Stochastic Approximate Gradient Descent via the Langevin Algorithm

Yixuan Qiu,[1] **Xiao Wang**[2]
[1]Department of Statistics and Data Science, Carnegie Mellon University, yixuanq@andrew.cmu.edu
[2]Department of Statistics, Purdue University University, wangxiao@purdue.edu

Abstract

We introduce a novel and efficient algorithm called the stochastic approximate gradient descent (SAGD), as an alternative to the stochastic gradient descent for cases where unbiased stochastic gradients cannot be trivially obtained. Traditional methods for such problems rely on general-purpose sampling techniques such as Markov chain Monte Carlo, which typically requires manual intervention for tuning parameters and does not work efficiently in practice. Instead, SAGD makes use of the Langevin algorithm to construct stochastic gradients that are biased in finite steps but accurate asymptotically, enabling us to theoretically establish the convergence guarantee for SAGD. Inspired by our theoretical analysis, we also provide useful guidelines for its practical implementation. Finally, we show that SAGD performs well experimentally in popular statistical and machine learning problems such as the expectation-maximization algorithm and the variational autoencoders.

Introduction

The stochastic gradient descent method (SGD, Bottou 2010; Bottou, Curtis, and Nocedal 2018) is one of the most popular and widely-used optimization techniques in large-scale machine learning problems. In many cases, the objective function one needs to optimize can be written as an expectation, $F(\theta) = \mathbb{E}[f(\theta;\xi)]$, over some random variable $\xi \in \mathbb{R}^r$ whose distribution is independent of the parameter vector $\theta \in \Theta \subset \mathbb{R}^p$. Under very mild regularity conditions, the true gradient of $F(\theta)$ is also an expectation, obtained as $g(\theta) \coloneqq \nabla F(\theta) = \mathbb{E}[\nabla f(\theta;\xi)]$. When the computational cost of $g(\theta)$ is massive, SGD makes use of the stochastic gradient, denoted by $\tilde{g}(\theta)$, to update the parameter vector. It has been well studied that by appropriately choosing the step sizes, SGD has good convergence properties (Robbins and Monro 1951). As an important special case, SGD is frequently used in the scenario where $F(\theta)$ is an average over the data points, $F(\theta) = n^{-1}\sum_{i=1}^{n} f(\theta; X_i)$. If data $X_1, \ldots, X_n$ are assumed to be independent and identically distributed, then an unbiased stochastic gradient can be trivially obtained as $\tilde{g}(\theta) = \nabla f(\theta; X_I)$, where I follows a uniform distribution on $\{1, 2, \ldots, n\}$.

However, there are a much broader class of problems where ξ follows a general probability distribution $\pi(\xi)$. Unlike the previous simple scenario, in many cases an unbiased stochastic gradient cannot be easily obtained due to the complexity of $\pi(\xi)$. If $\pi(\xi)$ is beyond the scope of standard distribution families, then some general-purpose sampling techniques such as Markov chain Monte Carlo (MCMC, Gilks, Richardson, and Spiegelhalter 1995; Metropolis et al. 1953; Hastings 1970; Geman and Geman 1984; Brooks et al. 2011) have to be adopted, which can be quite slow in practice.

In this article, we propose a novel and efficient algorithm called the stochastic approximate gradient descent (SAGD), as an alternative to SGD for cases where unbiased stochastic gradients cannot be trivially computed. The key idea of SAGD is to construct the stochastic gradient using the Langevin algorithm (Roberts and Tweedie 1996; Roberts and Stramer 2002; Cheng et al. 2018b), a sampling method whose statistical error can be rigorously quantified. In addition, we use an adaptive sampling scheme that allows larger errors in the early stage of the optimization, and gradually improves the precision as the procedure goes on.

These heuristics are formalized in the SAGD algorithm, and various theoretical results are developed to guarantee its convergence. Moreover, our analysis gives clear rates of the relevant hyperparameters, which provide useful guidelines for practical implementations of SAGD. The highlights and main contributions of this article are as follows:

- We develop a new computational framework for SGD problems in which a stochastic gradient cannot be trivially obtained. The proposed SAGD algorithm is fully automated with a solid convergence guarantee.
- New theoretical contributions are made to the underdamped Langevin algorithm for sampling from sophisticated distributions, which are of interest by their own.
- We discuss the application of the proposed SAGD framework in some important statistical and machine learning problems, including the expectation-maximization algorithm (EM algorithm), and the variational autoencoders (VAE). We show that SAGD is able to automate the EM algorithm for complex models and effectively remove the bias of VAE.

Notation: Throughout this article we adopt the following notation. Let $\mathbb{R}^r$ be the r-dimensional Euclidean space with the inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$. For matrices and higher-order tensors, $\| \cdot \|$ denotes the operator norm. Let $C \subset \mathbb{R}^r$ be a closed convex set, and then the notation $\mathcal{P}_C(x)$ means the projection of $x \in \mathbb{R}^r$ onto C. A mapping $\phi : \mathbb{R}^r \to \mathbb{R}^s$ is said to have polynomial growth if there exist a constant $C > 0$ and an integer $m \geq 0$ such that $\|\phi(x)\| \leq C(1 + \|x\|^m)$ for all $x \in \mathbb{R}^r$. The notation $\nabla^i \phi$ is used to denote the i-th derivative of a multivariate function $\phi : \mathbb{R}^r \to \mathbb{R}$, and in particular $\nabla^0 \phi \equiv \phi$. We use $\mathscr{C}^m_{poly}$ to denote the space of mapping ϕ such that ϕ is m-times differentiable, and ϕ and its derivatives have polynomial growth. A function $\phi : \mathbb{R}^r \to \mathbb{R}$ is said to be L-Lipschitz continuous if $|\phi(x) - \phi(y)| \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^r$.

