Faster Derivative-Free Stochastic Algorithm for Shared Memory Machines

Bin Gu [1] **Zhouyuan Huo** [1] **Cheng Deng** [2] **Heng Huang** [1 2]

Abstract

Asynchronous parallel stochastic gradient optimization has been playing a pivotal role to solve large-scale machine learning problems in big data applications. Zeroth-order (derivative-free) methods estimate the gradient only by two function evaluations, thus have been applied to solve the problems where the explicit gradient calculations are computationally expensive or infeasible. Recently, the first asynchronous parallel stochastic zeroth-order algorithm (AsySZO) was proposed. However, its convergence rate is $O(\frac{1}{\sqrt{T}})$ for the smooth, possibly non-convex learning problems, which is significantly slower than $O(\frac{1}{T})$ the best convergence rate of (asynchronous) stochastic gradient algorithm. To fill this gap, in this paper, we first point out the fundamental reason leading to the slow convergence rate of AsySZO, and then propose a new asynchronous stochastic zeroth-order algorithm (AsySZO+). We provide a faster convergence rate $O(\frac{1}{bT})$ (b is the mini-batch size) for AsySZO+ by the rigorous theoretical analysis, which is a significant improvement over $O(\frac{1}{\sqrt{T}})$. The experimental results on the application of ensemble learning confirm that our AsySZO+ has a faster convergence rate than the existing (asynchronous) stochastic zeroth-order algorithms.

1. Introduction

In the current big data era, the asynchronous parallel algorithms for stochastic optimization have achieved huge successes in machine learning community. Most of the asynchronous parallel stochastic algorithms are built on the first-order gradient or second-order information (*e.g.* Hessian matrix or its approximations) of the objective function. For example, Hogwild! (Recht et al., 2011) (the first asynchronous parallel stochastic gradient descent, SGD, algorithm) uses the first-order gradient to update the solution for smooth convex functions. Other variational asynchronous parallel SGD algorithms (Mania et al., 2015; Lian et al., 2015; Gu et al., 2018a; Zhao & Li, 2016; Gu et al., 2018b) also use the first-order derivative to update the solution for smooth convex or non-convex problems. The second-order information (*e.g.* approximated Hessian matrix) (Byrd et al., 2016) was also used to accelerate the optimization. To sum up, the asynchronous parallel stochastic gradient optimization has been playing a pivotal role for handling large-scale machine learning problems.

However, for a large number of machine learning problems, calculating the explicit gradient is computationally expensive. Specifically, as mentioned in (Wainwright & Jordan, 2008; Taskar et al., 2005), it is difficult to compute the explicit gradients for the objective functions of graphical model inference (Wainwright & Jordan, 2008) and structured-prediction (Taskar et al., 2005). Even worse, for several machine learning applications, the explicit gradient calculations are infeasible. For example, in bandit problems (Bubeck et al., 2012) and black box ensemble learning problems (Lian et al., 2016), it is infeasible to get the explicit gradients of the objective functions because only the observations of function values are available. In the above scenarios, zeroth-order (also called derivative-free) method is the best and only choice of the optimization, because zeroth-order method estimates the gradient only by two function evaluations. Thus, zeroth-order optimization is important and attracts a lot of attentions in recent machine learning research.

As discussed above, designing the asynchronous stochastic zeroth-order algorithms is crucial and desired for large-scale machine learning problems. However, existing works (Nesterov & Spokoiny, 2011; Ghadimi & Lan, 2013; Duchi et al., 2012; Bach & Perchet, 2016) mainly focus on the non-parallel (*i.e.*, sequential) stochastic zeroth-order algorithms (SZO). These works are summarized in Table 1. Specifically, Nesterov & Spokoiny (2011) first proposed SZO algorithm for convex problems, and proved the convergence rate

[1]Department of Electrical and Computer Engineering, University of Pittsburgh, PA, USA [2]School of Electronic Engineering, Xidian University, Xi'an, Shaanxi, China. Correspondence to: Heng Huang <henghuanghh@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Table 1. Representative zeroth-order stochastic algorithms. (S, NS, C and PNC are the abbreviations of smooth, non-smooth, convex and possibly non-convex, respectively. T is the iteration number, b is the mini-batch size, and β is the degree of smoothness of a function.)

Algorithm	Reference	Problem	Asynchronous	Accelerated	Mini-batch	Convergence rate
SZO	Nesterov & Spokoiny (2011)	C	No	No	No	$O(\frac{1}{\sqrt{T}})$
SZO	Ghadimi & Lan (2013)	S (PNC)	No	No	No	$O(\frac{1}{\sqrt{T}})$
SZO	Duchi et al. (2012)	S+NS & C	No	No	No	$O(\frac{1}{T}+\frac{1}{\sqrt{T}})$
SZO	Bach & Perchet (2016)	S & C	No	No	No	$O(\frac{1}{T})^{\frac{\beta-1}{2\beta}}, \beta>2$
AsySZO	Lian et al. (2016)	S (PNC)	Yes	No	No	$O(\frac{1}{T}+\frac{1}{\sqrt{T}})$
AsySZO+	Our	S (PNC)	Yes	Yes	Yes	$O(\frac{1}{bT})$

$O(\frac{1}{\sqrt{T}})$. Ghadimi & Lan (2013) proved the convergence rate of SZO as $O(\frac{1}{\sqrt{T}})$ for the smooth, possibly non-convex problems. Duchi et al. (2012) proposed the SZO algorithm for smooth and non-smooth convex problems, and proved the convergence rate $O(\frac{1}{T}+\frac{1}{\sqrt{T}})$. Bach & Perchet (2016) used the degree of smoothness to prove the convergence rate of SZO. They obtained the convergence rate $O(\frac{1}{T})^{\frac{\beta-1}{2\beta}}$ for the smooth and convex problems, where β is the degree of smoothness of a function.

