Approximating Distributions via Sampling (contd)

CS772A: Probabilistic Machine Learning

Piyush Rai

Recap: Metropolis-Hastings Sampling

- Based on sampling $\boldsymbol{z}^{(t+1)}$ in the chain using a proposal distribution $q(\boldsymbol{z}|\boldsymbol{z}^{(t)})$
  - Generate the next candidate $\boldsymbol{z}^*$ from $q(\boldsymbol{z}|\boldsymbol{z}^{(t)})$ and compute its acceptance probability

$$A(\boldsymbol{z}^*, \boldsymbol{z}^{(t)}) = \min\left(1, \frac{\tilde{p}(\boldsymbol{z}^*) q(\boldsymbol{z}^{(t)}|\boldsymbol{z}^*)}{\tilde{p}(\boldsymbol{z}^{(t)}) q(\boldsymbol{z}^*|\boldsymbol{z}^{(t)})}\right)$$

  - Draw $u \sim \text{Unif}(0,1)$.
    - If $A(\boldsymbol{z}^*, \boldsymbol{z}^{(t)}) > u$, accept $\boldsymbol{z}^*$, set $z^{(t+1)} = \boldsymbol{z}^*$
    - Otherwise reject $\boldsymbol{z}^*$
- This will generate a chain $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(T)}$ of samples
- Transition function (TF) of this Markov Chain
  - $T(\boldsymbol{z}^{(t)} \to \boldsymbol{z}^*) = A(\boldsymbol{z}^*, \boldsymbol{z}^{(t)}) q(\boldsymbol{z}^*|\boldsymbol{z}^{(t)})$ if state changed
  - $T(\boldsymbol{z}^{(t)} \to \boldsymbol{z}^*) = q(\boldsymbol{z}^{(t)}|\boldsymbol{z}^{(t)}) + \sum_{\boldsymbol{z}^* \neq \boldsymbol{z}^{(t)}} (1 - A(\boldsymbol{z}^*, \boldsymbol{z}^{(t)})) q(\boldsymbol{z}^*|\boldsymbol{z}^{(t)})$ otherwise
- This TF ensures that $p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z_p}$ is the "stationary distribution" of this Markov chain

MCMC: A Little Bit of Theory

- To construct a Markov Chain $\boldsymbol{z}^{(0)}, \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(T)}$ we need
  - An initial state distribution $p_{in}(\boldsymbol{z}^{(0)})$ to generate the first state in the chain
  - A Transition Function (TF) $T_\ell\left(\boldsymbol{z}^{(\ell)} \rightarrow \boldsymbol{z}^{(\ell+1)}\right)$ to generate subsequent states in the chain
  - Homogeneous Markov Chain: The TF is the same for all ℓ , i.e., $T_\ell = T$
- A distribution $\pi(\boldsymbol{z})$ is the stationary distribution of this Markov chain if

$$\pi(\boldsymbol{z}') = \int \pi(\boldsymbol{z}) T(\boldsymbol{z} \rightarrow \boldsymbol{z}') d\boldsymbol{z}$$

- Above implies that if $\boldsymbol{z}^{(t)} \sim \pi(\boldsymbol{z})$ then $\boldsymbol{z}^{(t+1)} \sim \pi(\boldsymbol{z})$ as $t \rightarrow \infty$
  - Thus samples in the chain eventually converge to samples from distribution $\pi(\boldsymbol{z})$
- In MCMC algos, we choose T so that $\pi(\boldsymbol{z})$ is the stationary distribution
  - MH by construction ensures that
- Detailed balance: $\pi(\boldsymbol{z}) T(\boldsymbol{z} \rightarrow \boldsymbol{z}') = \pi(\boldsymbol{z}') T(\boldsymbol{z}' \rightarrow \boldsymbol{z})$

MH transition function ensures detailed balance too

Note: Detailed balance is a sufficient but not necessary for stationary distribution to exist

