Bayesian Posterior Sampling via Mini-batch MCMC

Yang Gao
gaoyang10@mails.tsinghua.edu.cn
Department of Computer Science, Tsinghua University, Beijing, China

1. Preliminaries

We start to formalize the Bayesian posterior parameter sampling problem and introduce the Stochastic Gradient Descent and Metropolis-Hastings sampling method which are needed to motivate our "mini-batch MH" algorithm.

We have a large dataset X_N consisted of N i.i.d. data points $\{x_1, x_2, \cdots, x_N\}$, which is modeled as samples from a parametric model family M. M is parameterized by a vector $\theta \in R^D$. We choose a prior distribution $p(\theta)$ over θ and aim to draw samples from the parameter's posterior distribution $p_N(\theta) \equiv p(\theta|X_N) \propto p(\theta)p(X_N|\theta) = p(\theta)\prod_{i=1}^{N} p(x_i|\theta)$.

One famous class of methods to draw samples from a high dimensional distribution is Markov Chain Monte Carlo (MCMC) and Metropolis-Hastings (Hastings, 1970) is one of those methods. MH algorithm maintains a record of the current state θ^τ, where τ is a positive integer index, and generates the next state $\theta^{\tau+1}$ from a proposal distribution $q(\theta|\theta^\tau)$, which is accepted with a probability of $r \equiv \min(1, \frac{p_N(\theta)q(\theta|\theta^\tau)}{p_N(\theta^\tau)q(\theta^\tau|\theta)})$. And if it's rejected, the next state remains to be θ^τ. The distribution of θ^τ will approach $p_N(\theta)$ as $\tau \to \infty$. However, we can set τ to some adequate value to get a sample good enough for any practical application. For a machine learning problem with very large training dataset, the most time consuming step is the evaluation of $p_N(\theta)$ and $p_N(\theta^\tau)$, which involves processing through the whole dataset.

To alleviate the huge computational cost of $p_N(\theta)$, we bring in the mini-batch idea from SGD. In the optimization field, people are always interested in finding the maximum of the posterior $p_N(\theta)$ (MAP). A popular method called Stochastic Gradient Descent (SGD, Robbins & Monro, 1951) update parameter θ^τ as follows:

$$\Delta\theta^\tau = \frac{\epsilon^\tau}{2}(\nabla \log p(\theta^\tau) + \frac{N}{n}\sum_{i=1}^{n} \nabla log p(x_i^\tau|\theta^\tau))$$

where ϵ^τ is a sequence of step size, and $X^\tau = \{x_1^\tau, \cdots, x_n^\tau\}$ is a random subset of X_N with size n. The idea is to approximate the true gradient over the whole dataset with the gradient of a random subset. The noise introduced by the random subset could be averaged out across multiple iterations. The SGD algorithm can saves lots of computational effort on large dataset while it's still accurate enough. Thus we can use the same idea to approximate $p_N(\theta)$, which leads to "mini-batch MCMC" described in the next section.

2. Mini-batch MCMC

A natural combination of SGD and Metropolis-Hastings is to replace the computation of

$$p_N(\theta) = p(\theta)\prod_{i=1}^{N} p(x_i|\theta)$$

with

$$\mathrm{p_n}(\theta) = \mathrm{p}(\theta)\left(\prod_{i=1}^{n} p(x_i^\tau|\theta)\right)^{\frac{N}{n}}$$

where $\mathrm{X}^\tau = \{x_1^\tau, \cdots, x_n^\tau\}$ is a random subset of $\mathrm{X_N}$. We use the same subset X^τ to calculate $\mathrm{p_n}(\theta)$ and $\mathrm{p_n}(\theta^\tau)$, but different X^τ in each MH state change.

TODO(theoretical justification is still absent)

3. Choosing mini-batch size

TODO(show how to choose an adequate mini-batch size in a practical problem)

4. Experiments

4.1 Gaussian Mixtures

We first demonstrate our mini-batch MCMC method on a synthetic dataset and analyze the relation of the samples' standard deviation and autocorrelation with size of the mini-batch. We use the same example as Welling and Teh(2011). The class of model generating the training data is a Gaussian mixture of tied means, which involves only two parameters θ_1 and θ_2:

$$\theta_1 \sim N(0, \sigma_1^2); \theta_2 \sim N(0, \sigma_2^2)$$

$$\mathrm{x_i} \sim \frac{1}{2} N(\theta_1, \sigma_x^2) + \frac{1}{2} N(\theta_1 + \theta_2, \sigma_x^2)$$

where $\sigma_1^2 = 10, \sigma_2^2 = 1$ and $\sigma_x^2 = 2$. We select the model with $\theta_1 = 0$ and $\theta_2 = 1$ and try to sample from the posterior distribution of $\theta \equiv (\theta_1, \theta_2)$ based on a bunch of data points $\mathrm{X_N} = \{x_1, \cdots, x_N\}$.

