CPSG-MCMC: Clustering-Based Preprocessing method for Stochastic Gradient MCMC

Tianfan Fu
Shanghai Jiao Tong University

Zhihua Zhang
Peking University & Beijing Institute of Big Data Research

Abstract

In recent years, stochastic gradient Markov Chain Monte Carlo (SG-MCMC) methods have been raised to process large-scale dataset by iterative learning from small minibatches. However, the high variance caused by naive subsampling usually slows down the convergence to the desired posterior distribution. In this paper, we propose an effective subsampling strategy to reduce the variance based on a failed attempt to do importance sampling. In particular, before sampling, we partition the dataset with k-means clustering algorithm in a preprocessing step and use the fixed clustering throughout the entire MCMC simulation. Then during simulation, we approximate the gradient of log-posterior via summing the estimated gradient of each cluster. The resulting procedure is surprisingly simple without enhancing the complexity of the original algorithm during the sampling procedure. We apply our Clustering-based Preprocessing strategy on stochastic gradient Langevin dynamics, stochastic gradient Hamilton Monte Carlo and stochastic gradient Riemann Langevin dynamics. Empirically, we provide thorough numerical results to back up the effectiveness and efficiency of our approach.

1 Introduction

In recent years, scalable inference methods have been increasingly popular in the context of Bayesian learning. However, each iteration of typical Markov Chain Monte Carlo (MCMC) algorithms requires computations over the whole dataset in question. The computational complexity is prohibitively large in big data era. To address this issue, Stochastic Gradient MCMC (SG-MCMC) was first proposed in Welling and Teh [2011], based on Langevin dynamics [Rossky et al., 1978, Roberts and Stramer, 2002]. The general idea is to randomly choose a minibatch of data samples from the large-scale dataset, and then estimate the true gradient of the negative log-posterior from the minibatch instead of the entire data samples. In addition, Metropolis-Hastings (MH) correction step is thrown away. Subsequently, a great number of SG-MCMC methods were raised [Ahn et al., 2012, Patterson and Teh, 2013, Chen et al., 2014, Ding et al., 2014, Ahn et al., 2014, Teh et al., 2014, Chen et al., 2015a, Yi-An Ma, 2015, Chen et al., 2015b, Li et al., 2016a,b,c, Simsekli et al., 2016, Liu et al., 2016, Chen et al., 2016, Durmus et al., 2016]. This framework is popular thanks to its scalability in processing large scale datasets. However, most of these existing work in the framework stick to naive subsampling (also known as uniform sampling). The corresponding gradient estimator exhibits high variance, leading poor mixing rate.

In this paper, we focus on reducing variance of the stochastic gradient estimator, an orthogonal direction compared with prior works that insist on naive subsampling. As a first cut, we might resort to typical importance sampling strategy. Unfortunately, it is shown to be infeasible in large scale setting. We instead devise a novel scheme that partitions the dataset with k-means algorithm in a preprocessing step and use the fixed clustering throughout the entire MCMC simulation. During MCMC simulation, each time approximating the gradient of the negative log-posterior, we estimate the gradient of each cluster independently and sum them up. It is worth mentioning that the overhead of this scheme is negligible, validated empirically. We call it Clustering-based Preprocessing (CP) strategy. The empirical evaluation demonstrates both effectiveness and efficiency of the idea. The remainder of the paper is organized as follows: Section 2 briefly introduces stochastic gradient MCMC sampler. In Section 3, CP-SG-MCMC is derived based on a failed attempt to do importance sampling. Section 4 demonstrates the empirical results and Section 5 is devoted to the conclusion. Proof of theoretical results, additional empirical results and some backgrounds are provided in the supplementary materials.

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

2 Preliminaries

Suppose that N independent observations $x = \{x_1, \ldots, x_N\}$ are given, let $\theta \in \mathbb{R}^d$ be the parameter vector of interest and $p(\theta)$ be the prior distribution. Likelihood function is denoted as $p(x|\theta)$. The posterior distribution $p(\theta|x)$ satisfies the Bayes rule as

$$p(\theta|x) = \frac{p(\theta)p(x|\theta)}{p(x)} \propto p(\theta)p(x|\theta) = p(\theta)\prod_{i=1}^{N} p(x_i|\theta), \tag{1}$$

where $p(x) = \int_\theta p(\theta)p(x|\theta)d\theta$, independent of θ, is called normalizing constant. It is well-known that MCMC has been the main workhorse of Bayesian computation since 1990s. Recently, among various MCMC samplers, Langevin and Hamiltonian Monte Carlo [Roberts and Stramer, 2002, Neal, 2011], also known as diffusion based sampling methods, have become increasingly popular for efficient exploration of the state space. They are based on Itô diffusions as

$$dz_t = F(z_t)dt + \sigma(z_t)dW_t, \tag{2}$$

where $z \in \mathbb{R}^n$ denotes model states, t the time index, W_t represents Brownian motion, function $F : \mathbb{R}^n \to \mathbb{R}^n$ and $\sigma : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ (m not necessarily equal to n) are assumed to satisfy the usual Lipschitz continuity condition. In Bayesian inference, appropriate function F and σ are devised so that the marginal distribution of $\rho(z)$ (the stationary distribution the Itô diffusions) is equal to the posterior distribution we are interested in. For instance, in 1st-order Langevin dynamics (LD), $z = \theta$, $F = -\nabla_\theta U$ and $\sigma = \sqrt{2}I_d$, with I_d denoting the $d \times d$ identity matrix. 2nd-order Langevin dynamics correspond to $z = (\theta, p)$, $F = \begin{pmatrix} p \\ -Dp - \nabla_\theta U \end{pmatrix}$ and $\sigma = \sqrt{2D}\begin{pmatrix} 0 & 0 \\ 0 & I_d \end{pmatrix}$ for some $D > 0$. Here U, referred to as the potential energy, is the unnormalized negative log-posterior satisfying that $U(\theta) = -\log p(\theta|x) = -\log p(\theta) - \sum_{i=1}^{N} \log p(x_i|\theta) + C$, where $C = \log p(x)$ is a constant. $p \in \mathbb{R}^d$ is known as the momentum variable [Neal, 2011]. For ease of exposition, we take Metropolis-adjusted Langevin algorithm (MALA) [Rossky et al., 1978, Roberts and Stramer, 2002] as an example, which is based on Langevin dynamics. In MALA, the update of parameter can be described as

$$\begin{aligned}\theta_t =& \theta_{t-1} + \frac{\epsilon}{2}\Big[\nabla \log p(\theta_{t-1}) + \sum_{i=1}^{N} \nabla \log p(x_i|\theta_{t-1})\Big] \\ &+ \eta_t, \quad \text{where } \eta_t \sim \mathcal{N}(0, \epsilon I_d),\end{aligned} \tag{3}$$

along with an MH correction. Here ϵ represents the step-size, and we write $\nabla \log p(x|\theta) \triangleq \nabla_\theta \log p(x|\theta)$ for notational convenience. In addition, each sampling requires traversing the entire dataset due to the computation of both the gradient of log-posterior and acceptance probability in MH correction step. Hence it is computationally expensive in the context of large-scale inference. To tackle this problem, stochastic gradient Langevin dynamics (SGLD) was raised [Welling and Teh, 2011]. SGLD simply approximates the true gradient of the negative log-posterior over the whole dataset using the gradient computed on a subset of the entire data and throw away the MH correction step. Then the parameter is updated as

$$\begin{aligned}\theta_t =& \theta_{t-1} + \frac{\epsilon_t}{2}\Big[\nabla \log p(\theta_{t-1}) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{t_i}|\theta_{t-1})\Big] \\ &+ \eta_t, \quad \text{where } \eta_t \sim \mathcal{N}(0, \epsilon_t I_d),\end{aligned} \tag{4}$$

without MH correction. In the t-th iteration, a subset of n ($n \ll N$) data samples $\{x_{t_1}, x_{t_2}, \ldots, x_{t_n}\}$ are randomly drawn from the whole dataset. To ensure the convergence to the desired distribution without an MH correction procedure, the sequence of step-size $\{\epsilon_1, \epsilon_2, \ldots\}$ is required to satisfy the following property

$$\sum_{i=1}^{\infty} \epsilon_t = \infty, \quad \sum_{i=1}^{\infty} \epsilon_t^2 < \infty. \tag{5}$$

It can be proved that over multiple iterations the noise caused by the naive subsampling can be averaged out under certain assumptions [Teh et al., 2014, Chen et al., 2015a, Vollmer et al., 2015]. In recent years, numerous extensions of SGLD have been developed [Ahn et al., 2012, Patterson and Teh, 2013, Chen et al., 2014, Ding et al., 2014, Ahn et al., 2014, Teh et al., 2014, Chen et al., 2015a, Yi-An Ma, 2015, Chen et al., 2015b, Li et al., 2016a,b,c, Simsekli et al., 2016, Liu et al., 2016, Chen et al., 2016, Durmus et al., 2016], which are coined under the term Stochastic Gradient MCMC (SG-MCMC). However, they are based on naive subsampling, thus suffer from poor mixing rates when the variance of the gradient estimator is high. To handle the issue, we focus on reducing this variance.

