

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)
  - Assumes that the model is differentiable
- A lot of recent work on improving the basic SGLD to handle such limitations. Some examples
  - 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)
    - Uses a preconditioner matrix in the learning rate to improve convergence

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)
  - Assumes that the model is differentiable
- A lot of recent work on improving the basic SGLD to handle such limitations. Some examples
  - 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)
    - Uses a preconditioner matrix in the learning rate to improve convergence
    - This allows differet amounts of updates in different dimensions

Improvements to SLGD

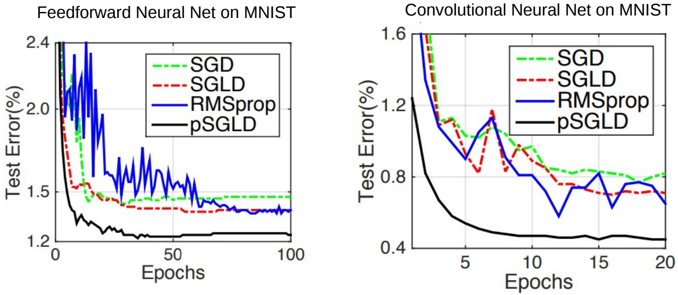
- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)
  - Assumes that the model is differentiable
- A lot of recent work on improving the basic SGLD to handle such limitations. Some examples
  - 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)
    - Uses a preconditioner matrix in the learning rate to improve convergence
    - This allows differet amounts of updates in different dimensions
  - Stoch. Grad. Riemannian Langevin Dynamics on the Probability Simplex (Patterson and Teh, 2013)

Improvements to SLGD

- The basic SGLD, although fairly simple, has many limitations, e.g.
  - Exhibits slow convergence and mixing. Uses same learning rate η_t in all dimensions of θ
  - Doesn't apply to models where θ is constrained (e.g., non-neg or prob. vector)
  - Assumes that the model is differentiable
- A lot of recent work on improving the basic SGLD to handle such limitations. Some examples
  - 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)
    - Uses a preconditioner matrix in the learning rate to improve convergence
    - This allows differet amounts of updates in different dimensions
  - Stoch. Grad. Riemannian Langevin Dynamics on the Probability Simplex (Patterson and Teh, 2013)
    - SLGD in Riemannian to handle constrained variables

Applications of SLGD

- Has become very popular recenty for Baysian neural networks and other complex Bayesian models
- Reason: We know how to do backprop, SLGD = 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

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

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

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

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

$$\bar{\theta}^2 = \frac{1}{T}\sum_{t=1}^{T}\theta_t^2,$$

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

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

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

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

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

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

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

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

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

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

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

- Note: If we want full cov., we can use a low-rank approx. of Σ (see Maddox et al for details)

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

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

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

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

- Note: If we want full cov., we can use a low-rank approx. of Σ (see Maddox et al for details)
- Why does this work?

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

$$\begin{aligned}\theta_{SWA} &= \frac{1}{T}\sum_{t=1}^{T}\theta_t \\ \bar{\theta}^2 &= \frac{1}{T}\sum_{t=1}^{T}\theta_t^2, \quad \Sigma_{\text{diag}} = \text{diag}(\bar{\theta}^2 - \theta_{SWA}^2) \\ p(\theta|\mathcal{D}) &\approx \mathcal{N}(\theta_{SWA}, \Sigma_{\text{diag}})\end{aligned}$$

- Note: If we want full cov., we can use a low-rank approx. of Σ (see Maddox et al for details)
- Why does this work? Reason: SGD is asymptotically Normal under certain conditions

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

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

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

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

- Note: If we want full cov., we can use a low-rank approx. of Σ (see Maddox et al for details)
- Why does this work? Reason: 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)

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Other Recent "SGD-inspired" Sampling Algorithms

- Can run SGD and use the SGD iterates $\theta_1, \theta_2, \ldots, \theta_T$ to construct a Gaussian approximation
- Recently Maddox et al (2019) proposed an idea based on stochastic weight averaging (SWA)
- If we want a Gaussian approximation with diagonal covariance, this is very easy

$$\begin{aligned}
\theta_{SWA} &= \frac{1}{T}\sum_{t=1}^{T}\theta_t \\
\bar{\theta}^2 &= \frac{1}{T}\sum_{t=1}^{T}\theta_t^2, \quad \Sigma_{\text{diag}} = \text{diag}(\bar{\theta}^2 - \theta_{SWA}^2) \\
p(\theta|\mathcal{D}) &\approx \mathcal{N}(\theta_{SWA}, \Sigma_{\text{diag}})
\end{aligned}$$