A Special Case of MH: Gibbs Sampling

- Goal: Sample from a joint distribution $p(\boldsymbol{z})$ where $\boldsymbol{z} = [z_1, z_2, \ldots, z_M]$
- Suppose we can't sample from $p(\boldsymbol{z})$ but can sample from each conditional $p(z_i|\boldsymbol{z}_{-i})$
  - In Bayesian models, can be done easily if we have a locally conjugate model
- For Gibbs sampling, the proposal is the conditional distribution $p(z_i|\boldsymbol{z}_{-i})$
- Gibbs sampling samples from these conditionals in a cyclic order

Hence no need to compute it

- Gibbs sampling is equivalent to MH sampling with acceptance prob. $= 1$

$$A(\boldsymbol{z}^*, \boldsymbol{z}) = \frac{p(\boldsymbol{z}^*)q(\boldsymbol{z}|\boldsymbol{z}^*)}{p(\boldsymbol{z})q(\boldsymbol{z}^*|\boldsymbol{z})} = \frac{p(z_i^*|\boldsymbol{z}_{-i}^*)p(\boldsymbol{z}_{-i}^*)p(z_i|\boldsymbol{z}_{-i}^*)}{p(z_i|\boldsymbol{z}_{-i})p(\boldsymbol{z}_{-i})p(z_i^*|\boldsymbol{z}_{-i})} = 1$$

where we use the fact that $\boldsymbol{z}_{-i}^* = \boldsymbol{z}_{-i}$

Since only one component is changed at a time

Gibbs Sampling: Sketch of the Algorithm

- M: Total number of variables, T: number of Gibbs sampling iterations

1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.

Assuming $\boldsymbol{z} = [z_1, z_2, \ldots, z_M]$

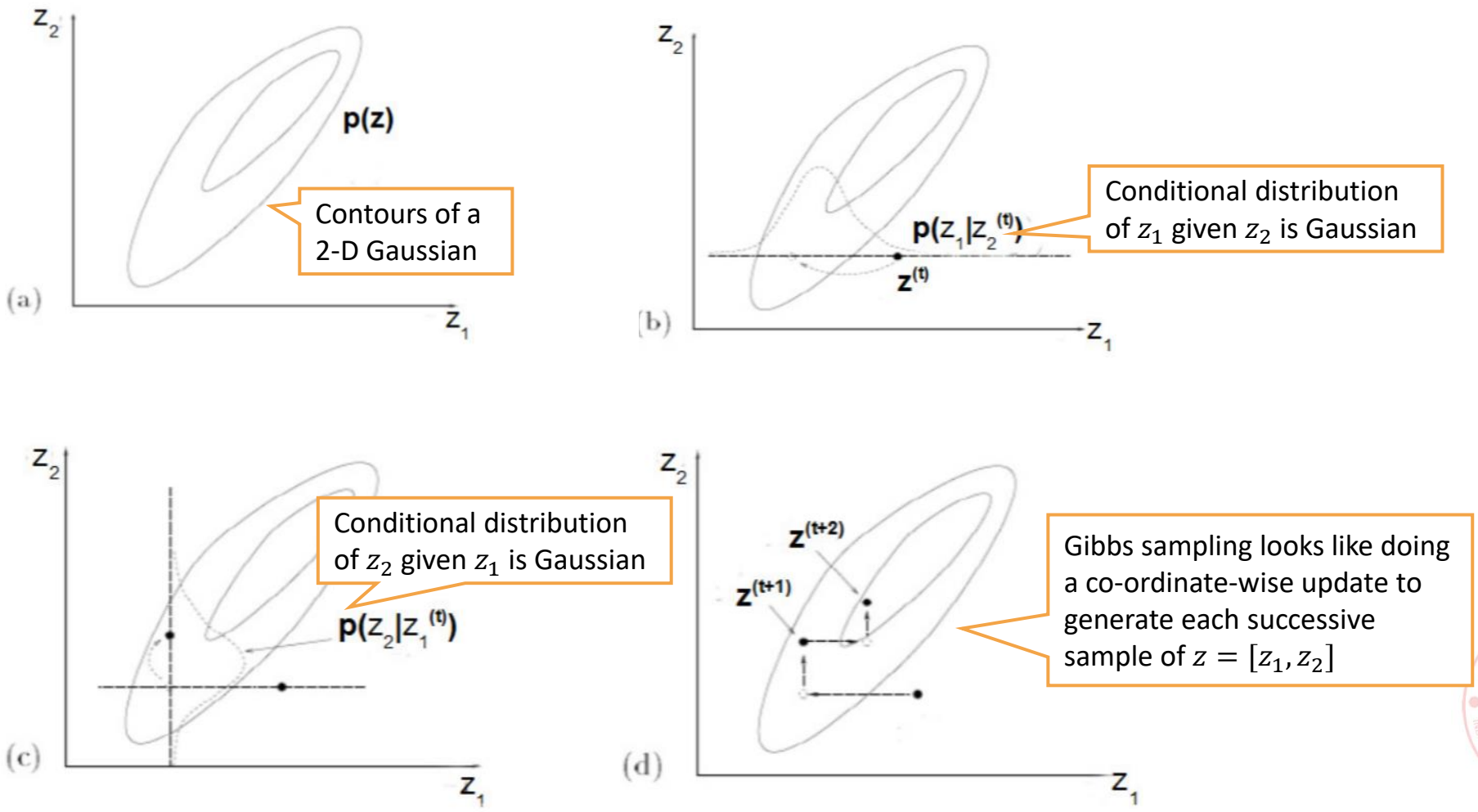
CP of each component of $\boldsymbol{z}$ uses the most recent values (from this or the previous iteration) of all the other components