A Visual Appearance

To get comparable results in each experiment group, we plot 1000 samples for every setting, and set the sample interval to 200 MH or mini-batch MCMC moves, which is shown to be enough in the following text. We also use the same θ^τ mean, $\sigma \mathrm{I}_2$ variance Gaussian as the proposal distribution for both methods, and adjust σ such that the final accept ratio r is about 0.4, which is required for the samples to have no bias. Initial values for θ are all set to $(0,0)$, and the first 1000 burn in moves are discarded.

This posterior distribution is interesting. Because, for small N, it has two modes at (0,1) and (1,-1) with strong negative correlation between the two parameters, but it will collapse into one peak around (0,1) when N become bigger. So it's challenging for mini-batch MCMC to get a single mode distribution for big N even if it use a rather small mini-batch size. See visual comparison between the samples generated by MH and our mini-batch MCMC method for N=2000 and N=10,000 in Figure 1. The mini-batch size is set to N/5.

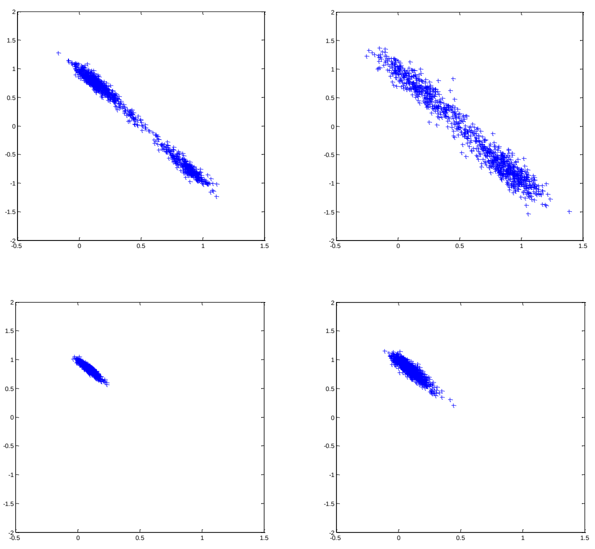

Figure 1. Samples generated by MH and mini-batch MCMC, where the x-axis denotes θ_1 and the y-axis stands for θ_2. The upper graphs are sampled from posterior of N=2000 training data, while the lower two have N=10,000 training data. The left column uses standard MH method, and the right column uses mini-batch MCMC. The mini-batch size is N/5.

The samples of mini-batch MCMC have bigger standard deviation than the standard MH method, but the basic shapes of the distributions are identical. The upper left and the lower right figure used the same computational resources, but powered by the mini-batch MCMC algorithm and more data, we're able to discover a single mode, which is close to the underlying parameter value, instead of two modes. The mini-batch MCMC is not a magic algorithm that preserves same sample accuracy with significantly less computational resource, rather, it attempts to take advantages of large data set and takes the right approximation to make Bayesian posterior sampling possible with comparable accuracy.

Order of STD Raise with Batch Size Reduce

To understand how the samples' standard deviation increases with the decrease of mini-batch size, we did a set of experiments. Recall that standard MH samples from the true posterior, which means that when the training data is shrunk to 1/k of original training data, the STD will be $\sqrt{k}$ times of the original STD. We want to discover similar relation for mini-batch MCMC, that is, if the mini-batch size is shrunk to 1/k of the original size, what the ratio of sample STD to original STD will be. We use a fix training dataset, and a set of mini-batch sizes. For each of the mini-batch size, we run the experiment 10 times and calculate each run's sample STD. A box plot is shown for 10 STDs from the same batch size (Figure 2).