To the best of our knowledge, the only work of asynchronous stochastic zeroth-order algorithm (AsySZO) was recently proposed in (Lian et al., 2016). Lian et al. (2016) proved that the convergence rate of AsySZO is $O(\frac{1}{T}+\frac{1}{\sqrt{T}})$ for the smooth, possibly non-convex problems, which basically can be viewed as $O\left(\frac{1}{\sqrt{T}}\right)$ because the term $\frac{1}{\sqrt{T}}$ dominates $\frac{1}{T}+\frac{1}{\sqrt{T}}$. As summarized in Table 1, the convergence rates of AsySZO and SZO (Nesterov & Spokoiny, 2011; Ghadimi & Lan, 2013; Duchi et al., 2012; Bach & Perchet, 2016) are of the same order of or lower than $O\left(\frac{1}{\sqrt{T}}\right)$ for the smooth functions without strongly convex assumption. The convergence rates of the state-of-the-art (asynchronous) stochastic gradient algorithms (Reddi et al., 2016; Lian et al., 2015; Gu et al., 2018a) are $O(\frac{1}{T})$ for the smooth, possibly non-convex problems, and $O(\frac{1}{bT})$ for mini-batch version with the mini-batch size of b, which are significantly faster than the convergence rate $O\left(\frac{1}{\sqrt{T}}\right)$ for AsySZO and SZO. Due to the deviation of the stochastic gradient induced by the zeroth-order approximation, and the inconsistent reading induced by asynchronous parallel computation, there is still no fast asynchronous stochastic zeroth-order algorithm with the guaranteed convergence rate $O(\frac{1}{T})$ or $O(\frac{1}{bT})$.

To fill this important gap, we first point out the fundamental reason leading to the slow convergence rate of AsySZO. After that, we propose a new asynchronous stochastic zeroth-order optimization algorithm, called as AsySZO+, and also show that our new AsySZO+ algorithm can achieve a faster convergence rate $O(\frac{1}{bT})$ (b is the mini-batch size) with rigorous theoretical analysis, which is a significant improvement over the convergence rate $O\left(\frac{1}{\sqrt{T}}\right)$ of AsySZO. To the best of our knowledge, AsySZO+ is the first asynchronous stochastic zeroth-order algorithm with the guaranteed convergence rate $O(\frac{1}{bT})$. The experimental results on the application of ensemble learning confirm that our AsySZO+ has a faster convergence rate than the existing (asynchronous) stochastic zeroth-order algorithms. Our preliminary theoretical results were published in arXiv.org (Gu et al., 2016).

Notations. To make the paper easier to follow, we provide the following notations:

- x denotes the vector data in the shared memory. If reading the vector x from the shared memory to the local memory, which is denoted as $\widehat{x}$.
- e_j is the N-dimensional zero vector except that the j-th coordinate is 1.
- $\nabla_j f(x)$ is the j-th coordinate of the gradient $\nabla f(x)$.
- $\{\mu_j\}_{j=1\ldots,N}$ are the approximate parameters for computing the zeroth-order gradient.
- $\{\gamma_t\}_{t=0,\ldots,m-1}$ are the decreasing steplengths in AsySZO, and γ is the steplength in AsySZO+.
- T, m and S are the total number of iterations, the number of iterations in the inner loop, and the number of iterations in the outer loop, respectively.
- Y is the size of the coordinate set J.
- L_0, L and $\widetilde{L}$ are the Lipschitz constants for $f_i(x)$, $\nabla f_i(x)$ and $\sum_{j=1}^N \nabla_j f_i^j(x)$ respectively.

Organization. We organize the rest of the paper as follows. In Section 2, we propose our new AsySZO+ algorithm. In Section 3, we prove the convergence rate of AsySZO+. In Section 4, we show the experimental results. Finally, we give concluding remarks in Section 5.

2. Faster Asynchronous Stochastic Zeroth-Order Algorithm

In this section, we first introduce the problem addressed in this paper, and then give a brief review of AsySZO algorithm. After that, we analyze the reason leading to the slow convergence rate of AsySZO. To address this challenging problem, we propose our new AsySZO+ algorithm.

2.1. Problem Statement

In theory, optimizing an expected risk minimization problem is the ultimate goal of many machine learning problems. However, due to the fact that the distribution of samples is unknown, optimizing the expected risk minimization problem is challenging. Because a huge number of samples are available in the current big data era, instead of optimizing the expected risk minimization problem, we usually optimize a finite-sum problem with deterministic functions as following:

$$\min_{x\in\mathbb{R}^N} f(x) = \frac{1}{l}\sum_{i=1}^{l} f_i(x)\,, \tag{1}$$

where each $f_i : \mathbb{R}^N \to \mathbb{R}$ is considered as a smooth, possibly non-convex function in this paper. Obviously, the empirical risk minimization problem is a special case of the problem (1). In addition to the empirical risk minimization problem, problem (1) summarizes an extensive number of important regularized learning problems, such as, ℓ_2-regularized logistic regression (Conroy & Sajda, 2012), ridge regression (Shen et al., 2013), least squares SVM (Suykens & Vandewalle, 1999) and so on.

2.2. Brief Review of AsySZO

To the best of our knowledge, AsySZO (Lian et al., 2016) is the first and only work of asynchronous stochastic zeroth-order algorithm, which is designed for the parallel environment with the shared memory, such as multi-core processors and GPU-accelerators. Specifically, AsySZO repeats the following three steps concurrently for each thread without any lock. First, AsySZO reads the vector x from the shared memory to the local memory. After that, AsySZO randomly chooses a component function f_i and a set of coordinates $J \subseteq \{1, ..., N\}$, and locally computes $G_J(\widehat{x}_t; f_i)$ by the zeroth-order method as an approximate estimation of the gradient $\nabla_J f_i(\widehat{x}_t)$:

$$\begin{aligned} & G_J(\widehat{x}_t; f_i) \\ = & \sum_{j\in J} \frac{N}{2Y\mu_j}\left(f_i(\widehat{x}_t + \mu_j e_j) - f_i(\widehat{x}_t - \mu_j e_j)\right) e_j\,, \end{aligned} \tag{2}$$

where μ_j is the approximate parameter for the j-th coordinate. Finally, AsySZO updates the set of coordinates J of the vector x in the shared memory by $(x_{t+1}^{s+1})_J \leftarrow \left((x_t^{s+1}) - \gamma_t G_J(\widehat{x}_t^{s+1}; f_i)\right)_J$, where γ_t is the steplength at the t-th iteration. The detailed description of AsySZO is summarized in Algorithm 1.