- Note: If we want full cov., we can use a low-rank approx. of Σ (see Maddox et al for details)
- Why does this work? Reason: 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 are now becoming popular for getting fast posterior approximations for complex models

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "auxiliary variable sampler" and incorporates gradient info

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "auxiliary variable sampler" and incorporates gradient info
- Uses the idea of simulating a Hamiltonian Dynamics of a physical system

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "auxiliary variable sampler" and incorporates gradient info
- Uses the idea of simulating a Hamiltonian Dynamics of a physical system
- Consider the target posterior $p(\theta|\mathcal{D}) \propto \exp(-U(\theta))$

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "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 θ as the position and $U(\theta) = -\log[p(\mathcal{D}|\theta)p(\theta)]$ is like "potential energy"

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "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 θ as the position and $U(\theta) = -\log[p(\mathcal{D}|\theta)p(\theta)]$ is like "potential energy"
- Let's introduce an auxiliary variable - the momentum $\boldsymbol{r}$ of the system

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "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 θ as the position and $U(\theta) = -\log[p(\mathcal{D}|\theta)p(\theta)]$ is like "potential energy"
- Let's introduce an auxiliary variable - the momentum $\boldsymbol{r}$ of the system
- Can now define a joint distribution over the position and momentum as

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

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "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 θ as the position and $U(\theta) = -\log[p(\mathcal{D}|\theta)p(\theta)]$ is like "potential energy"
- Let's introduce an auxiliary variable - the momentum $\boldsymbol{r}$ of the system
- Can now define a joint distribution over the position and momentum as

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

- $H(\theta, \boldsymbol{r}) = U(\theta) + \frac{1}{2}\boldsymbol{r}^\top M^{-1}\boldsymbol{r} = U(\theta) + K(\boldsymbol{r})$ is the total energy (potential + kinetic) of the system

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "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 θ as the position and $U(\theta) = -\log[p(\mathcal{D}|\theta)p(\theta)]$ is like "potential energy"
- Let's introduce an auxiliary variable - the momentum $\boldsymbol{r}$ of the system
- Can now define a joint distribution over the position and momentum as

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

- $H(\theta, \boldsymbol{r}) = U(\theta) + \frac{1}{2}\boldsymbol{r}^\top M^{-1}\boldsymbol{r} = U(\theta) + K(\boldsymbol{r})$ is the total energy (potential + kinetic) of the system
- $H(\theta, \boldsymbol{r})$ is also known as the Hamiltonian and constant w.r.t. time

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC (Neal, 1996) is an example of "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 θ as the position and $U(\theta) = -\log[p(\mathcal{D}|\theta)p(\theta)]$ is like "potential energy"
- Let's introduce an auxiliary variable - the momentum $\boldsymbol{r}$ of the system
- Can now define a joint distribution over the position and momentum as

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

- $H(\theta, \boldsymbol{r}) = U(\theta) + \frac{1}{2}\boldsymbol{r}^\top M^{-1}\boldsymbol{r} = U(\theta) + K(\boldsymbol{r})$ is the total energy (potential + kinetic) of the system
- $H(\theta, \boldsymbol{r})$ is also known as the Hamiltonian and constant w.r.t. time
- Given samples $(\theta, \boldsymbol{r})$ from joint $p(\theta, \boldsymbol{r})$, we can ignore $\boldsymbol{r}$ and θ will be a sample from $p(\theta|\mathcal{D})$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\frac{\partial \theta}{\partial t} = \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}}$$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\frac{\partial \theta}{\partial t} = \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}}$$
$$\frac{\partial \boldsymbol{r}}{\partial t} = -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}$$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$
\begin{aligned}
\frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}} \\
\frac{\partial \boldsymbol{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}
\end{aligned}
$$