Each iteration will give us one sample $\boldsymbol{z}^{(\tau)}$ of $\boldsymbol{z} = [z_1, z_2, \ldots, z_M]$

- Note: Order of updating the variables usually doesn't matter (but see "Scan Order in Gibbs Sampling: Models in Which it Matters and Bounds on How Much" from NIPS 2016)

Gibbs Sampling: A Simple Example

- Can sample from a 2-D Gaussian using 1-D Gaussians



Gibbs Sampling: Another Example

- Bayesian linear regression: $p(y_n|\boldsymbol{x}_n, \boldsymbol{w}, \beta) = \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1})$, $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \lambda^{-1}I)$, $p(\lambda) = \text{Gamma}(\lambda|a, b)$, $p(\beta) = \text{Gamma}(\beta|c, d)$. Gibbs sampler for $p(w, \lambda, \beta|\boldsymbol{X}, \boldsymbol{y})$ will be
- Initialize λ, β as $\lambda^{(0)}, \beta^{(0)}$. For iteration $t = 1,2, \ldots, T$
  - Generate a random sample of $\boldsymbol{w}$ by sampling from its CP as

$$\boldsymbol{w}^{(t)} \sim \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}^{(t-1)}, \boldsymbol{\Sigma}^{(t-1)}) \quad \text{where}$$

$$\boldsymbol{\Sigma}^{(t-1)} = \left(\beta^{(t-1)}\boldsymbol{X}^\top\boldsymbol{X} + \lambda^{(t-1)}\right)^{-1}$$

$$\boldsymbol{\mu}^{(t-1)} = \left(\boldsymbol{X}^\top\boldsymbol{X} + \frac{\lambda^{(t-1)}}{\beta^{(t-1)}}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

  - Generate a random sample of λ by sampling from its CP as

$$\lambda^{(t)} \sim \text{Gamma}\left(\lambda|a + \frac{D}{2}, b + \frac{\boldsymbol{w}^{(t)\top}\boldsymbol{w}^{(t)}}{2}\right)$$

  - Generate a random sample of β by sampling from its CP as

$$\beta^{(t)} \sim \text{Gamma}\left(\beta|c + \frac{N}{2}, d + \frac{\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^{(t)}\|^2}{2}\right)$$

Note: Assuming these are post-burn-in samples and thinning (if any) is also considered

- The posterior's approximation is the set of collected samples $\{\boldsymbol{w}^{(t)}, \lambda^{(t)}, \beta^{(t)}\}_{t=1}^{T}$

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 θ to denote all the unknowns

Can use automatic differentiation methods for this

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

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

Likelihood

Prior

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

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

And then accept/reject (MH)