3 Methodology

In naive subsampling, each data sample is treated equally and assigned the same sampling probability. First, we consider to employ importance sampling to cut down the variance. Unfortunately, we find that importance sampling strategy does not work here owing to the computational cost. To alleviate the computational burden of importance sampling, we are inspired by stratified sampling [Liu, 2008, Zhao and Zhang, 2014a] and explore the possibility of marrying it with SG-MCMC. Accordingly, the proposed method is referred to as Clustering-based Preprocessing Stochastic Gradient MCMC (CPSG-MCMC) approach. In particular, clustering precedure is performed to the data samples only once before simulation. During the sampling procedure, it is surprisingly easy to compute the new gradient estimator (simply merging the estimated gradient of each cluster) and would not bring extra overhead. Moreover, it is compati-

ble with almost all the existing stochastic gradient MCMC samplers. We also investigate its convergence properties.

3.1 Importance Sampling

In both Monte Carlo literature [Liu, 2008, Robert and Casella, 2013] and stochastic optimization community [Wang et al., 2013, Zhao and Zhang, 2014b, Loshchilov and Hutter, 2015, Bouchard et al., 2015], variance reduction is a hot topic. Among all the variance reduction techniques [Liu, 2008], importance sampling (IS) is a popular choice. In particular, we restrict our attention to the mini-batch with size 1. It is simple to generalize this case into any mini-batch size n ($n \leq N$). Given the i-th data sample with a sampling probability p_i (the corresponding weight is $\frac{1}{p_i}$), we have $\sum_{i=1}^N p_i = 1$ because the mini-batch size is 1. The estimated noisy gradient of the negative log-posterior can be represented as

$$\nabla \hat{U}(\theta) = -(p_i)^{-1} \nabla \log p(x_i|\theta) - \nabla \log p(\theta), \qquad (6)$$

where i is the index of the selected data sample. Now we demonstrate that this is an unbiased estimator of the true gradient of the negative log-posterior .

Lemma 3.1. *The expectation of estimated gradient $\nabla \hat{U}(\theta)$ governed in Equation* (6) *is equal to $\nabla U(\theta) = -\sum_{i=1}^N \nabla \log p(x_i|\theta) - \nabla \log p(\theta)$, the true gradient of negative log-posterior at θ for any θ. That is, $\mathbb{E}(\nabla \hat{U}(\theta)) = \nabla U(\theta)$.*

This result can be easily generalized into any mini-batch size $n \leq N$. In Equation (6), $\nabla \log p(\theta)$, the gradient of log-prior, is deterministic. Hence the variance reduction (minimization) problem is formulated as

$$\arg\min_{\forall i \in \{1,2,\ldots,N\},\ p_i \in (0,1);\ \sum_{i=1}^N p_i = 1} \mathbb{V}((p_i)^{-1} \nabla \log p(x_i|\theta)), \qquad (7)$$

where $\mathbb{V}(\cdot)$ represents the variance in this paper. The following lemma provides the optimal solution to the variance minimization problem.

Lemma 3.2. *The optimum of Problem* (7) *is*

$$p_i = \frac{\|\nabla \log p(x_i|\theta)\|}{\sum_{j=1}^N \|\nabla \log p(x_j|\theta)\|}, \quad \text{for } i = 1, 2, \ldots, N, \qquad (8)$$

where $\|\cdot\|$ represents the ℓ_2 norm for vector in this paper.

Though the optimum of Problem (7) can be found, the resulting algorithm is not practical because the sampling probabilities $\{p_i\}_{i=1}^N$ depend on θ, which means that in each iteration, we need to calculate $\{p_i\}_{i=1}^N$. From Equation (8), we know that calculating $\{p_i\}_{i=1}^N$ requires traversal over all the data samples and for each data sample x_i the gradient of log-likelihood $\nabla \log p(x_i|\theta)$ needs to be evaluated, which goes back to the original point (the complexity is equal to the Metropolis-adjusted Langevin algorithm described in Equation (3)). Thus, importance sampling is impractical in this scenario and we resort to an approximate solution to the variance reduction problem.

3.2 Clustering-based Preprocessing SG-MCMC

Though causing high variance, uniform sampling (naive sub-sampling) is the most efficient sampling method since it doesn't bring any extra overhead in computation. In contrast, importance sampling strategy, though computationally intensive, can significantly cut down the variance. The philosophy is: there is no free lunch. Naturally, we seek a tradeoff between these two schemes.

Comparing these two strategies more carefully, we observe that when the sampling probability of each sample is equal to $\frac{1}{N}$, the importance sampling reduces to uniform sampling. That is to say, uniform sampling assigns all samples the same weight, while importance sampling assigns each samples a specific weight.

Can we make a compromise here? We give a positive answer. Borrowing the idea from stratified sampling [Liu, 2008, Zhao and Zhang, 2014a], a natural proposal is to assign the same weight to "similar" points. The whole dataset can be divided into weighted groups via certain clustering method. Within a group the samples are drawn randomly. Both uniform and importance sampling can be seen as special cases of this method.

Formally speaking, we separate $1, \ldots, N$ into k clusters $C_1, C_2, \ldots, C_k$ (e.g., $l \in C_i$ means x_l in the i-th cluster C_i). The cardinality of C_i is denoted by n_i. That is, for $\forall\, i, j \in \{1, 2, \ldots, k\}$, we have

$$\begin{aligned} &C_i \subset \{1, 2, \ldots, N\}, \quad \cup_{i=1}^k C_i = \{1, 2, \ldots, N\}, \\ &|C_i| = n_i, \qquad\qquad C_i \cap C_j = \emptyset \ \text{ for } \forall\, i \neq j. \end{aligned} \qquad (9)$$

Given these k clusters, we will independently sample k subsets $B_1, \ldots, B_k$ with fixed sizes from $C_1, \ldots, C_k$, respectively. We denote the cardinality of B_i as b_i, which can be seen as the weight for the cluster C_i. The noisy gradient can be represented as

$$\nabla \bar{U}(\theta) = -\sum_{i=1}^k \frac{n_i}{b_i} \sum_{\substack{j \in B_i, \\ |B_i| = b_i}} \nabla \log p(x_j|\theta) - \nabla \log p(\theta). \qquad (10)$$

Now we show that this is an unbiased estimator for the true gradient $\nabla U(\theta)$ for any $C_1, \ldots, C_k$ satisfying the condition described in Equation (9).

Lemma 3.3. *The stochastic gradient described in Equation* (10) *is equal to the true gradient of negative log-posterior $\nabla U(\theta)$ in expectation for any θ. That is, $\mathbb{E}[\nabla \bar{U}(\theta)] = \nabla U(\theta)$.*

Now we want to minimize the variance of $\nabla\bar{U}(\theta)$ via choosing proper clusters $C_1,\ldots,C_k$ and the corresponding weights $b_1,\ldots,b_k$, which is reduced to the following optimization problem:

$$\arg\min_{C_1,\ldots,C_k;b_1,\ldots,b_k} \mathbb{V}(\nabla\bar{U}(\theta)|C_1,C_2,\ldots,C_k,b_1,b_2,\ldots,b_k), \tag{11}$$

where $C_1,\ldots,C_k$ are defined in Equation (9), $b_1,\ldots,b_k \in \mathbb{N}_+$, satisfy that $\sum_{i=1}^k b_i = b$. The mini-batch size b and the number of clusters k are prespecified.