Related Work

The two main ingredients of the proposed SAGD framework are SGD and the Langevin algorithm. SGD has been extensively studied in the literature, and recent research mainly focused on its acceleration, for example variance reduction methods (Johnson and Zhang 2013; Reddi et al. 2016), adaptive step sizes (Duchi, Hazan, and Singer 2011; Zeiler 2012), momentum methods (Kingma and Ba 2015; Luo et al. 2019), etc. In this article, the proposed SAGD framework is based on the original version of SGD, but it can be easily adapted to those acceleration methods.

The Langevin algorithm has two variants, the *overdamped* Langevin algorithm and the *underdamped* version. Most of the analysis in literature was based on the overdamped version (Dalalyan 2017; Durmus and Moulines 2019; 2017; Cheng and Bartlett 2018), whereas some recent research suggests that the underdamped version has faster convergence for some special classes of distributions (Cheng et al. 2018a; Ma et al. 2019). Due to this reason, we use the underdamped Langevin algorithm to develop the SAGD framework. As a byproduct, we have derived new results for the underdamped Langevin algorithm that complement prior art.

The Langevin algorithm can be compared to MCMC, as they are both useful sampling techniques. In fact, there are MCMC algorithms derived from the Langevin algorithm such as the Metropolis-based overdamped Langevin algorithm (Roberts and Tweedie 1996) and the underdamped Langevin MCMC (Cheng et al. 2018b). Nevertheless, the Langevin algorithm has several advantages in our problem. First, the Langevin algorithm skips the Metropolis adjustment step existing in most MCMC methods, which saves computational time and avoids duplicated values in the sample. Second, as our theoretical analysis shows, the Langevin algorithm has transparent hyperparameter setting and requires less manual intervention. The downside is the resulting bias of the Langevin algorithm, but the theoretical analysis shows that it does not harm the convergence of SAGD.

The idea to combine SGD with the Langevin algorithm has been seen in articles such as Xie et al. (2018) and Han et al. (2019), but in these works the Langevin algorithm was merely used as an MCMC-like sampling technique, and its statistical error and impact on the convergence of optimization were ignored. Instead, in SAGD the two ingredients are connected in a coherent way, with a rigorous theoretical analysis. One recent work that is similar to SAGD is De Bortoli et al. (2019), but the major difference is that they used the overdamped Langevin algorithm for sampling, whose theoretical analysis is very different from SAGD. Another direction of research that combines SGD and the Langevin algorithm is the stochastic gradient Langevin dynamics (SGLD, Welling and Teh 2011; Vollmer, Zygalakis, and Teh 2016). However, SGLD utilizes SGD to accelerate the Langevin sampling method, while our work aims at extending SGD by using the Langevin algorithm to construct the approximate gradient.

The Underdamped Langevin Algorithm

In this section we provide some background knowledge of the underdamped Langevin algorithm, and derive a few important results that are crucial to the convergence of SAGD. At a high level, the underdamped Langevin algorithm is an approach to obtaining approximate samples from a target distribution $\pi(\xi)$. In many cases, we can only compute $\pi(\xi)$ up to some normalizing constant, *i.e.*, we have access to $V(\xi) := -\log(\pi(\xi)) + C$, where C is free of ξ. Then the *underdamped Langevin diffusion* is defined by the following stochastic differential equation (SDE) for $W(t) = (\xi^{\mathrm{T}}(t), \rho^{\mathrm{T}}(t))^{\mathrm{T}} \in \mathbb{R}^{2r}$ with $\xi(t), \rho(t) \in \mathbb{R}^r$,

$$\mathrm{d}\xi(t) = \rho(t)\mathrm{d}t, \tag{1}$$

$$\mathrm{d}\rho(t) = -\gamma\rho(t)\mathrm{d}t - \nabla V(\xi(t))\mathrm{d}t + \sqrt{2\gamma}\mathrm{d}B(t), \tag{2}$$

$$\xi(0) = \xi_0,\ \rho(0) = \rho_0,\ t \geq 0,$$

where $\gamma > 0$ is a fixed constant but can be chosen arbitrarily, and $B(t)$ is an r-dimensional Brownian motion. Under mild conditions, Proposition 6.1 of Pavliotis (2014) shows that the invariant distribution of $W(t)$ is unique, with the density function

$$\pi_W(\xi, \rho) \propto \exp\{-V(\xi) - \|\rho\|^2/2\} \propto \pi(\xi) \cdot \exp(-\|\rho\|^2/2),$$

where ρ is an auxiliary variable, and our main interest is in ξ. The form of $\pi_W(\xi, \rho)$ indicates that ξ and ρ are independent with $\xi \sim \pi(\xi)$ and $\rho \sim N(0, I_r)$. That is, if we can solve the SDE exactly, then ξ follows the target distribution in the long run.

However, in general the solution to (1) and (2) has no closed form, so some discretization methods have to be adopted. Consider the following discretized chain for $W_k = (\xi_k^{\mathrm{T}}, \rho_k^{\mathrm{T}})^{\mathrm{T}}$, $k \geq 0$:

$$\xi_{k+1} = \xi_k + \delta\rho_k, \tag{3}$$

$$\rho_{k+1} = (1 - \gamma\delta)\rho_k - \delta \cdot \nabla V(\xi_k) + \sqrt{2\gamma\delta}\eta_k, \tag{4}$$

where δ is the step size, $\{\eta_k\}_{k=0}^{\infty} \overset{iid}{\sim} N(0, I_r)$, and η_k is independent of $\{W_k\}_{i=0}^{k-1}$. The iterations (3) and (4) are typically referred to as the *underdamped Langevin algorithm*.

The importance and usefulness of the $\{W_k\}$ sample will be illustrated in Theorem 2. Before that we need to first guarantee that $\{W_k\}$ is well defined and does not explode as time goes on. Formally, we show that under some mild conditions, $\{W_k\}$ is stable in the sense that it has finite moments

of any order, uniformly in the step count k. The result is summarized in Theorem 1, along with the assumptions we need to impose.

