CS267 Final Project: Distributed Stochastic Gradient Langevin Dynamics on MPI

Feynman Liang

2018/05/08

1 Introduction

In Bayesian statistics, one oftentimes need to estimate expectations with respect to a posterior distribution $\mathbb{P}(\theta \mid X)$ over parameters θ after observing data X. For example, the posterior mean

$$\hat{\theta}(X) = \mathbb{E}[\theta \mid X] \tag{1}$$

is a common quantity of interest because it minimizes mean squared error loss.

Bayesian statistical models consist of a specification of a prior distribution $\mathbb{P}(\theta)$ over the model parameters as well as a likelihood $\mathbb{P}(X \mid \theta)$ yielding the probability of the data X under parameter settings θ. By Bayes rule, the posterior distribution is given by

$$\mathbb{P}(\theta \mid X) = \frac{\mathbb{P}(X \mid \theta)\mathbb{P}(\theta)}{\int \mathbb{P}(X \mid \theta')\mathbb{P}(\theta')d\theta'} \tag{2}$$

Unfortunately, closed-form solutions of the integral in the denominator (aka the *partition function*) are not always available, precluding analytical computation of expectations with respect to the posterior.

In light of this complication, a common practice is to perform Monte Carlo. If a method to generate posterior samples $\theta_i \sim \mathbb{P}(\cdot \mid X)$ were available, then the law of large numbers and continuous mapping theorem can be used to argue that for any continuous function f

$$n^{-1}\sum_{i=1}^{n} f(\theta_i) \overset{a.s.}{\to} \mathbb{E}_{P(\cdot|X)} f(\theta_i) \tag{3}$$

as $n \to \infty$.

Generating posterior samples θ_i is again complicated by the intractable denominator in eq. (2), as the inability to compute probability densities / cumulative distribution functions prevents the usage of common sampling methods such as rejection sampling / inverse transform sampling. Instead, researchers have traditionally used Monte Carlo Markov Chain (MCMC) [Metropolis and Ulam, 1949] which avoids computation of the denominator.

While MCMC has enabled significant progress, it crucially requires a computation of an acceptance probability

$$\alpha_\theta = \frac{P(\theta' \mid X)}{P(\theta \mid X)} \tag{4}$$

where the entire dataset X is utilized. As data sets grow larger in this era of big data, this global computation required for each iteration of the algorithm becomes a bottleneck and limits the applicability of MCMC.

A recently developed method which avoids this problem is Stochastic Gradient Langevin Dynamics (SGLD) [Welling and Teh, 2011], which is capable of posterior sampling while leveraging only a subset of the data at each iteration. This property makes SGLD a prime candidate for solving the scalability problem in Bayesian statistics, resulting in a large amount of recent researcher interest and development of extensions such as distributed implementations (D-SGLD) [Ahn et al., 2014] and samplers where θ is restricted to the probability simplex (Riemannian Langevin dynamics, SGRLD) [Patterson and Teh, 2013].

2 Project Goals and Deliverables

We develop an implementation of D-SGLD which leverages MPI for coordination of multiple processors. We also investigate the following extensions and modifications:

- Trajectory sampling, to solve problems resulting from short communication cycles by trading off communication overhead with chain mixing rates
- Load balancing using trajectory lengths, to mitigate underutilization resulting from situations where different processors operate at different speeds or where data is not equally partitioned
- Riemannian Langevin dynamics [Patterson and Teh, 2013], to enable sampling of parameters θ constrained to a probability simplex ($0 \leq \theta_i \leq 1$ for all i, $\sum_i \theta_i = 1$) which is required for models such as latent Dirichlet allocation (LDA) [Blei et al., 2003]

In contrast to Ahn et al. [2014], where the implementation targeted a cloud environment (provided by AWS), our implementation uses OpenMPI and targets a HPC computing cluster (provided by NERSC). Compared to a HPC cluster, a cloud environment has higher communication costs (requiring network I/O) but greater scalability (can easily provision a large number of EC2 instances). As a result, we expect the effects of the above modifications to yield different tradeoffs than those reported in Ahn et al. [2014].