Lemma 3.4. *The variance of estimated gradient at θ, denoted $\mathbb{V}(\nabla\bar{U}(\theta)|C_1,C_2,\ldots,C_k,b_1,b_2,\ldots,b_k)$, can be further simplified to*

$$\begin{aligned}&\sum_{i=1}^k \frac{n_i}{b_i}\sum_{j\in C_i}\mathbb{V}[(\nabla\log p(x_j|\theta))]\\ =&\sum_{i=1}^k \frac{n_i}{b_i}\sum_{\substack{j\in C_i\\|C_i|=n_i}}\|\nabla\log p(x_j|\theta)-\frac{1}{n_i}\sum_{l\in C_i}\nabla\log p(x_l|\theta)\|^2.\end{aligned}$$

The simplified formulation is similar to a dynamically weighted k-means problem, where the weights of gradients in a cluster are the same and optimized with the clusters simultaneously. However, this clustering method is still based on the calculation of the gradient over the whole dataset and still θ-dependent. That means in each iteration, clustering procedure is required to be performed, which is computationally costly. To tackle this issue, we attempt to reformulate this problem. First, we state the following assumption.

Assumption 3.1. *The gradient of log-likelihood function, $\nabla\log p(x|\theta)$, is L-Lipschitz in x for fixed θ, where $L>0$. That is, if for all θ and all $x_1,x_2\in\mathbb{R}^d$, we have*

$$\|\nabla\log p(x_1|\theta)-\nabla\log p(x_2|\theta)\|\le L\|x_1-x_2\|,$$

where $\|\cdot\|$ represents the ℓ_2 norm for vector.

For canonical sampling, it is a mild assumption. Under this assumption, we try to obtain a surrogate function, which is an upper bound to the objective function in Problem (11) and easy to minimize.

Theorem 3.1. *Suppose Assumption 3.1 holds. Then the variance of estimated gradient can be upper-bounded as*

$$\begin{aligned}&\mathbb{V}(\nabla\bar{U}(\theta)|C_1,C_2,\ldots,C_k,b_1,b_2,\ldots,b_k)\\ =&\sum_{i=1}^k\frac{n_i}{b_i}\sum_{j\in C_i,|C_i|=n_i}\mathbb{V}(\nabla\log p(x_j|\theta))\\ \le& L^2\sum_{i=1}^k\frac{n_i}{b_i}\sum_{j\in C_i,|C_i|=n_i}\|x_j-\mu_i\|^2,\end{aligned}\tag{12}$$

where $\mu_i=\sum_{j\in C_i,|C_i|=n_i}x_j/n_i$ represents the center of the cluster C_i, the first equality follows from the conclusion of Lemma 3.4.

Strikingly, the resulting upper bound in RHS of Equation (12) is θ-independent, which allows us to solve it once and for all. Therefore, instead of solving Problem (11) directly, we take its upper bound as a surrogation. By minimizing the surrogation, we have an approximate solution to Problem (11). The minimization of the surrogate derived in Equation (12) lead to the following optimization problem:

$$\arg\min_{C_1,C_2,\ldots,C_k,b_1,b_2,\ldots,b_k}\sum_{i=1}^k\frac{n_i}{b_i}\sum_{j\in C_i,|C_i|=n_i}\|x_j-\mu_i\|^2, \tag{13}$$

where μ_i is defined in Theorem 3.1. The clusters $C_1,\ldots,C_k$ are obtained by standard k-means algorithm. Then the weights $b_1,\ldots,b_k$ can be obtained as follows.

Theorem 3.2. *For optimization problem described in Equation* (13)*, given the clusters $C_1,\ldots,C_k$, the solutions to the weights $b_1,\ldots,b_k$ are*

$$b_i=\frac{bn_i\sqrt{v_i}}{\sum_{j=1}^k n_j\sqrt{v_j}}, \tag{14}$$

where $v_i=\frac{1}{n_i}\sum_{j\in C_i}\|x_j-\frac{1}{n_i}\sum_{l\in C_i}x_l\|^2=\mathbb{V}(C_i)$. b is the size of the mini-batch satisfying that $b=\sum_{i=1}^k b_i$. Note that b_i represents the number of samples drawn from the cluster C_i and is usually fine-tuned to integer in practice. Both b and k are prespecified.

To conclude, we take the upper bound of the original variance (Problem (11)) as a surrogate. Then based on the minimization of the surrogate, we have an approximate solution to it. The detailed algorithm is presented in Algorithm 1. Clustering-based preprocessing step is performed once before MCMC simulation. Then during iterations, each time estimating gradient of the negative log-posterior, we approximate the gradient for each cluster in parallel (independently) and simply sum them up, as described in Equation (10). SGLD is taken as an instance here and CP strategy can be easily applied to all the stochastic gradient MCMC algorithms, say, stochastic gradient Hamiltonian Monte Carlo (SGHMC) [Chen et al., 2014], stochastic gradient Nose-Hoover thermostat (SGNHT) [Ding et al., 2014]. Furthermore, we only consider the fixed stepsize in this paper rather than annealed stepsize described in Equation (5), in both theoretical analysis and empirical evaluation.

3.3 K-means Algorithm and its scalability

Now we discuss more about k-means algorithm. Given a set of observations $x_1,\ldots,x_N$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the N observations into k ($\le N$) sets $C=\{C_1,\ldots,C_k\}$ so as to minimize the Within-Cluster Sum of Squares (WCSS), i.e., sum of distance functions of each point in the cluster

Algorithm 1 CP-SGLD

Input: Minibatch size b, Cluster number k, sequence of step size $\{\epsilon_t\}$, number of burn-in period p, initial value of parameter θ_0.

Output: $\theta_{p+1}, \theta_{p+2}, \ldots,$

```
Employ k-means algorithm to clustering the data samples into C_1, C_2, ..., C_k, and compute corresponding weights b_1, ..., b_k via Equation (14).
for t = 1, 2, ... do
    B_j = φ for all j ∈ {1, ..., k}.
    for j = 1 : k do
        for r = 1 : b_j do
            Draw i_r ∈ C_j without repetition, add into B_j.
        end for
        Compute the gradient for cluster j.
    end for
    Estimate ∇Ū(θ_{t-1}) using Equation (10).
    Draw the t-th sample using θ_t = θ_{t-1} − (ε_t/2)∇Ū(θ_{t-1}) + η_t, where η_t ~ N(0, ε_t I_d).
end for
```

to the k center. Planar k-means is proved to be an NP-hard problem [Mahajan et al., 2009]. Additionally, the algorithm scales badly in dimension of features and data number. To address these issues, in recent years, many variants have been developed to scale up k-means, say, parallel computation utilized in Balcan et al. [2013], random projection used in Boutsidis et al. [2010], Cohen et al. [2015]. Moreover, in Section 4.1, we conduct a series of experiment to show that the objective function will be significantly reduced within few iterations. Accordingly, we design a early termination strategy. The effectiveness of this strategy is validated empirically in Section 4.1.

3.4 Convergence Analysis

In this section, we provide the theoretical analysis of the proposed CP-SG-MCMC algorithm, which are based on the framework of Teh et al. [2014], Chen et al. [2015a]. For ease of exposition, we first define some notations and make some mild assumptions, following Chen et al. [2015a]. In Bayesian inference, the posterior average that we are interested in is defined as: $\bar{\phi} \triangleq \int_\theta \phi(\theta) p(\theta|x) d\theta$ for certain test function $\phi(\theta)$ of interest, which is assumed to be smooth. This integration is computationally intractable in general cases. Thus, it is common to approximate it using the sample average $\hat{\phi}_T = \frac{1}{T}\sum_{l=1}^{T} \phi(\theta_l)$, where $\{\theta_1, \ldots, \theta_T\}$ are samples generated by certain sampling method like MCMC. The precision of the true posterior average and its approximation is characterized by the expected difference between $\bar{\phi}$ and $\hat{\phi}_T$. As shown in Chen et al. [2015a], we require certain assumptions on ϕ. To show these assumptions, we define a functional ψ that solves the following Poission equation:

$$\mathcal{L}\psi(\theta_l) = \phi(\theta_l) - \bar{\phi}, \tag{15}$$

where $\mathcal{L}$ is the generator of the diffusion (2), defined for any compactly supported twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, such that,

$$\begin{aligned}\mathcal{L}f(z_t) &\triangleq \lim_{h \to 0^+} \frac{\mathbb{E}[f(z_{t+h})] - f(z_t)}{h} \\ &= (F(z_t)\cdot\nabla + \frac{1}{2}(\sigma(z_t)\sigma(z_t)^T) : \nabla\nabla^T) f(z_t)\end{aligned} \tag{16}$$

where $z \cdot y = z^T y$, $X : Y \triangleq \text{trace}(X^T Y)$, $h \to 0^+$ represents h approaches to zero along the positive real axis. Correspondingly, we denote the corresponding generators with stochastic gradient $\nabla\bar{U}(\theta_l)$ in the l-th iteration as $\tilde{\mathcal{L}}_l$. Summing over $l = 1, \ldots, T$ on Equation (15), this is equivalent to $\frac{1}{T}\sum_{l=1}^{T} \mathcal{L}\psi(\theta_l) = \hat{\phi}_T - \bar{\phi}$. The solution functional $\psi(\theta_l)$ characterizes the difference between $\phi(\theta_l)$ and the posterior average $\bar{\phi}$ for every θ_l. We assume that

