Monte Carlo Methods on High Dimensional Data

Rel Guzman-Apaza
Universidad Nacional de San Agustin, Arequipa, Peru

Markov Chain Monte Carlo (MCMC) simulation is a family of stochastic algorithms that are commonly used to approximate probability distributions by generating samples. The aim of this proposal is to deal with the problem of doing that job on a large scale because due to the increasing power computational demands of data being tall or wide, a study that combines statistical and engineering expertise can be made in order to achieve hardware-accelerated MCMC inference. In this work, I attempt to advance the theory and practice of approximate MCMC methods by developing a toolbox of distributed MCMC algorithms, and then a new method for dealing with large-scale problems will be proposed, or else a framework for choosing the most appropriate method will be established. Papers like [1] provide a comprehensive review of the existing literature regarding methods to tackle big data problems. My idea is to tackle divide and conquer approaches since they can work distributed in several machines or else Graphics Processing Unit (GPUs), so I cover the theory behind these methods; then, exhaustive experimental tests will help me compare and categorize them according to their limitations in wide and tall data by considering the dataset size n, sample dimension d, and number of samples T to produce.

1 MCMC for big data

MCMC is a method for approximating probability distributions of variables, and it is among many, the most popular because it normally provides both precise and pretty fast results. The most common methods for both probabilistically finding parameters θ and latent variables z are shown in the table below [2] that suggests that there are commonly two popular ways to do approximation: variational inference and Monte Carlo. Besides MCMC being slow for big data, methods like Metropolis Hastings generate correlated samples and are deficient when working with multi-modal distributions [3].

Accurate	Full Bayesian inference $p(z, \theta \mid \mathcal{X})$	Slow
	Variational inference: mean field $q(z)q(\theta) \approx p(z, \theta \mid \mathcal{X})$	
	MCMC $z_i, \theta_i \sim p(z, \theta \mid \mathcal{X})$	
	Expectation maximization $q(z)$, $\theta = \theta_{MAP}$	
	Variational EM: mean field $q_1(z_1), \ldots, q_d(z_d), \theta = \theta_{MAP}$	
Inaccurate	MCMC EM $z_i \sim p(z \mid \theta, \mathcal{X}), \theta = \theta_{\mathcal{MAP}}$	Fast

However, because Monte Carlo is more precise, then there are essentially two approaches to make it fast as well when dealing with big data in the literature: divide and conquer and subsampling-based. Divide and conquer approaches divide the initial dataset, run MCMC separately, and then combine results to obtain an approximation of the posterior while subsampling approaches simply aim at reducing the number of individual data point likelihood evaluations necessary at each iteration of the MCMC algorithm [1].

2 Large-scale MCMC

The general idea is to split the data $\mathcal{X}$ of size n across many machines, which is the divide and conquer approach. The data can be divided into S conditionally independent batches/data subsets $\mathbf{x}_1, \ldots, \mathbf{x}_S$, and one can run separate MCMC chains on each machine, and finally combine all the MCMC draws $p\left(\theta|\mathcal{X}\right) \propto \prod_{s=1}^{S} p(\mathbf{x}_s|\theta)p(\theta)^{1/S}$ where $p(\theta) = \prod_{s=1}^{S} p(\theta)^{1/S}$ to preserve the total prior. The time complexity to produce T samples on each subposterior $p(\theta|\mathbf{x}_s)$ would be $O(Tn/S)$, which is common to all divide-and-conquer MCMC algorithms [4], and it seems to be $O(dTn/S)$ when producing d-dimensional samples. At any rate, a question that arises is how to combine draw samples from different subposteriors, and some methods are minimally described below.

- A method called Consensus Monte Carlo assumes that each subposterior $p(\theta|\mathbf{x}_s) \sim N(\mu_s, \Omega_s)$. Then $p\left(\theta|\mathcal{X}\right) \propto \prod_{s=1}^{S} p(\theta|\mathbf{x}_s) = N(\mu, \Omega)$, and the proposal from [5] is to weight the averages of the subposterior draws.

- In [6], a kernel-based way to combine subposterior samples is proposed, and it also compares its performance with Consensus Monte Carlo.

- Finding the geometric median of subset posterior distributions is another method [7]. The algorithm consists of running MCMC on scaled subposteriors and returning a median subposterior of the S subposteriors. This median has two advantages: it provides a better approximation of the full posterior than the individual subposteriors, and it is more resistant to outliers.