Assumption 1. *(a) $V(x)$ is bounded from below, i.e., $V(x) \geq \nu_0$ for some constant $\nu_0 \in \mathbb{R}$ and all $x \in \mathbb{R}^r$. (b) The operator norm of the second derivative of V is bounded, i.e., $\|\nabla^2 V(x)\| \leq \nu$ for some constant $\nu > 0$ and all $x \in \mathbb{R}^r$. (c) $V(x) \in \mathscr{C}^{\infty}_{poly}$.*

For Assumption 1(a), we can assume $\nu_0 = 0$ without loss of generality. This is because we can always work on a scale-transformation of ξ, $\xi' = c\xi$, resulting in a transformed V, $V'(x) = V(x/c) + \log c$. In what follows we adopt this simplification, so that we have $V(x) \geq 0$.

Assumption 2. *There exist constants $\alpha > 0$ and $0 < \beta < 1$ such that for all $x \in \mathbb{R}^r$,*

$$\frac{1}{2}\langle \nabla V(x), x\rangle \geq \beta V(x) + \gamma^2 C_\beta \|x\|^2 - \alpha, \quad C_\beta = \frac{\beta(2-\beta)}{8(1-\beta)}.$$

Assumption 2 is a common and standard regularity condition on V coming from Mattingly, Stuart, and Higham (2002). We then have the following conclusion:

Theorem 1. *Suppose Assumptions 1 and 2 hold, and choose δ small enough such that $\delta \leq \min\{1/\gamma, \gamma/(2\nu), (D+1-\sqrt{D^2+1})/\gamma\}$, $D = \gamma^4 C_\beta/\nu^2$. Then for any fixed $l > 0$ and all $k \geq 0$, there exist constants $C = C(l,\delta) > 0$, $\lambda = \lambda(l,\delta) > 0$, and an integer $m = m(l) > 0$ such that*

$$\mathbb{E}\left(\|\xi_k\|^{2l} + \|\rho_k\|^{2l}\right) \leq C\left\{1 + \left(\|\xi_0\|^m + \|\rho_0\|^m\right) e^{-\lambda k}\right\}.$$

Next, we present the main result for the underdamped Langevin algorithm. Let $\varphi : \mathbb{R}^{2r} \to \mathbb{R}$ be a multivariate function with the notation $\varphi(w) \equiv \varphi(\xi, \rho)$, where $w = (\xi^{\mathrm{T}}, \rho^{\mathrm{T}})^{\mathrm{T}}$. Then define its expectation with respect to π_W as $\bar{\varphi} = \mathbb{E}_{\pi_W}\varphi := \int \varphi(\xi,\rho)\pi_W(\xi,\rho)\mathrm{d}\xi\mathrm{d}\rho$. It is easy to see that if $\varphi(w) = \nabla f(\theta;\xi)$, then $\bar{\varphi} = \mathbb{E}_{\pi_W}\varphi = \nabla F(\theta)$ is exactly the true gradient function we are interested in. Driven by the motivation to approximate $\bar{\varphi}$, Theorem 2 below shows that we can construct an estimator $\hat{\varphi}$ using the sequence $\{W_k\}$, where $\hat{\varphi} = K^{-1}\sum_{k=0}^{K-1}\varphi(W_k)$.

Theorem 2. *Let φ, $\bar{\varphi}$, and $\hat{\varphi}$ be defined as above, with $\varphi \in \mathscr{C}^{r+5}_{poly}$. Assume that the conditions in Theorem 1 hold. Then there exist constants $C_1 > 0$ and $C_2 > 0$ such that for any $\delta > 0$ in the range and any integer $K > 0$, we have*

$$|\mathbb{E}(\hat{\varphi}) - \bar{\varphi}| \leq C_1\left(\frac{1}{K\delta} + \delta\right),$$
$$\mathbb{E}\left[(\hat{\varphi} - \bar{\varphi})^2\right] \leq C_2\left(\frac{1}{K\delta} + \delta^2\right).$$

Theorem 2 shows that in general $\hat{\varphi}$ is a biased estimator for $\bar{\varphi}$, but its bias and mean squared error can be made arbitrarily small by appropriately choosing the algorithm parameters δ and K.

Here we make a few remarks about the results in this section. Theorem 1 is similar to Proposition 2.7 of Kopec (2015), but they use the implicit Euler scheme to discretize the Langevin SDE, which is computationally much harder. Therefore, Theorem 1 is a new result for the explicit Euler scheme given by (3) and (4). The rates in Theorem 2 are known results (Chen, Ding, and Carin 2015). However, in most prior art the assumptions to make Theorem 2 hold are highly non-trivial and very difficult to check for real machine learning models. For example, Chen, Ding, and Carin (2015) needs to assume that our conclusion in Theorem 1 holds, along with other technical conditions. In contrast, our assumptions are only made on the log-density function $V(\xi)$, which is the actual model that machine learning practitioners are given. In this sense, the results developed in this article have much broader practical use.

The benefit of our new results is that we can easily verify the assumptions for popular machine learning models. For example, the following corollary justifies the use of Langevin algorithm to sample from deep generative models (e.g. VAE). Consider a single-layer neural network $h(z) = a(Wz + b)$, where $z \in \mathbb{R}^r$, $b \in \mathbb{R}^m$, $W \in \mathbb{R}^{m\times r}$, and the activation function is $a(x) = \log(1 + e^x)$. Then we have the following result.

Corollary 1. *Assume that $Z \sim N(0, I_r)$ and $X|\{Z = z\} \sim N(h(z), \sigma^2 I)$, where σ^2 is a constant. Let $p(z|x)$ denote the conditional density of Z given $X = x$, and then $V(z) = -\log p(z|x)$ satisfies Assumptions 1 and 2.*

For brevity we omit the multi-layer case, but it can be analyzed similarly. In later part of this article we will discuss the application of SAGD to VAE model in more details.