- the corresponding SDE[1] of CP-SG-MCMC to be either elliptic or hypoelliptic. The ellipticity/hypoellipticity describes whether the Brownian motion is able to spread over the whole parameter space.
- ψ and its up to 3rd-order derivatives, $\mathcal{D}^k\psi$, are bounded by a function $\mathcal{V}$, i.e., $\|\mathcal{D}^k\psi\| \le C_k\mathcal{V}^{p_k}$ for $k = (0, 1, 2, 3)$, $C_k, p_k > 0$. Furthermore, the expectation of $\mathcal{V}$ on $\{\theta_l\}$ is bounded: $\sup_l \mathbb{E}\mathcal{V}^p(\theta_l) < \infty$, and $\mathcal{V}$ is smooth such that $\sup_{s\in(0,1)} \mathcal{V}^p(sX + (1-s)Y) \le C(\mathcal{V}^p(X) + \mathcal{V}^p(Y)), \forall\, X, Y, p \le \max\{2p_k\}$ for some $C > 0$.

Regarding to the numerical integrator, we focus on the 1-st order integrator, like forward-Euler integrator, which is commonly used, e.g., in Equation (3). Furthermore, the step-size, denoted h, is fixed during iterative learning. Now we investigate the convergence properties for CP-SG-MCMC algorithm in terms of the bias and mean square error (MSE).

Theorem 3.3. *Under the above assumptions, after T iterations, the bias of SG-MCMC algorithm is bounded as:*

$$\begin{aligned}|\mathbb{E}\hat{\phi}_T - \bar{\phi}| &= O(\frac{1}{Th} + \frac{\sum_l \|\mathbb{E}\Delta V_l\|}{T} + h) \\ &= O(\frac{1}{Th} + h),\end{aligned} \tag{17}$$

where $\Delta V_l \triangleq \mathcal{L} - \tilde{\mathcal{L}}_l$. The second equality follows from the fact that the gradient estimator used in CP-SG-MCMC algorithm is unbiased, as described in Lemma 3.3.

Remark. This results can be seen as a corollary of Theorem 2 in Chen et al. [2015a].

[1] stochastic differential equation

dataset	class	training/testing size	feature dim
usps	10	7,291/2,007	256
pendigits	10	7,494/3,498	16
covtype.binary	2	523,124/57,888	54
mnist	10	60,000/10,000	784
cifar	10	50,000/10,000	1024

Table 1: Bayesian Logistic Regression and Bayesian Neural Network: the datasets

Theorem 3.4. *Under the above assumptions, the MSE of CP-SG-MCMC after T iteration is bounded, for some constant $D_1 > 0$ independent of $\{T, h\}$, as:*

$$\mathbb{E}(\hat{\phi}_T - \bar{\phi})^2 \leq D_1\left(\frac{1}{Th} + \frac{\sum_l \mathbb{E}\|\Delta V_l\|^2}{T^2} + h^2\right). \quad (18)$$

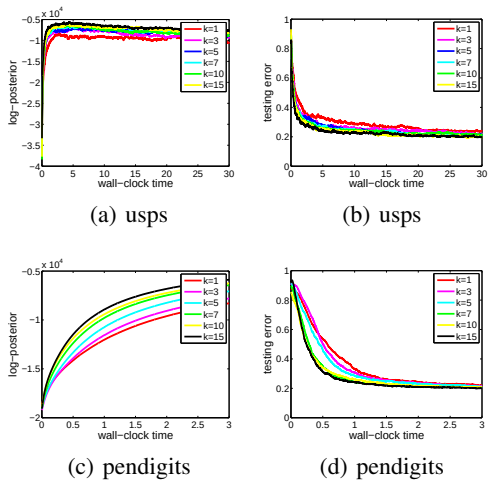
.

Remark. Theorem 3.4 shares a similar form as Theorem 3 in Chen et al. [2015a], but ΔV_l has different meanings. $\mathbb{E}\|\Delta V_l\|^2$ is approximately proportional to the variance of noisy gradient [Chen et al., 2015a]. If it is furtherly assumed that the variance is bounded, we have that $\frac{\sum_l \mathbb{E}\|\Delta V_l\|^2}{T^2} = O(\frac{1}{T})$. Compared with conventional SG-MCMC, $\mathbb{E}\|\Delta V_l\|^2$, is reduced owing to CP strategy and it would accelerate the convergence. Additionally, in the case where the variance is large enough and dominate the other terms in RHS of Equation (18), it will improved the constant and enjoys the same convergence rate ($O(\frac{1}{T})$) with SG-MCMC algorithm in terms of MSE.

4 Applications and Evaluations

In this section, we conduct applications of our CP strategy with empirical evaluations. In particular, we apply our CP strategy to stochastic gradient Langevin dynamics (SGLD) [Welling and Teh, 2011], stochastic gradient Hamilton Monte Carlo (SGHMC) [Chen et al., 2014] and stochastic gradient Riemann Langevin dynamics (SGRLD) [Patterson and Teh, 2013], and evaluate them on Bayesian logistic regression, Bayesian neural network and online latent dirichlet allocation, respectively. The original approaches (SGLD, SGHMC and SGRLD) are regarded as baseline methods, which are based on uniform sampling.

In this paper, as suggested by Chen et al. [2015a], Mandt et al. [2016], we adopt fixed learning rate for all CP-SG-MCMC and SG-MCMC algorithms instead of annealed stepsizes described in Equation (5) . It seems clear that the learning rate interacts in a nontrivial way with the variance of the gradient estimates. Thus we optimize the learning rate for each method. Moreover, for each task, we conduct 5 independent trials and report the average results (including figures). Before describing the empirical effect of our algorithm, we briefly introduce some evaluating metrics first:



(a) usps (b) usps

(c) pendigits (d) pendigits

Figure 1: Bayesian logistic regression. Results on MNIST and covtype are reported in Appendix. Note that when $k = 1$, CP-SGLD reduces to SGLD.

(1) Autocorrelation is the correlation of sample sequence with itself at different points in time. Informally, it is the similarity between observations as a function of the time lag between them. ρ_k denotes the autocorrelation at lag k.

(2) Effective sample size (ESS) is the common measurement, which summaries the amount of autocorrelation across different lags over all dimensions, defined as ESS $= \frac{n}{1+2\sum_{k=1}^{m}\rho_k}$, where n is number of samples generated by sampling method.

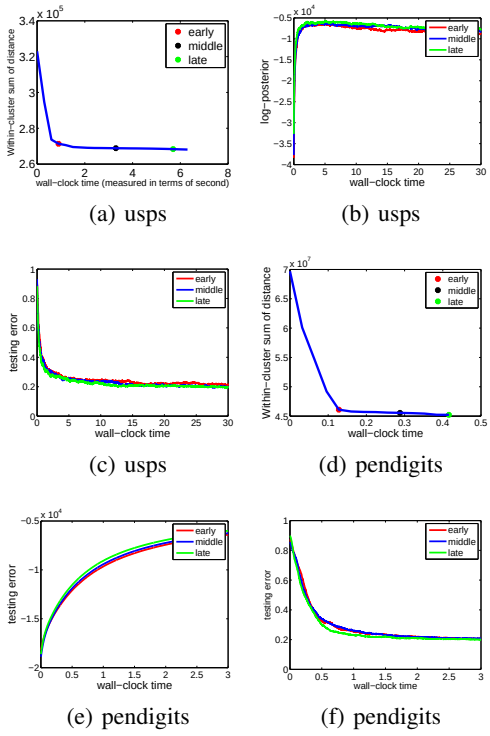
4.1 Bayesian Logistic Regression: Applying to stochastic gradient Langevin dynamics

We apply our CP strategy to SGLD [Welling and Teh, 2011] and evaluate it on the Bayesian multiclass logistic regression model. The experiments are conducted on a number of benchmark datasets. All of these datasets can be downloaded from the LIBSVM website[2]. Details about these datasets can be found in Table 1. They are chosen to cover various sizes of datasets. Let $x \in \mathbb{R}^d$ be a vector of feature values and $y = [y_1, \ldots, y_K]^T \in \mathbb{R}^K$ be a K-dimensional $0/1$ valued vector, where K is the number of classes. There exists a $k \in \{1, \ldots, K\}$ such that $y_k = 1$ and other coordinates are 0. Multiclass logistic regression is a conditional probability model of the form $p(y_k = 1|x, \mathbf{B}) = \frac{\exp(\beta_k^T x)}{\sum_{j=1}^{K}\exp(\beta_j^T x)}$ parametrized by the matrix $\mathbf{B} = [\beta_1, \beta_2, \ldots, \beta_K] \in \mathbb{R}^{d\times K}$. Each column of $\mathbf{B}$ corresponds to one class. The Gaussian prior is used, encouraging all the elements of $\mathbf{B}$ near 0. In the following experiment, for fair comparison, the size of minibatch is set to 100, fixed for each dataset.