3 Our Method

We follow the D-SGLD algorithm described by Ahn et al. [2014]. This is a data-parallel algorithm which first partitions data into disjoint shards $(X_s)_{s \in S}$ distributed across workers and then runs S parallel chains (one on each worker) using the update

$$\theta_{t+1} = \theta_t \frac{\epsilon_t}{2} \left(\nabla \log P(\theta) + N g(\theta_t, X_s) \right) + \nu_t \tag{5}$$

where $\nu_t \overset{\text{iid}}{\sim} N(0, \epsilon_t)$, $\epsilon_t \to 0$ sufficiently slowly, and $g(\theta_t, X_s)$ is an unbiased estimator of $\nabla \log P(\theta \mid X)$. Notice that the unbiased condition

$$\mathbb{E} g(\theta_t, X_s) = \nabla \log P(\theta \mid X) = \nabla \log P(\theta \mid \cup_{s \in S} X_s) \tag{6}$$

requires chains to be exchanged between different workers $s \in S$, resulting in communication.

4 Parallel chains and chain exchange

In this section and the following two, we use a two-component Gaussian mixture model (GMM) over $\mathbb{R}$ with priors $\theta_1 \sim \mathcal{N}(0, 10)$, $\theta_2 \sim \mathcal{N}(0, 1)$, and likelihood $x_i \overset{\text{iid}}{\sim} \frac{1}{2}\mathcal{N}(\theta_1, 2) + \frac{1}{2}\mathcal{N}(\theta_1 + \theta_2, 2)$. The posterior has two modes at $\theta = (0, 1)$ and $\theta = (-1, 1)$ and is negatively correlated, allowing easy verification of expected behavior.

As discussed in section 3, correctness of the algorithm requires the S parallel chains to be exchanged across workers. One possible implementation of $g(\theta_t, X_s)$ eq. (6) is to randomly sample a permutation $\sigma \in S^{|}S|$ and send the chain on worker s to $\sigma(s)$ after every iteration. Figure 1 illustrates the necessity of this exchange. Without it, each of the chains only samples from the posterior $\mathbb{P}(\theta \mid X_s)$ induced by the data local to the worker it is assigned to rather than the global posterior $\mathbb{P}(\theta \mid \cup_s X_s)$.

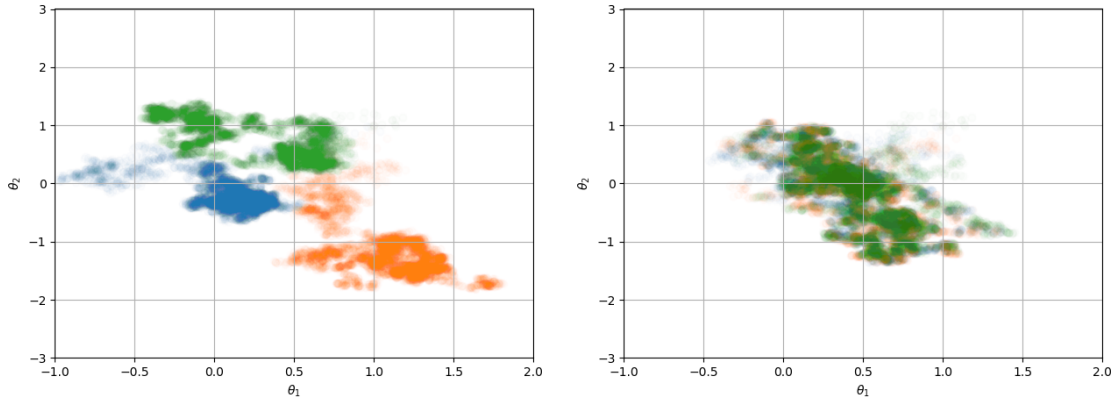

Figure 1: Samples drawn from local posteriors $P(\theta \mid X_s)$ (left, e.g. due to lack of chain exchange) may not well approximate the global posterior $P(\theta \mid \cup_s X_s)$ (right). Different colors corresponds to different chains.

As we increase the number of workers S, we expect the rate at which samples are collected to increase. Indeed, fig. 2 confirms that it takes less time to generate the same number of samples when we have more workers. However, the scaling is not very good as fig. 2 shows that the time to generate 800000 samples is only cut down by $\sim 1/3$ when three workers are added (perfect scaling would expect this to be a $2/3$ reduction). We can attribute this poor scaling to communication overhead, which we will address in the following section.