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

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\begin{aligned}\frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}} \\ \frac{\partial \boldsymbol{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}\end{aligned}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\begin{aligned} \frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}} \\ \frac{\partial \boldsymbol{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta} \end{aligned}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows
    - Initialize $\theta_0 = \theta^{(s-1)}$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\begin{aligned} \frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}} \\ \frac{\partial \boldsymbol{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta} \end{aligned}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows
    - Initialize $\theta_0 = \theta^{(s-1)}$, $\boldsymbol{r}_* \sim \mathcal{N}(0, \mathbf{I})$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\begin{aligned}\frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}} \\ \frac{\partial \boldsymbol{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}\end{aligned}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows
    - Initialize $\theta_0 = \theta^{(s-1)}$, $\boldsymbol{r}_* \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{r}_0 = \boldsymbol{r}_* - \frac{\rho}{2}\frac{\partial U}{\partial \theta}|_{\theta_0}$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\frac{\partial \theta}{\partial t} = \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}}$$
$$\frac{\partial \boldsymbol{r}}{\partial t} = -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows
    - Initialize $\theta_0 = \theta^{(s-1)}$, $\boldsymbol{r}_* \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{r}_0 = \boldsymbol{r}_* - \frac{\rho}{2} \frac{\partial U}{\partial \theta}|_{\theta_0}$
    - Do L "leapfrog" steps with learning rates $\rho_l = \rho$ for $\ell < L$, and $\rho_L = \rho/2$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\frac{\partial \theta}{\partial t} = \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}}$$
$$\frac{\partial \boldsymbol{r}}{\partial t} = -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows
    - Initialize $\theta_0 = \theta^{(s-1)}$, $\boldsymbol{r}_* \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{r}_0 = \boldsymbol{r}_* - \frac{\rho}{2}\frac{\partial U}{\partial \theta}|_{\theta_0}$
    - Do L "leapfrog" steps with learning rates $\rho_l = \rho$ for $\ell < L$, and $\rho_L = \rho/2$

$$\text{for } \ell = 1 : L,\ \theta_\ell = \theta_{\ell-1} + \rho \frac{\partial K}{\partial \boldsymbol{r}}|_{\boldsymbol{r}_{\ell-1}},\ \boldsymbol{r}_\ell = \boldsymbol{r}_{\ell-1} - \rho_\ell \frac{\partial U}{\partial \theta}|_{\theta_\ell}$$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\begin{aligned}\frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}} \\ \frac{\partial \boldsymbol{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}\end{aligned}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows
    - Initialize $\theta_0 = \theta^{(s-1)}$, $\boldsymbol{r}_* \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{r}_0 = \boldsymbol{r}_* - \frac{\rho}{2}\frac{\partial U}{\partial \theta}|_{\theta_0}$
    - Do L "leapfrog" steps with learning rates $\rho_l = \rho$ for $\ell < L$, and $\rho_L = \rho/2$

$$\text{for } \ell = 1 : L,\ \theta_\ell = \theta_{\ell-1} + \rho \frac{\partial K}{\partial \boldsymbol{r}}|_{\boldsymbol{r}_{\ell-1}},\ \boldsymbol{r}_\ell = \boldsymbol{r}_{\ell-1} - \rho_\ell \frac{\partial U}{\partial \theta}|_{\theta_\ell}$$

    - Perform MH accept/reject test on $(\theta_L, \boldsymbol{r}_L)$. If accepted, $\theta^{(s)} = \theta_L$

Hamiltonian/Hybrid Monte Carlo (HMC)

- How do we generate samples $(\theta, \boldsymbol{r})$ in HMC?
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

$$\begin{aligned}\frac{\partial \theta}{\partial t} &= \frac{\partial H}{\partial \boldsymbol{r}} = \frac{\partial K}{\partial \boldsymbol{r}} \\ \frac{\partial \boldsymbol{r}}{\partial t} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta}\end{aligned}$$

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$ by discretizing time
  - For $s = 1 : S$, sample as follows
    - Initialize $\theta_0 = \theta^{(s-1)}$, $\boldsymbol{r}_* \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{r}_0 = \boldsymbol{r}_* - \frac{\rho}{2}\frac{\partial U}{\partial \theta}|_{\theta_0}$
    - Do L "leapfrog" steps with learning rates $\rho_l = \rho$ for $\ell < L$, and $\rho_L = \rho/2$

$$\text{for } \ell = 1 : L,\ \theta_\ell = \theta_{\ell-1} + \rho \frac{\partial K}{\partial \boldsymbol{r}}|_{\boldsymbol{r}_{\ell-1}},\ \boldsymbol{r}_\ell = \boldsymbol{r}_{\ell-1} - \rho_\ell \frac{\partial U}{\partial \theta}|_{\theta_\ell}$$