Stochastic Approximate Gradient Descent

With the statistical properties of the underdamped Langevin algorithm studied in Theorem 2, the SAGD framework can then be readily developed. Recall that our target is to minimize the function $F(\theta) = \mathbb{E}[f(\theta;\xi)]$, whose true gradient $g(\theta) = \mathbb{E}[\nabla f(\theta;\xi)]$ is hard to compute exactly. Using the technique developed in the previous section, we can construct a stochastic gradient, $\tilde{g}(\theta) = K^{-1}\sum_{k=0}^{K-1}\nabla f(\theta;\xi_k)$, to approximate $g(\theta)$. Unlike most existing SGD settings, $\tilde{g}(\theta)$ is not an unbiased estimator for $g(\theta)$, as suggested by Theorem 2. Therefore, we refer to the optimization method based on such a $\tilde{g}(\theta)$ as the stochastic *approximate* gradient descent. The outline of SAGD is given in Algorithm 1.

Despite the fact that $\tilde{g}(\theta)$ is a biased estimator for the true gradient, we show that by carefully choosing the hyperparameters, we can actually guarantee the overall convergence of SAGD. Interestingly, the convergence rate for a convex objective function, in terms of the number of gradient updates, is the same as the vanilla SGD method with an order of $\mathcal{O}(1/\sqrt{T})$, as is shown in Theorem 3.

Assumption 3. *$f(\theta;\cdot) \in \mathscr{C}^{r+5}_{poly}$ for each $\theta \in \Theta$, and there exist a constant $C > 0$ and an integer $m \geq 0$ such that $\|\nabla^i f(\theta;\cdot)\| \leq C(1 + \|\cdot\|^m)$ for all $\theta \in \Theta$ and $0 \leq i \leq r+5$.*

Theorem 3. *Suppose that $F(\theta)$ is convex and L-Lipschitz continuous in $\theta \in \Theta$, and Θ is a closed convex set with diameter $D < \infty$. Also assume that Assumption 3 and the conditions in Theorem 1 hold. Then by choosing $\delta_t = C_1/\sqrt{t}$, $K_t = C_2 t$, and $\alpha_t = \alpha_0/\sqrt{t}$, where $C_1, C_2, \alpha_0 > 0$ are constants, we have $\mathbb{E}[F(\hat{\theta})] - F^* \leq \mathcal{O}(1/\sqrt{T})$.*

Algorithm 1: Stochastic approximate gradient descent for minimizing $F(\theta) = \mathbb{E}[f(\theta; \xi)]$

Input : T, $\{\alpha_t\}$, $\{\delta_t\}$, $\{K_t\}$, initial values θ_0, ξ_0, ρ_0
Output: Parameter estimate for θ
for $t = 0, 1, \ldots, T-1$ **do**
  $\xi_{t,0} \leftarrow \xi_0$, $\rho_{t,0} \leftarrow \rho_0$;
  for $k = 1, 2, \ldots, K_t - 1$ **do**
    $\xi_{t,k+1} \leftarrow \xi_{t,k} + \delta_t \rho_{t,k}$;
    Sample $\eta_{t,k} \sim N(0, I_r)$;
    $\rho_{t,k+1} \leftarrow$
    $(1 - \gamma\delta_t)\rho_{t,k} - \delta_t \cdot \nabla V(\xi_{t,k}) + \sqrt{2\gamma\delta_t}\eta_{t,k}$;
  $\tilde{g}_t(\theta) \leftarrow K_t^{-1} \sum_{k=0}^{K_t-1} \nabla f(\theta; \xi_{t,k})$;
  $\theta_{t+1} \leftarrow \mathcal{P}_\Theta(\theta_t - \alpha_t \cdot \tilde{g}_t(\theta_t))$;
return $\hat{\theta} = T^{-1} \sum_{t=1}^{T} \theta_t$

The significance of Theorem 3 is that it provides clear rates for the hyperparameters δ_t and K_t in the sampling algorithm, which are crucial for practical algorithm implementation but are typically missing in other MCMC-based methods. Of course, the preservation of the SGD rate is not without a price. Theorem 3 indicates that the number of inner iterations, *i.e.*, K_t in Algorithm 1, needs to increase with t. However, the developed error bounds are typically conservative, so for practical use, we advocate the following techniques to speed up SAGD: (1) An educated initial value ξ_0 can be used to initialize the Langevin algorithm, for example in VAE ξ_0 is sampled from the trained encoder; (2) A persistent Langevin Markov chain is stored during optimization, motivated by the persistent contrastive divergence (Tieleman 2008); (3) Some advanced gradient update schemes such as Adam can be used.

More generally, we consider objective functions that are nonconvex but smooth, and assume that $\Theta = \mathbb{R}^p$. Theorem 4 indicates that with a proper choice of hyperparameters, the algorithm again has a nice convergence property.

Theorem 4. *Suppose that $g(\theta)$ is G-Lipschitz continuous in θ, and assume that Assumption 3 and the conditions in Theorem 1 hold. Let $\delta_t = C_1 t^{-c}$, $K_t = C_2 t^{2c}$, and $\alpha_t = \alpha_0/t$ for some constants $0 < \alpha_0 < 1/(2G)$ and $C_1, C_2, c > 0$. Then we have $\liminf_{t\to\infty} \mathbb{E}[\|g(\theta_t)\|^2] = 0$.*

Theorem 4 is an analog to Theorem 4.9 of Bottou, Curtis, and Nocedal (2018). It is not meant to be the strongest conclusion, but to provide insights on the convergence property of SAGD for nonconvex objective functions.

Applications: EM Algorithm and VAE

Automated EM Algorithm

The SAGD framework is very useful for implementing an automated version of the EM algorithm (Dempster, Laird, and Rubin 1977). EM algorithm is a powerful and indispensable tool to solve missing data problems and latent variable models. Given the data set X that follows a probability distribution with density function $f(x; \theta)$, we are interested in computing the maximum likelihood estimator $\hat{\theta} = \arg\max_\theta \ell(\theta; x)$ for the unknown parameter vector θ, where $\ell(\theta; x) = \log[f(x; \theta)]$ is the log-likelihood function.