[2]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/

dataset	method	k-means/burn-in time	total runtime	$\sum_{i=1}^{\infty} \lvert\rho_k\rvert$	ESS	ESS per second	test error
usps	SGLD	-/9.53	45.98	5.2	351.40	7.63	0.2112
	CP-SGLD	0.97/4.53	47.0	3.5	562.50	11.96	0.2038
pendigits	SGLD	-/3.53	14.3	4.8	216.86	15.16	0.2287
	CP-SGLD	0.12/1.87	14.5	3.2	312.5	21.51	0.2196
covtype.binary	SGLD	-/18.9	115	5.7	305.6	2.65	0.2801
	CP-SGLD	9.7/12.3	124	4.7	353.6	2.84	0.2726
mnist	SGLD	-/25.4	153	6.9	256.0	1.67	0.2403
	CP-SGLD	15.7/16.4	168	6.6	316.4	1.94	0.2153

Table 2: Performance of Bayesian logistic regression. All the results have been averaged out after 5 different runs.

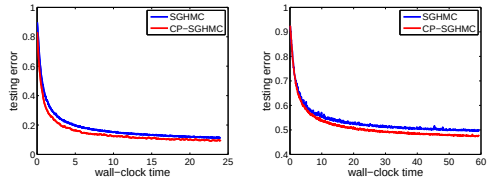


(a) usps (b) usps

(c) usps (d) pendigits

(e) pendigits (f) pendigits

Figure 2: Bayesian logistic regression. Results on mnist and covtype are reported in Appendix. It is shown that early termination strategy in k-means procedure would not degrade the performance. In this paper, the units of wall-clock time are all second.



(a) MNIST (b) CIFAR

Figure 3: Bayesian Neural Network

empirical effect of number of clusters

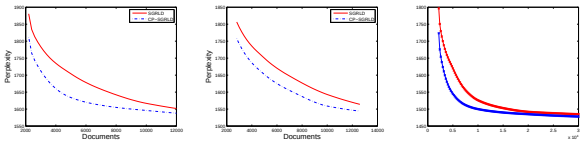
Obviously, the number of clusters k plays a key role in effect of our CP based algorithm. Here, we investigate the relationship between performance of CP-SG-MCMC algorithm and number of clusters. To do this, we run CP-SGLD algorithm with different number of clusters. The other settings, like mini-batch size, are all fixed. Both log-posterior[3] and testing error as a function of wall-clock time are recorded in Figure 1 for different number of clusters. We find that it would be better to increase the number of clusters k. Naturally, too large k is not feasible owing to the limitation of minibatch size. It is easy to find a reasonable k. We set k to 10 for following experiment.

k-means algorithm: terminate before convergence

Then we study the efficiency of k-means algorithm. Vanilla k-means algorithm need quite a few iterations to converge to the optimum. However, as shown in (a), (d) of Figure 2 and some literature about k-means [Cohen et al., 2015], we know that the result is close to the optimum with only a few iterations. Then, we attempt to terminate before convergence of k-means algorithm. That is, we use the clustering result after a small number of iterations of k-means algorithm for our gradient estimator and sampling procedure. We name this strategy "early termination".

To validate the effectiveness of this idea, for a given k-means procedure, we use the clustering result in different epochs (early, middle and late) to perform the sampling procedure. The results are shown in Figure 2. We find that the gap of results is negligible and the effect is desirable after performing k-means algorithm with a small number of iterations. Thus the preprocessing time will not bring too heavy computational burden as expected. This strategy is used throughout the experiment. From the above two parts, we find that k-means algorithm is robust and efficient. Then we provide an elaborate comparison of SGLD and CP-SGLD (using early termination and a reasonable k) in Table 2. We observe that CP-SGLD converges faster and achieves a significantly smaller and stable test error than SGLD on almost all datasets. Worth to note that thanks

[3]Here, log-posterior corresponds to the log of the unnormalized posterior, i.e., $\log p(\theta, x) = \log p(\theta) + \log p(x|\theta)$.



(a) ICML (b) NIPS (c) Wikipedia

Figure 4: Online LDA

to the early termination of k-means algorithms, the time cost by k-means is negligible compared with cost of the sampling procedure.

4.2 Bayesian Neural Network: Applying to stochastic gradient Hamilton Monte Carlo

Then we adapt our CP strategy to SGHMC [Chen et al., 2014] and then compare CP-SGHMC and SGHMC on Bayesian neural network model. The datasets we use are MNIST and CIFAR-10[4], both of which are standard datasets for deep neural network studies and described in Table 1. We train the neural network with one fully-connected hidden layer and 10 softmax output nodes. Sigmoid activation is used. We choose a zero-mean Gaussian prior for weight parameter of neural network. The numbers of nodes in hidden layer are 100 and 300 for MNIST and CIFAR-10, respectively. The sizes of minibatch and numbers of clusters are set to 200 and 20 respectively for both MNIST and CIFAR-10. Note that for fair comparison, SGHMC and CP-SGHMC adopt the same mini-batch size. Additionally, we perform 5-fold cross validation to select the best hyperparameter. In both SGHMC and CP-SGHMC, the MH correction step is ignored, following Chen et al. [2014]. Since our CP strategy does not cause additional overhead in computation and the weight parameter lies in high dimension space, the main computational bottleneck is the gradient evaluation, shared by both methods. Thus the two methods have almost the same computational complexity per iteration. For both methods, the test errors of different methods as a function of running time are provided in Figure 3, from which we can see that under the same setting, CP-SGHMC outperforms SGHMC in efficiency, effectiveness and stability owing to the reduction in variance.

4.3 Online Latent Dirichlet Allocation: Applying to stochastic gradient Riemann Langevin dynamics

Finally, we apply CP strategy to SGRLD [Patterson and Teh, 2013] and verify the effectiveness of the resulting CP-SGRLD on online Latent Dirichlet Allocation [Hoffman et al., 2010, Blei et al., 2003] tasks on a bunch of corpus. The datasets contain the ICML corpus, NIPS corpus, and Wikipedia corpus. Concretely, NIPS corpus is composed of a collection of NIPS papers from 2006-2015 and include 3073 documents while ICML corpus contains a collection of ICML papers from 2009-2015 and 1594 documents. Wikipedia corpus is much larger, contains 150, 000 documents from Wikipedia. More details about online LDA is described in Appendix. Naturally, the baseline method is SGRLD [Patterson and Teh, 2013], employing uniform sampling when approximating the gradient of log-posterior. The size of mini-batch is set to 50, 50 and 100 for ICML, NIPS, Wikipedia corpus, respectively. For CP-SGRLD, the number of clusters k is set to 5 for ICML and NIPS corpus and 15 for Wikipedia corpus. The other settings follow along Patterson and Teh [2013]. The metric used here is perplexity, the exponentiated cross entropy between the trained model probability distribution and the empirical distribution of the test data. For both approaches, we report average results of 5 random runs. The average runtime for the different methods are almost similar since the computation of stochastic gradient dominates in the computational time and CP strategy don't cause extra overhead. For both SGRLD and CP-SGRLD, the perplexities as a function of iteration number are reported in Figure 4, demonstrating that our CP-SGRLD is superior to SGRLD on this task.

Remark Since the learning rate is optimized for each methods, we always find that the hand-tuning optimized learning rates of CP-SG-MCMC methods are usually larger than that of vanilla SG-MCMC algorithms. Thus we would learn something from this: perhaps the variance reduction allows larger learning rates!

5 Conclusion

In this paper, we focus on reducing the variance of gradient estimator in SG-MCMC, an orthogonal direction of stochastic gradient MCMC samplers with previous works. Inspired by both importance sampling and stratified sampling, we have devised a novel algorithm referred to as clustering-based preprocessing stochastic gradient MCMC (CPSG-MCMC) method. It only requires precalculating the cluster of the observation instances before sampling, so no extra overhead is taken during sampling procedure. Moreover, it is compatible with all stochastic MCMC samplers and parallel computation. Experiments on several real datasets have been conducted to verify the effectiveness and stability of our approach.