Algorithm 1 Asynchronous Stochastic Zeroth-order Optimization (AsySZO)

Input: $\{\gamma_t\}_{t=0,\dots,m-1}$, $\{\mu_j\}_{j=1\dots,N}$, T, and Y.
Output: x_m.
1: Initialize $x_0 \in \mathbb{R}^N$.
2: *For each thread*, do:
3: **for** $t = 0, 1, 2, \dots, T-1$ **do**
4: Randomly select a component function f_i from $\{1, ..., l\}$ with equal probability.
5: Randomly choose a set of coordinates $J(t)$ of size Y from $\{1, ..., n\}$ with equal probability.
6: $(x_{t+1})_{J(t)} \leftarrow \left((x_t) - \gamma_t G_{J(t)}(\widehat{x}_t; f_i)\right)_{J(t)}$.
7: **end for**

2.3. Slow Convergence Rate of AsySZO

For the convergence of AsySZO, Lian et al. (2016) gave the following conclusion.

Theorem 1. *If we appropriately set the steplengths* $\{\gamma_t\}_{t=0,\dots,m-1}$ *for AsySZO, we have*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left\|\nabla f(x_t)\right\|^2 \le O(\frac{1}{T} + \frac{1}{\sqrt{T}}) + L^2\sum_{j=1}^{N}\mu_j^2\,. \tag{3}$$

According to Theorem 1, the convergence rate of AsySZO algorithm is $O(\frac{1}{T} + \frac{1}{\sqrt{T}})$, which is dominated by $O\left(\frac{1}{\sqrt{T}}\right)$. The convergence rate $O\left(\frac{1}{\sqrt{T}}\right)$ is significantly slower than $O(\frac{1}{T})$, which is the best convergence rate of (asynchronous) stochastic gradient algorithm. In the following, we will analyze the major reasons leading to the slow convergence rate of AsySZO.

First, we consider the zeroth-order approximation. As mentioned before, $G_J(x; f_i)$ is a zeroth-order approximation of $\nabla_J f_i(\widehat{x}_t)$. Thus, there would exist a deviation between $G_J(x; f_i)$ and $\nabla_J f_i(\widehat{x}_t)$ which is inherent from the zeroth-order approximation with the approximate parameters μ_j. According to Theorem 1, the zeroth-order approximation does not affect the convergence rate of AsySZO, which only makes the deviation of objective functions of (1) with $L^2\sum_{j=1}^{N}\mu_j^2$.

Second, we consider the stochastic approximation. $G_J(x; f_i)$ is a stochastic approximation of the full zeroth-order gradient $G_J(x; f)$. Thus, $G_J(x; f_i)$ has a large variance due to the random sampling similar to the SGD algorithm (Recht et al., 2011). To handle the large variance, AsySZO algorithm introduces a decreasing steplength

sequence $\{\gamma_t\}_{t=0,\ldots,m-1}$ to guarantee the convergence of AsySZO, which finally leads to a slow convergence rate.

Thus, the large variance of $G_J(x;f_i)$ is the fundamental reason leading to the slow convergence rate of AsySZO algorithm.

2.4. New AsySZO+ Algorithm

As mentioned above, the fundamental reason of the slow convergence of AsySZO is the large variance of $G_J(x;f_i)$. To speed up AsySZO, we use the techniques of variance reduction (SVRG) (Johnson & Zhang, 2013) and mini-batch (Li et al., 2014) to reduce the variance of stochastic zeroth-order gradients $G_J(x;f_i)$, and propose a faster asynchronous stochastic zeroth-order algorithm (AsySZO+).

Following the SVRG framework (Johnson & Zhang, 2013), AsySZO+ has two-layer loops. The outer layer is to parallelly compute the full zeroth-order gradient[1] $G(x^s;f) = \frac{1}{l}\sum_{i=1}^{l} G(x^s;f_i)$, where the superscript s denotes the s-th outer loop. The inner layer is parallelly repeating the following three steps:

1. **Read:** Read the vector x from the shared memory to the local memory without reading lock. We use $\widehat{x}_t^{s+1}$ to denote the value in the local memory, where the subscript t denotes the t-th inner loop.

2. **Compute:** Randomly choose a mini-batch $\mathcal{B}(t)$ of the component functions, and a set of coordinates $J(t)$ from $\{1,...,N\}$, and locally compute $\widehat{v}_{J(t)}^{s+1} = \frac{1}{|\mathcal{B}(t)|}\sum_{i\in\mathcal{B}(t)} G_{J(t)}(\widehat{x}_t^{s+1};f_i) - \frac{1}{|\mathcal{B}(t)|}\sum_{i\in\mathcal{B}(t)} G_{J(t)}(\widetilde{x}^s;f_i) + G_{J(t)}(\widetilde{x}^s;f)$ [2] to reduce the variance of stochastic zeroth-order gradients.

3. **Update:** Update the set of coordinates $J(t)$ of the vector x in the shared memory as $(x_{t+1}^{s+1})_{J(t)} \leftarrow \left((x_t^{s+1}) - \gamma\widehat{v}_{J(t)}^{s+1}\right)_{J(t)}$ without writing lock.

The detailed description of AsySZO+ is summarized in Algorithm 2.

Note that $\widehat{v}_{J(t)}^{s+1}$ computed locally is an approximation of the full zeroth-order gradient $G_J(\widehat{x}_t^{s+1};f)$. It is easy to verify that the expectation of $\widehat{v}_{J(t)}^{s+1}$ on $\mathcal{B}(t)$ is equal to $G_J(\widehat{x}_t^{s+1};f)$, *i.e.*, $\mathbb{E}_{\mathcal{B}(t)}\widehat{v}_{J(t)}^{s+1} = G_{J(t)}(\widehat{x}_t^{s+1};f)$. Thus,

[1]Given p threads, we partition the whole datasets into p parts equally, and each thread computes the gradient on each subset. Finally, we merge all the p results to get the full zeroth-order gradient.

[2]The space complexity of computing $G_J(\widetilde{x}^s;f)$ is $O(N)$, where N is the dimensionality of x. Specifically, we keep the zeroth-order gradient $G(\widetilde{x}^s;f)$ in memory. For any J, we select coordinates indexed by J from $G(x,f)$.