5 Trajectory sampling

As illustrated in fig. 2, exchanging chains after each iteration incurs significant communication overhead and impacts the algorithms scaling behavior. To circumvent this, SGLD can be modified to perform trajectory sampling. Instead of drawing a single sample, a trajectory of length τ is sampled on each worker prior to exhanging chains. This can reduce the amount of inter-worker communication by a factor of $1/\tau$, but it must be used with care as too little chain exchange results

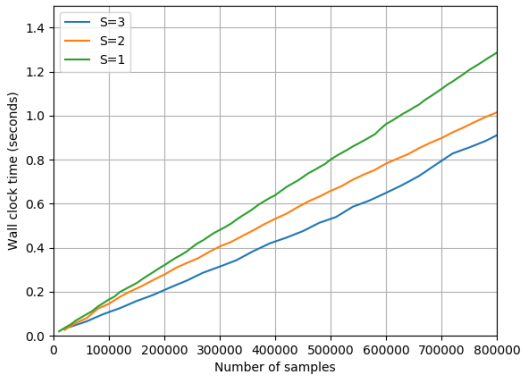

Figure 2: Speedup in sampling due to parallelism.

in biased samples (setting τ equal to the number of samples is equivalent to never exchanging chains).

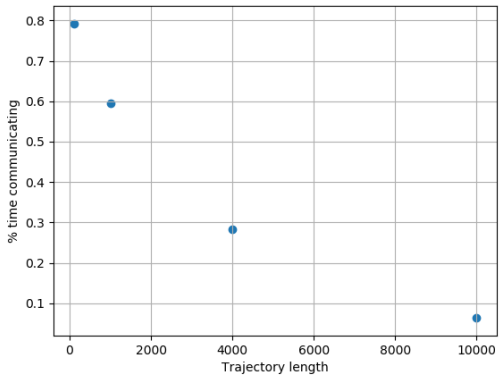

Figure 3: Increasing trajectory length reduces how often communication happens, resulting in less communication overhead.

Figure 3 shows the fraction of time spent communication during an execution of the algorithm. As expected, we see that longer trajectory lengths reduces this quantity and leads to a greater proportion of time spent generating samples.

Although we do not explore it in this work, one interesting direction for future work would be to explore the tradeoff between mixing rate and trajectory length in more detail. For example, one could try to optimize τ by choosing it such that it generates the largest effective sample size in a fixed amount of time.

6 Trajectory length load balancing

Workers may not all finish sampling trajectories at the same times. For example, workers may have different hardware and data partitioning may be imbalanced. As a result, faster workers must block on slower workers on every chain exchange, leading to low utilization (see fig. 4).

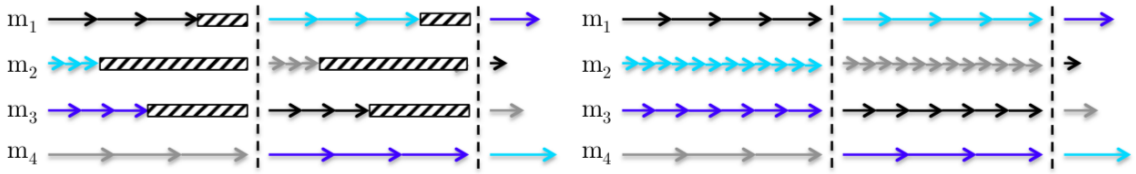

Figure 4: Figure from Ahn et al. [2014] illustrating trajectory length load balancing. The fast worker m_2 must sit idle waiting for the slow worker m_4 without load balancing (left), whereas after (right) it utilizes this time to generate a longer tarjectory.

To improve utilization, we can modify the global trajectory length τ into a worker-specific trajectory lengths $(\tau_s)_{s \in S}$ and use τ_s to balance work. Faster workers can be assigned a larger τ_s, allowing them to continue generating samples while waiting. We set these worker-specific trajectory lengths following Ahn et al. [2014] by reporting the time spent within the trajectory sampling loop to the master and adjusting τ_s in an inversely proportional manner. The results of this modification are shown in fig. 5, where we simulated imbalanced data partitioning by assigning 95% of the data to worker 1. The reduction in latencies after the first iteration (where initial timing results are gathered) confirm that trajectory load balancing results in improved utilization.