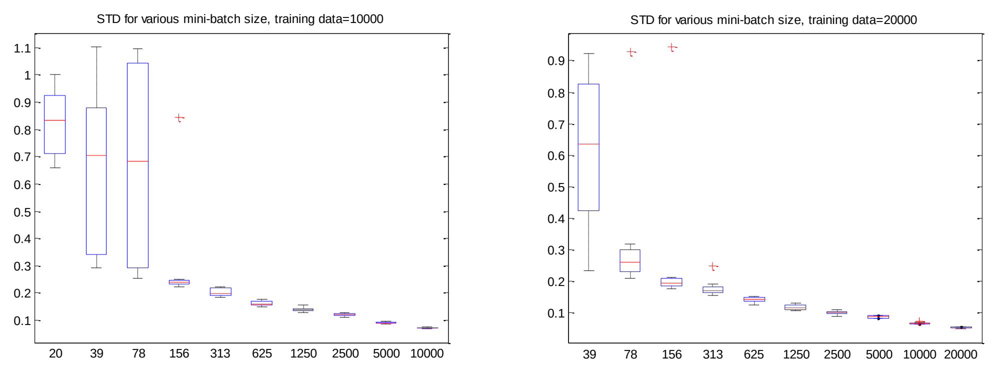

Figure 2. Standard deviation for various mini-batch sizes. X-axis is different mini-batch sizes and Y-axis is the STD for the samples. For each mini-batch size, sample STDs of 10 independent runs are plotted in a box plot. The red line indicates median of 10 STDs, the upper and lower blue lines are two quartiles, two black lines are the min and max of STDs, and red plus signs are outliers. The Left graph use a training data of 10,000, while the right one uses 20,000.

When the mini-batch size is less or equal to 78, the samples will have two modes, which makes their comparison to STD of bigger mini-batch size insignificant. After removing data of the first few mini-batch sizes, we can clearly see that the remaining STDs are decreasing in a uniform manner. If we draw a similar curve for the standard MH algorithm, where the x-axis should be changed to training set size, the curve would be proportional to $\mathrm{N}^{-0.5}$, as we have mentioned above. So we have strong incentive to assume that this curve is proportional to $\mathrm{N}^{-\mathrm{k}}$, where k is to be determined. After applying logarithm to the median of each group of 10 STDs, and use linear regression to fit them, we have found that for the left plot k=0.28 with $\mathrm{R}^2 = 0.991$, and for the right one, k=0.264 with $\mathrm{R}^2 = 0.989$. Thus, we conjecture that k will approach 0.25 as $\mathrm{N} \to \infty$. Further theoretical understanding is desirable.

If our conjecture actually holds, then the samples generated from mini-batch MCMC will have STD of $\alpha\frac{\mathrm{p}^{1/4}}{N^{1/2}}$, where α is a constant only depending on the model, N is the training data size, p is the proportion of mini-batch size to N. This result has very important practical application. If we have a very large dataset, where exact posterior sampling is not possible, we can use a $\mathrm{p} = 5^4 = 625$ mini-batch MCMC algorithm, which will enjoy about 625 times faster in speed, and get approximate samples with about 5 times STD of the true posterior.

Autocorrelation Analysis

Mini-batch MCMC generated a slightly more correlated sequence than standard MH method. For this Gaussian mixture model, we will show that the correlation time will be at most 2 times more, if mini-batch size is chosen appropriately.

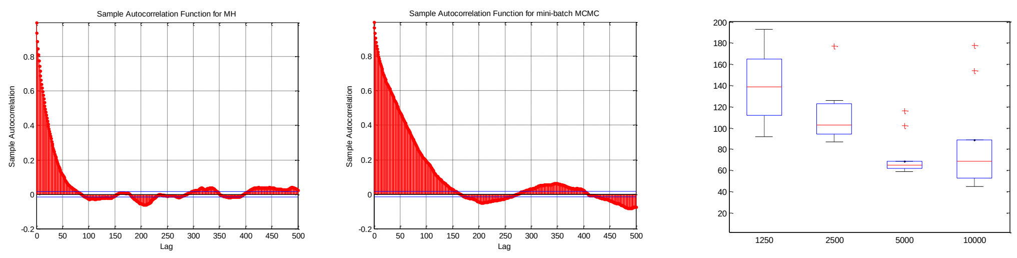

Figure 3. Sample autocorrelation for two MH (left, N=10000) and mini-batch MCMC (middle, N=10000, batch=1000). X-axis is lags for autocorrelation and Y-axis is the correlation coefficient. The right one is box plot for lags needed to make samples not correlated (y-axis) of different mini-batch size (x-axis), and the training set is fixed to N=10000. Only parameter θ_1's autocorrelation is calculated, because the other one's plot is similar.

The two methods' plot of autocorrelation versus lags is shown in Figure 3. We can see that original MH method takes about 100 lags to burn in, while an N/10 mini-batch MCMC takes 150 lags to burn in. Comparing to the 10 times computational saving for each lag, the 50% more lags is not too much. The right-most plot in Figure 3 shows for big range of mini-batch sizes, how the uncorrelated time will change. We can conclude that for this Gaussian mixture model, the correlation time would not be much more than the standard MH method. Furthermore, the situation of the phase after burn in is similar, i.e. mini-batch MCMC will take a few more steps to mix. (Mini-batch sizes less than 1000 are not shown, due to the fact that if the mini-batch is too small, the samples no longer have a single peak, and the comparison between those two conditions makes no sense.)