Algorithm 2 New Asynchronous Stochastic Zeroth-Order Algorithm with Variance Reduction and Mini-Batch (AsySZO+)

Input: m, S, γ, $\{\mu_j\}_{j=1\ldots,N}$, and Y.
Output: x^S.

```
Initialize x^0 ∈ R^N.
for s = 0, 1, 2, ..., S − 1 do
    x̃^s ← x^s
    All threads parallelly compute the full approximate gradient G(x̃^s; f) = Σ_{i=1}^{l} (1/l) G(x̃^s; f_i)
    For each thread, do:
    for t = 0, 1, 2, ..., m − 1 do
        Randomly sample a mini-batch B(t) from {1, ..., l} with equal probability.
        Randomly choose a set of coordinates J(t) of size Y from {1, ..., n} with equal probability.
        Compute v̂_{J(t)}^{s+1} = (1/|B(t)|) Σ_{i∈B(t)} G_{J(t)}(x̂_t^{s+1}; f_i) −
            (1/|B(t)|) Σ_{i∈B(t)} G_{J(t)}(x̃^s; f_i) + G_{J(t)}(x̃^s; f).
        (x_{t+1}^{s+1})_{J(t)} ← ((x_t^{s+1}) − γ v̂_{J(t)}^{s+1})_{J(t)}.
    end for
    x^{s+1} ← x_m^{s+1}
end for
```

$\widehat{v}_{J(t)}^{s+1}$ is called a stochastic approximation of the full zeroth-order gradient $G_{J(t)}(\widehat{x}_t^{s+1};f)$, and $\widehat{v}_{J(t)}^{s+1}$ has smaller variance than $G_J(x;f_i)$. More importantly, we provide an upper bound to $\sum_{t=0}^{m-1}\mathbb{E}\left\|\widehat{v}_t^{s+1}\right\|^2$ in Lemma 2. The upper bound shows that $\widehat{v}_{J(t)}^{s+1}$ would vanish after a large number of iterations. Thus, we do not use a decreasing steplength sequence $\{\gamma_t\}_{t=0,\ldots,m-1}$ to guarantee the convergence of AsySZO+. In our AsySZO+, γ is set as a fixed constant, which is totally different to AsySZO. The detailed setting of γ is given in Theorem 2. The differences between AsySZO and AsySZO+ are summarized in Table 1.

3. Faster Convergence Rate

In this section, we prove the convergence rate of AsySZO+ (Theorem 2) which improves the convergence rate of asynchronous stochastic zeroth-order algorithm from $O(\frac{1}{\sqrt{T}})$ to $O(\frac{1}{bT})$. To the best of our knowledge, this is the first work to achieve the convergence rate $O(\frac{1}{bT})$ for the asynchronous stochastic zeroth-order algorithm.

We first give the basic Lipschitz smooth assumptions, and then discuss two major difficulties for proving the convergence rate of AsySZO+. Finally, we prove the convergence rate of AsySZO+.

3.1. Lipschitz Smooth Assumptions

We give the Lipschitz smooth assumptions (*i.e.*, Assumptions 1 and 2) for $f_i(x)$ and $\nabla f_i(x)$, which is widely used in the optimization analysis (Mania et al., 2015; Lian et al., 2015; Gu et al., 2018a; Zhao & Li, 2016).

Assumption 1 (Lipschitz constant for $f_i(x)$)**.** *L_0 is the Lipschitz constant for f_i ($\forall i \in \{1, \cdots, l\}$) in (1). $\forall x$ and $\forall y$, we have that*

$$|f_i(x) - \nabla f_i(y)| \leq L_0 \|x - y\| \tag{4}$$

Assumption 2 (Lipschitz constant for $\nabla f_i(x)$)**.** *L is the Lipschitz constant for $\nabla f_i(x)$ ($\forall i \in \{1, \ldots, l\}$) in (1). Thus, for all x and y, L-Lipschitz smooth can be presented as*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L \|x - y\| \tag{5}$$

Equivalently, L-Lipschitz smooth can also be written as the formulation (6).

$$f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{L}{2} \|x - y\|^2 \tag{6}$$

3.2. Difficulties for Convergence Rate Analysis

There are two major difficulties for proving the convergence rate of AsySZO+. One is the deviation of the gradient induced by the zeroth-order approximation. Another is the inconsistent reading induced by the asynchronous parallel computation.

3.2.1. Deviation of Zeroth-Order Gradient

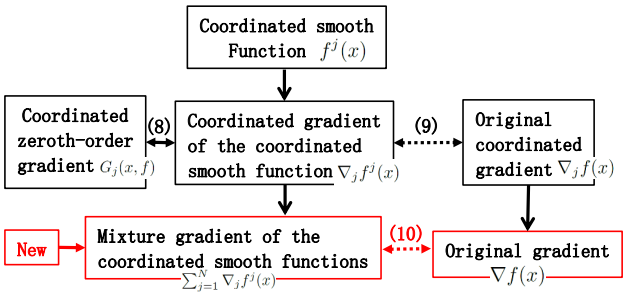


Figure 1. Coordinated smooth function $f^j(x)$ bridges between zeroth-order gradient $G_j(x, f)$ and original gradient $\nabla f(x)$. The mixture gradient $\sum_{j=1}^N \nabla_j f_i^j(x)$ of the coordinated smooth functions is new in our AsySZO+.

To address the difficulty of the deviation on the zeroth-order gradient, we introduce a coordinated smooth function $f^j(x)$ to bridge between zeroth-order gradient and original gradient (see Figure 1). The definition of coordinated smooth function $f^j(x)$ is presented as follows.

Definition 1 (Coordinated Smooth Function)**.** *Given a function $f(x)$ and predefined approximation parameters $\{\mu_j\}_{j=1,\ldots,N}$, the coordinated smooth function $f^j(x)$ is an average smoothing version of $f(x)$ w.r.t. the j-th dimension via a window size $2\mu_j$ as following.*

$$\begin{aligned} f^j(x) &= \mathbb{E}_{v \sim U_{[-\mu_j, \mu_j]}} f(x + v e_j) \\ &= \frac{1}{2\mu_j} \int_{-\mu_j}^{\mu_j} f(x + v e_j) dv \end{aligned} \tag{7}$$

where $v \sim U_{[-\mu_j, \mu_j]}$ means that v follows the uniform distribution over the interval $[-\mu_j, \mu_j]$.