However, in many cases the computation of the marginal distribution $f(x; \theta)$ is intractable, but with an additional random vector U, the complete log-likelihood $L(\theta; x, u) = \log[f(x, u; \theta)]$ is simple. This phenomenon typically happens when U represent missing data or latent variables in the model. The EM algorithm computes $\hat{\theta}$ in an iterative way. Given the current value of θ, denoted by θ_k, the EM algorithm proceeds by the following two steps:

- **Expectation Step (E-step)**: Compute the expected value of $L(\theta; x, U)$ with respect to the conditional distribution of U given $X = x$ under the current parameter estimate θ_k, and define the function $Q(\theta; \theta_k) = \mathbb{E}_{U|X=x,\theta_k}[L(\theta; x, U)]$.
- **Maximization Step (M-step)**: Update the estimate of θ by maximizing the Q function: $\theta_{k+1} = \arg\max_\theta Q(\theta; \theta_k)$.

The EM algorithm has a remarkable monotonicity property, *i.e.*, the marginal log-likelihood $\ell(\theta; x)$ is always nondecreasing on the $\{\theta_k\}$ sequence. Due to such nice properties, the EM algorithm has been the standard optimization technique for Gaussian mixture models and many other missing data models. However, one serious problem of the EM algorithm is that the expectation defining the Q function usually has no simple closed form, so the Monte Carlo EM algorithm (MCEM, Wei and Tanner 1990; Levine and Casella 2001) proposes to use Monte Carlo methods to approximate the expectation. Using MCMC to approximate the expectation in the E-step is not a new idea, but what really matters is how to properly choose the hyperparameters to guarantee the convergence of the M-step.

In this sense, Theorem 3 provides a clear way to make the EM algorithm effectively automated. It is easy to see that the target distribution $\pi(\xi)$ is $p(u|x; \theta_k)$, the conditional density of U given $X = x$, which is proportional to the joint density of (X, U) under θ_k. Therefore, we can define $V(\xi) = -L(\theta_k; x, \xi)$, and then apply Algorithm 1 to directly solve the M-step, whose convergence is readily guaranteed. Finally, one only needs to create an outer loop to iteratively update the $\{\theta_k\}$ sequence, until some convergence condition is met.

Debiased VAE

The automated EM algorithm can be further used to improve the popular VAE model (Kingma and Welling 2014). VAE has the same goal of seeking the maximum likelihood estimator for θ, but it uses the variational Bayes technique to maximize a lower bound of $\ell(\theta; x)$. Let $q(u|x)$ be any conditional density function, and then VAE maximizes the function $\tilde{\ell}(\theta; x)$, defined by

$$\tilde{\ell}(\theta; x) = \mathbb{E}_{u\sim q}[\log p(x|u; \theta)] - \mathrm{KL}[q(u|x)\|p(u)], \quad (5)$$

where $p(u)$ is the marginal density of u, and $p(x|u; \theta)$ is the conditional distribution of X given $U = u$. In most VAE settings, $U \sim N(0, I)$, and $q(u|x)$ is taken to be a normal distribution whose mean and variance parameters are represented by a deep neural network. VAE has been successfully

applied to many problems, but its most critical weakness is that VAE does not maximize the exact log-likelihood, which induces a bias in the final θ.

Here we show that using the SAGD framework, the bias of VAE can be removed via an additional refining step. First, it is easy to show that $\tilde{\ell}(\theta; x)$ has another representation, $\tilde{\ell}(\theta; x) = \ell(\theta; x) - \mathrm{KL}[q(u|x) \| p(u|x; \theta)]$. That is, if the distribution $q(u|x)$ matches the true $p(u|x; \theta)$, then $\tilde{\ell}(\theta; x)$ is the genuine log-likelihood function $\ell(\theta; x)$. In this case, the objective function (5) can be optimized via an EM algorithm with a Q function $Q(\theta; \theta_k) = \mathbb{E}_{U \sim p(u|x;\theta_k)}[\log p(x|u; \theta) + \log p(u)]$, and we update the current parameter θ_k by a gradient move

$$\theta_{k+1} = \theta_k + \alpha_k \cdot [\partial Q(\theta; \theta_k)/\partial \theta]|_{\theta=\theta_k},$$

with the true expectation replaced by the Langevin-based approximate gradient. The consequence of this refining step is that we are now optimizing the true log-likelihood function $\ell(\theta; x)$ instead of the lower bound $\tilde{\ell}(\theta; x)$, and hence the bias of VAE is removed.

We emphasize that we do not position SAGD as a replacement for VAE; in fact, VAE is computationally more efficient and has a lower variance. Instead, the major virtue of SAGD is its bias-correction capacity that fixes the intrinsic gap between the evidence lower bound of VAE and the true likelihood. Therefore, we suggest using VAE to pre-train models, and then fine-tuning the generative network using SAGD due to its theoretical guarantee.

Numerical Experiments

EM Algorithm

In this section we use numerical experiments to demonstrate the applications of SAGD in EM algorithm and VAE as discussed in the previous section. First consider a simple model such that the parameter estimation procedure can be easily visualized. Assume that given latent variables $Z_1, \ldots, Z_n \overset{iid}{\sim} N(0, 1)$, the data are independently generated as $X_i | \{Z_1 = z_1, \ldots, Z_n = z_n\} \sim \mathsf{Gamma}(10 \cdot \sigma(a + bz_i))$, where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function, and $\mathsf{Gamma}(s)$ stands for a gamma distribution with shape parameter s. The target is to estimate the unknown parameters $\theta = (a, b)$ from the observed data $X_1, \ldots, X_n$. In our simulation, the true parameters are set to $a = 2$ and $b = 0.5$, and a sample size $n = 100$ is used to simulate X_i.

