Sampling (Contd.) and Gradient-based Monte Carlo

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

April 3, 2018

Sampling Methods: Label Switching Issue

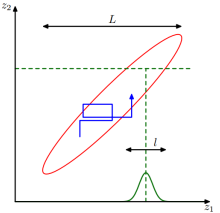
- A subtle but important issue
- Suppose we are given samples $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(S)}$ from the posterior $p(\mathbf{Z}|\mathbf{X})$
- We can't always simply "average" them to get the "posterior mean" $\bar{\mathbf{Z}}$
- Reason: Non-identifiability of latent variables in models that have multiple posterior modes
- Example: In a clustering model (e.g., GMM), the likelihood is invariant to how we label clusters
- Therefore averaging latent variables across samples can be meaningless
- Quantities not affected by permutations of latent variables can be safely averaged
  - E.g., probability that two points belong to the same cluster (e.g., in GMM)

MCMC and Random Walk

- MCMC methods use a proposal distribution to draw the next sample given the previous sample

$$\theta^{(t)} \sim \mathcal{N}(\theta^{(t-1)}, \sigma^2)$$

- .. and then we accept/reject (if doing MH) or always accept (if doing Gibbs sampling)
- Such proposal distributions typically lead to a random-walk behavior (e.g., a zig-zag trajectory in Gibbs sampling) and may lead to very slow convergence (pic below: $\theta = [z_1, z_2]$)



- Can be especially critical when the components of θ are highly correlated

Using Gradient Information in Sampling

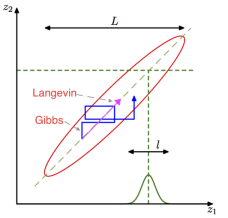
- Two prominent methods
  - Langevin Dynamics based Monte Carlo
  - Hamiltonian Monte Carlo or "Hybrid" Monte Carlo (HMC)
- Both methods incorporate the posterior's gradient in their proposal distribution

Using Gradient Info via Langevin Dynamics

- Constructs proposal distribution using gradient of the log-posterior
- Gradient of the log-posterior: $\nabla_\theta \log \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \nabla_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]$
- Now let's construct a proposal and generate a random sample as follows

$$\begin{aligned} \theta^* &= \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} \\ \theta^{(t)} &\sim \mathcal{N}(\theta^*, \eta) \qquad \text{(and then accept/reject using an MH step)} \end{aligned}$$

- This method is called Langevin dynamics (Neal, 2010). Has its origins in statistical Physics.



Langevin Dynamics (Contd)

- Note that the updates of θ can also be written in the form

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \eta)$$

- After this update, we accept/reject $\theta^{(t)}$
- Equivalent to gradient-based MAP estimation with added noise (plus the accept/reject step)
- The random noise ensures that we aren't stuck just on the MAP estimate but explore the posterior
- A few technical conditions (Welling and Teh, 2011)
  - The noise variance needs to be controlled (here, we are setting it to twice the learning rate)
  - As $\eta \to 0$, the acceptance probability approaches 1 and we can always accept
- Note that the procedure is almost as fast as MAP estimation!

Stochastic Gradient (Online) Langevin Dynamics

- Allows scaling up MCMC algorithms by processing data in small minibatches
- Stocahstic Gradient Langevin Dynamics (SGLD) is one such example
- Basically an online extension of the Langevin Dynamics method we saw earlier
- Given minibatch $\mathcal{D}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$. Then the (stochastic) Langevin dynamics update is

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

- Basically, instead of doing gradient descent, SGLD does stochastic gradient descent + MH
  - Valid under some technical conditions on learning rate, variance of proposal distribution, etc.
- Recent flurry of work on this topic (see "Bayesian Learning via Stochastic Gradient Langevin Dynamics" by Welling and Teh (2011) and follow-up works)

SGLD: Some Comments

- Very easy to implement (only need to compute gradients of log-lik and log-prior)
- If not doing accept/reject, we just need to do the following for each minibatch of data

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

- It's just like SGD updates (+added Gaussian noise). Highly scalable even when N is very large
  - Almost as efficient as doing MAP estimation using stochastic gradient methods
- Applies to non-conjugate models easily (so long as we can take derivatives)
- Several improvements on SGLD in the past couple of years
  - Better choice of learning rate and pre-conditioners for improving convergence
  - Extending to the case when θ has some constraints (e.g., a point on simplex)
  - Theoretical analysis and justification for the "correctness" of the procedure
- Many recent applications to models such as logistic regression, matrix factorization, etc.

Using Gradient Info via Hamiltonian/Hybrid Monte Carlo (HMC)

- 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"
- Assume the momentum of the system to be $\boldsymbol{r}$
- Can now define a joint distribution over the position and momentum as

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

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

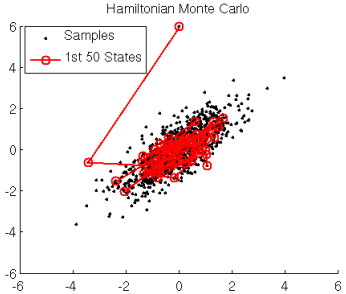
Using Gradient Info via Hamiltonian/Hybrid Monte Carlo (HMC)

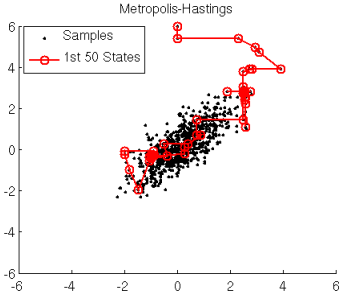
- Fact: The total energy $H(\theta, \boldsymbol{r}) = U(\theta) + K(\boldsymbol{r})$ is always constant
- Given an initial $(\theta, \boldsymbol{r})$, Hamiltonian Dynamics defines how $(\theta, \boldsymbol{r})$ changes w.r.t. continuous time t

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

- We can use these equations to update $(\theta, \boldsymbol{r}) \rightarrow (\theta^*, \boldsymbol{r}^*)$
- The step of generating $(\theta^*, \boldsymbol{r}^*)$ usually requires discretization of the continuous time
  - Many ways, such as "leapfrog" method (won't discuss the details here, will provide reference)
- HMC typically has very low rejection rate (that too, primarily due to discretization error)
- A lot of renewed interest in HMC (you may check out NUTS - No U-turn Sampler)
  - Probabilistic Programming packages such as Stan/Edward contain implementations of HMC
- Can also do online HMC (Stochastic Gradient HMC).

Illustration: HMC vs vanilla MH to Sample from 2D Gaussian





Some Comments

- Sampling is a very general technique for generating samples from a distribution
- Looked at some MCMC algorithms, such as MH and Gibbs sampling
  - MH is very general. Gibbs is very easy to derive is local conditionals can be found easily
- MCMC is random-walk based method, so convergence can be slow
- Using gradient information can make convergence faster (e.g., Langevin methods, HMC)
- MCMC can be scaled up to large datasets (online and distributed MCMC methods)
- Some other issues of MCMC
  - Assessing convergence can be hard (there exist methods but beyond the scope of this class)
  - Representation of the distribution requires storing all the samples (also, making predictions can be slow since it requires empirically averaging the predictions computed using each MCMC sample)