η set s.t. acceptance rate is around 0.6

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

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

Helps also incorporate the curvature info of the posterior

If gradient is pre-multiplied by a preconditioner matrix $M(\theta^{(t-1)})$: Simplified Manifold MALA

One option to use for $M(\theta^{(t-1)})$ is the second derivative of the unnorm. posterior

And then accept/reject (MH)

Known as Metropolis-Adjusted Langevin Algorithm (MALA)

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

"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

And then accept/reject (MH)

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

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. 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 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:

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

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 η_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)

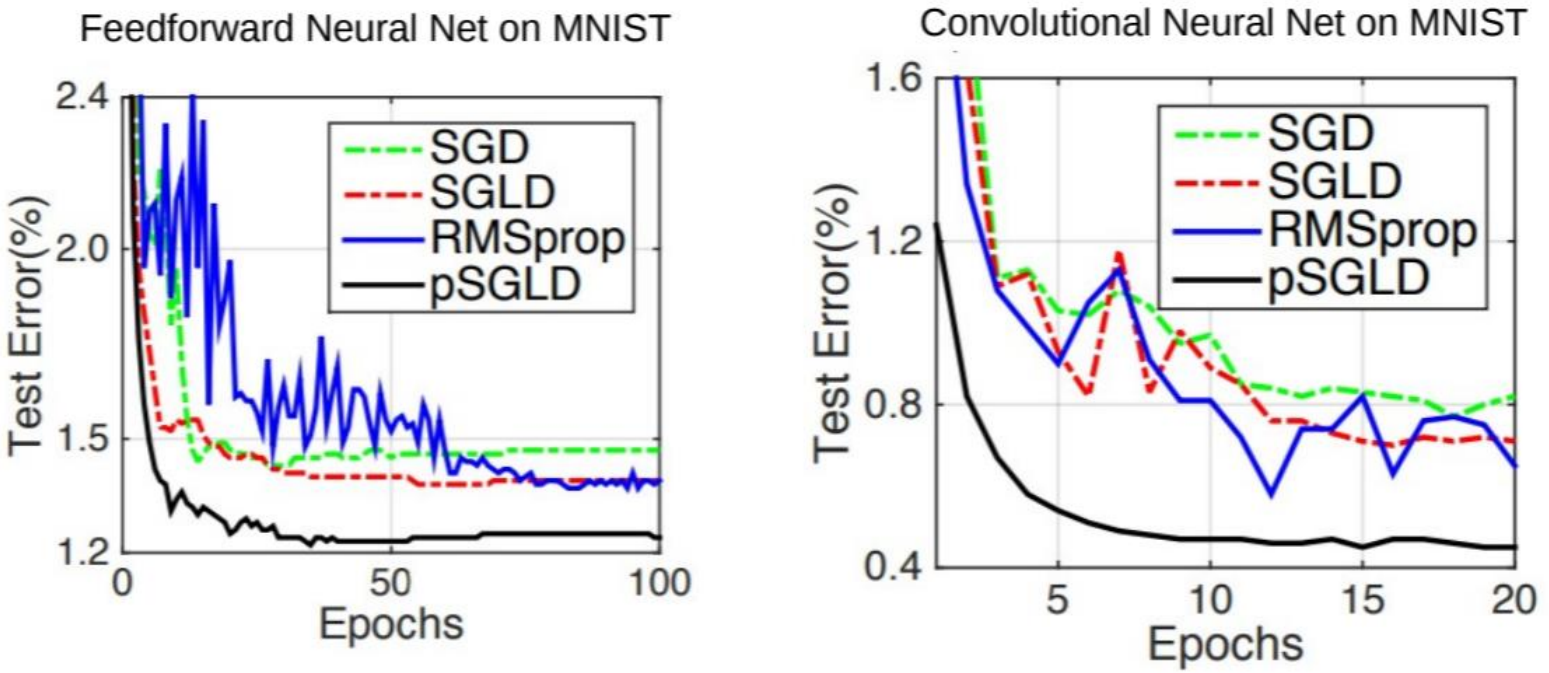
Improvements to SGLD

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

Based on reparametrizing the constrained variables to make them unconstrainted

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 avging (SWA)