    - Perform MH accept/reject test on $(\theta_L, \boldsymbol{r}_L)$. If accepted, $\theta^{(s)} = \theta_L$
- The momentum forces exploring different regions instead of getting driven to regions where MAP is

Hamiltonian/Hybrid Monte Carlo (HMC)

- HMC typically has very low rejection rate (that too, primarily due to discretization error)

Hamiltonian/Hybrid Monte Carlo (HMC)

- 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, so difficult to tune

Hamiltonian/Hybrid Monte Carlo (HMC)

- 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, so difficult to tune
- A lot of renewed interest in HMC (you may check out NUTS - No U-turn Sampler)

Hamiltonian/Hybrid Monte Carlo (HMC)

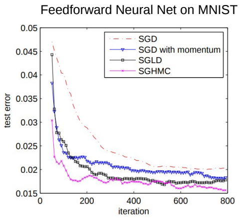
- HMC typically has very low rejection rate (that too, primarily due to discretization error)
- Performance can be sensitive to L (no. of leapfrog steps) and step-sizes, so difficult to tune
- A lot of renewed interest in HMC (you may check out NUTS - No U-turn Sampler)
    - Prob. Prog. packages e.g., Tensorflow Probability, Stan, etc, contain implementations of HMC

Hamiltonian/Hybrid Monte Carlo (HMC)

- 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, so difficult to tune
- A lot of renewed interest in HMC (you may check out NUTS - No U-turn Sampler)
  - Prob. Prog. packages e.g., Tensorflow Probability, Stan, etc, contain implementations of HMC
- Can also do online HMC (Stochastic Gradient HMC - Chen et al, 2014)

Hamiltonian/Hybrid Monte Carlo (HMC)

- 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, so difficult to tune
- A lot of renewed interest in HMC (you may check out NUTS - No U-turn Sampler)
  - Prob. Prog. packages e.g., Tensorflow Probability, Stan, etc, contain implementations of HMC
- Can also do online HMC (Stochastic Gradient HMC - Chen et al, 2014)
- An illustration: SGHMC vs some other methods on MNIST classification



(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)

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

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Parallel/Distributed MCMC

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

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

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

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Parallel/Distributed MCMC

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

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

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

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

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Parallel/Distributed MCMC

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

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

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

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

where $p^{(j)}(\theta|\mathbf{X}^{(j)}) \propto p(\theta)^{1/J} \prod_{\boldsymbol{x}_n \in \mathbf{X}^{(j)}} p(\boldsymbol{x}_n|\theta)$ is the "subset posterior"

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Parallel/Distributed MCMC

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

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

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

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

where $p^{(j)}(\theta|\mathbf{X}^{(j)}) \propto p(\theta)^{1/J} \prod_{\boldsymbol{x}_n \in \mathbf{X}^{(j)}} p(\boldsymbol{x}_n|\theta)$ is the "subset posterior"

- Assume $\{\theta_{j,t}\}_{t=1}^{T}$ to be the set of T MCMC samples generated by the j^{th} machine

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Parallel/Distributed MCMC

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

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

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

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

where $p^{(j)}(\theta|\mathbf{X}^{(j)}) \propto p(\theta)^{1/J}\prod_{\boldsymbol{x}_n \in \mathbf{X}^{(j)}} p(\boldsymbol{x}_n|\theta)$ is the "subset posterior"

- Assume $\{\theta_{j,t}\}_{t=1}^{T}$ to be the set of T MCMC samples generated by the j^{th} machine
- We need a way to combine these subset posteriors using a "consensus"

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

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

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

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

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

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

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

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

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

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

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

$$\begin{aligned}
\bar{\Sigma}_j &= \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
\Sigma &= (\Sigma_0^{-1} + \sum_{j=1}^{J} \bar{\Sigma}_j^{-1})^{-1} \quad (\Sigma_0 \text{ is the prior's covariance}) \\
W_j &= \Sigma(\Sigma_0^{-1}/J + \bar{\Sigma}_j^{-1})
\end{aligned}$$