For a single variable pair (x, z), it is easy to show that the complete log-likelihood function is $L(\theta; x, z) = -z^2/2 + (s - 1)\log(x) - \log\{\Gamma(s)\} + C$, where $s = 10 \cdot \sigma(a + bz)$, $\Gamma(\cdot)$ is the gamma function, and C is a constant. The EM algorithm is then used to solve this problem as follows. In the k-th M-step, we fix parameter estimate at $\theta_k = (a_k, b_k)$, and then optimize the objective function $Q(\theta; \theta_k) = \mathbb{E}_{Z|X=x,\theta_k}[L(\theta; x, Z)]$ using SAGD. The next θ value is set to the optimum of $Q(\theta; \theta_k)$.

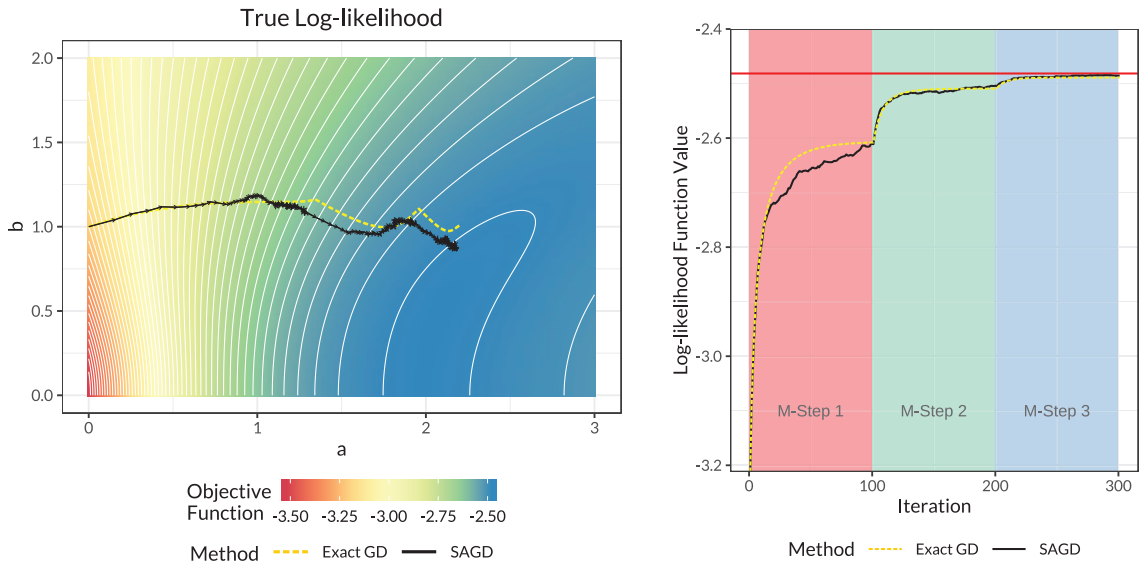
Since for this model we can evaluate the true derivatives of $Q(\theta; \theta_k)$ using numerical integration, it is of interest to compare SAGD with the exact gradient descent (GD) method. We set the initial value to be $\theta_0 = (0, 1)$, and run both SAGD and exact GD for $T = 100$ iterations in each M-step, with a constant step size $\alpha_t = 0.2$. For SAGD, Langevin parameters are specified as $\delta_t = 0.1/\sqrt{t}$ and $K_t = t + 20$, with the first 100 Langevin iterations discarded as burn-in, similar to that in MCMC. Figure 1(a) demonstrates the path of (a, b) values on the surface of the true log-likelihood function after three M-steps, and Figure 1(b) gives the log-likelihood values at each gradient update. Clearly, Figure 1 shows that the path of SAGD nicely approximates that of exact GD, which further verifies the validity of the SAGD algorithm.

Debiased VAE

In the second experiment, we use synthetic data to show that even in the simplest setting, VAE can lead to biased distribution estimation, but its bias can be effectively corrected by SAGD. The observed data are generated as follows: given independent latent variables $Z_i \sim \pi(z)$, we set $X_i = Z_i + e_i$, where $e_i \sim N(0, 1)$ is independent of Z_i, $i = 1, 2, \ldots, n$. The target is to recover the unknown latent distribution $\pi(z)$ from $X_1, \ldots, X_n$. In the VAE framework, we first represent Z by a deep neural network transformation $Z = h_\theta(U)$, where $U \sim N(0, 1)$, and then we have $p(x|u; \theta) \equiv N(h_\theta(u), 1)$. Once the neural network function h_θ has been learned, we can simulate random variates of $Z = h_\theta(U)$ by generating random $U \sim N(0, 1)$, and $\pi(z)$ is approximated by the empirical distribution of a large sample of Z. Therefore, by evaluating the quality of the Z sample, we can study the accuracy of the learned h_θ function.

In our experiment, we consider three true latent distributions and generate the corresponding data sets: (a) $\pi = N(1, 0.5^2)$; (b) an exponential distribution of mean 2; (c) a mixture of normal distributions, $\pi = 0.4 \cdot N(0, 0.5^2) + 0.6 \cdot N(3, 0.5^2)$. For each case, we first train a VAE model with 5000 iterations, and then fine-tune the neural network parameter θ by running the following four training algorithms for additional 1000 iterations: (a) VAE; (b) the importance weighted autoencoders (IWAE, Burda, Grosse, and Salakhutdinov 2016) with $k = 50$ importance samples; (c) Hamiltonian Monte Carlo (HMC) to approximate the true gradient; (d) SAGD. In HMC we use the same step size and chain length as SAGD, and run $L = 5$ leapfrog steps to get each proposal. After training is finished, we simulate random variates of Z, and compare its empirical distribution $\hat{\pi}$ with the true latent distribution π. The Kolmogorov–Smirnov distance and 1-Wasserstein distance between $\hat{\pi}$ and π are computed. Figure 2 shows the data distribution, true latent distribution π, and the estimated $\hat{\pi}$ in each setting, based on one simulated data set of sample size $n = 1000$.