Acknowledgement

This work has been supported by the National Natural Science Foundation of China (No. 61572017), Natural Science Foundation of Shanghai City (No. 15ZR1424200), 973 Program (No. 2015CB856000) and Microsoft Research Asia Collaborative Research Award.

[4]https://www.cs.toronto.edu/ kriz/cifar.html

References

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.

PJ Rossky, JD Doll, and HL Friedman. Brownian dynamics as smart Monte Carlo simulation. *The Journal of Chemical Physics*, 69(10):4628–4633, 1978.

Gareth O Roberts and Osnat Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

Sungjin Ahn, Anoop Korattikara Balan, and Max Welling. Bayesian posterior sampling via Stochastic Gradient Fisher Scoring. In *ICML*, 2012.

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.

Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *ICML*, 2014.

Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems*, pages 3203–3211, 2014.

Sungjin Ahn, Babak Shahbaba, Max Welling, et al. Distributed stochastic gradient MCMC. In *ICML*, pages 1044–1052, 2014.

Yee Whye Teh, Alexandre Thiéry, and Sebastian Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *arXiv preprint arXiv:1409.0578*, 2014.

Changyou Chen, Nan Ding, and Lawrence Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pages 2278–2286, 2015a.

Emily Fox Yi-An Ma, Tianqi Chen. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, 2015.

Changyou Chen, David Carlson, Zhe Gan, Chunyuan Li, and Lawrence Carin. Bridging the gap between stochastic gradient MCMC and stochastic optimization. *arXiv preprint arXiv:1512.07962*, 2015b.

Chunyuan Li, Changyou Chen, David E. Carlson, and Lawrence Carin. Preconditioned Stochastic Gradient Langevin Dynamics for deep neural networks. In *AAAI*, 2016a.

Wenzhe Li, Sungjin Ahn, and Max Welling. Scalable MCMC for mixed membership stochastic blockmodels. In *AISTATS*, 2016b.

Chunyuan Li, Changyou Chen, Kai Fan, and Lawrence Carin. High-order stochastic gradient thermostats for bayesian learning of deep models. In *AAAI*, 2016c.

Umut Simsekli, Roland Badeau, Ali Taylan Cemgil, and Gael Richard. Stochastic Quasi-Newton Langevin Monte Carlo. In *ICML*, 2016.

Chang Liu, Jun Zhu, and Yang Song. Stochastic gradient Geodesic MCMC methods. In *Advances in Neural Information Processing Systems*, 2016.

Changyou Chen, Nan Ding, Chunyuan Li, Yizhe Zhang, and Lawrence Carin. Stochastic gradient MCMC with stale gradients. In *Advances in Neural Information Processing Systems*, 2016.

Alain Durmus, Umut Simsekli, Eric Moulines, Roland Badeau, and Gael Richard. Stochastic gradient Richardson-Romberg Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, 2016.

Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

Sebastian J Vollmer, Konstantinos C Zygalakis, et al. (non-) asymptotic properties of stochastic gradient Langevin dynamics. *arXiv preprint arXiv:1501.00438*, 2015.

Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

Peilin Zhao and Tong Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv preprint arXiv:1405.3080*, 2014a.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Chong Wang, Xi Chen, Alex J Smola, and Eric P Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189, 2013.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling. *arXiv preprint arXiv:1401.2753*, 2014b.

Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Accelerating stochastic gradient descent via online learning to sample. *arXiv preprint arXiv:1506.09016*, 2015.

Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pages 274–285. Springer, 2009.

Maria-Florina F Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general

topologies. In *Advances in Neural Information Processing Systems*, pages 1995–2003, 2013.

Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. Random projections for k-means clustering. In *Advances in Neural Information Processing Systems*, pages 298–306, 2010.

Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 163–172. ACM, 2015.

Stephan Mandt, Matthew D Hoffman, and David M Blei. A variational analysis of stochastic gradient algorithms. *arXiv preprint arXiv:1602.02666*, 2016.

Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

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

Appendix

Additional Empirical Results

5.1 Details about online LDA

In LDA, each topic is associated with a distribution over words, with β_{kw} the probability of word w under topic k. Every document is related to a mixture of topic, with $\pi_k^{(d)}$ the probability of topic k of document d. Documents are generated by first selecting a topic $z_j^{(d)} \sim \pi^{(d)}$ for the j-th word and then drawing the specific word from the topic as $x_j^{(d)} \sim \beta_{z_j^d}$, where $\pi^{(d)}$ and β_k are given Dirichlet priors.

Background

In this section, we will introduce related background, including importance sampling and stratified sampling.

Variance reduction is commonly used in Monte Carlo computationsLiu [2008], Robert and Casella [2013] and contains a bunch of classical methods. In this paper, we restrict our attention on two highly-related variance reduction methods, importance sampling and stratified sampling. These two methods can also be applied in stochastic optimization problemZhao and Zhang [2014a,b].

Importance Sampling

Importance sampling are derived from the Monte Carlo computation [Liu, 2008]. Suppose we are interested in the quantity as follows:

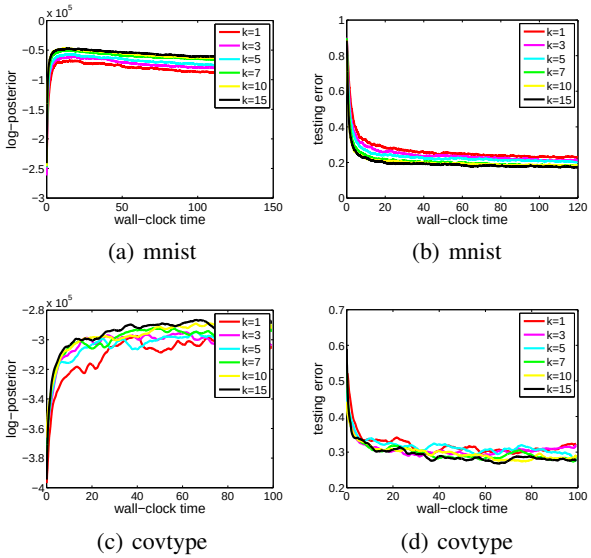
$$Q = \int_X h(x)\pi(x)dx = E_\pi[h(x)] \tag{19}$$

where $\pi(x)$ is a probability distribution. The vanilla Monte Carlo is to draw m random samples from $\pi(x)$ and simply compute the average $h(x)$. However, when the $\pi(x)$ is difficult to sample, a trial distribution $g(x)$ is defined, which is easy to sample. The importance weight is defined as $w(x) = \frac{\pi(x)}{g(x)}$. Then draw from $g(x)$, compute the weighted average $w(x)h(x)$. It is easy to verify that the estimated quantity equal to the desired quantity in expectation.

In this paper, the space X contain N points, and the $\pi(x)$ equals to uniform distribution $[\frac{1}{N}, \ldots, \frac{1}{N}]$. Other quantity can also be extended into discrete case similarly.

Stratified Sampling

Stratified sampling [Liu, 2008] can be viewed as a special importance sampling method with its trial density $g(x)$ constructed as a piecewise constant function. Suppose we wish to evaluate the quantity $\int_X f(x)dx$. If possible, we want to break the region X into the union of k disjoint subregions, $D_1, \ldots, D_k$, so that within each subregion, the function $f(x)$ is relatively homegeneous(close to be a constant). Then we perform random sampling in each subregion D_i and obtain the estimation of each subregional integral $\int_{D_i} f(x)dx$ with a small variance. The integral over X can be approximated by the sum of these subregional integration. It can be easily proved that the new estimator is an unbiased estimator as well as lower variance.



(a) mnist (b) mnist

(c) covtype (d) covtype

Figure 5: Performance on Bayesian logistic regression on MNIST and covtype. Note that when $k = 1$, CP-SGLD reduces to SGLD.