The relationship between the coordinated zeroth-order gradient $G_j(x, f)$ and the coordinated gradient of the coordinated smooth function $\nabla_j f^j(x)$ is obtained as follows (see Figure 1):

$$\begin{aligned} \nabla_j f^j(x) &= \frac{1}{2\mu_j} \int_{-\mu_j}^{\mu_j} \nabla_j f(x + v e_j) dv \\ &= \frac{e_j}{2\mu_j} \left(f(x + \mu_j e_j) - f(x - \mu_j e_j) \right) \\ &= N G_j(x, f) \end{aligned} \tag{8}$$

The relationship between $\nabla_j f^j(x)$ and $\nabla_j f(x)$ is provided in (9) (see Figure 1) which is proved in (26) of (Lian et al., 2016).

$$\mathbb{E}_j \|\nabla_j f^j(x) - \nabla_j f(x)\| \leq \frac{L^2 \sum_{j=1}^N \mu_j^2}{4N} \stackrel{\text{def}}{=} \frac{\omega}{4} \tag{9}$$

where $\omega = \frac{L^2 \sum_{j=1}^N \mu_j^2}{N}$.

To prove the convergence rate of AsySZO+, we define a mixture gradient $\sum_{j=1}^N \nabla_j f^j(x)$ in Definition 2, which is new in our AsySZO+.

Definition 2 (Mixture Gradient)**.** *Given the coordinated smooth functions $f^j(x)$, $j = 1, \ldots, N$, the mixture gradient is a mixture gradient on the coordinated smooth functions as $\sum_{j=1}^N \nabla_j f^j(x)$.*

Because $f_i^j(x)$ ($i = 1, \ldots, l$) is an average smoothing version of $f_i(x)$ with the window size $2\mu_j$ as defined in (7), it is reasonable to assume that there exists a proportional constant $\widehat{L}$ between the original gradient $\nabla f_i(x)$ and the mixture gradient $\sum_{j=1}^N \nabla_j f_i^j(x)$ (see Figure 1), as Assumption 3. Specifically, if $[\mu_1, \mu_2, \ldots, \mu_N] = \mathbf{0}$, it is easy to verify that $\widehat{L} = 1$. If $\mu_j = \infty$ for all $j = 1, \ldots, N$, it is easy to verify that $\widehat{L} = 0$.

Assumption 3. *For any smooth function $f_i(x)$, ($i = 1, \ldots, l$), there exists a proportional constant $\widehat{L}$ between the original gradient $\nabla f_i(x)$ and the mixture gradient $\sum_{j=1}^N \nabla_j f_i^j(x)$ such that*

$$\left\| \sum_{j=1}^N \nabla_j f_i^j(x) \right\| \leq \widehat{L} \|\nabla f_i(x)\| . \tag{10}$$

In addition, it is reasonable to have a Lipschitz constant $\widetilde{L}$ on the mixture gradient $\sum_{j=1}^{N}\nabla_j f_i^j(x)$. Lemma 1 shows that there exists a Lipschitz constant $\widetilde{L}$ ($\widetilde{L} \leq \min\{NL, \sum_{j=1}^{N}\frac{L_0}{\mu_j}\}$) for the mixture gradient $\sum_{j=1}^{N}\nabla_j f_i^j(x)$ ($\forall i \in \{1,\ldots,l\}$), which is proved in Appendix.

Lemma 1 (Lipschitz constant for $\sum_{j=1}^{N}\nabla_j f_i^j(x)$)**.** *There exists a Lipschitz constant $\widetilde{L}$ for the mixture gradient $\sum_{j=1}^{N}\nabla_j f_i^j(x)$ ($\forall i \in \{1,\ldots,l\}$), such that, $\forall x$ and $\forall y$, we have*

$$\left\|\sum_{j=1}^{N}\nabla_j f_i^j(x) - \sum_{j=1}^{N}\nabla_j f_i^j(y)\right\| \leq \widetilde{L}\|x-y\|\,, \qquad (11)$$

where $\widetilde{L} \leq \min\{NL, \sum_{j=1}^{N}\frac{L_0}{\mu_j}\}$.

Remark 1. *Specifically, if $[\mu_1, \mu_2, \ldots, \mu_N] = \mathbf{0}$, it is easy to verify that $\widetilde{L} = L$. If $\mu_j = \infty$ for all $j = 1,\ldots,N$, it is easy to verify that $\widetilde{L} = 0$. Both the facts are consistent with the conclusion of Lemma 1.*

3.2.2. Inconsistent Reading

For the second difficulty, we give a reasonable representation to x_t^{s+1} which allows the conflicts of writing operations for different threads. Because AsySZO+ does not use the reading and writing locks, the vector $\widehat{x}_t^{s+1}$ read into the local memory may be inconsistent to the vector x_t^{s+1} in the shared memory, which means that some components of $\widehat{x}_t^{s+1}$ are different to the ones in x_t^{s+1}. It is the so-called inconsistent reading. For our AsySZO+, we give a reasonable representation to x_t^{s+1} which allows the conflicts of writing operations for different threads. The representation to x_t^{s+1} is written as:

$$x_t^{s+1} = \widehat{x}_t^{s+1} - \gamma\sum_{t'\in K(t)} B_{t'}^{s+1}\widehat{v}_{J(t')}^{s+1}\,, \qquad (12)$$

where $K(t)$ is a set of inner iterations with $t' \leq t-1$, $B_{t'}^{s+1}$ is a diagonal matrix with diagonal entries as either 1 or 0 (0 denotes that the corresponding coordinate is overwritten by other thread, and 1 denotes that the corresponding coordinate is written successfully by the current thread). It is reasonable to assume that there exists an upper bound τ such that $\tau \geq t - \min\{t'|t' \in K(t)\}$ (*i.e.*, Assumption 4).

Assumption 4 (Bound of delay)**.** *There exists an upper bound τ such that $\tau \geq t - \min\{t'|t' \in K(t)\}$ for all inner iterations t in AsySZO+.*

It is worth pointing out that, AsySZO (Lian et al., 2016) assumes $x_t = \widehat{x}_t - \gamma_t\sum_{t'\in K(t)} G_{J(t')}(\widehat{x}_t; f_i)$. This representation could not formulate the conflicts of two writing operations. Even worse, based on this representation, (Lian et al., 2016) implicitly assumes that they use the writing lock in the analysis.

3.3. Convergence Rate Analysis

In this section, we prove that the convergence rate of AsySZO+ is $O(\frac{1}{bT})$ (Theorem 2 and Corollary 1). Due to the limited space, all detailed proofs in this paper are presented in our Appendix.

Before providing Theorem 2, we first give an upper bound of $\sum_{t=0}^{m-1}\mathbb{E}\left\|\widehat{v}_t^{s+1}\right\|^2$ in Lemma 2.