- Fit J Gaussians, one for each $\{\theta_{j,1}, \ldots, \theta_{j,T}\}$ and take their product

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

$$
\begin{aligned}
\bar{\Sigma}_j &= \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
\Sigma &= (\Sigma_0^{-1} + \sum_{j=1}^{J} \bar{\Sigma}_j^{-1})^{-1} \quad (\Sigma_0 \text{ is the prior's covariance}) \\
W_j &= \Sigma(\Sigma_0^{-1}/J + \bar{\Sigma}_j^{-1})
\end{aligned}
$$

- Fit J Gaussians, one for each $\{\theta_{j,1}, \ldots, \theta_{j,T}\}$ and take their product

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

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

$$
\begin{aligned}
\bar{\Sigma}_j &= \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
\Sigma &= (\Sigma_0^{-1} + \sum_{j=1}^{J} \bar{\Sigma}_j^{-1})^{-1} \quad (\Sigma_0 \text{ is the prior's covariance}) \\
W_j &= \Sigma(\Sigma_0^{-1}/J + \bar{\Sigma}_j^{-1})
\end{aligned}
$$

- Fit J Gaussians, one for each $\{\theta_{j,1}, \ldots, \theta_{j,T}\}$ and take their product

$$
\begin{aligned}
\bar{\mu}_j &= \text{sample mean of } \{\theta_{j,1}, \ldots, \theta_{j,T}\}, \quad \bar{\Sigma}_j = \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
\hat{\Sigma}_J &= (\sum_{j=1}^{J} \bar{\Sigma}_j^{-1})^{-1}
\end{aligned}
$$

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

$$\begin{aligned}
\bar{\Sigma}_j &= \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
\Sigma &= (\Sigma_0^{-1} + \sum_{j=1}^{J} \bar{\Sigma}_j^{-1})^{-1} \quad (\Sigma_0 \text{ is the prior's covariance}) \\
W_j &= \Sigma(\Sigma_0^{-1}/J + \bar{\Sigma}_j^{-1})
\end{aligned}$$

- Fit J Gaussians, one for each $\{\theta_{j,1}, \ldots, \theta_{j,T}\}$ and take their product

$$\begin{aligned}
\bar{\mu}_j &= \text{sample mean of } \{\theta_{j,1}, \ldots, \theta_{j,T}\}, \quad \bar{\Sigma}_j = \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\
\hat{\Sigma}_J &= (\sum_{j=1}^{J} \bar{\Sigma}_j^{-1})^{-1}, \quad \hat{\mu}_J = \hat{\Sigma}_J(\sum_{j=1}^{J} \bar{\Sigma}_j^{-1} \bar{\mu}_j) \quad \text{(cov and mean of prod. of Gaussians)}
\end{aligned}$$

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

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

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

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

- Fit J Gaussians, one for each $\{\theta_{j,1}, \ldots, \theta_{j,T}\}$ and take their product

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

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

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

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

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

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

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

- Fit J Gaussians, one for each $\{\theta_{j,1}, \ldots, \theta_{j,T}\}$ and take their product

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

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

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

- For detailed proof and other more sophisticated ways, please refer to the provided reading

*Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Computing Consensus Samples: Some Methods

- Weighted avg: $\hat{\theta}_t = \sum_{j=1}^{J} W_j \theta_{j,t}$ where W_j can be learned. Assuming Gaussian prior and lik.

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

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

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

- Fit J Gaussians, one for each $\{\theta_{j,1}, \ldots, \theta_{j,T}\}$ and take their product

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

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

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

- For detailed proof and other more sophisticated ways, please refer to the provided reading
- Note: VI can also be parallelized using similar techniques

* Patterns of Scalable Bayesian Inference (Angelino et al, 2016)

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- Conjugate models with one "main" parameter: Straightforward posterior updates

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- Conjugate models with one "main" parameter: Straightforward posterior updates
- MLE-II/MAP-II: Often useful for estimating the hyperparameters

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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 ans sampling methods can be used to get full posterior for complex models

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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 ans 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)

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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 ans 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

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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 ans 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)

Inference Methods: Summary

- MLE/MAP: Straightforward for differentiable models (can even use automatic diffentiation)
- 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 ans 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 approximations