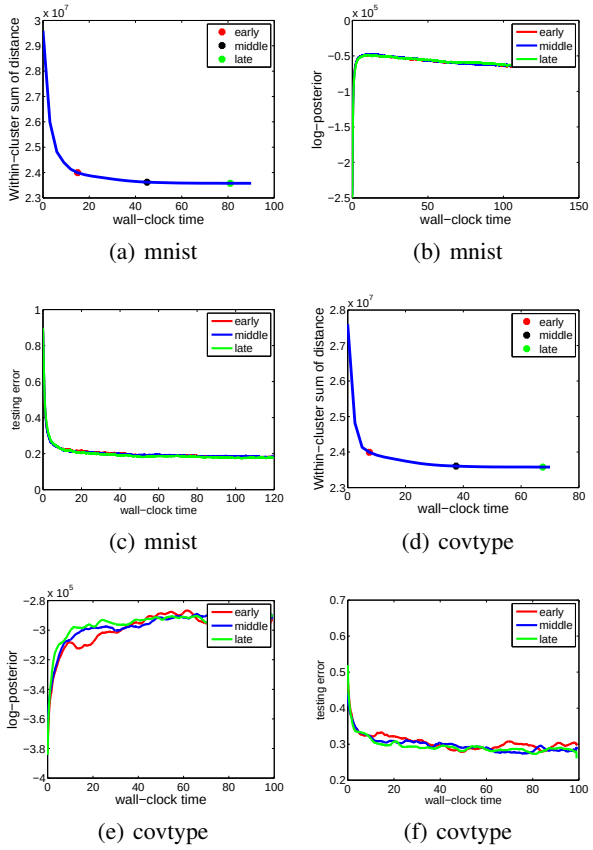
In this paper, each subregion represents a cluster in discrete scenario. Stratified sampling provides an approximation for the optimal trial distribution $\pi(x)$, reduces the total variance by cluster similar x in a subregion. Among the three methods—importance sampling, stratified sampling and uniform sampling, Importance sampling provides estimator with least variance while uniform sampling largest. In contrast, uniform sampling is most efficient while importance sampling most cumbersome. The moral is this: there is no free lunch. And this philosophy will be used in this paper. The details will be shown in the next section.

Proof

In this section, we provide elaborate proof for all the theoretical work. First we describe the celebrated Cauchy-Schwarz Inequality, which is frequently used in proof.

Lemma 5.1. *The Cauchy–Schwarz inequality states that for all n-dimensional real vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, of an inner product space it is true that*

$$|\langle \mathbf{u}, \mathbf{v} \rangle|^2 \leq \langle \mathbf{u}, \mathbf{u} \rangle \cdot \langle \mathbf{v}, \mathbf{v} \rangle,$$



(a) mnist (b) mnist

(c) mnist (d) covtype

(e) covtype (f) covtype

Figure 6: Performance if Bayesian logistic regression on MNIST and covtype. It is shown that early termination would not degrade the performance.

or equivalently,

$$(\sum_{i=1}^{n} u_i v_i)^2 \leq (\sum_{i=1}^{n} u_i^2)^2 (\sum_{i=1}^{n} v_i^2)^2.$$

Furthermore, the equality holds only when either $\mathbf{u}$ *or* $\mathbf{v}$ *is a multiple of the other.*

Proof of Lemma 3.1

Proof. According to the definition of estimated gradient $\mathbb{E}(\nabla \hat{U}(\theta))$, we have that

$$\begin{aligned}
&\mathbb{E}(\nabla \hat{U}(\theta)) \\
=&-\sum_{i=1}^{N}(p_i)[(p_i)^{-1}\nabla \log p(x_i|\theta)] - \nabla \log p(\theta) \\
=&\nabla U(\theta).
\end{aligned}$$

This completes the proof. □

Proof of Lemma 3.2

Proof. Owing to the fact that $\mathbb{V}(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2$ and the expectation of $\hat{U}(\theta_t)$ equal to the true gradient and can be seen as a constant, Equation (7)

$$\arg\min_{\forall i \in \{1,2,\ldots,N\},\ p_i \in (0,1);\ \sum_{i=1}^{N} p_i = 1} \mathbb{V}((p_i)^{-1}\nabla \log p(x_i|\theta)),$$

is equivalent into the following form:

$$\arg\min_{\forall i \in \{1,2,\ldots,N\},\ p_i \in (0,1);\ \sum_{i=1}^{N} p_i = 1} \sum_{i=1}^{N}((p_i)^{-1}\|\nabla \log p(x_i|\theta_t)\|^2) \tag{20}$$

For notational convenience, here, we use $\|\cdot\|$ to represent the l_2-norm for vector.

Then using Cauchy-Schwarz inequality to solve the optimization problem in Equation (20), as follows:

$$\begin{aligned}
&\sum_{i=1}^{N}((p_i)^{-1}\|\nabla \log p(x_i|\theta_t)\|^2) \\
=&(\sum_{i=1}^{N}(p_i)^{-1}\|\nabla \log p(x_i|\theta_t)\|^2)(\sum_{i=1}^{N} p_i) \\
\geq&(\sum_{i=1}^{N}\sqrt{(p_i)^{-1}\|\nabla \log p(x_i|\theta_t)\|^2 p_i})^2 \qquad (21) \\
=&(\sum_{i=1}^{N}\sqrt{\|\nabla \log p(x_i|\theta_t)\|^2})^2 \\
=&(\sum_{i=1}^{N}\|\nabla \log p(x_i|\theta_t)\|)^2
\end{aligned}$$

The equality is obtained only when there exists a constant c such that

$$\begin{pmatrix} (p_1)^{-1}\|\nabla \log p(x_1|\theta_t)\|^2 \\ \vdots \\ (p_N)^{-1}\|\nabla \log p(x_N|\theta_t)\|^2 \end{pmatrix} = c \begin{pmatrix} p_1 \\ \vdots \\ p_N \end{pmatrix}, \tag{22}$$

which is equivalent to

$$\begin{aligned} &(p_1)^{-2}\|\nabla \log p(x_1|\theta_t)\|^2 = \dots \\ &= (p_N)^{-2}\|\nabla \log p(x_N|\theta_t)\|^2. \end{aligned} \tag{23}$$

Under such case, the variance is minimized, so the optimal strategy is to choose p_i satisfy that

$$p_i = \frac{\|\nabla \log p(x_i|\theta_t)\|}{\sum_{j=1}^{N} \|\nabla \log p(x_j|\theta_t)\|}, \tag{24}$$

for $\forall\, i \in \{1, 2, \dots, N\}$.

This completes the proof.

□

Proof of Lemma 3.3

Proof. According to Equation (10), we directly expand $\nabla \hat{U}(\theta_t)$ and have that

$$\begin{aligned} &\mathbb{E}[\nabla \hat{U}(\theta_t)] \\ =& -\mathbb{E}[\sum_{i=1}^{k} \frac{n_i}{b_i} \sum_{\substack{j\in B_i \\ |B_i|=b_i}} \nabla \log p(x_j|\theta_t)] - \nabla \log p(\theta) \\ =& -\sum_{i=1}^{k} \frac{n_i}{b_i} \mathbb{E}[\sum_{\substack{j\in B_i \\ |B_i|=b_i}} \nabla \log p(x_j|\theta_t)] - \nabla \log p(\theta) \\ =& -\sum_{i=1}^{k} \frac{n_i}{b_i} \sum_{\substack{j\in B_i \\ |B_i|=b_i}} \mathbb{E}[\nabla \log p(x_j|\theta_t)] - \nabla \log p(\theta) \\ =& -\sum_{i=1}^{k} \frac{n_i}{b_i} \sum_{\substack{j\in C_i \\ |C_i|=n_i}} [\frac{b_i}{n_i} \nabla \log p(x_j|\theta_t)] - \nabla \log p(\theta) \\ =& \sum_{i=1}^{N} \nabla \log p(x_i|\theta_t) - \nabla \log p(\theta) \\ =& \nabla U(\theta_t) \end{aligned} \tag{25}$$

where the fourth equality follows from the fact that within each cluster C_i, each data sample is selected with equal probability at b_i/n_i. The fifth equality follows from the properties of clusters $C_1, \dots, C_k$ described in Equation (9).

