Computational Statistics & Machine Learning

Lecture 13

Langevin Markov chain Monte Carlo II - Applications

Mark Girolami
mag92@cam.ac.uk

Department of Engineering
University of Cambridge

April 26, 2022

Overview

- Langevin MCMC for Bayesian inference
- Modern large-scale Bayesian ML Applications

Langevin schemes for Bayesian inference

Assume that our target distribution is

$$\pi(x) \propto p(x) \prod_{i=1}^{n} L(y_i|x),$$

where $p(x)$ is the prior distribution over x and $L(y_i|x)$ is the likelihood for data vector y_i.

- Large scale probabilistic models
- Bayesian neural networks

As we have seen, we can implement the Langevin schemes with the gradient

$$\nabla \log \pi(x) := \nabla \log p(x) + \sum_{i=1}^{n} \nabla \log L(y_i|x).$$

But what if $n \gg 1$ as in modern applications?

Langevin schemes for Bayesian inference

Using the recursion

$$X_{k+1} = X_k + \gamma \left(\nabla \log p(X_k) + \sum_{i=1}^{n} \nabla \log L(y_i|X_k) \right) + \sqrt{2\gamma} Z_k$$

will provide us samples from our target π. We can use

- MALA
- ULA

Given the availability of automatic differentiation, one can handle very complex likelihoods L automatically.

- This enables the use Bayesian neural networks.

But what if $n \gg 1$?

Langevin schemes for scalable Bayesian inference

Problems with MALA in modern ML

If we wanted to apply MALA, we run into two problems:

- The gradient computation for the proposal is too costly

$$\nabla \log \pi(x) := \nabla \log p(x) + \sum_{i=1}^{n} \nabla \log L(y_i|x),$$

as for large n, the large sum has to be recomputed every iteration.

- $\mathcal{O}(n)$ complexity every iteration.

Langevin schemes for scalable Bayesian inference

Problems with MALA in modern ML

- Another issue specific to MALA is the computation of acceptance probabilities:

$$\alpha := \min\left\{1, \frac{\pi(\overline{X}_{k+1})q(X_k|\overline{X}_{k+1})}{\pi(X_k)q(\overline{X}_{k+1}|X_k)}\right\},$$

  where $\pi(x) \propto p(x)\prod_{i=1}^{n} L(y_i|x)$, an unnormalised evaluation is enough, but still needs $\mathcal{O}(n)$ computations.
- While we can avoid $\mathcal{O}(n)$ acceptance probability cost (see next slide), we cannot avoid $\mathcal{O}(n)$ gradient computation.

Langevin schemes for scalable Bayesian inference

How can we avoid $\mathcal{O}(n)$ gradient computation?

- This would enable us to employ ULA-like Metropolis free schemes.

Recall

$$\nabla \log \pi(x) := \nabla \log p(x) + \sum_{i=1}^{n} \nabla \log L(y_i|x).$$

Since the bottleneck in this gradient is the sum term

- Can we estimate the sum cheaply?
- What kind of property of the estimator would preserve the same properties as the ULA?
  - Unbiasedness.

Langevin schemes for scalable Bayesian inference

An unbiased gradient can be computed if we *subsample* data, i.e., on a mini-batch chosen at random.

- At iteration k, choose an index set $\mathcal{I}_k \subset \{1, \ldots, n\}$.
- Approximate the gradient at X_k (the state of the chain)

$$\nabla \widehat{\log \pi(X_k)} = \nabla \log p(X_k) + \frac{n}{|\mathcal{I}_k|} \sum_{i \in \mathcal{I}_k} \nabla \log L(y_i | X_k).$$

where $|\mathcal{I}_k|$ is the size of the mini-batch.

One can check the unbiasedness

$$\mathbb{E}\left[\nabla \widehat{\log \pi(x)}\right] = \nabla \log \pi(x).$$

Langevin schemes for scalable Bayesian inference

Run a ULA chain with $\widehat{\nabla \log \pi(X_k)}$?

- The scheme is called stochastic gradient Langevin dynamics (SGLD)

$$X_{k+1} = X_k + \gamma \widehat{\nabla \log \pi(X_k)} + \sqrt{2\gamma} W_{k+1},$$

  where $\mathbb{E}[\widehat{\nabla \log \pi(X_k)}] = \nabla \log \pi(X_k)$.
- Variants of the SGLD are de facto choice to train Bayesian NNs.
- Each iteration costs the size of the mini-batch $\mathcal{O}(|\mathcal{I}_k|)$.
- Similar convergence guarantees as ULA, i.e.

$$W_2(\pi_k, \pi) \leq \mathcal{O}(e^{-\gamma m k} + \gamma^{1/2}).$$

  for m-strongly log-concave π.

But the variance of the gradient estimates should be controlled!

Langevin schemes for scalable Bayesian inference

The idea of using stochastic (mini-batch) gradients in gradient-based MCMC is general.