Lemma 2. *If $Y - 2N\widetilde{L}^2\gamma^2\tau^2 > 0$, under Assumptions 3 and 4, we have that*

$$\sum_{t=0}^{m-1}\mathbb{E}\left\|\widehat{v}_t^{s+1}\right\|^2 \leq \frac{2Y}{Y - 2N\widetilde{L}^2\gamma^2\tau^2}\cdot \qquad (13)$$
$$\sum_{t=0}^{m-1}\left(\frac{2N\widetilde{L}^2}{b}\left\|x_t^{s+1} - \widetilde{x}^s\right\|^2 + 2\widehat{L}\mathbb{E}\left\|\nabla f(x_t^{s+1})\right\|^2\right)$$

Remark 2. *The upper bound in (13) shows that $\widehat{v}_{J(t)}^{s+1}$ would vanish after a large number of iterations. Thus, the steplength γ in our AsySZO+ can be set as a fixed constant instead of a decreasing steplength sequence $\{\gamma_t\}_{t=0,\ldots,m-1}$ used in AsySZO. The detailed setting of γ is given in Theorem 2.*

Because $f(x)$ is possibly non-convex, the global optimum solution cannot be guaranteed by AsySZO+. Thus, in our analysis, we use the gradient $\nabla f(x_t^{s+1})$ to analyze the convergence rate of AsySZO+ as following.

Theorem 2. *Let $c_m = 0$, $\gamma = \frac{u_0 b}{\widetilde{L}l^\alpha}$, $\beta_t = \frac{\widetilde{L}N^2}{Y}$, $0 < \alpha < 1$, $0 < u_0 < 1$, $c_t = c_{t+1}(1 + \gamma\beta_t) + \left(\frac{c_{t+1}N\gamma^2}{Y} + \frac{LY\gamma^2}{2N} + \frac{\gamma^3 NL^2\tau^2}{Y}\right)\frac{4YN\widetilde{L}^2}{b(Y-2N\widetilde{L}^2\gamma^2\tau^2)}$ for $t = 0,\ldots,m-1$, $\sigma = u_0\left(\frac{1}{2} - \left(\frac{13LYu_0 b}{5N\widetilde{L}} + \frac{8NL^2\tau^2u_0^2b^2}{5Y\widetilde{L}}\right)4\widehat{L}\right)$. Under Assumptions 2, 3 and 4, if $\tau < \min\left\{\frac{\frac{5\widetilde{L}N^2}{2Y} - \frac{2LY^2}{N^2}}{8L^2u_0 b}, \frac{(\frac{1}{8\widetilde{L}} - \frac{13LYu_0 b}{5N\widetilde{L}})5Y\widetilde{L}}{8NL^2u_0^2b^2}\right\}$, $\frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\mathbb{E}\left\|\nabla f(x_t^{s+1})\right\|^2$ in AsySZO+ satisfies the bound:*

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\mathbb{E}\left\|\nabla f(x_t^{s+1})\right\|^2 \qquad (14)$$
$$\leq \frac{\widetilde{L}l^\alpha\left(f(x^0)) - \mathbb{E}(f(x^S))\right)}{\sigma bT} + \frac{Nu_0\omega}{4\sigma}$$

Remark 3. *Theorem 2 shows that, AsySZO+ converges to a stationary point with the convergence rate $O(\frac{1}{bT})$. Note that the delay parameter τ is reflected in the parameter σ. In general , if τ is larger, σ is smaller.*

Remark 4. *If we force AsySZO+ to run on one core, we obtain a non-parallel* (i.e.*, sequential) version of AsySZO+, which is called as SZO+ in this paper. The convergence rate*

of SZO+ can be obtained according to Theorem 2 by setting $\tau = 0$.

Corollary 1. *Let* $c_m = 0$, $\gamma = \frac{u_0 b}{\widetilde{L} l^\alpha}$, $\beta_t = \frac{\widetilde{L} N^2}{Y}$, $0 < \alpha < 1$, $0 < u_0 < 1$, σ *is a small value independent to* l, *and* $c_t = c_{t+1}(1 + \gamma\beta_t) + \left(\frac{c_{t+1} N\gamma^2}{Y} + \frac{LY\gamma^2}{2N} + \frac{\gamma^3 NL^2\tau^2}{Y}\right) \frac{4YN\widetilde{L}^2}{b(Y - 2N\widetilde{L}^2\gamma^2\tau^2)}$ *for* $t = 0, \ldots, m-1$, $\sigma = u_0\left(\frac{1}{2} - \left(\frac{13LYu_0 b}{5N\widetilde{L}} + \frac{8NL^2\tau^2 u_0^2 b^2}{5Y\widetilde{L}}\right) 4\widehat{L}\right)$.

Under Assumptions 2, 3 and 4, if $\omega = 0$, *and* $\tau < \min\left\{\frac{\frac{5\widetilde{L}N^2}{2Y} - \frac{2LY^2}{N^2}}{8L^2 u_0 b}, \frac{\left(\frac{1}{8\widetilde{L}} - \frac{13LYu_0 b}{5N\widetilde{L}}\right) 5Y\widetilde{L}}{8NL^2 u_0^2 b^2}\right\}$, $\frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \mathbb{E}\left\|\nabla f(x_t^{s+1})\right\|^2$ *in AsySZO+ satisfies the bound:*

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \mathbb{E}\left\|\nabla f(x_t^{s+1})\right\|^2 \leq \frac{\widetilde{L} l^\alpha \left(f(x^0)) - \mathbb{E}(f(x^S))\right)}{\sigma b T}. \tag{15}$$

4. Experiments

In this section, we first describe the experimental setup. After that, we provide our experimental results with discussions.

4.1. Experimental Setup

We will provide the details of experiment design and algorithm implementations in this subsection.