Figure 2 reflects the following remarkable results. For the normal case, VAE has little bias, since the true conditional distribution $p_{U|X}(u|x)$ is indeed normal, which is well characterized by the encoder. However, in other two cases, neither the latent distribution $\pi(z)$ nor $p_{U|X}(u|x)$ is normal, and hence VAE gives highly biased estimates for π. For the three debiasing methods, the refining steps indeed reduce the bias of VAE. However, the debiasing effect of HMC is smaller than that of SAGD, even though theoretically they are similar. HMC also takes more computing time due to the



(a) (b)

Figure 1: (a) The paths of (a, b) on the surface of the true log-likelihood function. (b) The log-likelihood function value versus the number of gradient updates. The horizontal line on the top stands for the maximum log-likelihood value.

leapfrog steps and the calculation of acceptance probability. For IWAE, it tends to overly truncate the support of the distribution and exaggerate the density of modes. Overall, SAGD provides the most favorable bias reduction results.

Table 1: Mean and standard errors (in parentheses) of Kolmogorov–Smirnov distance (D) and 1-Wasserstein distance (W) between $\hat{\pi}$ and π across replications.

		VAE	IWAE	HMC	SAGD
Normal	D	0.033 (0.0024)	0.043 (0.0035)	0.032 (0.0019)	0.033 (0.0020)
	W	0.052 (0.0040)	0.066 (0.0047)	0.050 (0.0032)	0.052 (0.0035)
Exp(2)	D	0.095 (0.0018)	0.059 (0.0039)	0.085 (0.0016)	0.064 (0.0029)
	W	0.226 (0.0079)	0.125 (0.0091)	0.161 (0.0051)	0.115 (0.0082)
Mixture	D	0.127 (0.0027)	0.098 (0.0061)	0.104 (0.0015)	0.085 (0.0019)
	W	0.320 (0.0025)	0.165 (0.0059)	0.276 (0.0021)	0.197 (0.0031)
High-Dim.	D	0.093 (0.0010)	0.080 (0.0012)	0.092 (0.0008)	0.065 (0.0005)
	W	0.222 (0.0027)	0.158 (0.0029)	0.212 (0.0030)	0.093 (0.0015)

To further take into account the randomness in data generation, we simulate 30 replications of the data set in each setting, and compute the mean and standard errors of the distance metrics, shown in Table 1. Also included in this table is a high-dimensional data set with sample size $n = 10000$ and dimension $p = 100$: each dimension independently follows an exponential distribution with mean 2. We pre-train this data set using VAE for 10000 epochs, and then fit each method with 1000 more epochs. The mean and standard errors are computed over all dimensions. Both Figure 2 and Table 1 indicate that SAGD provides good bias reduction results for VAE in both simple and high-dimensional settings, which highlights the importance of optimizing the correct objective function in model fitting.

Generative Model for MNIST Data

In the last experiment, we consider the MNIST handwritten digits data set, and fit generative models on it. The dimension of the latent space is set to 20, and the generative network is a combination of convolutional filters and fully-connected layers. We first train a VAE model for 500 epochs with a batch size of 200, and then run SAGD for 100 epochs for fine-tuning. In SAGD, twenty independent chains are used to compute the approximate gradient, each with five burn-in's.

Since SAGD basically refines the generative network of VAE, we can directly compare their output images. We randomly generate 100 digits from the trained VAE model and the debiased model, respectively, and in Figure 3 we show some representative pairs of generated digits, with VAE-trained ones on the top, and SAGD-refined ones on the bottom. It is clear that the SAGD refining step improves the quality of the generated images. For example, in the first column of Figure 3, VAE shows an ambiguous digit between "9" and "7", but the refined one is a definite "7".

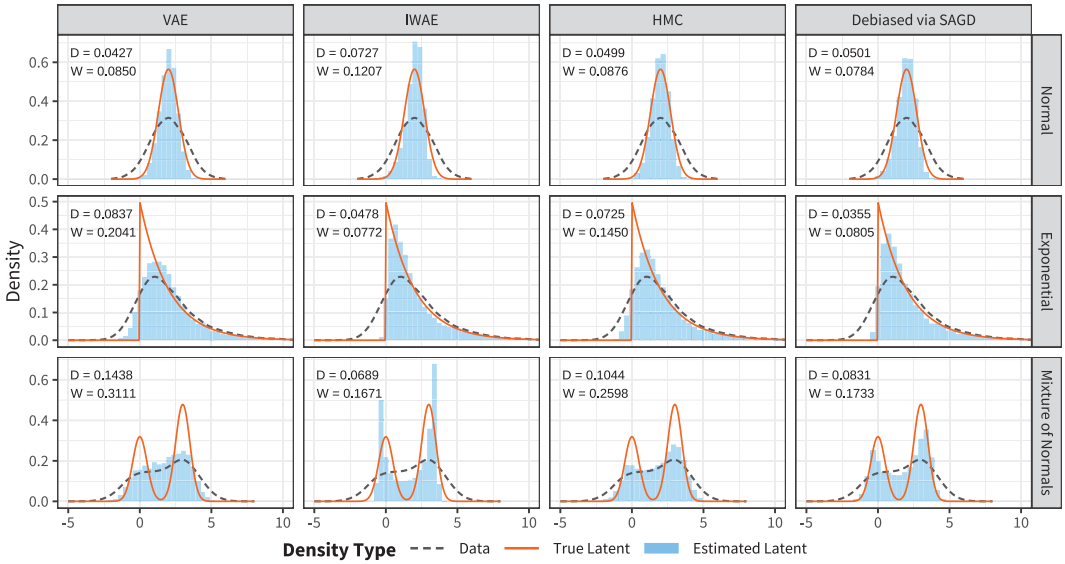


Figure 2: A demonstration of the data distribution (black dashed curves), true latent distribution ($\pi(z)$, red solid curves), and estimated latent distributions ($\hat{\pi}(z)$, blue histograms). Each row in the plot matrix corresponds to a true latent density setting. The text on the top-left corner of each plot gives the Kolmogorov–Smirnov distance (D) and 1-Wasserstein distance (W) between π and $\hat{\pi}$.