□

Proof of Lemma 3.4

Proof. $\mathbb{V}(\nabla \bar{U}(\theta_t)|C_1, C_2, \dots, C_k, b_1, b_2, \dots, b_k)$, the new variance, can be represented as the sum of weighted within-clustering variance, as following:

$$\begin{aligned} &\mathbb{V}(\nabla \bar{U}(\theta_t)|C_1, C_2, \dots, C_k, b_1, b_2, \dots, b_k) \\ =& \mathbb{V}[\sum_{i=1}^{k} \frac{n_i}{b_i} \sum_{\substack{j\in B_i \\ |B_i|=b_i}} (\nabla \log p(x_j|\theta_t))] \\ =& \sum_{i=1}^{k} (\frac{n_i}{b_i})^2 \mathbb{V}[\sum_{j\in B_i | B_i|=b_i} \nabla \log p(x_j|\theta_t)] \\ =& \sum_{i=1}^{k} (\frac{n_i}{b_i})^2 \sum_{\substack{j\in C_i \\ |C_i|=n_i}} \\ & \frac{b_i}{n_i} \|\nabla \log p(x_j|\theta_t) - \frac{1}{n_i} \sum_{l\in C_i} \nabla \log p(x_l|\theta_t)\|^2 \\ =& \sum_{i=1}^{k} (\frac{n_i}{b_i})^2 \sum_{\substack{j\in C_i \\ |C_i|=n_i}} \frac{b_i}{n_i} \|\nabla \log p(x_j|\theta_t) - \mu_{C_i}\|^2 \\ =& \sum_{i=1}^{k} \frac{n_i}{b_i} \sum_{\substack{j\in C_i \\ |C_i|=n_i}} \|\nabla \log p(x_j|\theta_t) - \frac{1}{n_i} \sum_{l\in C_i} \nabla \log p(x_l|\theta_t)\|^2, \\ =& \sum_{i=1}^{k} \frac{n_i}{b_i} \sum_{\substack{j\in C_i \\ |C_i|=n_i}} \mathbb{V}[(\nabla \log p(x_j|\theta_t))], \end{aligned} \tag{26}$$

where $\mu_{C_i} \triangleq \frac{1}{n_i} \sum_{\substack{j\in C_i \\ |C_i|=n_i}} \nabla \log p(x_j|\theta_t)$, represents the mean gradient of cluster C_i. The second equality (the third line of the above equation) follows from the fact that $B_1, \dots, B_k$ are independent with each other, without any overlapping. The third equality (the fourth line) follows from the fact that for cluster C_i, mini-batch with size of b_i is randomly drawn. Thus the variance regarding cluster B_i is related to the variance of cluster C_i.

□

Proof of Theorem 3.1

Proof. We expand $\mathbb{V}(\nabla\bar{U}(\theta_t)|C_1, C_2, \ldots, C_k, b_1, b_2, \ldots, b_k)$ according to the definition of variance, and have

$$\begin{aligned}
&\mathbb{V}(\nabla\bar{U}(\theta_t)|C_1, C_2, \ldots, C_k, b_1, b_2, \ldots, b_k)\\
=&\sum_{i=1}^{k}\frac{n_i}{b_i}\sum_{\substack{j\in C_i,\\ |C_i|=n_i}}\mathbb{V}(\nabla\log p(x_j|\theta_t))\\
=&\sum_{i=1}^{k}\frac{n_i}{b_i}\sum_{\substack{j\in C_i,\\ |C_i|=n_i}}\|\nabla\log p(x_j|\theta_t)-\frac{1}{n_i}\sum_{l}\nabla\log p(x_l|\theta_t)\|^2\\
\leq&\sum_{i=1}^{k}\frac{n_i}{b_i}\sum_{\substack{j\in C_i,\\ |C_i|=n_i}}\\
&\|\nabla\log p(x_j|\theta_t)-\nabla\log p(\frac{1}{n_i}\sum_{\substack{l\in C_i,\\ |C_i|=n_i}}x_l|\theta_t)\|^2\\
=&\sum_{i=1}^{k}\frac{n_i}{b_i}\sum_{\substack{j\in C_i,\\ |C_i|=n_i}}\|\nabla\log p(x_j|\theta_t)-\nabla\log p(\mu_i|\theta_t)\|^2\\
\leq&L^2\sum_{i=1}^{k}\frac{n_i}{b_i}\sum_{\substack{j\in C_i,\\ |C_i|=n_i}}\|x_j-\mu_i\|^2
\end{aligned}\tag{27}$$

where $\mu_i=\sum_{\substack{j\in C_i,\\ |C_i|=n_i}}x_j/n_i$. The first equality follows from the results of Lemma 3.4. The first inequality (the fourth line) follows from the fact that given $x_1,\ldots,x_n\in\mathbb{R}^d$, we have

$$\frac{1}{n}\sum_{i=1}^{n}x_i=\arg\min_y\sum_{i=1}^{n}\|x_i-y\|_2^2.\tag{28}$$

Equivalently,

$$\|x_i-\frac{1}{n}\sum_{i=1}^{n}x_i\|_2^2\leq\sum_{i=1}^{n}\|x_i-y\|_2^2\quad\text{for }\forall y.\tag{29}$$

Hence, we can take the summation into the log-probability and have that

$$\begin{aligned}
&\|\nabla\log p(x_j|\theta_t)-\frac{1}{n_i}\sum_{l\in C_i}\nabla\log p(x_l|\theta_t)\|^2\\
\leq&\|\nabla\log p(x_j|\theta_t)-\nabla\log p(\frac{1}{n_i}\sum_{l\in C_i}x_l|\theta_t)\|^2.
\end{aligned}\tag{30}$$

Furthermore, the last inequality holds due to the definition of L-Lipschitz.

□

Proof of Theorem 3.2

First, we restate the Theorem 3.2.

Theorem 5.1. *For optimization problem described in Equation* (13) *as*

$$\arg\min_{C_1,C_2,\ldots,C_k,b_1,b_2,\ldots,b_k}\sum_{i=1}^{k}\frac{n_i}{b_i}\sum_{j\in C_i,|C_i|=n_i}\|x_j-\mu_i\|^2,$$

where $\mu_i=\sum_{j\in C_i,|C_i|=n_i}x_j/n_i$ *represents the center of the cluster* C_i*, defined in Theorem 3.1. Given the clusters* $C_1,\ldots,C_k$*, the solutions to the weights* $b_1,\ldots,b_k$ *are*

$$b_i=\frac{bn_i\sqrt{v_i}}{\sum_{j=1}^{k}n_j\sqrt{v_j}},$$

where $v_i=\frac{1}{n_i}\sum_{j\in C_i}\|x_j-\frac{1}{n_i}\sum_{l\in C_i}x_l\|^2=\mathbb{V}(C_i)$.

$\mathbb{V}(C_i)$ *corresponds to the variance of data samples in cluster* C_i *and* b *is the size of the mini-batch satisfying that* $b=\sum_{i=1}^{k}b_i$*. Note that* b_i *represents the number of samples drawn from the cluster* C_i *and is usually fine-tuned to integer in practice. Both* b *and* k *are prespecified.*

Proof. Since the clusters $C_1,\ldots,C_k$ are given, we reformulate the optimization as

$$\arg\min_{b_1,b_2,\ldots,b_k}\sum_{i=1}^{k}\frac{n_i^2}{b_i}v_i.\tag{31}$$

Furthermore, the size of minibatch b is given and we have that

$$\sum_{i=1}^{k}b_i=b.\tag{32}$$

Thus, we have that

$$\begin{aligned}
&\sum_{i=1}^{k}\frac{n_i^2}{b_i}v_i\\
=&\frac{1}{b}(\sum_{i=1}^{k}\frac{n_i^2}{b_i}v_i)(\sum_{i=1}^{k}b_i)\\
\geq&\frac{1}{b}(\sqrt{\frac{n_i^2}{b_i}v_ib_i})^2\\
=&\frac{1}{b}(\sum_{i=1}^{k}\sqrt{n_i^2v_i})^2\\
=&\frac{1}{b}(\sum_{i=1}^{k}n_i\sqrt{v_i})^2,
\end{aligned}\tag{33}$$

where the inequality follows from the Cauchy-Schwarz Inequality. The equality holds only when there exists a con-

stant c such that

$$\begin{pmatrix} \frac{n_1^2}{b_1} v_1 \\ \vdots \\ \frac{n_N^2}{b_N} v_k \end{pmatrix} = c \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix}, \tag{34}$$

which is equivalent to

$$\frac{n_1^2}{b_1^2} v_1 = \frac{n_2^2}{b_2^2} v_2 = \cdots = \frac{n_k^2}{b_k^2} v_k. \tag{35}$$

Under such case, the optimum is reached, so the optimal strategy is to choose b_i satisfying that

$$b_i \propto n_i \sqrt{v_i} \tag{36}$$

for $\forall\, i \in \{1, 2, \ldots, N\}$. Combining the constaint that

$$\sum_{i=1}^{k} b_i = b, \tag{37}$$

we have that

$$b_i = \frac{n_i \sqrt{v_i}}{\sum_{j=1}^{k} n_j \sqrt{v_j}}. \tag{38}$$

Furthermore, in practice, b_i is usually finetuned to integrator for convenience.

This completes the proof.

□