4.1.1. Design of Experiments

In the experiments, we apply our AsySZO+ to an application of ensemble learning. As mentioned in (Dror et al., 2012), KDD-Cup 2011 challenged the community to identify user tastes in music by leveraging Yahoo! Music user ratings. The winning team from National Taiwan University created 237 models based on various machine learning algorithms (Chen et al., 2011). They combined the 237 models based on neural networks and binned linear regression, and created an ensemble model. We were able to obtain the predicted ratings of $N = 237$ individual models on the KDD-Cup Track 1 testing data set from the NTU KDD-Cup team, which is a matrix A with 6,005,940 rows (corresponding to the 6,005,940 samples in the testing data set) and 237 columns (corresponding to the outputs of 237 models on all samples in the testing set). We expect to learn an ensemble model which is a linear combination of these 237 models as Ax, where x includes the combination coefficients. The evaluation criteria for the combination coefficients x is to minimize the average error on the validation set as:

$$f(x) = \frac{1}{l}\sum_{i=1}^{l} (A_i x - c_i)^2 \tag{16}$$

where $l = 6,005,940$ is the size of the validation set, c is the corresponding true ratings in the validation set, and $A_i x$ is the predicted rating for the i-th sample by our ensemble model. For the KDD-Cup competition, we cannot obtain the true ratings. We only can obtain the output of function $f(x)$. Obviously, in this case, we cannot compute the gradient of $f(x)$. Thus, the zeroth-order approach is the only choice for optimizing the evaluation function $f(x)$. We compare our AsySZO+ algorithm with AsySZO in the application of ensemble learning. We also compare our SZO+ algorithm with SZO for non-parallel condition.

4.1.2. Implementation

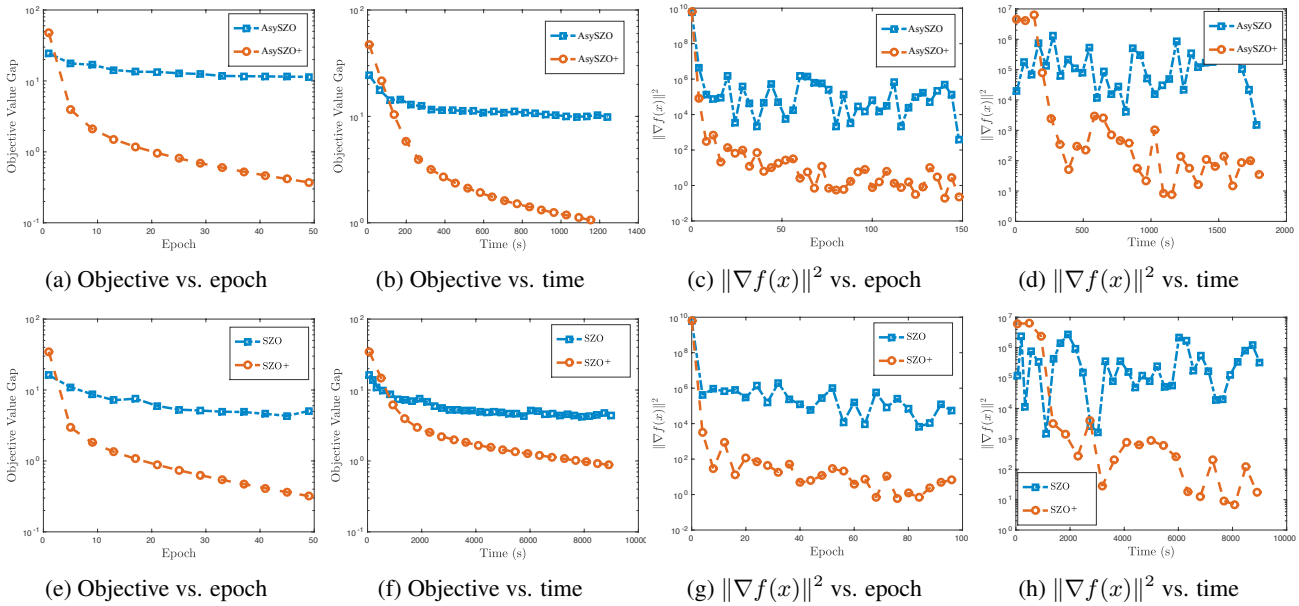
Our experiments are performed on a 32-core two-socket Intel Xeon E5-2699 machine where each socket has 16 cores. We implement our AsySZO+ in C++, where the shared memory parallel computation is handled via OpenMP (Chandra, 2001). Similarly, we implement AsySZO using C++ and OpenMP. We implement SZO+ and SZO by forcing AsySZO+ and AsySZO to run on one core.

4.2. Experimental Results and Discussions

We first compare the experimental results of AsySZO+ and AsySZO algorithms using 8 cores. The experimental results are plotted in Figures 2a-2d. Specifically, Figures 2a and 2b show the convergence of objective value of (16) vs. the epoch and running time, respectively, for AsySZO+ and AsySZO. Figures 2c and 2d compare the convergence of $\|\nabla f(x)\|^2$ of (16) vs. the epoch and running time, respectively, for AsySZO+ and AsySZO. The results confirm that our new AsySZO+ has a faster convergence rate than existing AsySZO.

We also conduct the experiments of SZO+ and SZO algorithms using 1 core, and Figures 2e-2h display the empirical results. Figures 2e and 2f present the convergence of objective value of (16) vs. the epoch and running time, respectively, for SZO+ and SZO. Figures 2g and 2h illustrate the convergence of $\|\nabla f(x)\|^2$ of (16) vs. the epoch and running time, respectively, for SZO+ and SZO algorithms. The results confirm that our new SZO+ has a faster convergence rate than existing SZO.

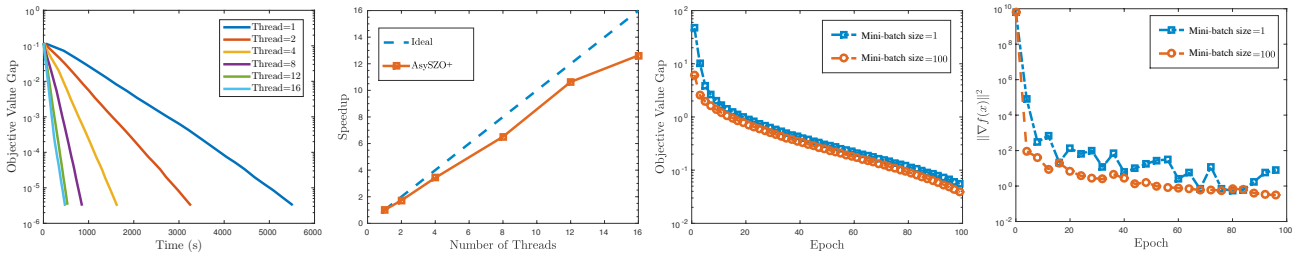
To evaluate the scalability of our AsySZO+ algorithm, we perform AsySZO+ on 1, 2, 4, 8, 12 and 16 cores, respectively, to observe the speedup of AsySZO+. Figures 3a and 3b display the speedup results of AsySZO+, which show that AsySZO+ can have a near-linear speedup on a parallel system with shared memory. The reason is that we do not