Figure 3: Representative examples from randomly generated digits that show significant improvement after the refining step using SAGD.

Conclusion

In this article we have developed the SAGD framework for optimizing objective functions that can be expressed as a mathematical expectation with intractable gradients. SAGD uses the Langevin algorithm to construct an approximate gradient in each iteration, whose accuracy is carefully controlled. Theoretical analysis shows that SAGD has the same convergence property as SGD, and more importantly, all the hyperparameters of SAGD are transparent so that the algorithm can be practically implemented. We have successfully applied SAGD to both the automated EM algorithm and the debiased VAE. To summarize, SAGD is an alternative to the ordinary SGD in a broader realm, and it is hoped that SAGD can be used to solve more statistical and machine learning problems both efficiently and reliably.

We mention two future directions for the research on SAGD. First, one might be interested in improving the Langevin algorithm, as the assumptions we have made are mild yet not the weakest. A second direction is to study the convergence of SAGD combined with various acceleration techniques, such as the momentum methods.

References

Bottou, L.; Curtis, F. E.; and Nocedal, J. 2018. Optimization methods for large-scale machine learning. *SIAM Review* 60(2):223–311.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer. 177–186.

Brooks, S.; Gelman, A.; Jones, G.; and Meng, X.-L. 2011. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC.

Burda, Y.; Grosse, R. B.; and Salakhutdinov, R. 2016. Importance weighted autoencoders. In *4th International Conference on Learning Representations*.

Chen, C.; Ding, N.; and Carin, L. 2015. On the convergence of stochastic gradient mcmc algorithms with high-order integrators. In *Advances in Neural Information Processing Systems 28*, 2278–2286.

Cheng, X., and Bartlett, P. L. 2018. Convergence of langevin mcmc in kl-divergence. In *Proceedings of Algorithmic Learning Theory*, volume 83 of *Proceedings of Machine Learning Research*, 186–211.

Cheng, X.; Chatterji, N. S.; Abbasi-Yadkori, Y.; Bartlett, P. L.; and Jordan, M. I. 2018a. Sharp convergence rates for langevin dynamics in the nonconvex setting. *arXiv preprint arXiv:1805.01648*.

Cheng, X.; Chatterji, N. S.; Bartlett, P. L.; and Jordan, M. I. 2018b. Underdamped langevin mcmc: A non-asymptotic analysis. In Bubeck, S.; Perchet, V.; and Rigollet, P., eds., *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, 300–323.

Dalalyan, A. S. 2017. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 79(3):651–676.

De Bortoli, V.; Durmus, A.; Pereyra, M.; and Vidal, A. F. 2019. Efficient stochastic optimisation by unadjusted langevin monte carlo. application to maximum marginal likelihood and empirical bayesian estimation. *arXiv preprint arXiv:1906.12281*.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 39(1):1–38.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Durmus, A., and Moulines, É. 2017. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability* 27(3):1551–1587.

Durmus, A., and Moulines, É. 2019. High-dimensional bayesian inference via the unadjusted langevin algorithm. *Bernoulli* 25(4A):2854–2882.

Geman, S., and Geman, D. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(6):721–741.

Gilks, W.; Richardson, S.; and Spiegelhalter, D. 1995. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC.

Han, T.; Nijkamp, E.; Fang, X.; Hill, M.; Zhu, S.-C.; and Wu, Y. N. 2019. Divergence triangle for joint training of generator model, energy-based model, and inference model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 8670–8679.

Hastings, W. K. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika* 57(1):97–109.

Johnson, R., and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, 315–323.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 1–13.

Kingma, D. P., and Welling, M. 2014. Stochastic gradient vb and the variational auto-encoder. In *2nd International Conference on Learning Representations*.

Kopec, M. 2015. Weak backward error analysis for langevin process. *BIT Numerical Mathematics* 55(4):1057–1103.

Levine, R. A., and Casella, G. 2001. Implementations of the monte carlo em algorithm. *Journal of Computational and Graphical Statistics* 10(3):422–439.

Luo, L.; Xiong, Y.; Liu, Y.; and Sun, X. 2019. Adaptive gradient methods with dynamic bound of learning rate. In *7th International Conference on Learning Representations*.

Ma, Y.-A.; Chatterji, N.; Cheng, X.; Flammarion, N.; Bartlett, P.; and Jordan, M. I. 2019. Is there an analog of nesterov acceleration for mcmc? *arXiv preprint arXiv:1902.00996*.

Mattingly, J. C.; Stuart, A. M.; and Higham, D. J. 2002. Ergodicity for sdes and approximations: locally lipschitz vector fields and degenerate noise. *Stochastic Processes and their Applications* 101(2):185–232.

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; and Teller, E. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21(6):1087–1092.

Pavliotis, G. A. 2014. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*, volume 60. Springer.

Reddi, S. J.; Hefny, A.; Sra, S.; Poczos, B.; and Smola, A. 2016. Stochastic variance reduction for nonconvex optimization. In *Proceedings of the 33nd International Conference on Machine Learning*, 314–323.

Robbins, H., and Monro, S. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 22(3):400–407.

Roberts, G. O., and Stramer, O. 2002. Langevin diffusions and metropolis-hastings algorithms. *Methodology and Computing in Applied Probability* 4(4):337–357.

Roberts, G. O., and Tweedie, R. L. 1996. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli* 2(4):341–363.

Tieleman, T. 2008. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, 1064–1071.

Vollmer, S. J.; Zygalakis, K. C.; and Teh, Y. W. 2016. Exploration of the (non-) asymptotic bias and variance of stochastic gradient langevin dynamics. *The Journal of Machine Learning Research* 17(1):5504–5548.

Wei, G. C., and Tanner, M. A. 1990. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American statistical Association* 85(411):699–704.

Welling, M., and Teh, Y. W. 2011. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 681–688.

Xie, J.; Lu, Y.; Gao, R.; and Wu, Y. N. 2018. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.