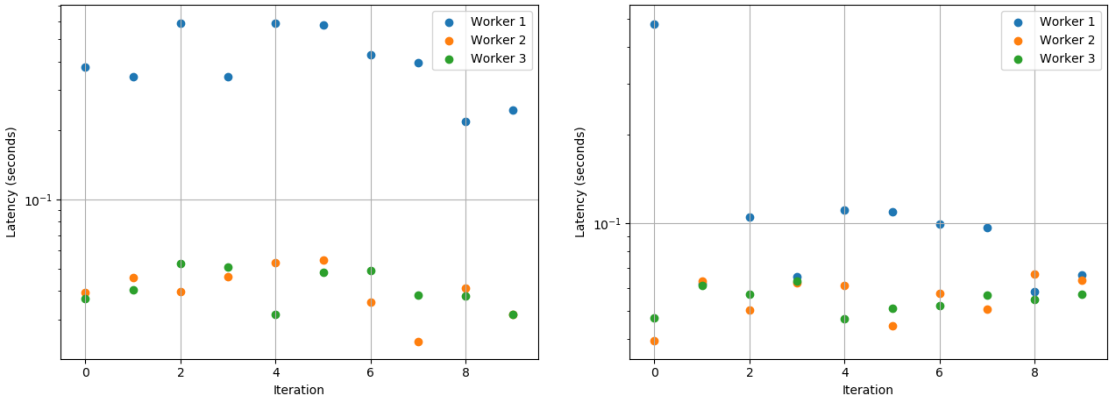

Figure 5: Before (top) and after (bottom) trajectory length load balancing. Experiments used the GMM with an imbalanced data partitioning which assigned 95% to worker 1. The lower variance in latency after the first iteration (where initial timing results are gathered) confirm that trajectory length load balancing results in improved utilization.

7 Latent Dirichlet Allocation with Riemannian Langevin Dynamics

So far, we have only explored a toy GMM. One real world application of posterior sampling is inference in LDA [Blei et al., 2003], a probabilistic topic model used to discover structure in documents. As "topics" in LDA represent discrete distributions over words, they are constrained to lie on the probability simplex. Accordingly, we leverage stochastic gradient Riemannian Langevin dynamics [Patterson and Teh, 2013] to sample the probability simplex and adapt it to run in a distributed MPI environment over partitioned data. The result of training a LDA model using our implementation is shown in fig. 6, where we see that the parameters sampled are decreasing in perplexity. This indicates that the sampler is being drawn towards a posterior mode, and suggests that the implementation is behaving as expected.

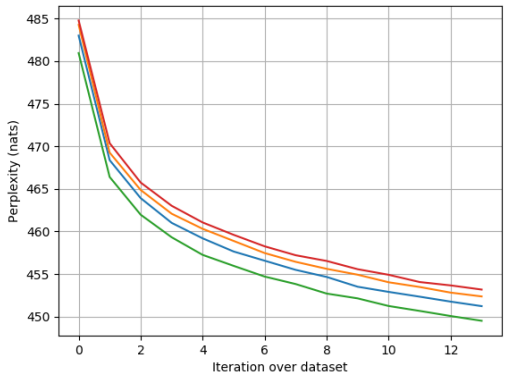

Figure 6: Distributed SGRLD training on NIPS corpus (1740 documents, 19889 unique words) of a 10 topic LDA model. Each color corresponds to a different chain.

8 Conclusion

We provide an implementation of D-SGLD on top of MPI and investigate the performance characteristics of various optimizations. Our results demonstrate that D-SGLD enjoys significant speedups due to parallelism, can sample posteriors even when data is partitioned disjointedly across workers, and can be controlled using trajectory lengths to trade off between communication versus mixing time as well as load balance workers.

References

Sungjin Ahn, Babak Shahbaba, and Max Welling. Distributed stochastic gradient mcmc. In *International conference on machine learning*, pages 1044–1052, 2014.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Sam Patterson and Yee Whye Teh. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems*, pages 3102–3110, 2013.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.