(a) Objective vs. epoch (b) Objective vs. time (c) $\|\nabla f(x)\|^2$ vs. epoch (d) $\|\nabla f(x)\|^2$ vs. time

(e) Objective vs. epoch (f) Objective vs. time (g) $\|\nabla f(x)\|^2$ vs. epoch (h) $\|\nabla f(x)\|^2$ vs. time

Figure 2. Comparison of convergence of AsySZO+ (or SZO+) and of AsySZO (or SZO). (a-d) display the experimental results of AsySZO+ and AsySZO using 8 cores. (e-h) compare the results of SZO+ and SZO using 1 core.



(a) Objective vs. time (b) Speedup of AsySZO+ (c) Objective vs. epoch (d) $\|\nabla f(x)\|^2$ vs. epoch

Figure 3. (a) and (b) show the speedup of AsySZO+ with different cores. (c) and (d) show the convergence of AsySZO+ with different mini-batch sizes 1 and 100.

use any lock in the implementation of AsySZO+.

To check the speedup of the mini-batch method, we perform AsySZO+ algorithm with different mini-batch sizes 1 and 100. Figures 3c and 3d present the convergence of objective value and $\|\nabla f(x)\|^2$ of the function (16) vs. the epoch with the mini-batch sizes 1 and 100, respectively, for AsySZO+. The results confirm that the mini-batch method can speed up the asynchronous stochastic zeroth-order algorithm. However, it cannot provide a significant improvement compared with the SVRG technique. The reason is that, more variance is reduced for stochastic zeroth-order gradients, more speedup can be achieved for the stochastic optimization algorithm. Therefore, the mini-batch technique cannot reduce the variance of stochastic zeroth-order gradients as significantly as the SVRG technique.

5. Conclusion

The convergence rate of the newly proposed AsySZO algorithm (Lian et al., 2016) is $O(\frac{1}{\sqrt{T}})$, which is significantly slower than $O(\frac{1}{T})$, the best convergence rate of (asynchronous) stochastic gradient algorithm. To fill this important gap, in this paper, we first point out the fundamental reasons leading to the slow convergence rate of AsySZO. After that, we propose a faster asynchronous stochastic zeroth-order optimization algorithm (AsySZO+). We prove that our AsySZO+ has a faster convergence rate $O(\frac{1}{bT})$ (b is the mini-batch size) via rigorous theoretical analysis, which is a significant improvement on the convergence rate $O\left(\frac{1}{\sqrt{T}}\right)$ for AsySZO. The experimental results on the application of ensemble learning confirm that our AsySZO+ has a faster convergence rate than the existing (asynchronous) stochastic zeroth-order algorithms.

Acknowledgement

B.G. and Z.H. were partially supported by U.S. NIH R01 AG049371, NSF IIS 1302675, IIS 1344152, DBI 1356628, IIS 1619308, IIS 1633753. C.D. was partially supported by the National Natural Science Foundation of China under Grants 61572388, 61703327, the National Key Research and Development Program of China (2017YFE0104100), and in part by the Key R&D Program-The Key Industry Innovation Chain of Shaanxi under Grant 2017ZDCXL-GY-05-04-02.

References

Bach, F. and Perchet, V. Highly-smooth zero-th order online optimization. In *Conference on Learning Theory*, pp. 257–283, 2016.

Bubeck, S., Cesa-Bianchi, N., et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5 (1):1–122, 2012.

Byrd, R. H., Hansen, S., Nocedal, J., and Singer, Y. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

Chandra, R. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

Chen, P.-L., Tsai, C.-T., Chen, Y.-N., Chou, K.-C., Li, C.-L., Tsai, C.-H., Wu, K.-W., Chou, Y.-C., Li, C.-Y., Lin, W.-S., et al. A linear ensemble of individual and blended models for music rating prediction. In *Proceedings of the 2011 International Conference on KDD Cup 2011-Volume 18*, pp. 21–60. JMLR. org, 2011.

Conroy, B. and Sajda, P. Fast, exact model selection and permutation testing for l2-regularized logistic regression. In *Artificial Intelligence and Statistics*, pp. 246–254, 2012.

Dror, G., Koenigstein, N., Koren, Y., and Weimer, M. The yahoo! music dataset and kdd-cup'11. In *KDD Cup*, pp. 8–18, 2012.

Duchi, J. C., Bartlett, P. L., and Wainwright, M. J. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Gu, B., Huo, Z., and Huang, H. Zeroth-order asynchronous doubly stochastic algorithm with variance reduction. *arXiv preprint arXiv:1612.01425*, 2016.

Gu, B., Huo, Z., and Huang, H. Asynchronous doubly stochastic group regularized learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1791–1800, 2018a.

Gu, B., Miao, X., Huo, Z., and Huang, H. Asynchronous doubly stochastic sparse kernel learning. In *AAAI*, 2018b.

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

Li, M., Zhang, T., Chen, Y., and Smola, A. J. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670. ACM, 2014.

Lian, X., Huang, Y., Li, Y., and Liu, J. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 2737–2745, 2015.

Lian, X., Zhang, H., Hsieh, C.-J., Huang, Y., and Liu, J. A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order. In *Advances in Neural Information Processing Systems*, pp. 3054–3062, 2016.

Mania, H., Pan, X., Papailiopoulos, D., Recht, B., Ramchandran, K., and Jordan, M. I. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv preprint arXiv:1507.06970*, 2015.

Nesterov, Y. and Spokoiny, V. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, pp. 1–40, 2011.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 693–701, 2011.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pp. 314–323, 2016.

Shen, X., Alam, M., Fikse, F., and Rönnegård, L. A novel generalized ridge regression method for quantitative genetics. *Genetics*, 193(4):1255–1268, 2013.

Suykens, J. A. and Vandewalle, J. Least squares support vector machine classifiers. *Neural processing letters*, 9 (3):293–300, 1999.

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pp. 896–903. ACM, 2005.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Zhao, S.-Y. and Li, W.-J. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.