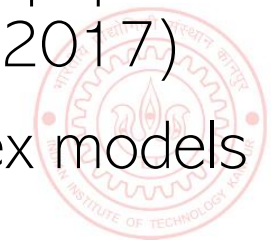
$$\theta_{SWA} = \frac{1}{T}\sum_{t=1}^{T}\theta_t$$

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

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

Approach known as SWA-Gaussian (SWAG)

- If we want full cov., we can use a low-rank approx. of $\boldsymbol{\Sigma}$ (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



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 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 "position" then $U(\theta) = -\log p(\mathcal{D}|\theta)p(\theta)$ 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

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

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

Constant w.r.t. time

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

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

Generating Samples in HMC

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

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

$$\left(H(\theta, \boldsymbol{r}) = U(\theta) + \tfrac{1}{2}\boldsymbol{r}^\top M^{-1}\boldsymbol{r} = U(\theta) + K(\boldsymbol{r}) \right)$$

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

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

- 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 $\rho_\ell = \rho$ for $\ell < L$ and $\rho_L = \rho/2$
    - For $\ell = 1{:}L$

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

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

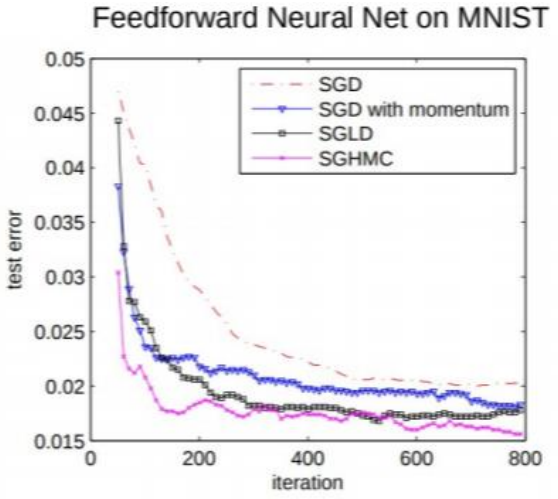
  - Perform MH accept/reject test on $(\boldsymbol{\theta}_L, \boldsymbol{r}_L)$. If accepted $\theta^{(s)} = \theta_L$

A single sample generated by taking L steps

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

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)

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

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

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

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

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

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 on θ

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

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

$$\begin{aligned} \bar{\mu}_j &= \text{sample mean of } \{\theta_{j,1}, \ldots, \theta_{j,T}\}, \quad \bar{\Sigma}_j = \text{sample covariance of } \{\theta_{j,1}, \ldots, \theta_{j,T}\} \\ \hat{\Sigma}_J &= (\sum_{j=1}^{J} \bar{\Sigma}_j^{-1})^{-1}, \quad \hat{\mu}_J = \hat{\Sigma}_J(\sum_{j=1}^{J} \bar{\Sigma}_j^{-1}\bar{\mu}_j) \quad \text{(cov and mean of prod. of Gaussians)} \\ \hat{\theta}_t &\sim \mathcal{N}(\hat{\mu}_J, \hat{\Sigma}_J), t = 1, \ldots, T \quad \text{(the final consensus samples)} \end{aligned}$$

- 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 $p(\boldsymbol{Z}|\boldsymbol{X})$ by another distribution $q(\boldsymbol{Z}|\boldsymbol{\phi})$
- Sampling uses $\boldsymbol{S}$ samples $\boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}, \ldots, \boldsymbol{Z}^{(S)}$ to approximate $p(\boldsymbol{Z}|\boldsymbol{X})$
- 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 $\boldsymbol{\phi}$
  - 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: $p(x_*|X) = \int p(x_*|Z)p(Z|X)dZ \approx \frac{1}{S}\sum_{s=1}^{S} p\left(x_*|Z^{(s)}\right)$

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

Closed form if integral is tractable (otherwise Monte Carlo avg still needed for PPD)

Compressing the S samples into something more compact

- 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