- Combining subposteriors using Gaussian Processes (GPs) is a more recent idea. In [4], the authors first run an MCMC method on each subposterior and obtain draws $\theta_s^{(i)}$ and $\log p(\theta_s^{(i)}|\mathbf{x}_s)$ evaluations for $i = 1, \ldots, I$, which is an arbitrary set of size I.

  $$\mathcal{D}_s = \left\{\theta_s^{(1:I)}, \log p(\theta_s^{(1:I)}|\mathbf{x}_s)\right\}.$$

  Then, a noise-free GP regression is fit to $\mathcal{D}_s$ with response $\log p(\theta_s^{(1:I)}|\mathbf{x}_s)$. The predictive distribution for the log subposterior at a new set of parameter values $\theta^{(1:J)}$ is $\log p_s(\theta^{(1:J)})|\mathcal{D}_s \sim GP\left(\mu_s(\theta^{(1:J)}), \Sigma_s(\theta^{(1:J)})\right)$. Finally, a sum of subposterior GPs approximates the full data $\log p(\theta|\mathbf{y})$

  $$\log p(\theta^{(1:J)}|\mathbf{x})|\mathcal{D} \sim GP\left(\sum_{s=1}^{S} \mu_s(\theta^{(1:J)}), \sum_{s=1}^{S} \Sigma_s(\theta^{(1:J)})\right)$$

- In that same paper [4], a variant of the Metropolis Hastings algorithm is used. It is the popular No-U-Turn Sampler (NUTS) sampler, which is an adaptively tuned Hamiltonian Monte Carlo (HMC), and a GP-HMC sampler is proposed for big data.

An overall comparison of state of the art methods in various dataset size, sample dimensions, and the number of CPUs and GPUs is presented in [8]. Methods like the ones above aren't mentioned although the categories described are pretty clear.

3 Experiments and methodology

At big-data scale, many every method has its weaknesses and strengths, and a way to compare them is by considering how well they perform for different amounts of data n and descriptors or features d. This

n	d	Single CPU	Single GPU	CPU Cluster	GPU Cluster
Small	Small	Standard MCMC	GPU-accelerated particle filters	Independent parallel chains	Not needed
Small	Big	HMC or Gibbs sampling	HMC or GPU-accelerated Gibbs sampling	Asynchronous Gibbs sampling	No Bayesian method
Big	Small	Continuous-time MCMC with subsampling and control variates	Continuous-time MCMC or GPU-accelerated Gibbs sampling	Asynchronous Gibbs sampling or methods based on sharing	No Bayesian method
Big	Big	Hopeless: point estimates only	Model-specific	Model-specific	No Bayesian method

study is in progress and will first categorize the different ways of combining subposterior draws and review various types of constraints in those methods and their characteristics such as time complexity. Some of the methods are open source, others part of libraries like Edward [9]. Second, based on this understanding, a decent methodology for dealing with large-scale problems will be proposed. This whole study is being conducted taking into account the drawbacks of MCMC methods: speed, autocorrelation, performance in multi-modal distributions, so the third step will be to use proper evaluation methods.

References

[1] R. Bardenet, A. Doucet, and C. Holmes, "On markov chain monte carlo methods for tall data," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1515–1557, 2017.

[2] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[3] H. Tak, X.-L. Meng, and D. A. van Dyk, "A repelling-attracting metropolis algorithm for multimodality," *Journal of Computational and Graphical Statistics*, no. just-accepted, 2017.

[4] C. Nemeth, C. Sherlock *et al.*, "Merging mcmc subposteriors through gaussian-process approximations," *Bayesian Analysis*, 2017.

[5] S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch, "Bayes and big data: The consensus monte carlo algorithm," *International Journal of Management Science and Engineering Management*, vol. 11, no. 2, pp. 78–88, 2016.

[6] W. Neiswanger, C. Wang, and E. Xing, "Asymptotically exact, embarrassingly parallel mcmc," *arXiv preprint arXiv:1311.4780*, 2013.

[7] S. Minsker, S. Srivastava, L. Lin, and D. Dunson, "Scalable and robust bayesian inference via the median posterior," in *International Conference on Machine Learning*, 2014, pp. 1656–1664.

[8] D. Draper and A. Terenin, "Comment: A brief survey of the current state of play for bayesian computation in data science at big-data scale," *arXiv preprint arXiv:1712.00849*, 2017.

[9] D. Tran, M. D. Hoffman, R. A. Saurous, E. Brevdo, K. Murphy, and D. M. Blei, "Deep probabilistic programming," *arXiv preprint arXiv:1701.03757*, 2017.